0 x = } 0 1 0 :) 1 0

(y = 0 * > 0 1 : 1 @ [/

1 @ 1 0 : < / 0 * 0 1 # %

PROGRAMOZÁS GYEREKEKNEK

1 @ 1 0 : < / 0 *

LÉPÉSRŐL LÉPÉSRE

0 1 : / y # } : 1 1 0 /

x 0 @ (1 0 1 0 0 1 < $

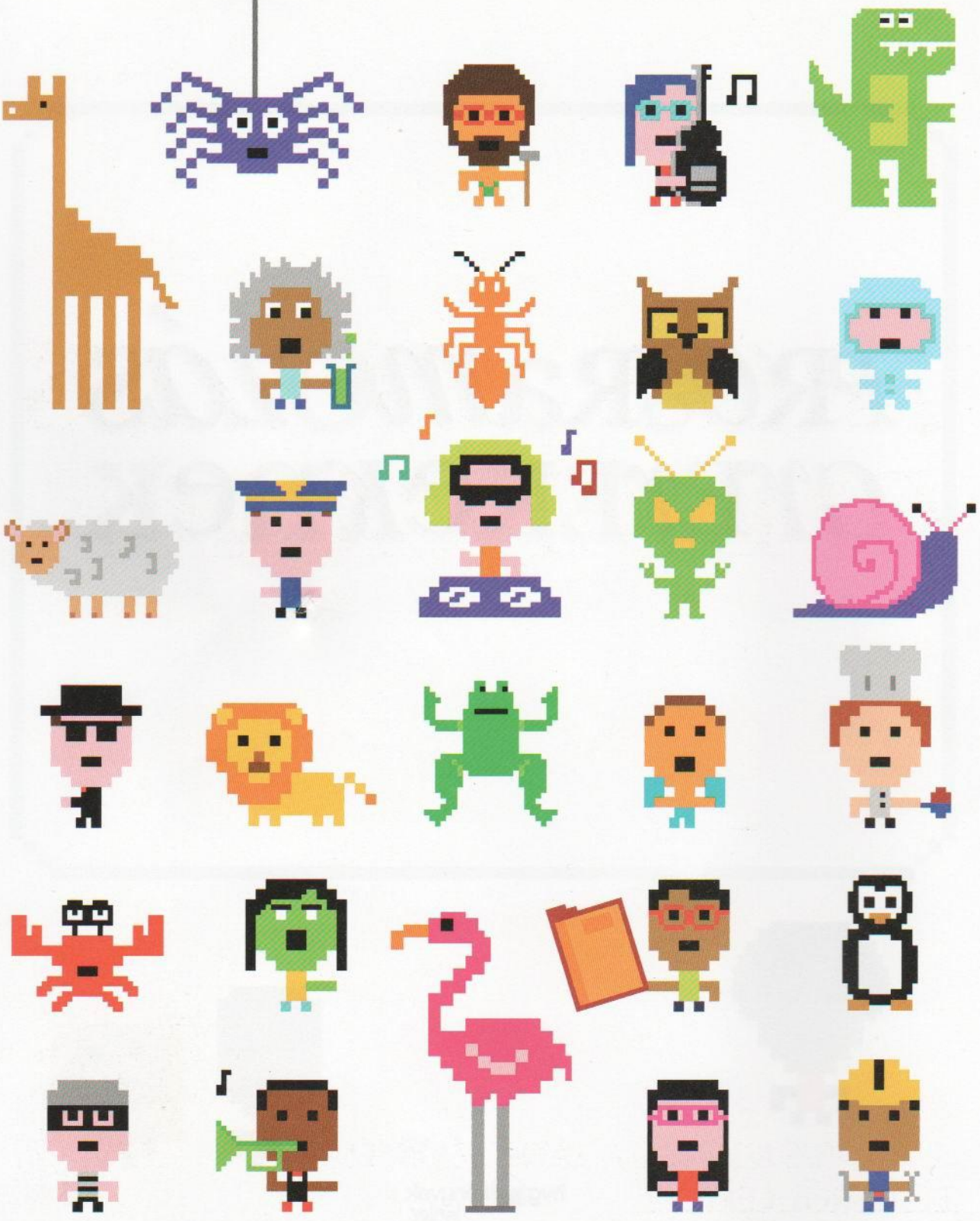
1 } 0 \ = * 0 : 1 > 1)

A BINÁRIS KÓDTÓL A JÁTÉKKÉSZÍTÉSIG

) 1 0

CAROL VORDERMAN

1 @ [/



PROGRAMOZÁS GYEREKEKNEK

A BINÁRIS KÓDTÓL A JÁTÉKKÉSZÍTÉSIG
LÉPÉSRŐL LÉPÉSRE

LONDON, NEW YORK, MELBOURNE,
MUNICH AND DELHI

A fordítás alapja:
Carol Vorderman: *Computer Coding for Kids: A Unique Step-By-Step Visual Guide From Binary Code to Building Games*
First published in Great Britain, London, 2014

Szakmai lektor: Molnár-Sáska Gábor
Szerkesztette: Mártonfi Attila

HVG Könyvek
Kiadóvezető: Budaházy Árpád
Felelős szerkesztő: Szűcs Adrienn

ISBN 978-963-304-320-2

Kiadja a HVG Kiadó Zrt., Budapest, 2016
Felelős kiadó: Szauer Péter

www.hvgkonyvek.hu

Nyomdai előkészítés: HVG Press Kft.
Felelős vezető: Tóth Péter

Nyomás: TBB, Szlovákia

CAROL VORDERMAN Nagy-Britannia egyik legnépszerűbb műsorvezetője, matematikai tudása közismert. A Cambridge-i Egyetemen szerzett mérnöki diplomát. Carol komoly érdeklődést mutat a programozás iránt, és meggyőződéssel vallja, hogy minden gyereknek lehetőséget kellene adni, hogy ezt az értékes tudást elsajátíthassa. Tudományokat népszerűsítő tévéműsorok házigazdája volt a BBC-n és egyéb brit csatornákon. Legyen szó akár 26 éves tudományos műsorvezetői múltjáról, akár arról, hogy az elmúlt évtized egyik legnépszerűbb ismeretterjesztő írója lett, vagy hogy David Cameron brit miniszterelnöknek a matematikaoktatás jövőjéről ad tanácsot, Carol szenvedélyesen elkötelezett a matematika, a tudomány és a technológia érdekes, közérthető módon történő népszerűsítése mellett.

DR. JON WOODCOCK az Oxfordi Egyetemen szerzett fizikusi diplomát, majd Londonban PhD-fokozatot asztrofizikából. Nyolcéves korában kezdett el programozni, és azóta a legkisebb mikrokontrollerektől a szuperszámítógépekig mindenen dolgozott. Munkái között akad összetett világűr-szimuláció, informatikai nagyvállalatok részére végzett kutatás, de még hulladékokból készített intelligens robot is. Szenvedélye a tudomány és a technológia oktatása, előadások tartása a világűrről, valamint programozó szakkörök szervezése. Számos tudományos és technikai könyv társszerzője.

SEAN MCMANUS kilencéves korában tanult meg programozni a Logo nyelvet használva. Ma elismert szakíró és újságíró. Olyan népszerű könyvek szerzője, mint a *Scratch Programming in Easy Steps*, a *Web Desing in Easy Steps* és a *Raspberry Pi For Dummies*. A www.sean.co.uk weboldalán Scratch-játékok és további hasznos tippek találhatók.

CRAIG STEELE a számítástechnika-oktatás szakértője. A fiatalok számára ingyenes programozóklubokat szervező CoderDojo Scotland projekt menedzsere. Korábban skót tudományos, oktatási és kutatóintézetek részére dolgozott. Első számítógépe egy ZX Spectrum volt.

CLAIRE QUIGLEY informatikusi diplomát, majd PhD-címet szerzett a Glasgow-i Egyetemen. Dolgozott a Cambridge-i Egyetem számítástechnikai laboratóriumában, valamint részt vett egy projektben, amelynek célja az általános iskolás diákok algoritmikus gondolkodásának kialakítása. A skóciai CoderDojo Scotland programozóklub mentora.

DANIEL MCCAFFERTY informatikusként végzett a Strathclyde-i Egyetemen, azóta a világ vezető befektetési bankjai számára készít számítógépes programokat. Szabadidejében a skóciai CoderDojo Scotland programozóklub mentora.

TARTALOM

4 A SZÁMÍTÓGÉP BELSEJE

5 PROGRAMOZÁS A VALÓSÁGBAN

ELŐSZÓ

Alig néhány éve a számítógép-programozás még egy titokzatos, csak a beavatottak számára elérhető képességnek tűnt. A legtöbb embernek igencsak furcsa volt, hogy a programozás öröm forrása lehet. A világ azonban megváltozott. Az internet, az e-mail, a közösségi hálók, az okostelefonok és az appok berobbantak a köztudatba, és örökre megváltoztatták az életünket.

Ma már természetesnek vesszük, hogy a számítógépek a mindennapok részei. Telefonhívás helyett sms-t küldünk vagy a közösségi hálót használjuk. A vásárlástól és a szórakozástól kezdve a hírekig és játékokig mindenevők vagyunk, amit csak a számítógép kínál. De nemcsak használhatjuk ezt a technológiát, alkothatunk is általa. Ha megtanulunk programozni, saját magunk is készíthetünk digitális remekműveket.

Minden számítógépet olyan programsorok vezérelnek, amelyeket valaki egyszer már megírt. Elsőre egy érthetetlen, idegen nyelvnek tűnhet, de bárki könnyen megtanulhatja. Sokak szerint a programozás a 21. század egyik legfontosabb képessége.

A programozás elsajátítása rengeteg örömmel jár, mert a tanulási folyamat minden szintjén állandó a sikerélmény. Saját játékokat készíteni élvezetesen egyszerű és magával ragadóan kreatív tevékenység. Ez talán az első olyan tudomány, amely egyesíti a művészetet, a logikát, a történetmesélést és az üzleti gondolkodást.

Ezenfelül rendkívül hasznos is. Fejleszti a logikus gondolkodást és a problémamegoldó képességet. Ez a tudás nem csupán a műszaki tudományokban, hanem szinte bármilyen munkaterületen létfontosságú. A programozási ismeretekkel rendelkezők iránti kereslet hatalmas mértékben növekszik világszerte, és nincs elegendő képzett szakember. Tanulj meg programozni, és a digitális világ kinyílik előtted!

Carol Vorderman

CAROL VORDERMAN

Hogyan használd a könyvet?

A könyv bemutatja a programozás megértéséhez szükséges alapelveket. Ezeket játékos példák segítségével a gyakorlatban is kipróbálhatod. Minden téma egyszerű, könnyen követhető lépésekre van bontva.

A pixeles figurák hasznos tanácsokat adnak.

Az egyes témákat példák és gyakorlatok mutatják be

A „LÁSD MÉG" feliratú kiemelt részek kapcsolódó témákra utalnak

A színes rajzok segítik megérteni a leírtakat

Programok és blokkok magyarázata lépésről lépésre

Pontos utasítások, hogy mikor mire kell kattintani

Az egyes lépések részletes magyarázatai

BUBORÉKPUKKASZTÓ

Két pont távolságának kiszámítás

Ebben a játékban és sok másikban is fontos, hog
távolságát. Ezt a Pitagorasz-tétel segítségével fo

11 Ez a függvény kiszámolja két objektum távolságát. Írd ezt a függvényt a 9. pontban írt függvény után!

```
from math import sqrt
def távolság(id1, id2):
    x1, y1 = koord_kér(id1)
    x2, y2 = koord_kér(id2)
    return sqrt((x2 - x1)**2 + (y2
```


Bújócska

Üdvözlünk a speciális effektek világában! Figyeld meg, hogy a lila „Kinézet" blokkok használatával a szereplők minként tűnnek el, majd jelennek meg újra, nőnek meg és mennek össze, halványulnak el és válnak élessé újra.

LÁSD MÉG
‹ 38–39 Mozgásban
Üzenetküldés 70–71 ›

Szereplők elrejtése

A „tűnj el" blokk segítségével elrejtheted a szereplőt. Továbbra is mozog a játéktéren, de nem látszik, amíg a „jelenj meg" blokk újra előre nem hozza.

A „tűnj el" blokk elrejti a szereplőt a játékban

▷ **Látható és láthatatlan**
A szereplő elrejtéséhez használd a „tűnj el" blokkot! Ha újra látni akarod, a „jelenj meg" blokk előhozza. Mindkét blokkot a „Kinézet" csoportban találod.

tűnj el

jelenj meg

▽ **Bujkáló macska**
Próbáld ki ezt az utasítássorozatot a macskával! Eltűnik, majd újra megjelenik, de akkor is folyamatosan mozog, amikor nem látszik.

-ra kattintva
mindig
várj 1 mp-et — Láthatatlanná teszi a macskát
tűnj el
fordulj 90 fokot — Elforgatja a macskát az óramutató járásával megegyező irányban
menj 100 lépést — A macska tovább mozog akkor is, amikor láthatatlan
várj 1 mp-et
jelenj meg — Újra láthatóvá teszi a macskát

TANÁCSOK

Szereplők láthatósága

Válassz ki egy szereplőt a listából! A kis „i" betűre kattintva nyisd meg a beállításokat! Itt a „látható" jelölőnégyzettel tudod beállítani, hogy a szereplő látható legyen-e, vagy sem.

Sprite1
x: 84 y: −59 irány: −90°
forgási stílus:
játékos vonszolhatja:
látható:

Láthatatlan szereplő megjelenítése

Méretek és hatások

Utasításokkal változtathatod a szereplők méretét, és különleges hatásokat, effekteket is alkalmazhatsz.

méret változzon 10 — Pozitív számokkal növeled, negatívokkal pedig csökkented a szereplő méretét

méret legyen 100 % — A nagyobb számok növelik, a kisebbek csökkentik a szereplő méretét. A 100% az eredeti méret

△ **Szereplő méretének megváltoztatása**
Ez a két blokk használható a szereplők méretének változtatására adott értékkel vagy százalékosan

Az összes hatás törlése

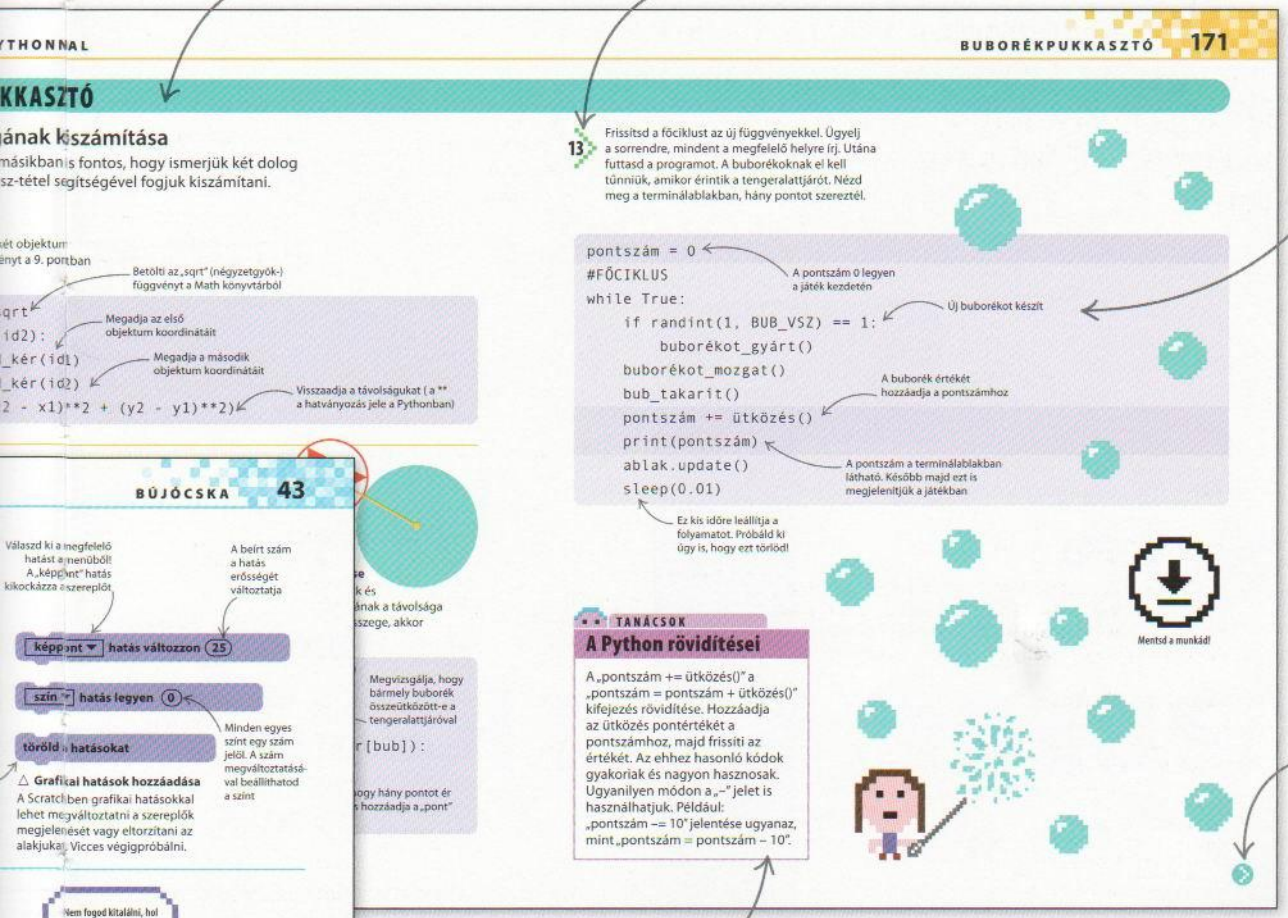
Teleportálás hatások használatával

Válassz ki egy szellem szereplőt a „Fantázia" kategóriából, és add hozzá az alábbi utasítássorozatot! Kattintásra a szellem teleportálni fog.

e szereplőre kattintáskor
töröld a hatásokat
ismételd 20
szellem hatás változzon 5 — A „szellem" hatás elhalványítja a szereplőt. Ha 20-szor megismétled, a szereplő teljesen eltűnik
csússz 0,1 mp-ig x: véletlen −150 és 150 y: véletlen −150 és 150 — Ez a „Műveletek" blokk véletlenszerű vízszintes pozíciót határoz meg
ismételd 20
szellem hatás változzon −5 — Ez a blokk jeleníti meg újra a szereplőt

A könyvben hét projekt található. A projektoldalakat kék csík jelöli

Világos magyarázatok vezetnek lépésről lépésre végig a projekt során



Az utasítássorok magyarázatainak köszönhetően nem lehet hibázni

Ez a nyíl jelzi, hogy a projekt a következő oldalon folytatódik

A keretes szövegek szaktanácsokat, magyarázatokat és összefoglalókat tartalmaznak

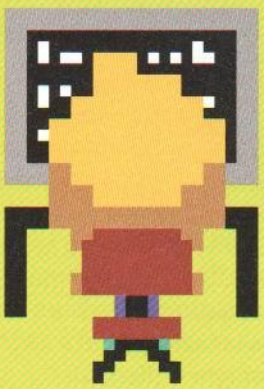
TANÁCSOK

Mentés

Ez a jel emlékeztet arra, hogy gyakran mentsd el a munkád, nehogy elvesszen valami, ha esetleg lefagy a számítógép.

Mentsd el a munkád!

Mi a programozás?



Mi a számítógépes program?

LÁSD MÉG

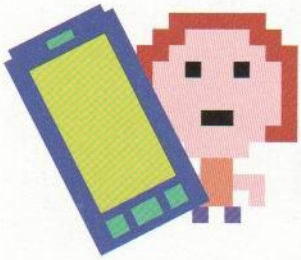
Gondolkodj úgy, mint egy számítógép! **16–17 ›**

Legyél te is programozó! **18–19 ›**

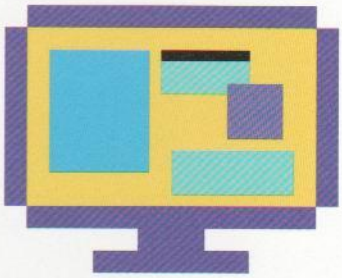
A számítógépes program utasítások sorozata, amelyeket követve a gép elvégez egy feladatot. A programozás olyan, lépésről lépésre végrehajtandó utasítások megírását jelenti, amelyekkel megmondjuk a számítógépnek, hogy mit csináljon.

Számítógépes programok mindenhol

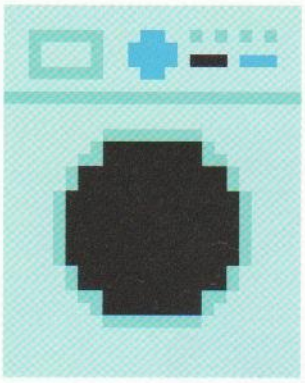
A számítógépes programok körülvesznek bennünket. Ezek vezérlik a legtöbb eszközt és kütyüt, programozók által írt utasításokat követnek.



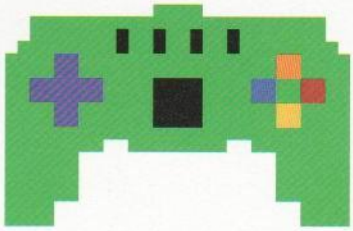
◁ **Mobiltelefonok**
Programok teszik lehetővé a telefonhívást vagy az sms-küldést is. Ha felhívunk valakit, program keresi meg a telefonszámát.



△ **Szoftver**
Mindazt, amit egy számítógéppel meg tudunk csinálni – mint például az internet böngészése, a szövegszerkesztés vagy a zenehallgatás – programozók által megírt program teszi lehetővé.

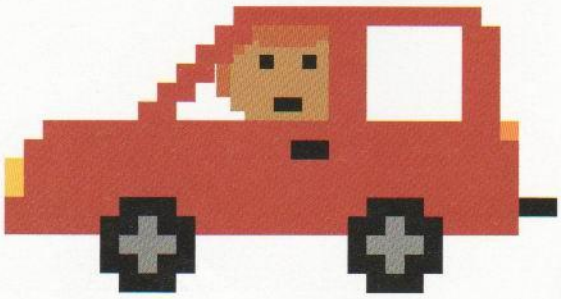


△ **Mosógépek**
A mosógépek különböző programokat hajtanak végre. Számítógépes program ellenőrzi a víz hőmérsékletét és a mosási időt.



◁ **Játékok**
A játékkonzolok tulajdonképpen számítógépek, amelyeken játékprogramok futnak. A grafikát, a hangot és a vezérlést is számítógépes kód hozza létre.

▷ **Autók**
Sok autóban számítógép figyeli a sebességet, a hőmérsékletet vagy az üzemanyagszintet. A biztonság érdekében a számítógép a fék működésében is szerepet játszik.



Hogyan működik a számítógépes program?

A számítógép nagyon okosnak tűnhet, de valójában csak egy doboz, amely gyorsan és pontosan követi az utasításokat. Az emberi értelem az, ami képes létrehozni a programokat a legkülönfélébb feladatok végrehajtására.

1 A számítógép nem tud gondolkodni

A számítógép semmit nem tud megtenni anélkül, hogy a programozó el ne látná a megfelelő utasításokkal.

Utasítások nélkül a számítógép tanácstalan

2 A programozás

Részletes utasítások sorozatával, azaz program megírásával lehet a számítógépet vezérelni. Ha az utasítások pontatlanok, a számítógép sem azt fogja tenni, amit várunk tőle.

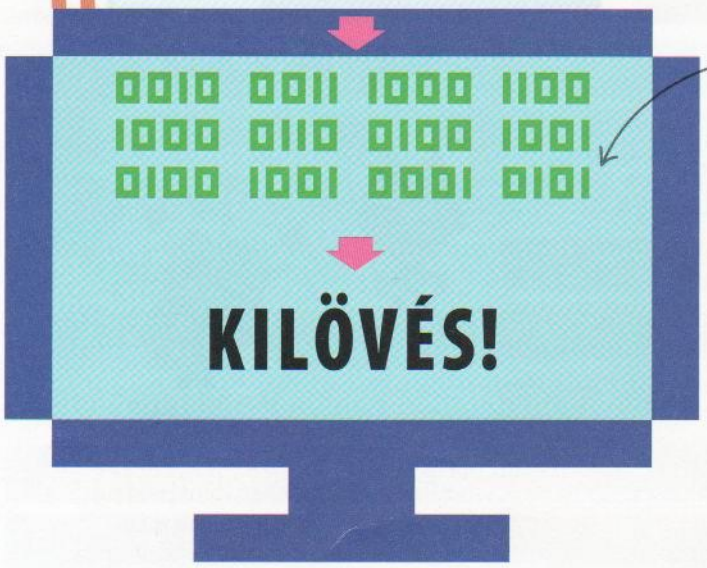
Ez egy visszaszámlálást végző program

```
for count in range(10, 0, -1):
    print("Visszaszámlálás", count)
```

3 Programnyelvek

A számítógép csak olyan nyelven írt parancsokat tud végrehajtani, amelyet megért. A programozó dolga kiválasztani a legalkalmasabb nyelvet az adott feladatra.

```
for count in range(10, 0, -1):
    print("Visszaszámlálás", count)
```



Minden programot végül lefordítanak bináris kódra, amely csak egyesekből és nullákból áll

FOGALOMTÁR

Hardver és szoftver

Hardvernek nevezzük a számítógép minden kézzelfogható részét (pl. áramkörök, vezetékek, billentyűzet, képernyő).
A szoftver a számítógépen futó összes programot jelenti.
A számítógép a szoftver és a hardver együttműködésével képes a feladatok ellátására.

Gondolkodj úgy, mint egy számítógép!

LÁSD MÉG
❮ 14–15 Mi a számítógépes program?
Legyél te is programozó! 18–19 ❯

A programozónak meg kell tanulnia számítógép módjára gondolkodni. Minden feladatot olyan kis lépésekre bontva kell a gépnek megadni, hogy egyértelmű és pontos legyen a végrehajtása.

A robot gondolkodása

Képzelj el egy éttermet, ahol a felszolgáló egy robot! A robot agya egy számítógép. Ahhoz, hogy kivigye az ételt az étterem konyhájából az asztaloknál ülő vendégeknek, utasításokat kell adni. Először a feladatot olyan kis lépésekre kell bontani, hogy a számítógép megértse.

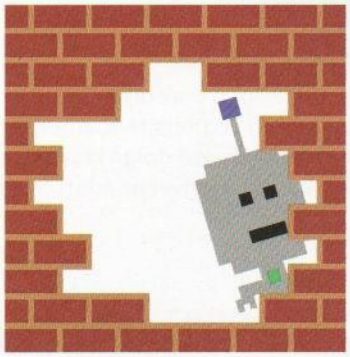
FOGALOMTÁR

Algoritmus

Az algoritmus egy feladat megoldására alkalmazott elemi lépések sorozata. A program egy programnyelv segítségével leírt algoritmus.

1 **1. számú felszolgálóprogram**
Ezt a programot végrehajtva a robot felemeli az ételt, áttöri a konyha falát, és leteszi a tányért a földre. Az algoritmus nem volt elég részletes.

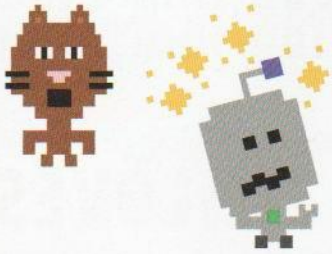
1. Emeld fel az ételt!
2. Menj el a konyhából az asztalig!
3. Tedd le az ételt!



◁ **Katasztrófa!**
Az utasítások nem voltak egyértelműek: elfelejtettük megmondani a robotnak, hogy használja az ajtót. Ez az embernek természetes, de a gép nem tud gondolkodni.

2 **2. számú felszolgálóprogram**
Ezúttal megmondtuk a robotnak, hogy használja az ajtót, de utána megbotlott a macskában, és összetörte a tányért.

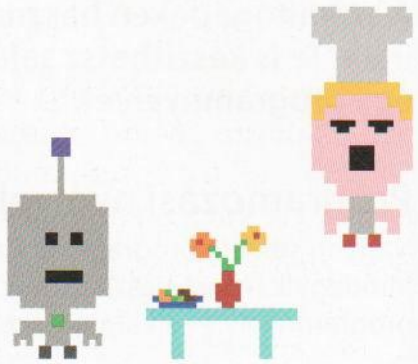
1. Emeld fel a tányért az étellel!
2. Menj el a konyhából az asztalig:
   - Menj a konyhaajtóhoz!
   - Menj az ajtótól az asztalig!
3. Tedd le a tányért az asztalra a vendég elé!



△ **Még nem tökéletes**
A robot nem tudja, hogyan kezelje az olyan akadályokat, mint például egy macska. A programnak még részletesebb utasításokból kell állnia.

3 **3. számú felszolgálóprogram**
Ebben a változatban a robot sikeresen célba juttatja a tányért, elkerülve minden akadályt. Ezután azonban ott marad az asztalnál, miközben a kihordásra váró étel gyűlik a konyhában.

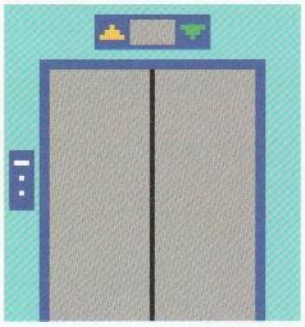
1. Emeld fel a tányért az étellel, és vidd mindig vízszintesen!
2. Menj el a konyhából az asztalig:
    - Menj a konyhaajtóhoz,
        - közben figyelj az akadályokra, és kerüld ki őket!
    - Menj az ajtótól az asztalig,
        - közben figyelj az akadályokra, és kerüld ki őket!
3. Tedd le a tányért az asztalra a vendég elé!



△ **Sikerült?**
A robot végre biztonságban felszolgálja az ételt. Arra azonban elfelejtettük utasítani, hogy utána menjen vissza a konyhába a következő tányérért.

Valós példa

A pincérrobotot csak elképzeltük, de az ilyen algoritmusok igen gyakoriak. Például egy számítógép-vezérelt lift hasonló problémákkal néz szembe. Felfelé menjen, vagy lefelé? Melyik emeleten álljon meg?



1. Várj, amíg az ajtó becsukódik!
2. Várj, amíg megnyomják a gombot!
    - Ha a kiválasztott emelet feljebb van, mint a jelenlegi:
        - menj felfelé!
    - Ha a kiválasztott emelet lejjebb van, mint a jelenlegi:
        - menj lefelé!
3. Ha a jelenlegi emelet azonos a kiválasztottal, állj meg!
4. Nyisd ki az ajtót!

◁ **Liftprogram**
A lift kifogástalan és biztonságos működéséhez minden lépésnek pontosnak, egyértelműnek kell lennie, és minden lehetőséget kezelnie kell. A programozóknak meg kell bizonyosodniuk arról, hogy megfelelő algoritmust készítettek.

Legyél te is programozó!

A számítógépeken használt programokat programozók írják. Te is készíthetsz saját programokat, ha megtanulsz egy programnyelvet.

LÁSD MÉG
Mi a Scratch? 22–23 ›
Mi a Python? 86–87 ›

Programozási nyelvek

Nagyon sokféle programozási nyelv közül lehet választani. Mindegyik másra használható. Íme, néhány népszerű programnyelv és alkalmazási területei:

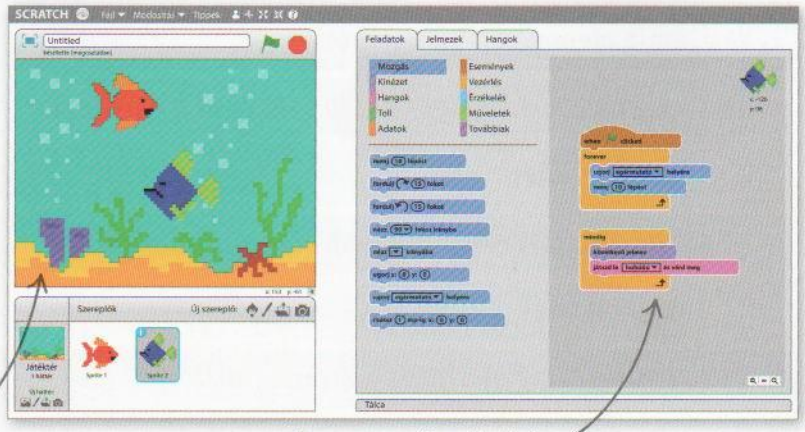
C	Hatékony programnyelv számítógépes operációs rendszerek írására.
Ada	Űrhajók, műholdak és repülőgépek irányítására alkalmas.
Java	Számítógépeken, tableteken és mobiltelefonokon egyaránt használható.
MATLAB	Sok számítási feladatot tartalmazó programokhoz ideális.
Ruby	Sok információ automatikus kezelése a weboldalakon.
JavaScript	Interaktív weboldalak készítése.

Scratch Vizuális programnyelv, amely kiváló a programozás megtanulására. Ezzel foglalkozik a könyv első része.

Python Egy nagyon sokoldalú, úgynevezett szkriptnyelv. A könyv második része erről szól.

Mi a Scratch?

A Scratch kiváló a programozás alapjainak elsajátításához. Az utasítások begépelése helyett a programokat utasításblokkokból lehet összeépíteni. Egyszerű a használata, ráadásul megtanítja a programozás alapelveit.



Itt látható a program működésének eredménye

A program színes blokkok összeillesztésével készül

Mi a Python?

A Python programnyelvet világszerte használják játékok, különféle eszközök és weboldalak készítésére. Érdemes megtanulni, mert nagyon sokféle programot lehet készíteni vele. Angol szavak és jelek keveréke.

IDLE File Edit Shell Debug Window Help

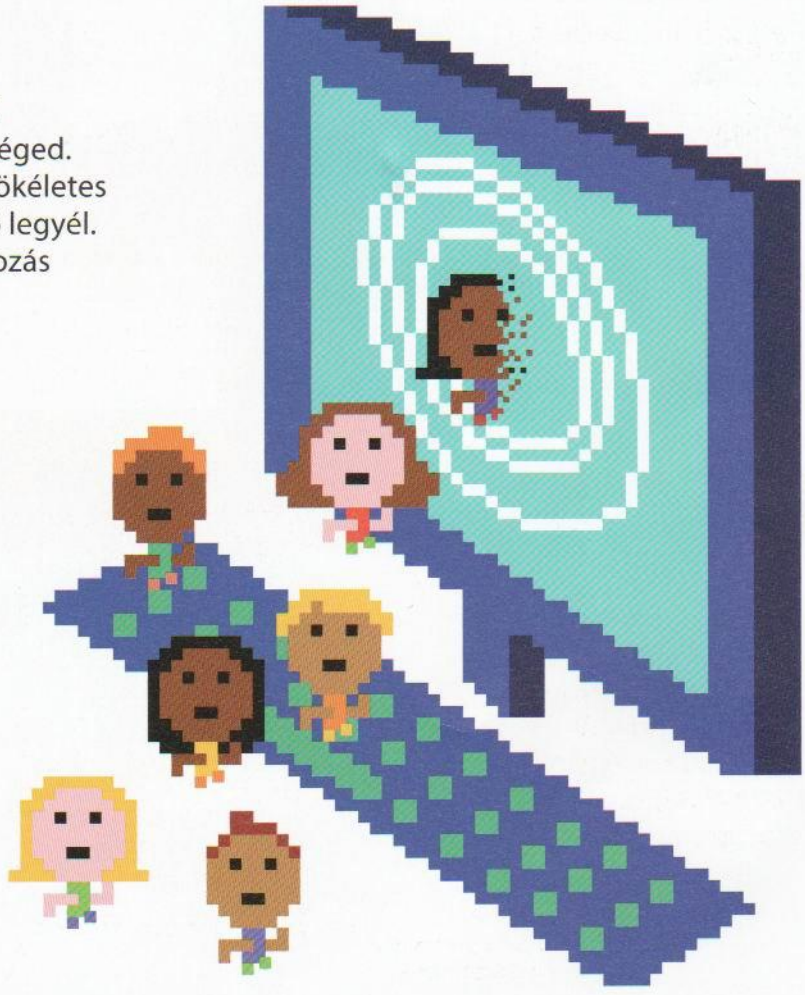
Kísértetház

```
# Kísértetház
from random import randint
print('Kísértetház')
bátor_vagyok = True
pontszám = 0
while bátor_vagyok:
    szellem_ajtó = randint(1, 3)
    print('Három ajtó van előtted...')
```

Python nyelven írt program

Láss hozzá!

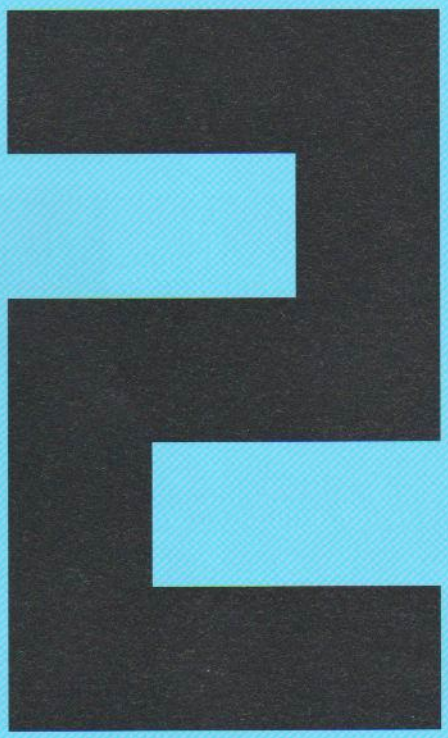
Ideje elkezdeni a programozást. Mindössze egy internetkapcsolattal rendelkező számítógépre van szükséged. A könyv a Scratch nyelvvel indul – tökéletes kezdet, hogy akár profi programozó legyél. Ugorj fejest a számítógép-programozás izgalmas világába!



TANÁCSOK

Kísérletezz bátran!

Programozóként kísérletezhetsz a programjaiddal. A tanulás egyik legjobb módja, ha megfigyeled, mi történik, amikor valamin változtatsz. Próbálgatással új megoldásokat fedezhetsz fel, sokat tanulhatsz a programozásról és még élvezni is fogod.



Kezdjük a Scratchcsel!

Mi a Scratch?

A Scratch egy vizuális programnyelv, amellyel egyszerű a programozás. Sokféle szórakoztató és érdekes program készítésére alkalmas.

LÁSD MÉG

A Scratch telepítése és indítása **24–25** ❯

A Scratch felülete **26–27** ❯

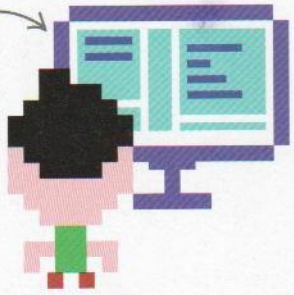
Színes blokkok és utasítások **30–31** ❯

A Scratch alapjai

A Scratch kiváló eszköz játékok és animációk készítésére. Menő grafikák és hangok gyűjteménye (könyvtárak), amelyekkel kedvedre kísérletezhetsz.

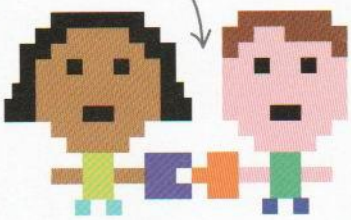
1 Kezdj el programozni!
A Scratch egy programnyelv. Könnyű a használata, nem kell sokat gépelni.

Készítsd el az első programodat Scratchben!



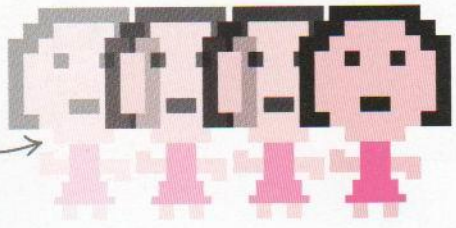
2 Illeszd össze a blokkokat!
A Scratchben színes blokkok alkotják a programot. Az egymáshoz illesztett blokkok képezik az utasítássorozatokat.

A blokkok úgy illeszkednek, mint a puzzle



3 Irányítsd és beszéltesd a szereplőket!
A programokban különböző szereplőket használhatsz, például embereket, állatokat vagy járműveket. A szereplőket lehet mozgatni és beszéltetni.

A szereplők tudnak sétálni, futni és táncolni is



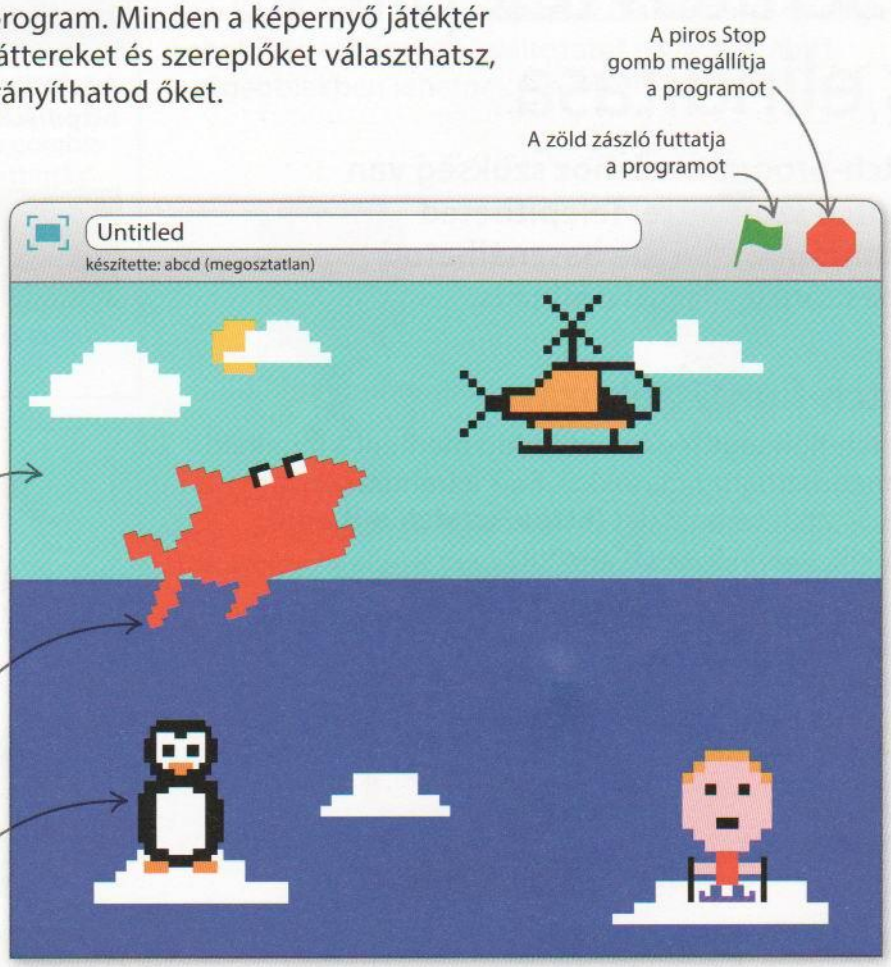
FOGALOMTÁR

Miért Scratch a neve?

A scratch eredetileg egy zenei kifejezés. A Scratch programnyelv is lehetővé teszi, hogy képek, zenék és programok újrakeverésével (remix) új programokat készíts.

Egy tipikus Scratch-program

Íme egy Scratch-példaprogram. Minden a képernyő játéktér nevű részén történik. Háttereket és szereplőket választhatsz, majd a programoddal irányíthatod őket.



▷ **A program futtatása**

A program elindítását futtatásnak nevezzük. Futtatáshoz kattints a játéktér feletti zöld zászlóra!

▷ **Program mozgatja a szereplőket**

A program a Scratch blokkjaiból áll össze, ennek hatására mászkál a cápa a játéktéren. A „következő jelmez" blokkal nyitja és zárja a száját minden mozdulatára.

A „mindig" blokk mozgatja folyamatosan a szereplőt

JEGYEZD MEG!

Projektek

A Scratchben a lementett munkákat projekteknek nevezzük. A projekt szereplőket, háttereket, hangokat és utasítássorozatokat tartalmaz. Amikor később újra megnyitod, mindent úgy találsz, ahogy elmentetted. A Scratch-projekt valójában egy számítógépes program.

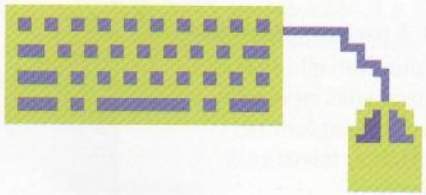
A Scratch telepítése és elindítása

Scratch-programozáshoz szükség van a Scratch szoftverre. Telepítheted a számítógépedre, de használhatod böngészőn keresztül online is.

JEGYEZD MEG!

A Scratch hivatalos weboldala

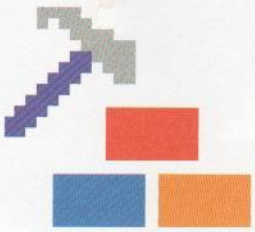
A Scratch weboldala:
http://scratch.mit.edu/

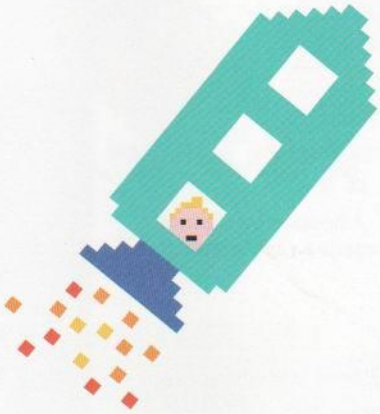


Scratch-fiók létrehozása

A Scratch-fiók segítségével el tudod menteni a munkáid, vagy megoszthatod másokkal. Fiók létrehozásához nyisd meg a Scratch weboldalát **(http://scratch.mit.edu),** és kattints a „Join Scratch" menüre.

▷ **Első lépések**
A Scratch beállítása különbözik az internetes (online), valamint a letöltött (offline) változat esetén.





1 Beállítások

Online

Nyisd meg a **http://scratch.mit.edu** weboldalt, és kattints a „Join Scratch" menüre. Töltsd ki az adatlapot felhasználónév és jelszó létrehozásához. Ehhez kérd a szüleid engedélyét.

Offline

Töltsd le a Scratch telepíthető változatát a **http://scratch.mit.edu/scratch2download/** linkről. A telepítőprogram lefuttatása után a Scratch ikon megjelenik az Asztalon.

2 A Scratch elindítása

A Scratch weboldalán kattints a „Sign in" menüpontra. A felhasználóneved és jelszavad megadásával jelentkezz be. Ezután kattints az „Alkoss!" menüpontra, és már készítheted is az első programodat.

A program elindításához kattints duplán az Asztalon lévő Scratch ikonra. Miután elindult, elkezdhetsz programozni. A magyar nyelvű programozáshoz a földgömb ikonra kattintva válaszd ki a magyar nyelvet.

TANÁCSOK

Az egér használata

A „kattintás" azt jelenti, hogy az egér bal gombjával kattintunk. Amennyiben egynél több gombja van az egérnek, a „jobb kattintás" az egér jobb gombjával való kattintást jelenti. Ha csak egy gomb van az egéren, a jobb kattintást úgy lehet elérni, hogy kattintás közben lenyomva tartjuk a „Ctrl" billentyűt.

Scratch-verziók

Ez a könyv a Scratch legújabb, 2.0-s verzióját mutatja be. Lehetőség szerint ezt a változatot használd, mert a régebbiekben lehetnek kisebb eltérések.

△ **Scratch 1.4**
A régebbi verzióban a játéktér a képernyő jobb oldalán van.

△ **Scratch 2.0**
A legújabb verzióban van néhány új utasítás, és a játéktér a bal oldalon található.

3 A munka mentése

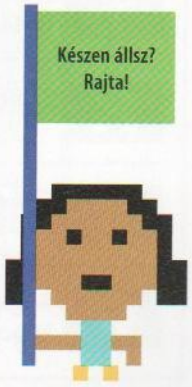
Ha be vagy jelentkezve a fiókodba, a Scratch automatikusan elvégzi a mentést. Korábbi projekted megnyitásához kattints a jobb felső sarokban a felhasználóneved alatti „Saját dolgaim" menüpontra.

Kattints a „Fájl" menüre, és válaszd a „Mentés most" menüpontot. Kérdezd meg a számítógép tulajdonosától, hogy hová mentheted a munkáidat.

4 Operációs rendszerek

A Scratch online változata használható Windows, Ubuntu és Mac alapú gépeken. Használatához Adobe Flash szükséges, így néhány tableten nem alkalmazható.

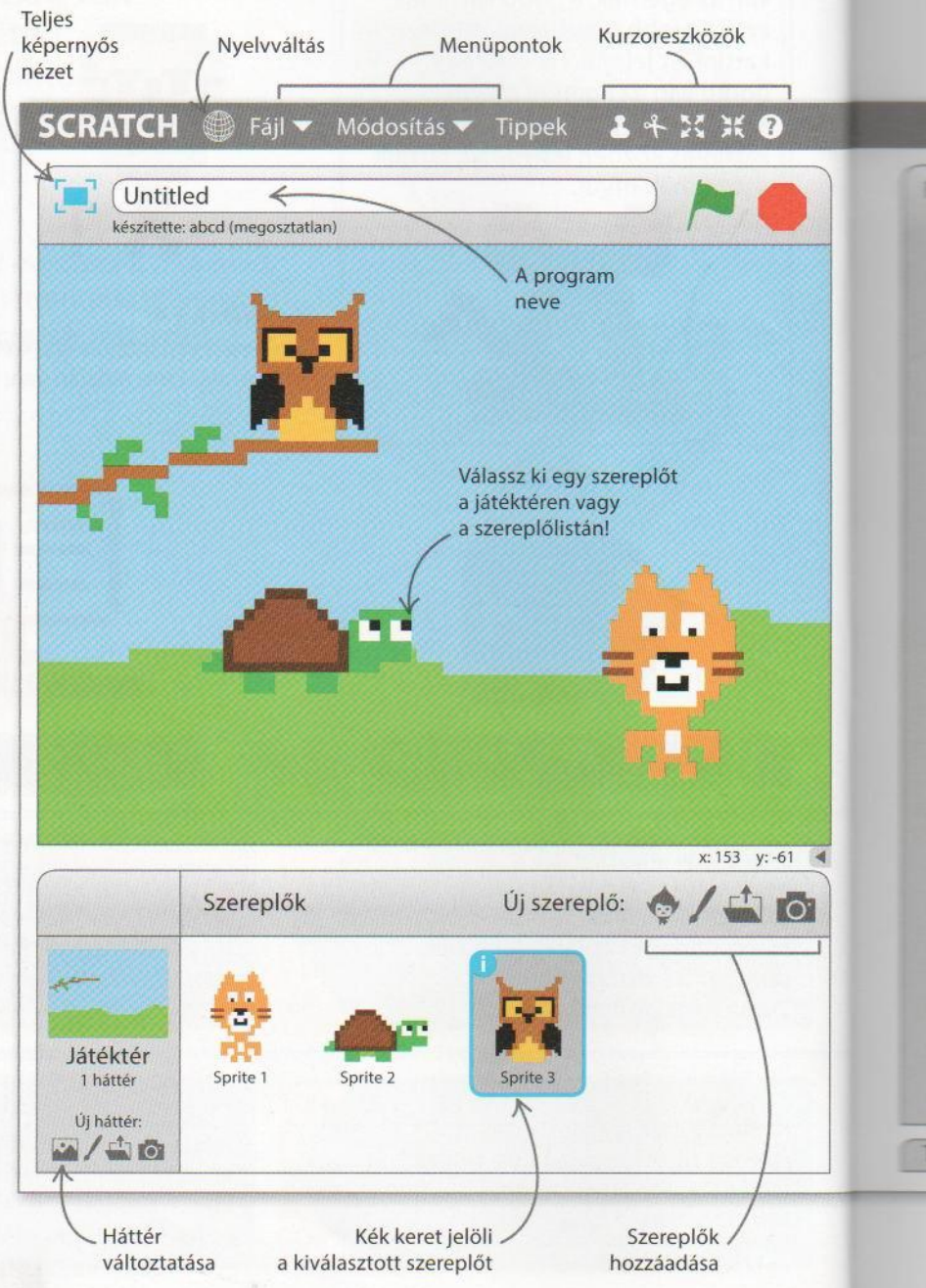
A Scratch offline változata jól működik Windows és Mac alapú gépeken. Ubuntu esetén javasolt az online változat használata.



A Scratch felülete

Íme a Scratch képernyő elrendezése, azaz felülete. Balra van a játéktér, a jobb oldalon készülnek a programok.

▽ **Kísérletezz!**
Kattintsd végig a gombokat és a füleket, hogy megismerkedj a Scratch felületével. A későbbi projektek segítenek megtanulni, mit hogyan használj.



TANÁCSOK

Menü és eszközök

MENÜPONTOK

Ez történik a képernyő felső részén lévő menüpontokban

Fájl — **Mentés** vagy új projekt indítása.

Módosítás — **Törlés visszavonása** vagy nézetváltás.

Tippek — **Ha elakadsz,** itt találsz segítséget.

KURZORESZKÖZÖK

Kattints a használni kívánt eszközre, majd arra a szereplőre vagy utasításra, amelyre a művelet vonatkozik!

Szereplő vagy utasítás **duplázása.**

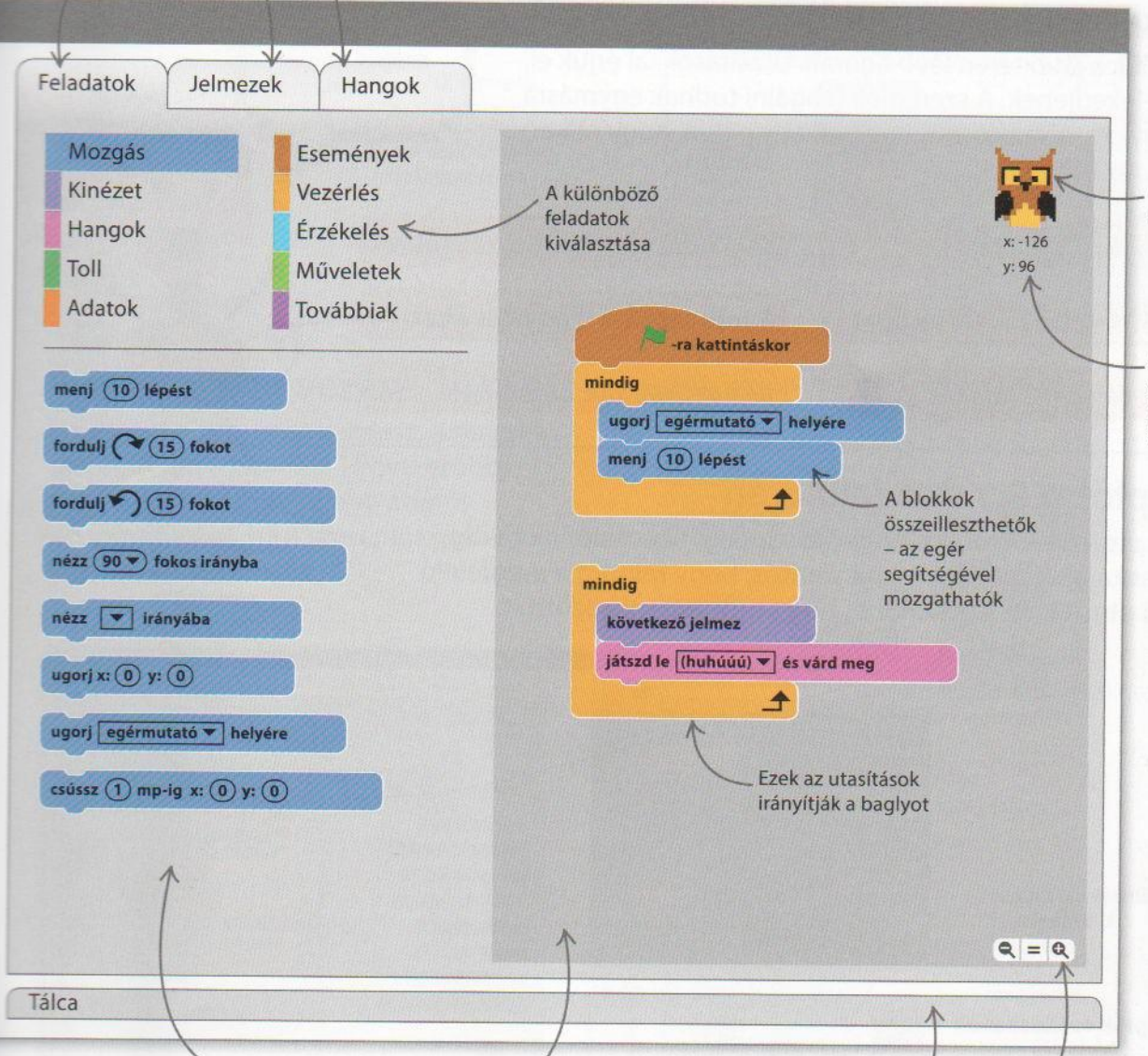
Szereplő vagy utasítás **törlése.**

Szereplő **nagyítása.**

Szereplő **kicsinyítése.**

Segítség kérése a blokkal kapcsolatban.

▷ **Scratch-térkép**
A játéktéren futnak a programok. Külön helyen találhatók a blokkok és a szereplők. A programterületen készülnek a programok.



Szereplők

LÁSD MÉG
‹ 26–27
A Scratch felülete
Jelmezek 40–41 ›
Bújócska 42–43 ›

A szereplők a Scratch alapelemei. Minden Scratch-program szereplőkből és az őket irányító utasításokból áll. A „Menekülj a sárkánytól!" program (32–37. o.) szereplője a macska, a sárkány és a fánk.

Mire képesek a szereplők?

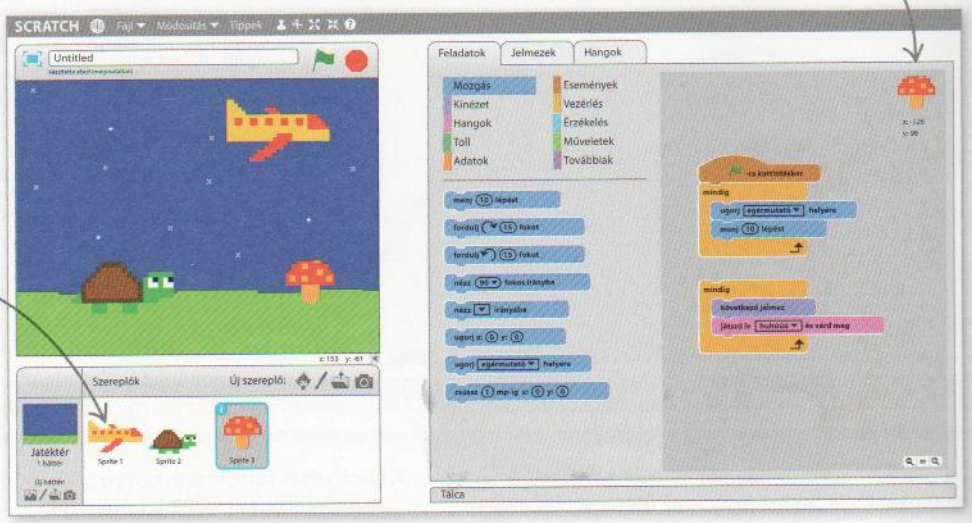
A szereplők a játéktéren lévő figurák. Utasításokkal érjük el, hogy cselekedjenek. A szereplők reagálni tudnak egymásra vagy a program kezelőjére. Íme, néhány példa, hogy miket tudnak csinálni:

- **Mozognak a játéktéren**
- **Megváltoztatják a külsejüket**
- **Hangot és zenét játszanak le**
- **Valamihez hozzáérve, reagálnak**
- **A felhasználó irányítja őket**
- **Buborékokban beszélnek**

Szereplők a Scratch felületén

Minden projektnek több szereplője van, egy szereplőnek pedig saját utasítássorozatai lehetnek. Fontos, hogy mindig a megfelelő szereplő utasításait használd, és hogy tudj váltani közöttük.

A megjelenített utasítások erre a szereplőre vonatkoznak



Kattintással válaszd ki a megfelelő szereplőt!

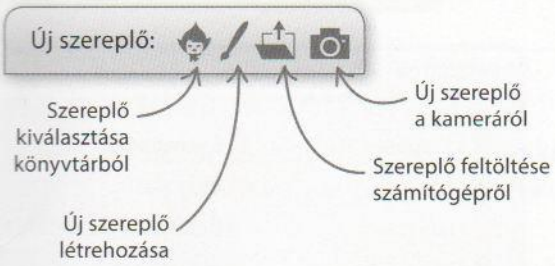
▷ **Szereplők és utasítások**
Egy programban sok szereplő lehet, mindegyik saját utasítássorozattal.

Szereplők létrehozása és szerkesztése

A játékok sokkal érdekesebbek, ha több szereplőjük van, akiket üldözhetünk, kicselezhetünk vagy eltalálhatunk. Szereplőket egyszerű létrehozni, másolni vagy törölni.

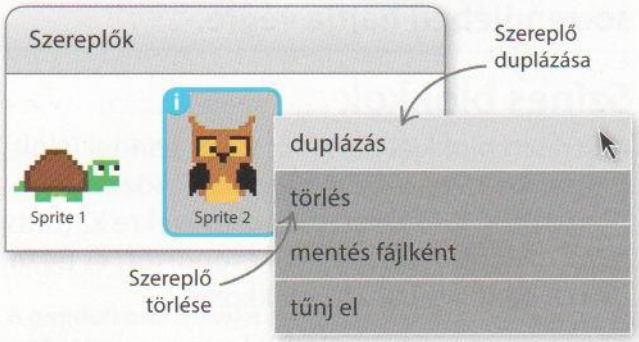
▽ Szereplő létrehozása

Használd a szereplőlista feletti gombokat szereplő hozzáadására vagy létrehozására.



▽ Szereplő másolása vagy törlése

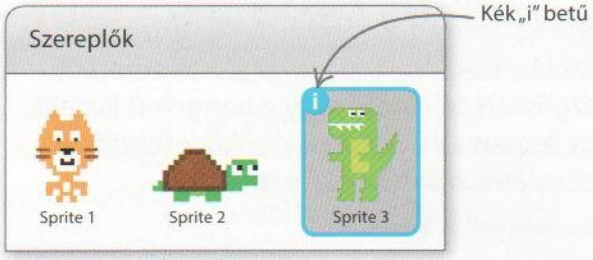
Egy szereplő és a hozzá tartozó utasítássorozatok másolásához kattints a listában a szereplőre jobb egérgombbal, és válaszd a „duplázás" menüpontot.



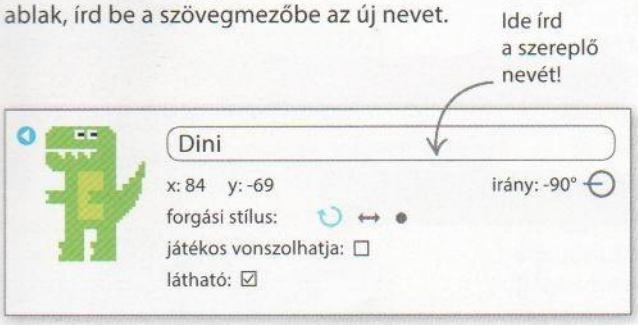
Szereplő elnevezése

Ha egy új programot kezdesz, a macska szereplő neve „Sprite 1". Egyszerűbb programozni és könnyebb átlátni a programokat, ha a szereplőknek kifejező neveket adsz.

1 **Válaszd ki a szereplőt!**
Kattints az elnevezni kívánt szereplőre, majd a sarokban lévő kék „i" betűre.



2 **Változtasd meg a nevét!**
Miután megnyílt az információs ablak, írd be a szövegmezőbe az új nevet.



3 **Az átnevezés befejezése**
Kattints a szereplőtől balra lévő kék nyílra, hogy bezárd az információs ablakot.

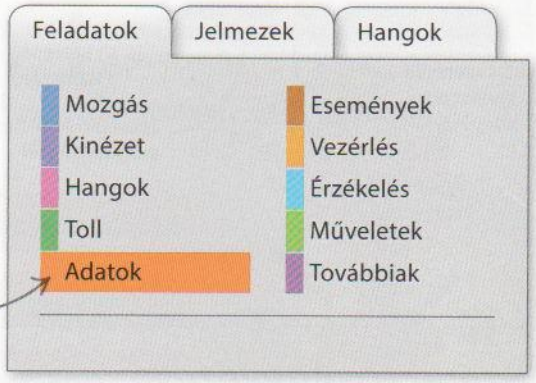
Színes blokkok és utasítások

LÁSD MÉG
❮ 26–27
A Scratch felülete
Menekülj a sárkánytól!
32–37 ❯

A blokkok különböző színűek, attól függően, hogy milyen típusú utasítást jelölnek. Egymáshoz illesztve programokat képeznek, utasításait a gép a blokkok sorrendjében hajtja végre.

Színes blokkok

A Scratch blokkjai tíz, különböző színnel jelölt csoportba tartoznak. A csoportok között a „Feladatok" fülön lévő csoportnevekre kattintva lehet váltani. Válassz ki egy csoportot, és nézd meg a benne található blokkokat.

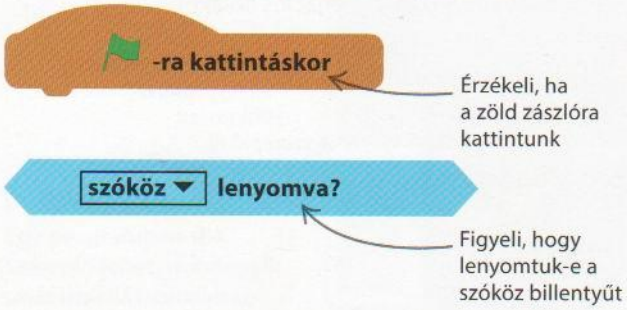


A blokkok feladata

A blokkok egyes csoportjai különböző feladatokat látnak el. Vannak, amelyek a szereplők mozgásáért felelnek, vagy a hangokat kezelik, vagy éppen az események bekövetkeztével kapcsolatos döntéseket hoznak.

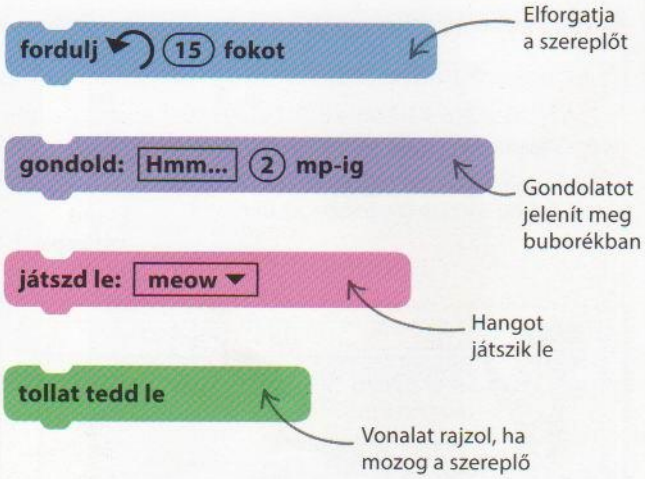
▽ **Események és érzékelés**

A barna „Események" csoport indítóblokkokat tartalmaz. A világoskék „Érzékelés" csoport blokkjai a billentyűzetet és az egeret érzékelik, illetve azt, hogy a szereplő hozzáér-e valamihez.

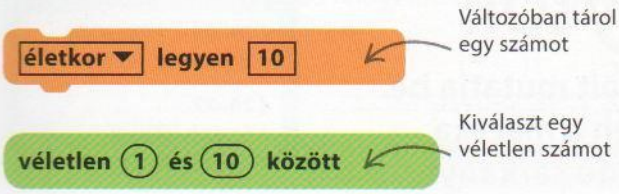


▽ **Mozgás, kinézet, hangok és toll**

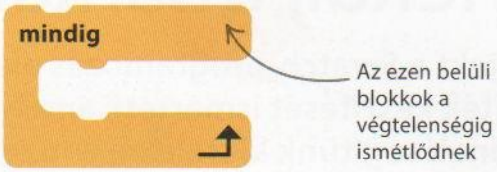
Ezek a blokkok irányítják a szereplőket – ez a program kimenete. Válassz egy szereplőt, és próbáld ki az egyes blokkokat!



▽ **Adatok és műveletek**
A narancssárga „Adatok" és a zöld „Műveletek" blokkok számokat és szavakat tárolnak és kezelnek.

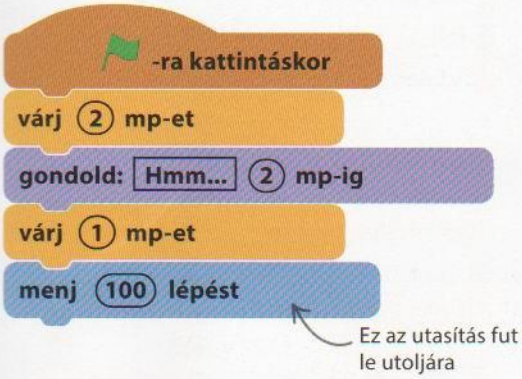


▽ **Vezérlés**
A „Vezérlés" blokkjai döntenek az utasítások végrehajtásáról. Az utasítások ismétlésére is használhatók.



Az utasítások végrehajtása

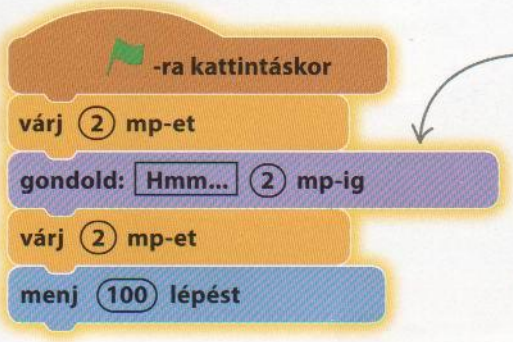
Egy program futásakor a Scratch a blokkokon lévő utasításokat hajtja végre. A legfelsővel kezdi, és lefelé halad.



◁ **A gondolkodó macska**
Ha ezt a programot a macska szereplővel használod, a macska először várakozik két másodpercet, gondolkodik további kettőt, újra vár egyet, majd továbbáll.

Futó programok

Az éppen végrehajtás alatt lévő utasítás körvonala világít. A zöld zászlóra kattintva futtathatod a programot, de egy tetszőleges blokkra kattintva is elindítható az utasítássorozat.



◁ **Tesztelés**
Ellenőrizd a program helyes futását úgy, hogy rákattintasz.

JEGYEZD MEG!

Program leállítása

Egy program valamennyi utasítássorozatának leállításához kattints a játéktér felett lévő hatszögletű piros Stop gombra. Közvetlenül az indító zöld zászló mellett találod.

1. PROJEKT

Menekülj a sárkánytól!

Ez a projekt a Scratch-programozás alapjait mutatja be. Olyan játék készítését ismerteti, amelyben a macska szereplőnek segítünk kicselezni a tűzokádó sárkányt.

LÁSD MÉG

24–25 A Scratch telepítése és indítása

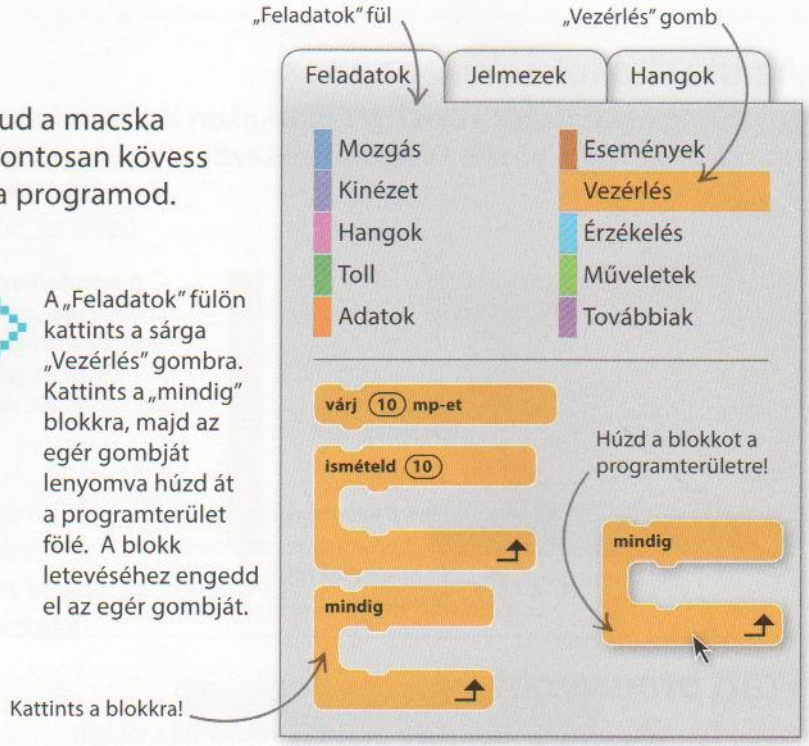
26–27 A Scratch felülete

Mozgasd a macskát!

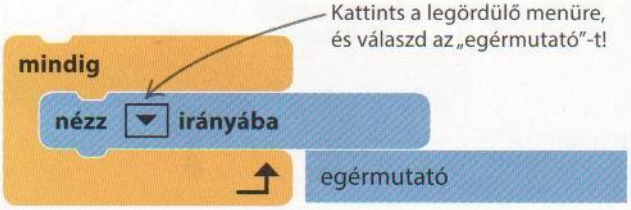
Először azt mutatjuk meg, hogyan tud a macska mozogni az egérmutatót követve. Pontosan kövess minden lépést, hogy jól működjön a programod.

1 Indítsd el a Scratchet. A „Fájl" menüben válaszd az „Új" menüpontot. A macska szereplő megjelenik a játéktéren.

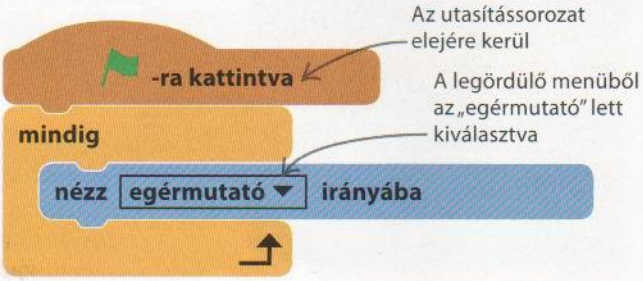
2 A „Feladatok" fülön kattints a sárga „Vezérlés" gombra. Kattints a „mindig" blokkra, majd az egér gombját lenyomva húzd át a programterület fölé. A blokk letevéséhez engedd el az egér gombját.



3 Kattints a kék „Mozgás" gombra a „Feladatok" fülön, hogy megjelenjenek a kék blokkok. A „nézz … irányába" blokkot húzd a programterületre, és illeszd a „mindig" blokk belsejébe. Válaszd ki a legördülő menü „egérmutató" lehetőségét.

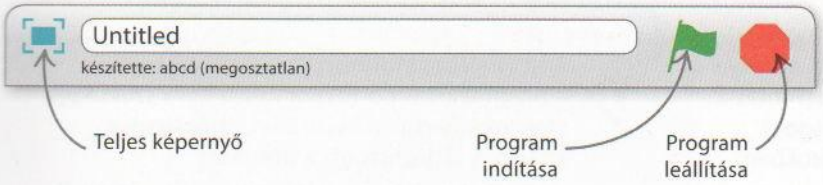


4 Kattints az „Események" gombra. Húzd a „zöld zászlóra kattintva" blokkot a programterületre, és illeszd a program tetejére.



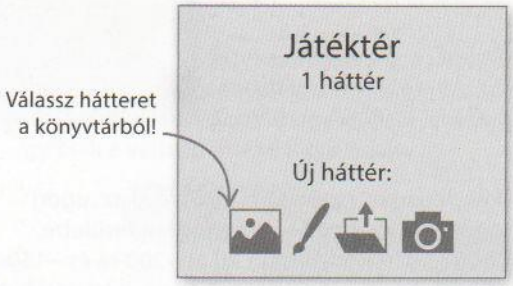
5 Kattints a játéktér felső részén lévő zöld zászlóra, és teszteld a programot! Ahogy mozgatod az egeret, aszerint mozog a macska is.

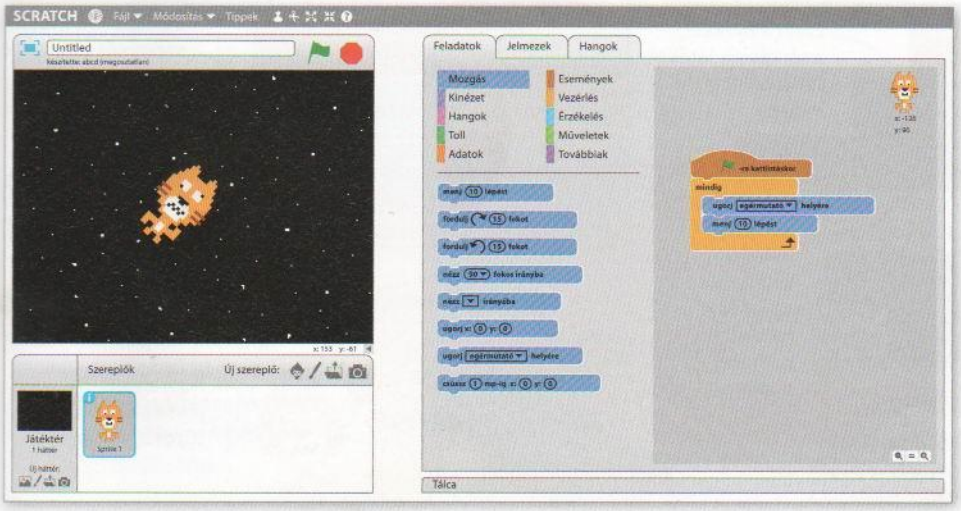
Mozgasd az egeret, és figyeld, hogyan fordul a macska!



6 Kattints újra a „Mozgás" gombra, és a „menj 10 lépést" blokkot húzd a „mindig" blokk belsejébe. Kattints a zöld zászlóra, és a macska elkezdi követni az egérmutatót.

7 A szereplők mögötti képet háttérnek nevezzük. A szereplők listájától balra található gombbal lehet új hátteret választani. Kattints rá, és válaszd az „Űr" témát. Kattints a csillagos háttérre (stars), majd a jobb alsó „OK" gombra.





◁ Macska az űrben
A játéktér most így néz ki. Futtasd a programot, és a macska a világűrben fogja üldözni az egérmutatót.

Az online Scratch automatikusan menti a munkád. Offline esetben kattints a „Fájl" menü „Mentés másként" menüpontjára!

MENEKÜLJ A SÁRKÁNYTÓL!

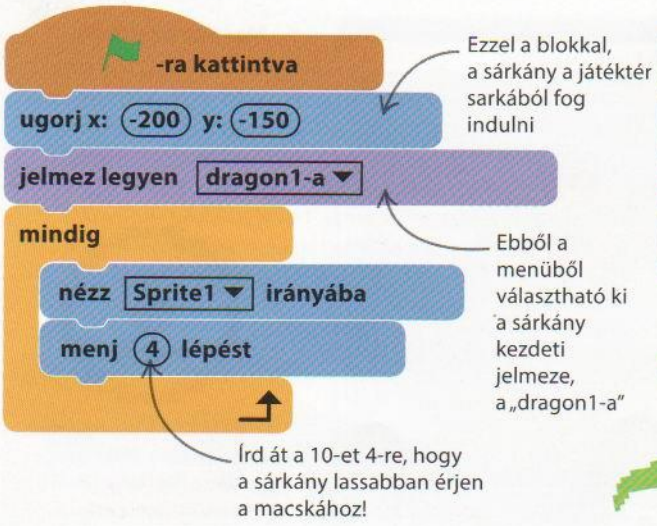
Add hozzá a tűzokádó sárkányt!

Most, hogy a macska már kergeti az egeret, jöhet a macskát üldöző sárkány. Ne hagyd, hogy elkapja, a végén még megperzseli!

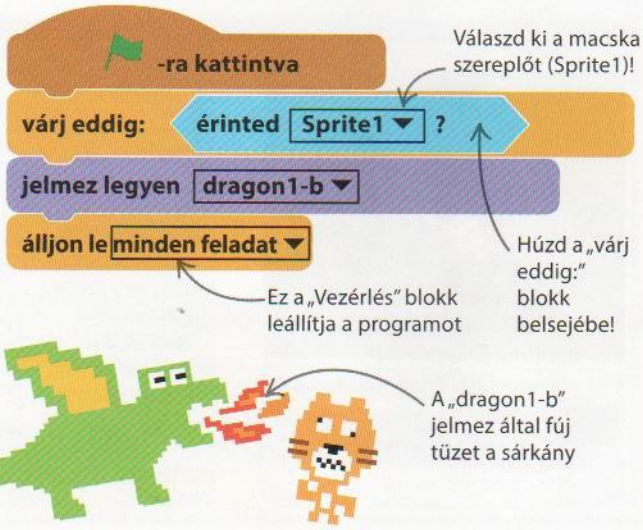
8 A szereplők listája felett van egy gomb, amellyel új szereplőt választhatsz. Kattints rá, és a „Fantázia" kategóriából válaszd ki a sárkányt (Dragon). Kattints az „OK" gombra a jobb alsó sarokban.

9 Add az alábbi utasítássorozatot a sárkányhoz. A megfelelő csoportokból válaszd ki a szükséges blokkokat, és húzd össze őket. Ettől kezdve a sárkány üldözni fogja a macskát.

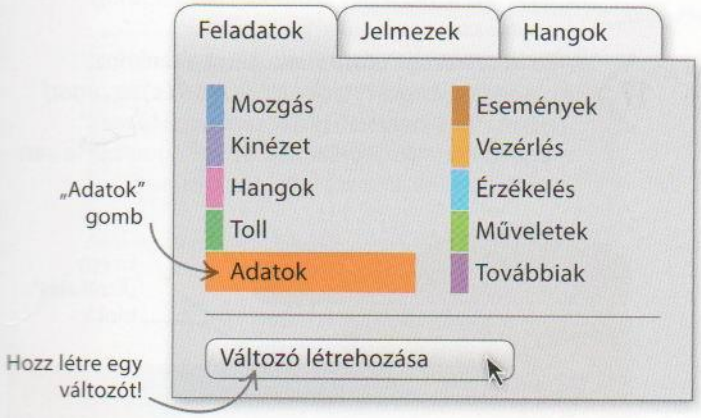
10 A kék „Mozgás" csoportból válaszd ki az „ugorj x: 0 y: 0" blokkot. Helyezd a programterületre, és a benne lévő számokat írd át –200-ra és –150-re. A lila „Kinézet" csoportból húzd át a „jelmez legyen" blokkot.



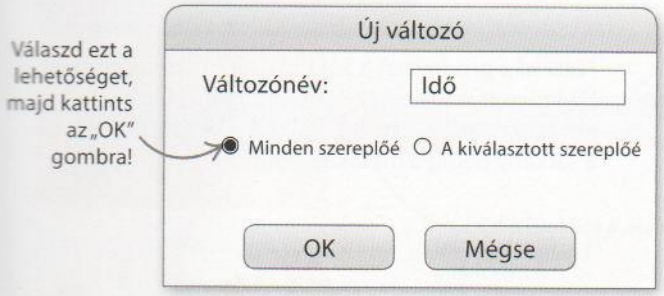
11 Válaszd ki a sárkány szereplőt, és add hozzá ezt az utasítássorozatot a programterülethez. A „várj eddig:" blokkot a „Vezérlés", az „érinted:" blokkot pedig az „Érzékelés" alatt találod. A sárkány tüzet fog fújni, ha hozzáér a macskához.



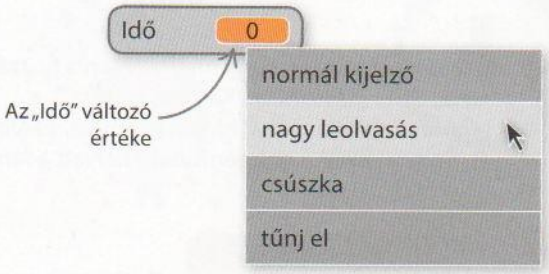
12 Programozáskor az adatok tárolására változókat használunk. Most egy változót fogunk használni, hogy megmérjük, egy játékos mennyi ideig tudja elkerülni a sárkányt. Kattints az „Adatok" gombra, majd a „Változó létrehozása" menüpontra.



13 Nevezd el a változót „Idő"-nek, válaszd ki alatta a „Minden szereplőé" gombot, és kattints az „OK" gombra. Így az összes szereplő használhatja a változót.



14 A változó neve és értéke megjelenik a játéktér sarkában egy kis dobozban. Jobb kattintás után válaszd a „large redout" (nagy kijelző) lehetőséget, így csak a változó értéke fog látszani.



15 A létrehozott változó miatt új blokkok jelennek meg az „Adatok" csoportban. Húzd az „(Idő) legyen (0)" és „(Idő) változzon (1)" blokkokat a program-területre, és hozd létre ezt az utasítássorozatot. Bármelyik szereplőhöz hozzáadhatod.

Mentsd el a munkád!

TANÁCSOK

Nehezíts a játékon!

Változtass a szereplők sebességén vagy méretén!

Gyorsítsd a sárkányt:

Növeld vagy csökkentsd a sárkány méretét:

Kattints erre a gombra, hogy a szereplő nagyobb legyen!

Kattints erre a gombra, hogy a szereplő kisebb legyen!

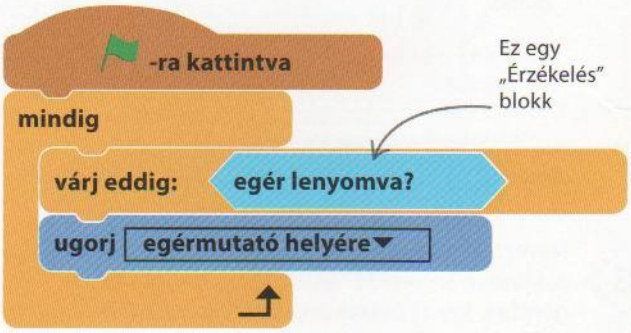
MENEKÜLJ A SÁRKÁNYTÓL!

Adj hozzá egy finom fánkot!

A Scratch könyvtárában sok szereplőt találsz.
Tedd érdekesebbé a játékot: adj a programhoz egy fánkot, amelyet a macska üldöz!

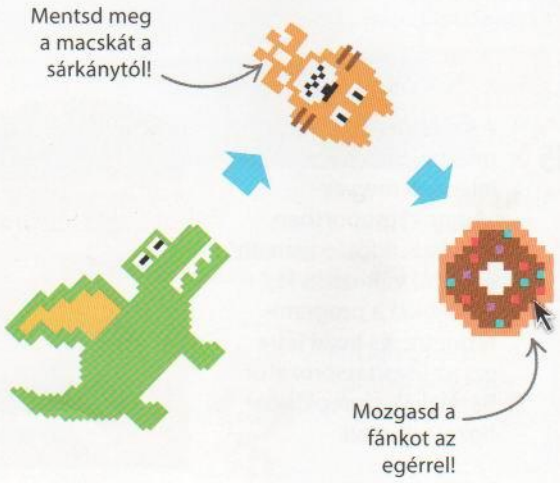
16 Az „Új szereplő" menü első gombjával nyisd meg a könyvtárat, és válaszd ki a „Dolgok" kategóriából a fánkot (Donut). Kattints az „OK" gombra.

17 Add hozzá ezt az utasítássorozatot a fánkhoz. Az „egér lenyomva?" blokk az „Érzékelés", az „ugorj (egérmutató) helyére" blokk pedig a „Mozgás" csoportban található. Amikor az egér gombja le van nyomva, a fánk követni fogja a mozgását.

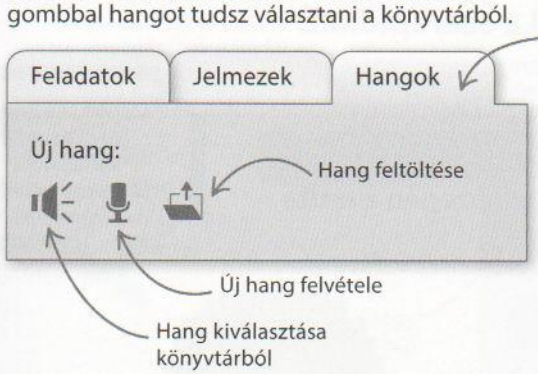


18 Jelöld ki a macskát, hogy megjelenjen a hozzá tartozó utasítássorozat. Kattints a „nézz (egérmutató) irányába" blokk menüjére, és állítsd át, hogy a macska az egérmutató helyett a fánkot (Donut) kövesse.

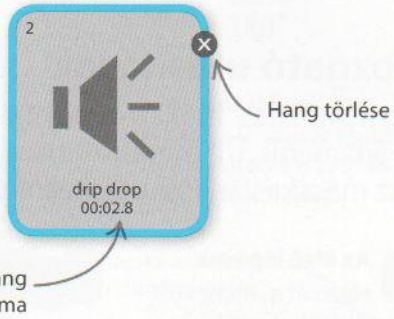
19 Futtasd a programot a zöld zászlóra kattintva. Ha lenyomva tartod az egér gombját, a fánk követi az egérmutatót, a macska a fánkot, a sárkány pedig a macskát üldözi.



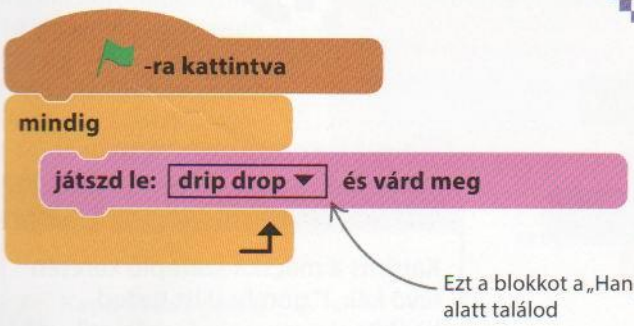
20 Most pedig adj hozzá zenét! Kattints a „Hangok" fülre a blokkok felett. Minden szereplő saját hangkészlettel rendelkezik, ezeket itt szabályozhatod. Az „Új hang" menü alatti bal gombbal hangot tudsz választani a könyvtárból.



21 Válaszd ki a „drip drop" hangot, és kattints az „OK" gombra. Miután a hangot hozzáadtad a macskához, megjelenik a „Hangok" fülön.



22 Kattints a „Feladatok" fülre, hogy visszatérj a programokhoz. Add ezt az utasítássorozatot a macskához, hogy folyamatosan szóljon a zene. Jó szórakozást a játékhoz.



JEGYEZD MEG!

Amit már tudsz

Ez a projekt megmutatta, mire képes a Scratch. A következőket tanultad meg eddig;

Program létrehozása: Blokkok összeillesztésével játékprogramot írtál.

Elemek hozzáadása: Szereplőket és hátteret adtál hozzá.

Szereplők mozgatása: Elérted, hogy a szereplők kergessék egymást.

Változó használata: Időmérőt készítettél a játékhoz.

Jelmezek használata: Különféle jelmezek használatával változtattál a sárkány kinézetén.

Zene hozzáadása: Hangot választottál ki, és ezt a program futás közben lejátszotta.

Mozgásban

LÁSD MÉG
‹ 28–29 Szereplők
Koordináták 56–57 ›

A játékprogramok gyakran szólnak lövöldözésről, üldözésről és menekülésről. A szereplők általában futnak vagy száguldoznak valamivel. Egy izgalmas játék készítéséhez először is meg kell tanulni, hogyan lehet mozgatni a szereplőket.

Mozgató utasítások

A kék „Mozgás" blokkok irányítják a szereplőket. Kezdj egy új projektet a „Fájl" menü „Új" menüpontjára kattintva! A játéktér közepén a tettre kész macska várja az utasításokat.

A Scratch nem engedi, hogy a szereplők kisétáljanak a játéktérről, így sosem tűnhetnek el.

1 Az első lépések
Húzd át a „menj 10 lépést" blokkot a kék „Mozgás" csoportból a játéktérre. Valahányszor a blokkra kattintasz, a macska megmozdul.

2 Tartsd mozgásban!
A sárga „mindig" blokkot húzd át a „menj 10 lépést" blokk köré. A zöld zászlóra kattintva a macska elmegy a játéktér széléig.

3 Visszapattanás
Húzd a „ha szélen vagy, pattanj vissza" blokkot a „mindig" blokk belsejébe. A macska ezután mindig visszafordul, ha a játéktér szélére ér. Visszafelé mindig fejjel lefelé közlekedik.

TANÁCSOK

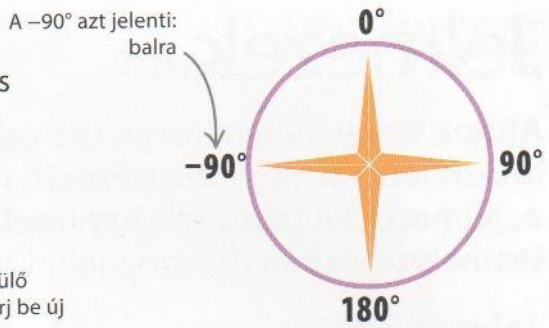
Forgási stílus

Kattints a macska szereplő keretén lévő kék „i" gombra! Itt tudod beállítani a szereplő forgási stílusát – hogy ne fejjel lefelé mászkáljon.

- A macska mindig arra néz, amerre megy, de időnként fejjel lefelé.
- A macska balra vagy jobbra néz, de mindig lábbal lefelé mozog.
- A macska egyáltalán nem fordul el.

Melyik irányba?

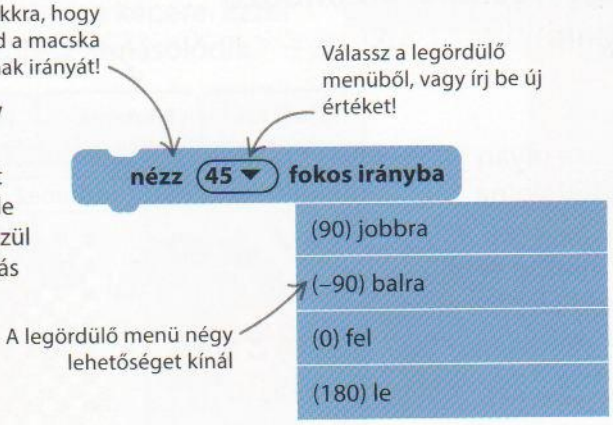
A macska most büszkén járkál jobbra és balra, de arra is rá tudod venni, hogy fel és le, vagy akár átlósan mozogjon. A „Mozgás" blokkokkal valódi macska-egér játékot készíthetsz.



△ **Iránytű**
Az irányokat fokokban mérjük, a 0° felfelé mutat. Az irány értéke –179° és +180° között lehet.

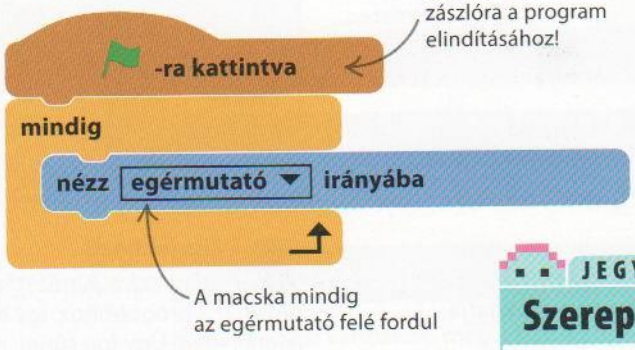
4 A megfelelő irány

Húzd a „nézz 90 fokos irányba" blokkot a játéktérre, és gördítsd le a menüjét. Négy irány közül választhatsz, de – kattintás után – beírhatsz más értéket is.



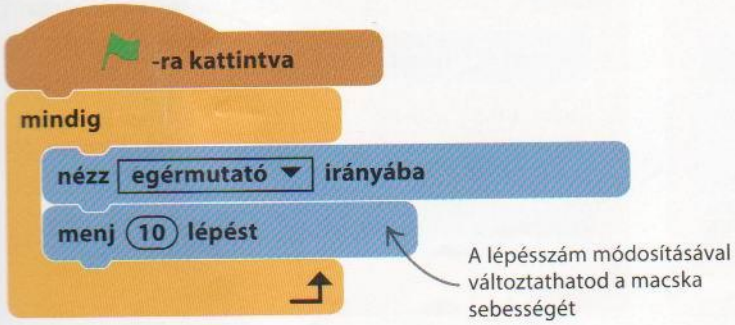
5 Macska és egér

Távolítsd el az utasítássorozatból a „menj 10 lépést" és a „ha szélen vagy, pattanj vissza" blokkokat. Húzd a „mindig" belsejébe a „nézz … irányába" blokkot, és a menüjéből válaszd ki az „egérmutató" lehetőséget.



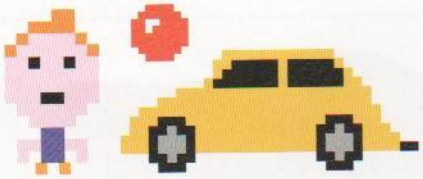
6 Kapd el az egeret!

El tudja kapni a macska az egeret? Húzd a „menj 10 lépést" blokkot is a „mindig" blokk belsejébe. A macska elindul az egérmutató felé.



JEGYEZD MEG!

Szereplők



A Scratch objektumai a szereplők, amelyeket irányíthatsz (*lásd* 28–29. o.). Minden új projekt a macskával indul, de választhatsz autót, dínót, táncost és még sok mást a könyvtárból. Saját szereplőket is tervezhetsz.

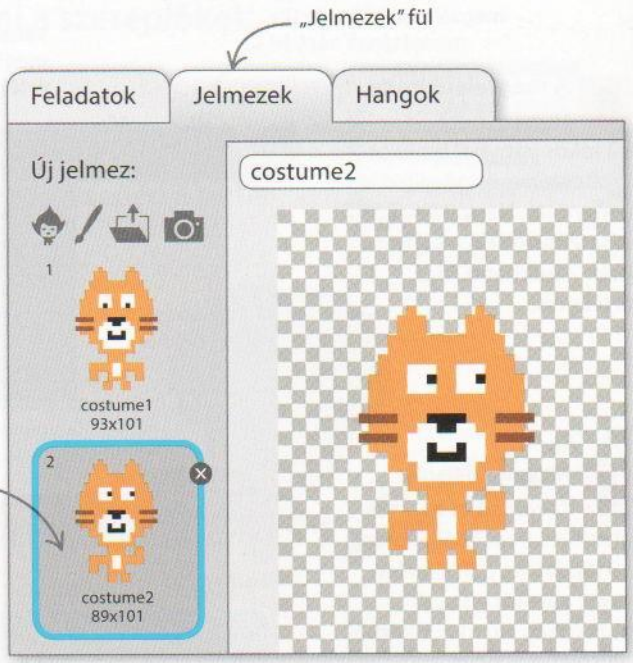
Jelmezek

LÁSD MÉG
❮ 38–39
Mozgásban
Üzenetküldés
70–71 ❯

Ahhoz, hogy megváltoztassuk egy szereplő küllemét, arckifejezését vagy testtartását, meg kell változtatni a „jelmezét". A jelmezek a szereplőket különböző testhelyzetekben ábrázolják.

Jelmezváltás

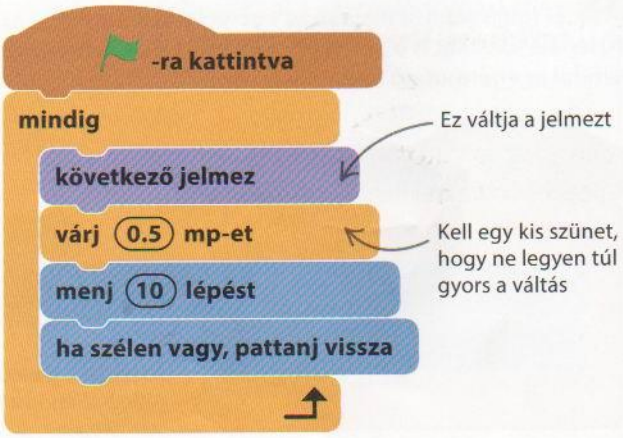
A különböző jelmezek váltakozása olyan látszatot kelt, mintha megmozdult volna a szereplő keze vagy lába. A macska két jelmezének váltogatása olyan, mintha sétálna. Kezdj egy új projektet, és próbáld ki!



1 Különböző jelmezek
Kattints a „Jelmezek" fülre. Itt találod a macska jelmezeit, amelyeken két különböző pozícióban van a keze és a lába.

2 Sétáljon a macska!
Az alábbi utasítássorozat hatására a macska úgy sétál, hogy közben nem mozgatja a végtagjait.

3 Válts jelmezt!
Add hozzá a „Kinézet" csoport „következő jelmez" blokkját a programhoz, így minden lépésnél más jelmez jelenik meg. Úgy fog tűnni, mintha a macska járás közben mozgatná a kezét és a lábát.



Táncoló balerina

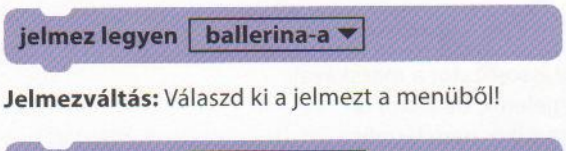
Most nézzünk egy kis balettet! Válaszd ki a könyvtárból a balerina (Ballerina) szereplőt. Kattints a macskára a szereplőlistában, majd húzd át a teljes utasítássorozatát a balerina képére. Ezzel a megfelelő programrész átmásolódik.

△ **A balerina utasítássorozata**
A balerinát ugyanazok az utasítások mozgatják, mint a macskát. Neki azonban négy jelmeze van, és így alakul ki a mozdulatsor.

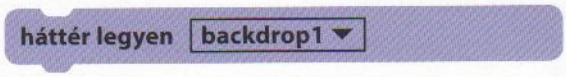
TANÁCSOK

Jelmezváltás

Ha a jelmezek közül csak néhányat szeretnél váltogatni, használd a „jelmez legyen" blokkot! Ezzel kiválaszthatod a megfelelő jelmezeket.



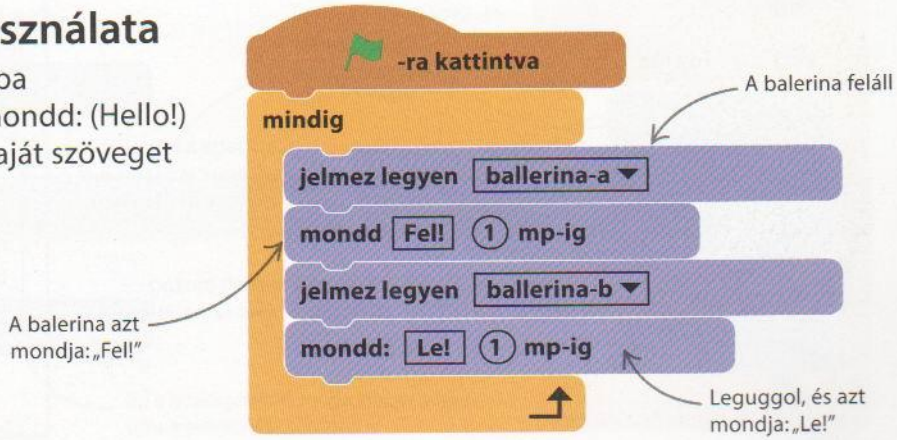
Jelmezváltás: Válaszd ki a jelmezt a menüből!



Háttérváltás: Változtasd meg a hátteret!

Szövegbuborékok használata

Adj szöveget a szereplők szájába jelmezváltáskor! Használd a „mondd: (Hello!) (2) mp-ig" blokkot, és írj bele saját szöveget a szereplő számára!



Bújócska

Üdvözlünk a speciális effektek világában! Figyeld meg, hogy a lila „Kinézet" blokkok használatával a szereplők minként tűnnek el, majd jelennek meg újra, nőnek meg és mennek össze, halványulnak el és válnak élessé újra.

LÁSD MÉG
❮ 38–39
Mozgásban
Üzenetküldés
70–71 ❯

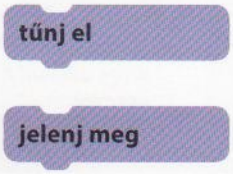
Szereplők elrejtése

A „tűnj el" blokk segítségével elrejtheted a szereplőt. Továbbra is mozog a játéktéren, de nem látszik, amíg a „jelenj meg" blokk újra előre nem hozza.

A „tűnj el" blokk elrejti a szereplőt a játékban

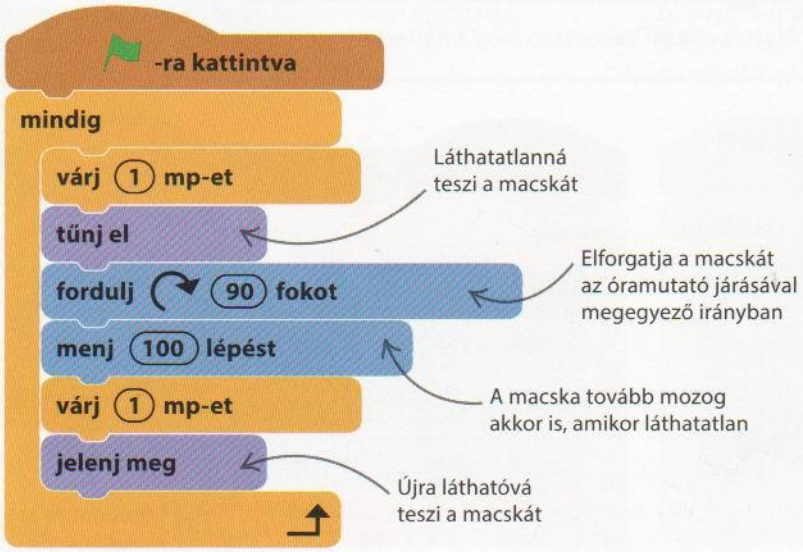
▷ **Látható és láthatatlan**

A szereplő elrejtéséhez használd a „tűnj el" blokkot! Ha újra látni akarod, a „jelenj meg" blokk előhozza. Mindkét blokkot a „Kinézet" csoportban találod.



▽ **Bujkáló macska**

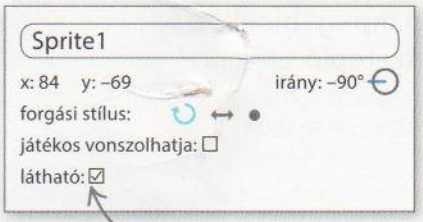
Próbáld ki ezt az utasítássorozatot a macskával! Eltűnik, majd újra megjelenik, de akkor is folyamatosan mozog, amikor nem látszik.



TANÁCSOK

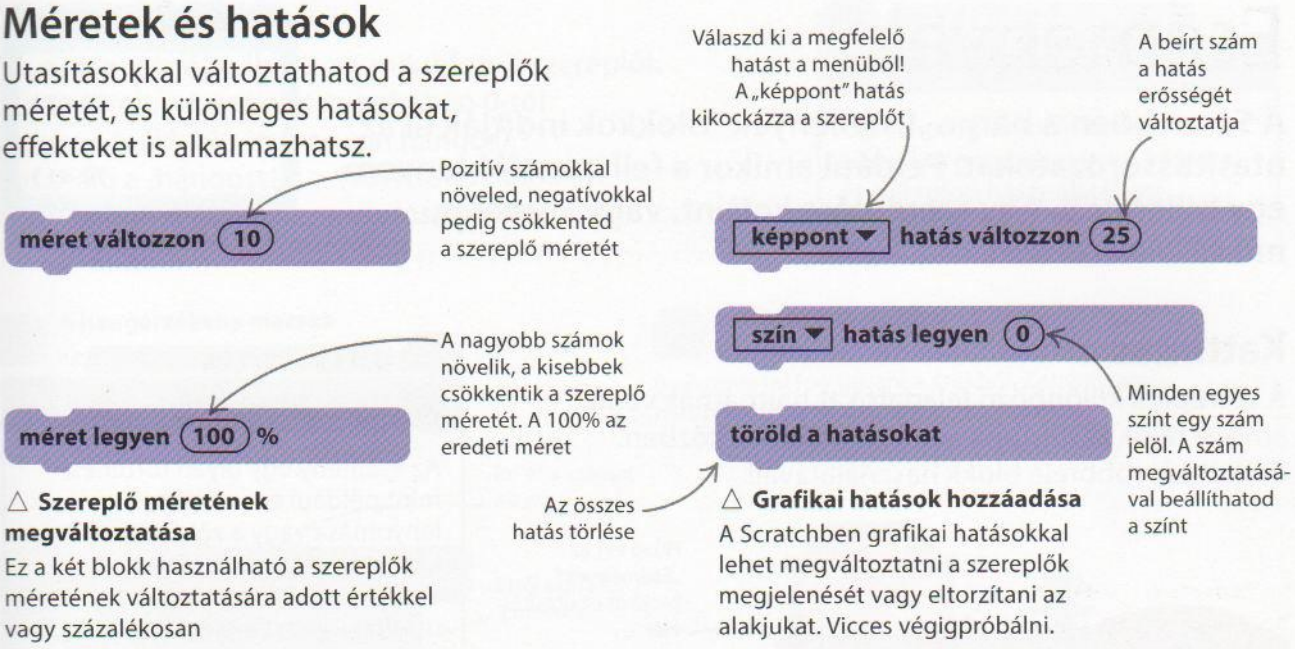
Szereplők láthatósága

Válassz ki egy szereplőt a listából! A kis „i" betűre kattintva nyisd meg a beállításokat! Itt a „látható" jelölőnégyzettel tudod beállítani, hogy a szereplő látható legyen-e, vagy sem.



Méretek és hatások

Utasításokkal változtathatod a szereplők méretét, és különleges hatásokat, effekteket is alkalmazhatsz.

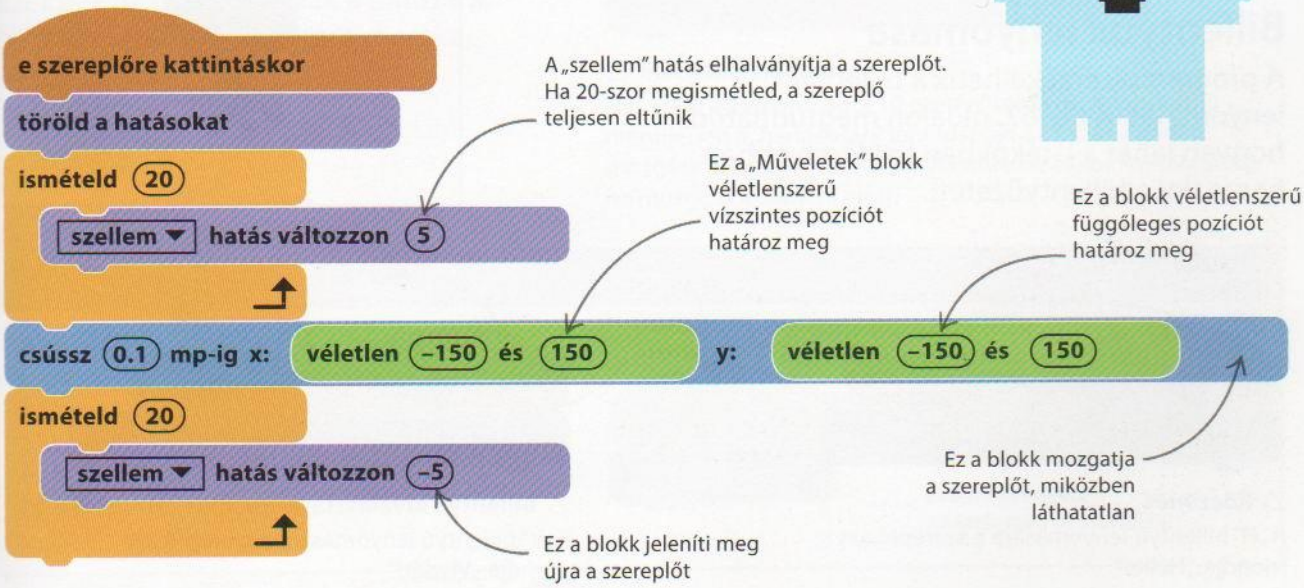


△ **Szereplő méretének megváltoztatása**
Ez a két blokk használható a szereplők méretének változtatására adott értékkel vagy százalékosan

△ **Grafikai hatások hozzáadása**
A Scratchben grafikai hatásokkal lehet megváltoztatni a szereplők megjelenését vagy eltorzítani az alakjukat. Vicces végigpróbálni.

Teleportálás hatások használatával

Válassz ki egy szellem szereplőt a „Fantázia" kategóriából, és add hozzá az alábbi utasítássorozatot! Kattintásra a szellem teleportálni fog.



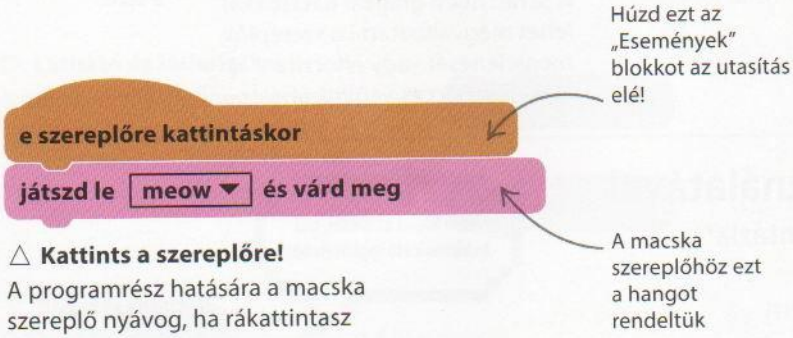
Események

A Scratchben a barna „Események" blokkok indítják el az utasítássorozatokat. Például amikor a felhasználó lenyom egy billentyűt, egy szereplőre kattint, vagy kamerát, mikrofont használ.

LÁSD MÉG
Érzékelés 66–67 ❯
Üzenetkülgés 70–71 ❯

Kattintás

A szereplők különböző feladatokat hajthatnak végre, amikor rájuk kattintunk a program futása közben. Próbáld ki többféle blokk használatával!

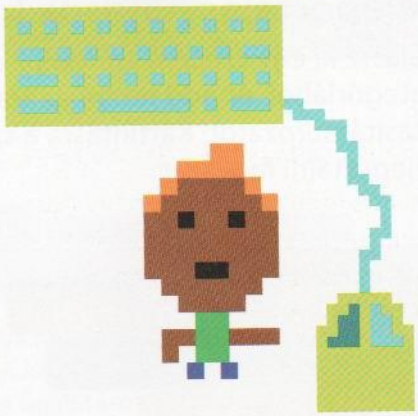


△ **Kattints a szereplőre!**
A programrész hatására a macska szereplő nyávog, ha rákattintasz

FOGALOMTÁR

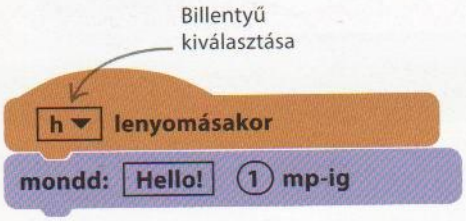
Mi az esemény?

Az esemény egy olyan történés, mint például egy billentyű lenyomása vagy a zöld zászlóra kattintás. Az eseményeket érzékelő blokkokat az utasítássorozat elejére kell illeszteni. Az adott esemény hatására a program elindul.



Billentyűk lenyomása

A programok érzékelhetik a billentyűk lenyomását. A 66–67. oldalon megtudhatod, hogyan lehet a játékokban hatékonyabban használni a billentyűzetet.



△ **Köszönés**
A „H" billentyű lenyomására a szereplő azt mondja: „Hello!"

△ **Billentyű kiválasztása**
A „V" billentyű lenyomására a szereplő azt mondja: „Viszlát!"

Hangesemények

Ha a számítógépedben van mikrofon, a szereplők érzékelhetik a hangerőt a szobában 0-tól (nagyon halk) 100-ig (nagyon hangos). Használd a „hangosság > (10) esetén" blokkot, hogy hanggal indítsd az utasítássorozatot.

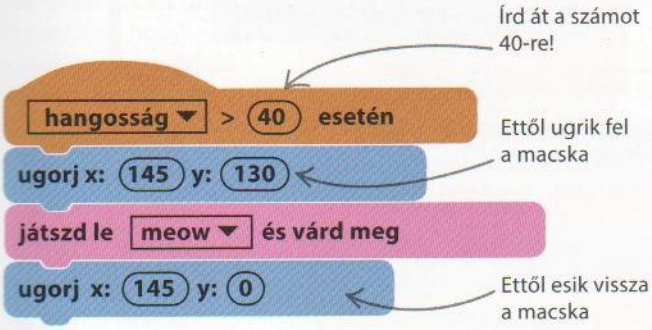
TANÁCSOK

Engedélykérés

Online programozás esetén a Scratch engedélyt kér a kamera és mikrofon használatához. Kattints az „Allow"-ra!

1 A hangérzékeny macska

Kezdj egy új projektet, és válaszd ki a „room3" hátteret. Húzd a macskát a székre, és add hozzá az alábbi utasítássorozatot.

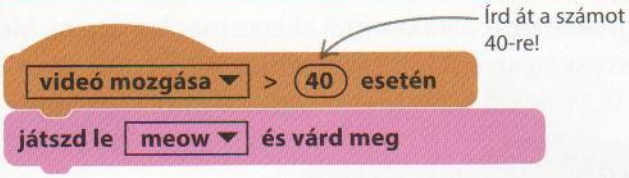


2 Kiálts rá a macskára!

Kiabálj a mikrofonba, és a macska rémülten nyávogva fel fog ugrani a székből. Zenére vagy más hangokra is reagál, ha elég hangos.

Webkamerás mozgásérzékelő

Ha van webkamerád, azt is használhatod. Ezeknek az utasításoknak a hatására, ha a kamerába integetsz, a macska nyávogni fog.



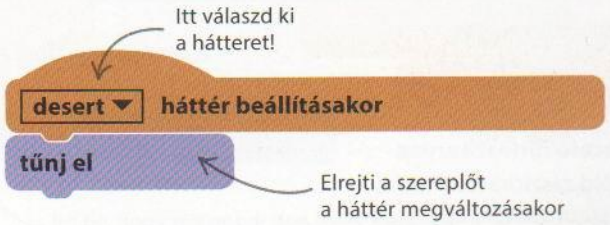
△ **Mozgásérzékelés**

Használd a „hangosság > 10 esetén" blokkot! A legördülő menüjében a „hangosság" lehetőséget állítsd át a „videó mozgása" lehetőségre. Az utasítássorozat akkor fog elindulni, ha mozogsz a kamera előtt.

TANÁCSOK

Háttérváltás

A szereplők reagálhatnak a háttér megváltozására. Például lehet olyan háttered, amelynek hatására eltűnik a szereplő. Tölts fel egy új hátteret, és használd a „(backdrop1) háttér beállításakor" blokkot!



Egyszerű ciklusok

A ciklus egy önmagát ismétlő programrész. A ciklusblokkokból (a „Vezérlés" csoportból) kiderül, hogy mit és hányszor kell ismételni. Így nem kell ugyanazokat a blokkokat többször behúzni.

LÁSD MÉG
Összetett ciklusok **68–69** ›
Ciklusok a Pythonban **122–123** ›

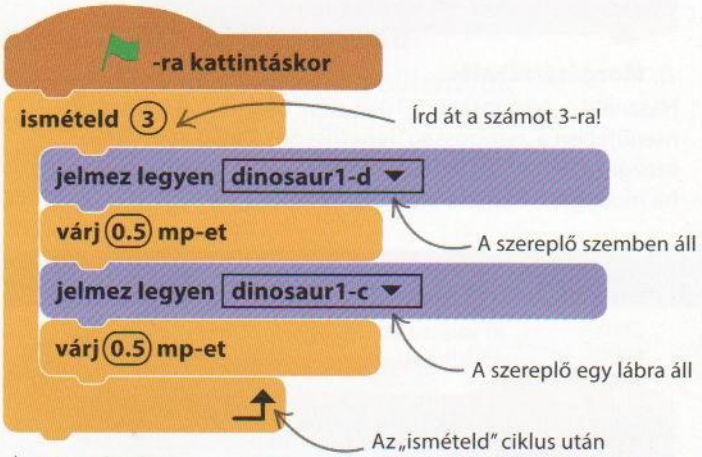
A „mindig" ciklus

Amit egy „mindig" ciklusba húzunk, állandóan ismétlődni fog. Nincs értelme utána további blokkokat illeszteni, mert a „mindig" ciklus sosem ér véget.

△ **Állandóan ismétlődik**
Ha a ciklus utolsó blokkja is lefutott, kezdődik elölről.

Az „ismételd" ciklus

Használd az „ismételd 10" blokkot, ha egy műveletet meghatározott alkalommal akarsz megismételni. Módosítsd az értéket a kívánt ismétlésszámra. Add a „Dinosaur1" szereplőt egy új projekthez, és készítsd el az alábbi utasítássorozatot!

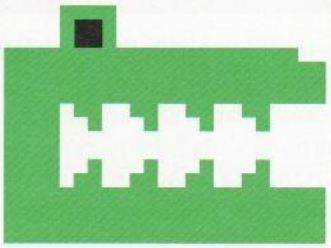


△ **Táncoló dinoszaurusz**
Ha a zöld zászlóra kattintasz, a dinoszaurusz táncol. Háromszor ismétli meg a mozdulatot.

JEGYZED MEG!

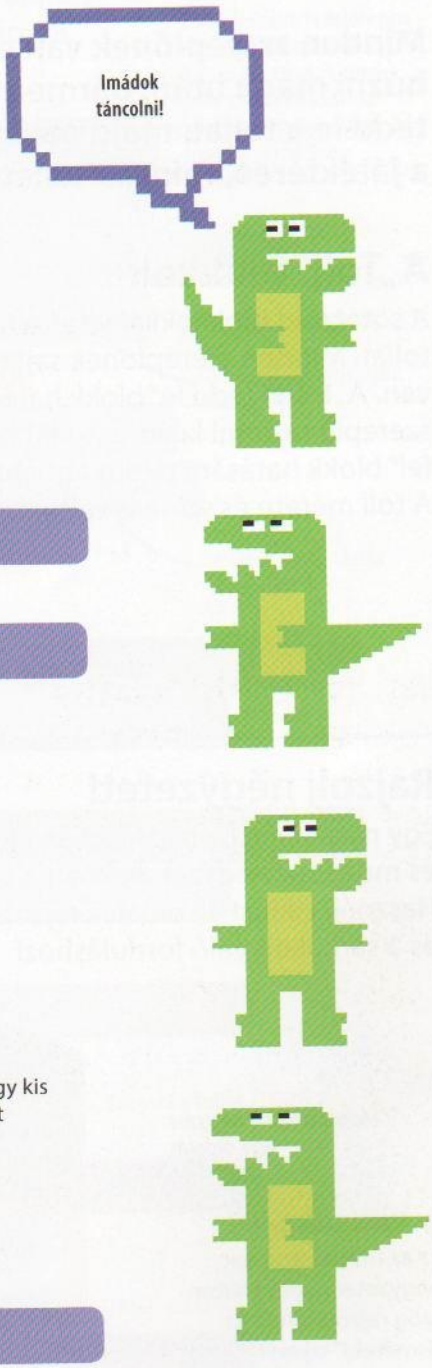
A ciklusblokkok alakja

A ciklusblokkok olyan alakúak, mint egy állkapocs. Húzd bele az ismétlendő blokkokat, és a ciklus körülöleli őket. Ha több blokkot építesz be, kitágul, hogy meglegyen a szükséges hely.



Egymásba ágyazott ciklusok

A ciklusokat „egymásba ágyazhatjuk", vagyis egymás belsejébe építhetjük őket. Ebben az utasítássorozatban a dinoszaurusz úgy járja a táncot, hogy jobbra-balra megy, majd egy pillanatra elgondolkozik. Miután kifújta magát, újra táncolni kezd, és csak akkor fejezi be, ha a piros Stop gombra kattintasz.



▷ **Ciklusok a ciklusokban**
A „mindig" ciklusban több „ismételd" ciklus található. Figyelj arra, hogy a blokkok a megfelelő ciklusba kerüljenek, különben a program nem fog helyesen működni.

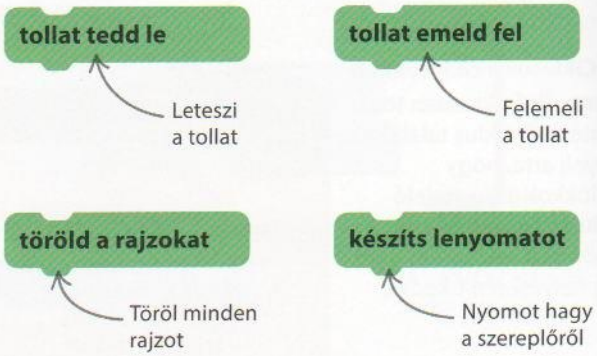
Tollak és teknőcök

LÁSD MÉG
‹ 44–45 Események
‹ 46–47 Egyszerű ciklusok

Minden szereplőnek van egy tolla, amellyel vonalat tud húzni maga után, bármerre halad. Ha rajzolni szeretnél, tedd le a tollat, majd mozgasd végig a szereplőt a játéktéren, mintha tollat húznál egy papíron.

A „Toll" blokkok

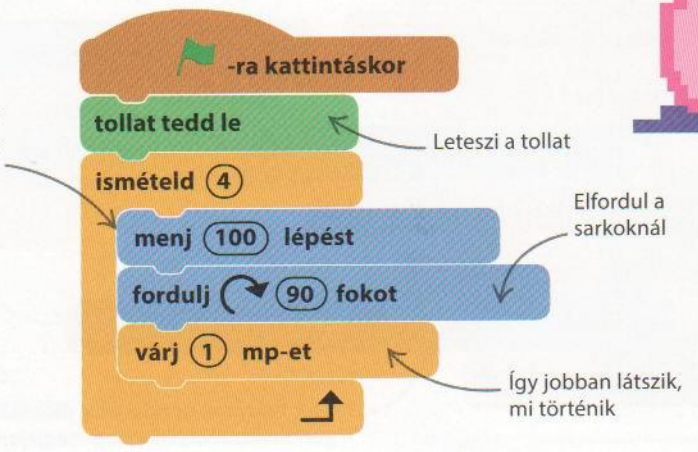
A sötétzöld blokkokkal vezérelheted a tollat. Minden szereplőnek saját tolla van. A „tollat tedd le" blokk hatására a szereplő rajzolni kezd, a „tollat emeld fel" blokk hatására pedig abbahagyja. A toll mérete és színe is változtatható.



△ **Játék a tollakkal**
Próbálgasd, hogyan tudsz rajzolni a „Toll" blokkokkal!

Rajzolj négyzetet!

Egy négyzet rajzolásához tedd le a tollat, és mozgasd négyzet alakban a szereplőt! Használj ciklust az oldalak rajzolásához és a sarkokon való forduláshoz!



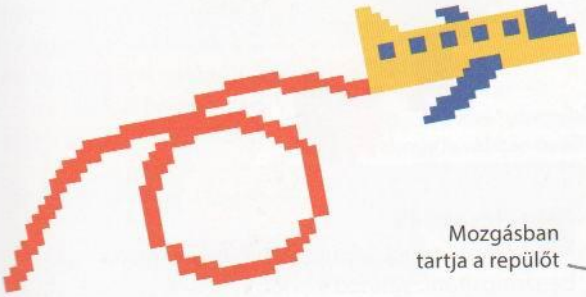
▷ **Válts alakzatot!**
Ez az utasítássorozat négyzetet rajzol. Háromszög rajzolásához az „ismételd" ciklust állítsd háromra, és a fordulást állítsd 90-ről 120 fokra.

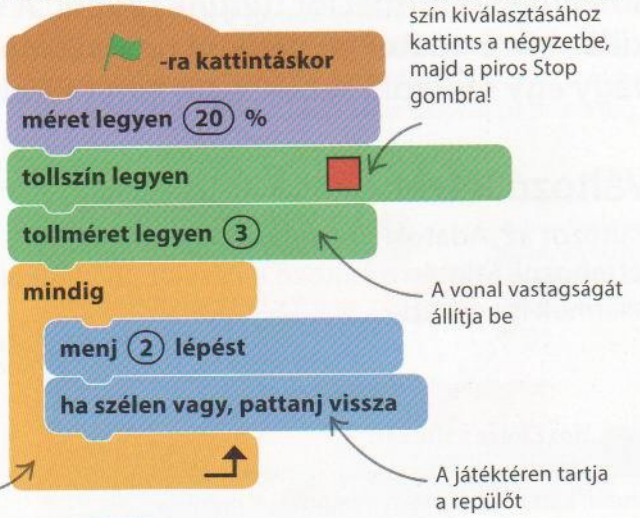
Égre írás

Ebben a programban egy repülőt irányíthatsz. Ahogy repül, kondenzcsíkot húz maga után, így írhatsz az égre. Kezdj egy új projektet! Add hozzá a repülő szereplőt (Airplane), és az alábbi utasítássorozatot!

▷ **Magasan szállni**

A jobbra és balra nyilakkal irányíthatod a repülőt. A füstöt az „a" billentyűvel kapcsolhatod be, a „z"-vel pedig ki. A szóköz leütésével újra tiszta lesz az ég.





Csak a Scratch felületén megjelenő színeket használhatod. A piros szín kiválasztásához kattints a négyzetbe, majd a piros Stop gombra!

A vonal vastagságát állítja be

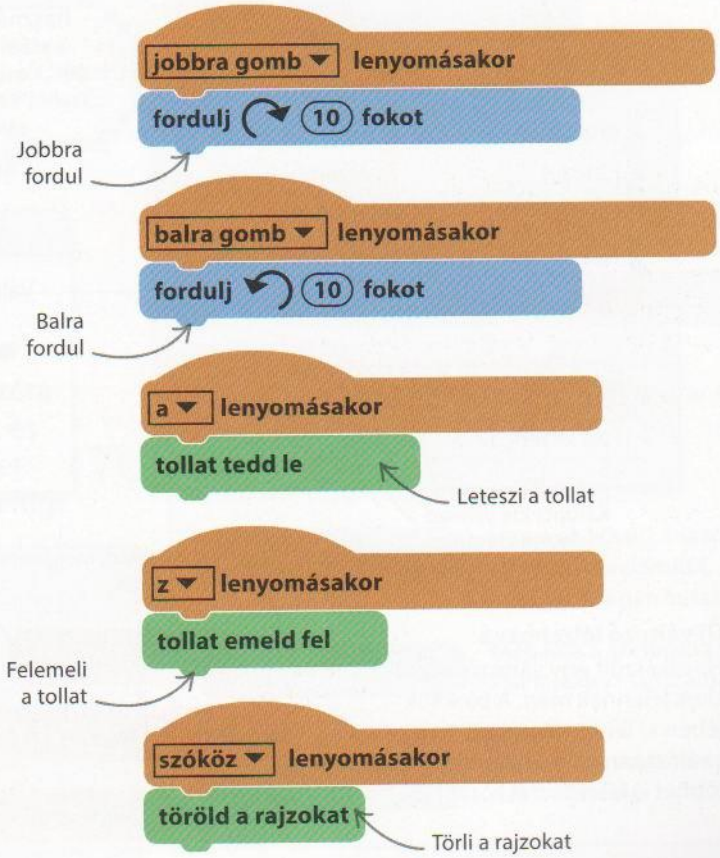
Mozgásban tartja a repülőt

A játéktéren tartja a repülőt

FOGALOMTÁR

Teknőcgrafika

A szereplővel rajzolást teknőcgrafikának nevezzük. Nevét egy teknőcnek nevezett robotról kapta, amely a földön mozogva képeket rajzolt. Az első programozási nyelv, amely teknőcgrafikát használt, a Logo volt.



Jobbra fordul

Balra fordul

Leteszi a tollat

Felemeli a tollat

Törli a rajzokat

Változók

LÁSD MÉG
Matematika 52–53 ❯
Változók a Pythonban 108–109 ❯

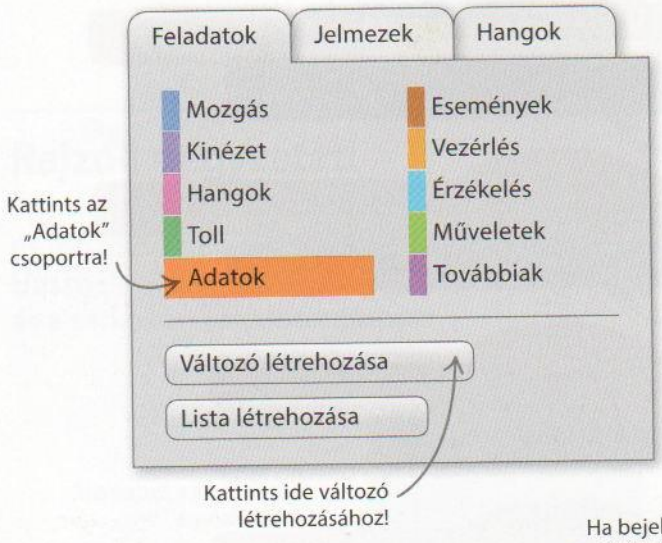
A programozásban változón egy megnevezett helyet értünk, amelyben információt tudunk tárolni. A változókban különböző adatokat, például pontszámot, a játékos nevét vagy egy szereplő sebességét lehet tárolni.

Változó létrehozása

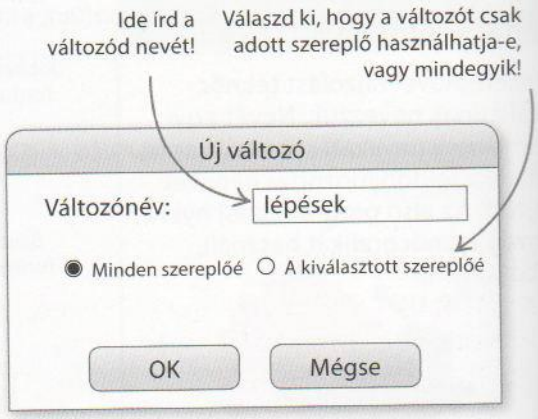
Változót az „Adatok" csoport blokkjaival lehet létrehozni. Miután a változó elkészült, új blokkok jelennek meg alatta.

◁ **Adatok tárolása**
A változók olyanok, mint a dobozok, amelyekben különféle, a programban használt adatokat tárolhatsz.

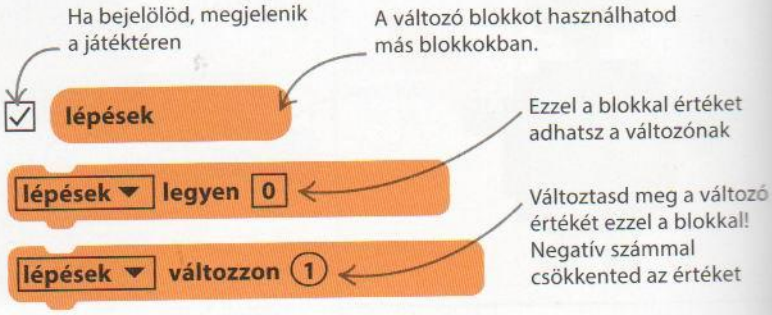
1 Hozz létre változót!
Először válaszd az „Adatok" csoportot. Ezután kattints a „Változó létrehozása" gombra.



2 Nevezd el!
Olyan nevet adj neki, amellyel könnyű beazonosítani. Válaszd ki, hogy mely szereplők használhatják, majd kattints az „OK" gombra.



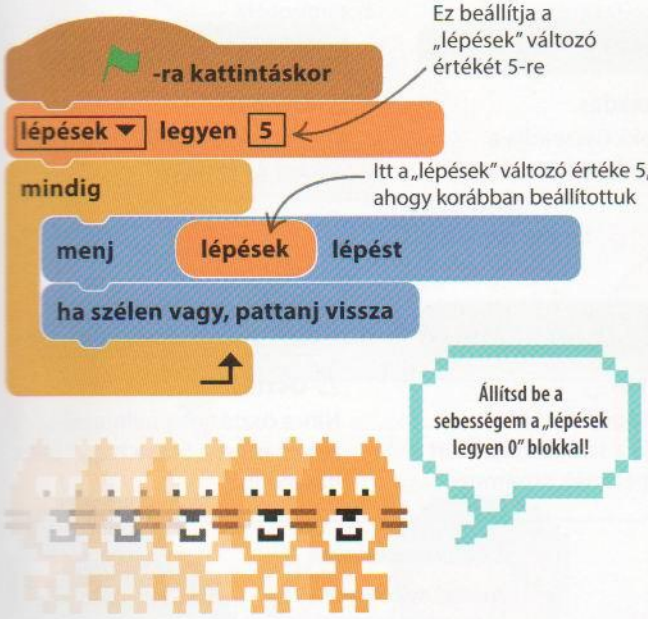
3 Új változó létrehozva
Ha elkészült egy változó, akkor új blokkok jelennek meg. A blokkok menüjében ki lehet választani, hogy melyik változóra vonatkozzanak, feltéve, hogy többet is létrehoztál korábban.



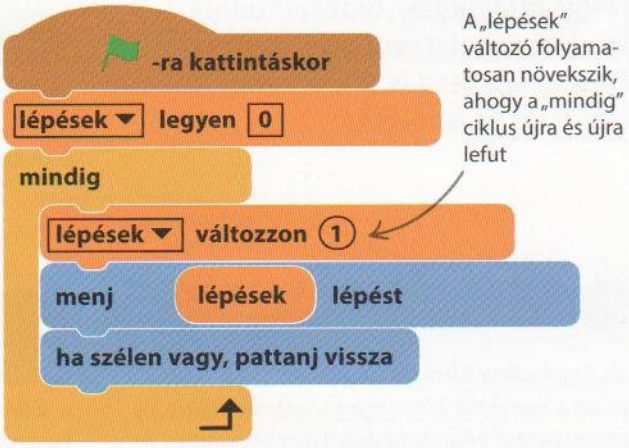
A változók használata

Változókkal módosítható egy-egy szereplő sebessége.
Ez az egyszerű utasítássorozat megmutatja, hogyan.

1 Állítsd be a változó értékét!
Hozd létre ezt az utasítássorozatot. A „lépések legyen 0" blokkot használd, és módosítsd az értékét 5-re. Húzd a „menj 10 lépést" blokkot az utasítássorozatba, és húzd a „lépések" változóblokkot a „10" helyére.



2 Módosítsd a változó értékét!
Használd a „lépések változzon 1" blokkot a „lépések" változó értékének 1-gyel való növelésére. Illeszd a „mindig" blokkon belülre, így a macska folyamatosan gyorsulni fog.



Változó törlése

Ha nincs többé szükséged egy változóra, kattints rá jobb gombbal a blokknál, és válaszd a „változó törlése" lehetőséget. A benne tárolt információ is törlődni fog.

TANÁCSOK

Beépített változók

Vannak a Scratch által beállított beépített változók, amelyeket nem tudunk módosítani. Ennek ellenére ezek is változók, hiszen az értékük változhat.

távolság — Nyomon követi a szereplő távolságát valamitől, például az egérmutatótól.

jelmez sorszáma # — Megadja a szereplő által viselt jelmez sorszámát.

irány — Megadja, melyik irányba néz a szereplő.

Matematika

A változókban tárolt számokkal (*lásd* 50–51. o.) a Scratch képes mindenféle műveletet elvégezni, a „Műveletek" blokkok segítségével.

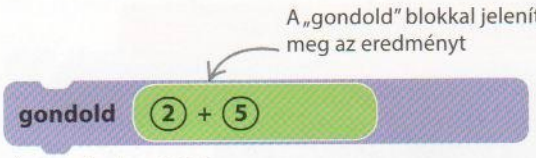
LÁSD MÉG
〈 50–51 Változók
Matematika a Pythonban 112–113 〉

Műveletek

Négy „Műveletek" blokkal tudjuk az alapműveleteket elvégezni. Ezek az összeadás, a kivonás, a szorzás és az osztás.

△ **Összeadás**
A „+" blokk összeadja a blokkban lévő két számot.

△ **Kivonás**
A „–" blokk kivonja a második számot az elsőből.



△ **Eredmény kiírása**
Húzd a „gondold" blokkot a programterületre, és illeszd bele a „+" blokkot. Adj össze két számot, és látható lesz, amint a szereplő az eredményre gondol.

△ **Szorzás**
A számítógép a „*" jelet használja a szorzáshoz, mert a „x"-et betűnek értelmezi.

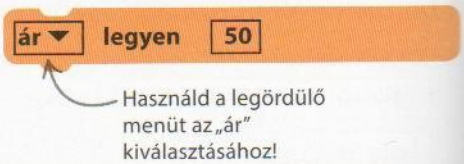
△ **Osztás**
Nincs osztásjel a billentyűzeten, ezért a Scratch a „/" jelet használja.

Eredmények a változókban

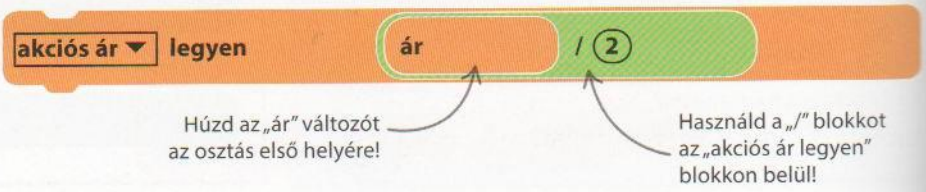
Bonyolultabb számolásoknál, mint például egy termék akciós árának a kiszámítása, számok helyett használhatsz változót a műveleten belül. Az eredményt is elmentheted egy változóba.

1 Készíts változókat!
Menj az „Adatok" csoportba, és készíts két változót – „akciós ár" és „ár".

2 Állítsd be az árat!
Válaszd az „ár legyen" blokkot, és állítsd be a termék árát 50-re.

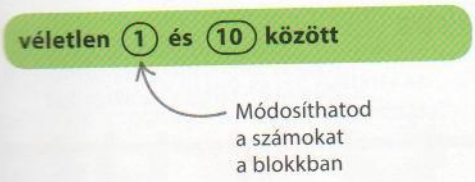


3 Számold ki az akciós árat!
Ezzel az utasítással kiszámolhatod az ár felét, és beállíthatod ezt akciós árnak.



Véletlen számok

A „véletlen" blokkal két megadott érték közé eső véletlen számot tudunk előállítani. Használhatjuk például kockadobós játékhoz vagy egy szereplő jelmezének a változtatására.

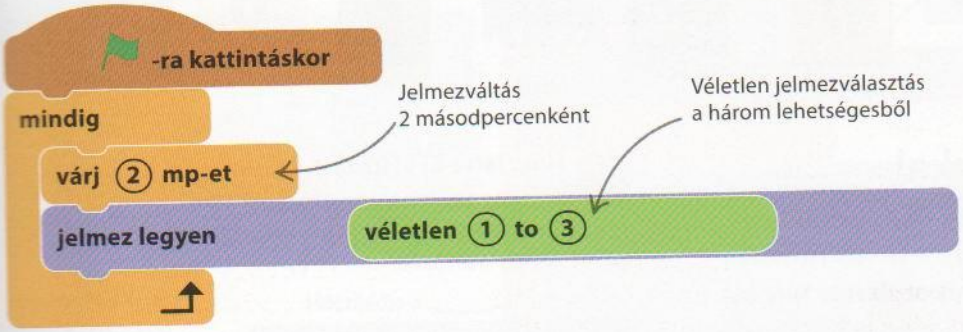


◁ **Válassz egy véletlen számot!**
Egy véletlen hónap kiválasztásához írd át az értékeket 1-re és 12-re!

TANÁCSOK

Játékok

A számítógépek gyakran használnak véletlen számokat, hogy váratlan események legyenek a játékokban. Például egy földönkívüli megjelenhet véletlenszerű helyeken vagy véletlenszerű idő elteltével. A véletlen számokat használhatod kockadobáshoz vagy egy szereplő véletlenszerű jelmezének kiválasztására is.

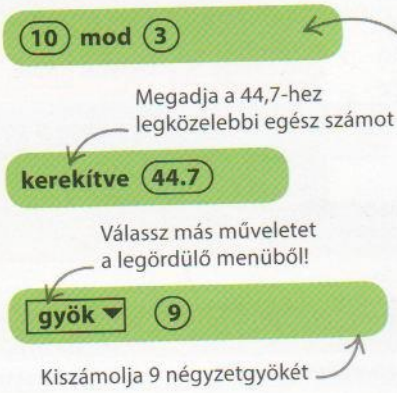


◁ **Jelmezváltás**
Ez az utasítássorozat véletlenszerűen változtatja a szereplő jelmezét 2 másodpercenként

◁ **Véletlen jelmezek**
Jelmezek váltásával imitálhatjuk egy szereplő mozgását, illetve különböző ruhákba öltöztethetjük.

Bonyolultabb matek

Az egyszerű „Műveletek" blokkokkal a legtöbb számítást el lehet végezni, de a Scratchben van lehetőség bonyolultabb műveletek elvégzésére is. A „mod" blokk eloszt két számot, és megadja a maradékot. A „kerekítve" blokk a beírt számot egész számra kerekíti, a „gyök" blokk pedig a négyzetgyökét adja meg.

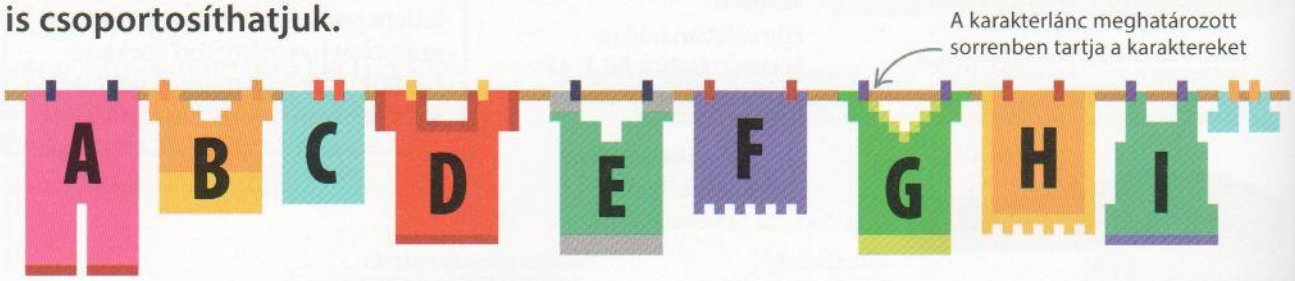


◁ **Még több matek**
A „Műveletek" csoportban bonyolultabb matematikai műveletek is találhatók.

Karakterláncok és listák

A programozásban a betűk és szimbólumok sorozatát karakterláncnak (sztringnek) nevezzük. Egy karakterlánc tetszőleges karaktereket tartalmazhat (szóközt is), és bármilyen hosszú lehet. A karakterláncokat listákba is csoportosíthatjuk.

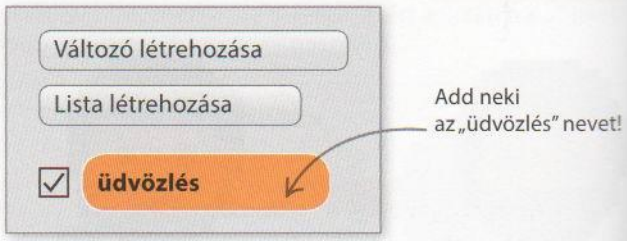
LÁSD MÉG
‹ 50–51 Változók
Karakterláncok a Pythonban 114–115 ›



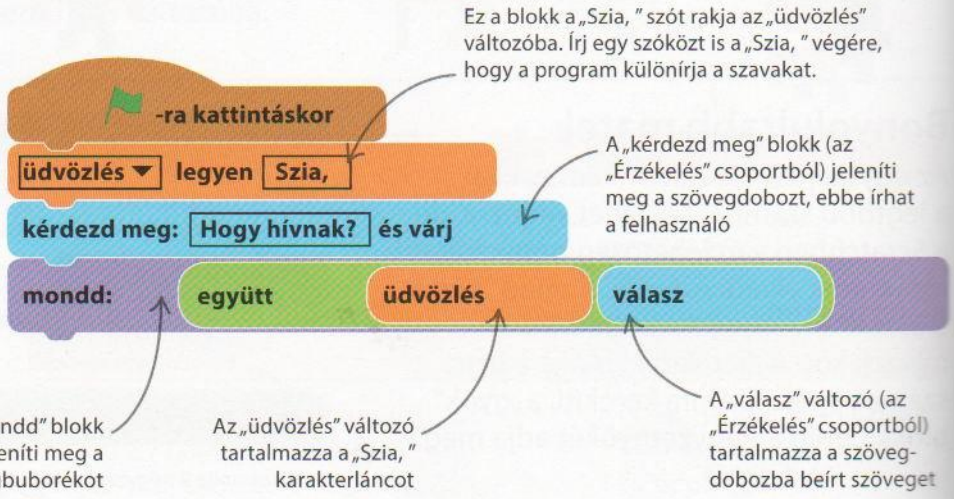
Feladatok szavakkal

A programokban gyakran kell szavakat megjegyezni, például egy játékos nevét. Ezeket a szavakat változóban lehet tárolni. A Scratch-programokban meg lehet kérdezni a felhasználót, a választ pedig egy felbukkanó szövegdobozba lehet beírni. A következő utasítássorozat megkérdezi a felhasználó nevét, és utána köszön neki.

1 Hozz létre új változót!
Kattints az „Adatok" csoportra, majd a „Változó létrehozása" gombra. Hozd létre az „üdvözlés" változót.



2 Kérdezés
Ezzel az utasítássorozattal a szereplő feltesz egy kérdést. Amit a felhasználó a szövegdobozba ír, az egy új, „válasz" nevű változóba kerül. Ezután az utasítássorozat az „üdvözlés" és a „válasz" változók kombinálásával köszön a felhasználónak.

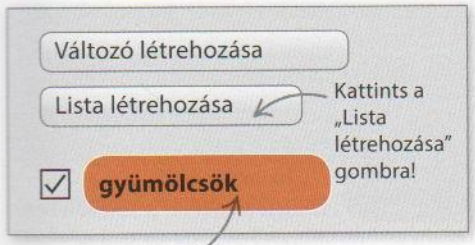


Lista készítése

A változók akkor hasznosak, ha egyetlen dolgot szeretnénk megőrizni. Ha viszont sok hasonló adatot tárolnánk, akkor inkább listákat használunk. A listák sok adatot (számokat és karakterláncokat) képesek egyszerre tárolni – például az összes rekordot egy játékban. Az alábbi program bemutatja a lista használatát.

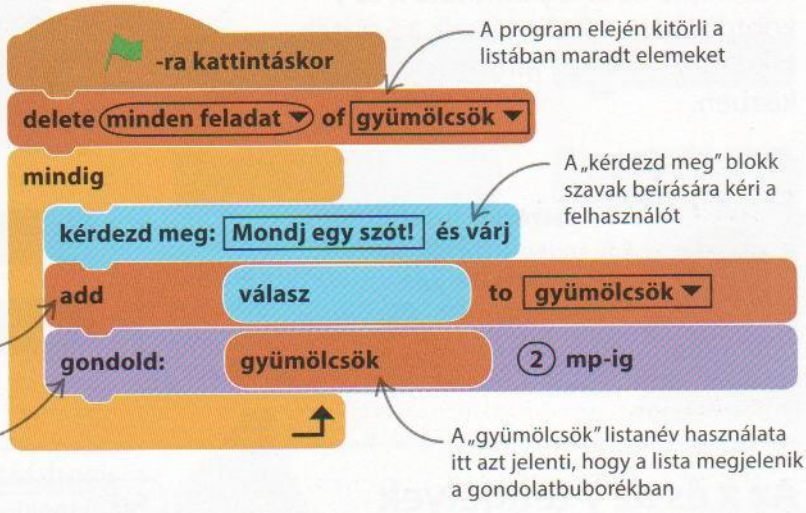
1 Hozz létre új listát!

Kezdj egy új projektet. Kattints az „Adatok" csoportban a „Lista létrehozása" gombra. Add a listának a „gyümölcsök" nevet.



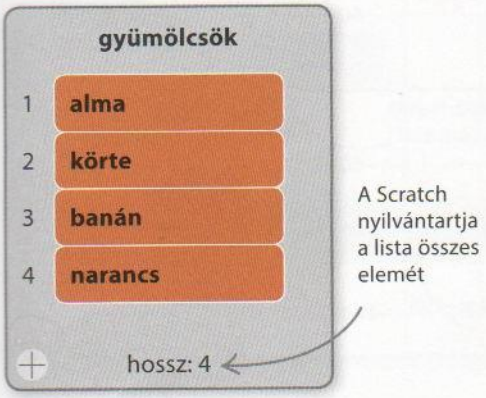
2 A lista használata

Ez az utasítássorozat arra kéri a felhasználót, hogy írjon szavakat egy listába. Minden szó megjelenik egy gondolatbuborékban, miután bekerült a listába.



3 A lista megjelenítése

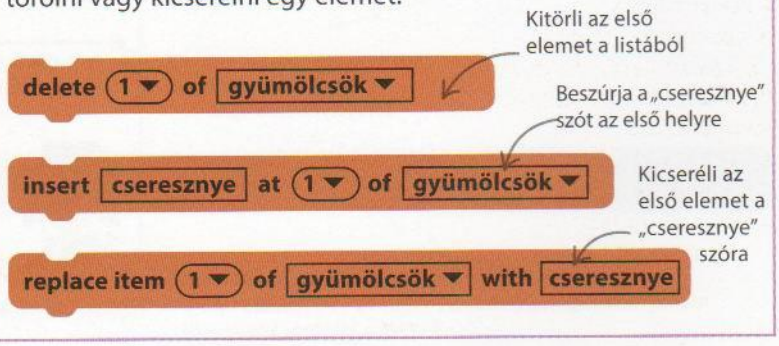
Ha bejelölöd a lista melletti négyzetet, a játéktéren megjelenik a lista tartalma. Így minden egyes új elemet láthaszt, amint bekerül a listába.



TANÁCSOK

Játék a listákkal

Ezek a blokkok megváltoztatják egy lista tartalmát. A listában minden elemhez tartozik egy szám – az elsőhöz 1-es, és így tovább. Ezeknek a számoknak a segítségével lehet beszúrni, törölni vagy kicserélni egy elemet.



Koordináták

Ahhoz, hogy egy szereplőt egy meghatározott helyre tegyünk, koordinátákat használunk. A koordináták olyan számpárok, amelyek pontosan megadják a szereplő helyzetét vízszintesen (x) és függőlegesen (y).

LÁSD MÉG
❮ 38–39 Mozgásban
❮ 52–53 Matematika

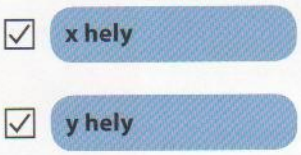
Az x és az y koordináták

A szereplő és az egérmutató x és y koordinátája mindig látszik a Scratch felületén. Hasznos információ progamírás közben.

x: 240 y: 180

△ **Az egérmutató helye**
Az egérmutató koordinátái a játéktér jobb alsó sarkában találhatók. Mozgasd az egeret a játéktéren, és figyeld, hogyan változnak a koordinátái!

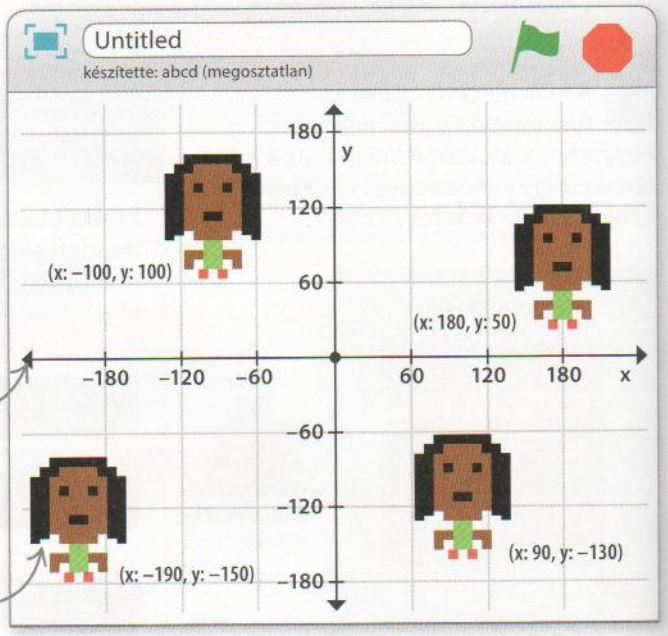
◁ **A szereplők helye**
Egy szereplő koordinátái a programterület jobb felső sarkában láthatók.



◁ **Koordináták megjelenítése a játéktéren**
Jelöld be a négyzeteket az „x hely" és az „y hely" blokkok mellett, hogy megjelenjenek a játéktéren!

Az x és az y tengelyek

Egy pont helyzetének meghatározásához le kell számolni a lépéseket vízszintes és függőleges irányban a játéktér közepétől kezdve. A vízszintes irány az x, a függőleges az y tengely. Negatív számokkal mozoghatsz balra vagy lefelé.

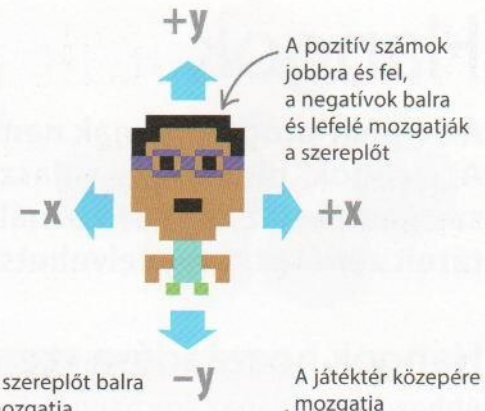


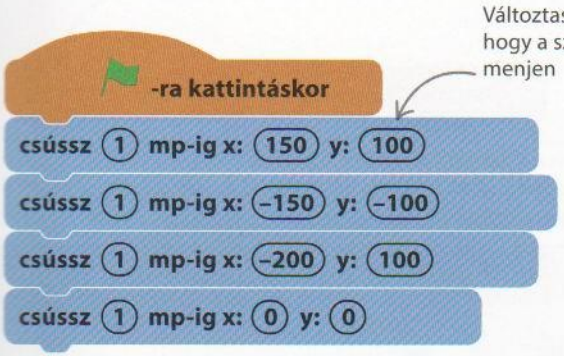
A játéktér az x és y tengelyeken alapul

Ez a szereplő 190 lépésre van balra (–190) és 150 lépésre lefelé (–150) a játéktér közepétől

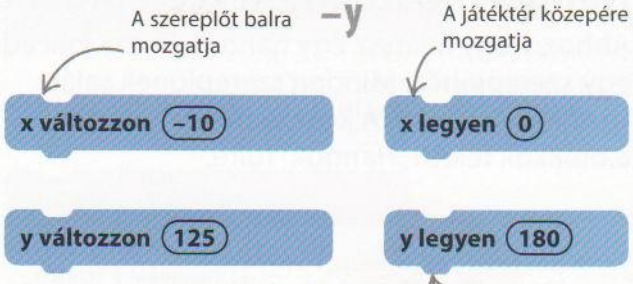
A szereplő irányítása

A koordináták segítségével egy pontosan meghatározott helyre mozgathatunk egy szereplőt. Nem számít, hogy milyen közel vagy távol van az a pont. A „csússz 1 mp-ig x: 0 y: 0" a „Mozgás" csoportból egyenletesen csúsztatja oda a szereplőt.





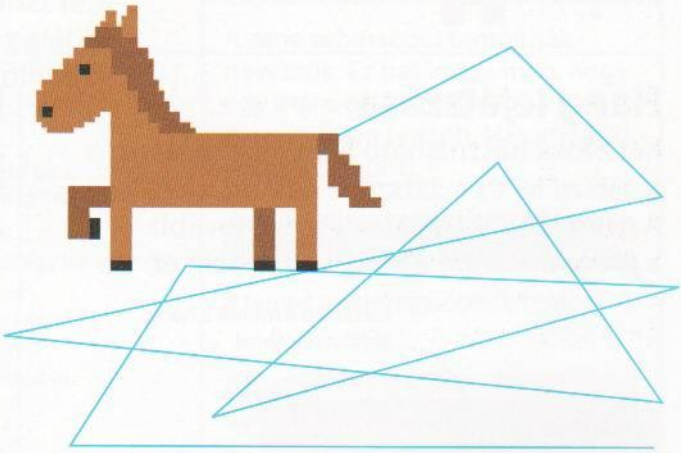
△ **Irányítsd a szereplőt utasításokkal!**
Kitalálod, hogy milyen útvonalon fog mozogni a szereplő? Próbáld ki, és nézd meg!

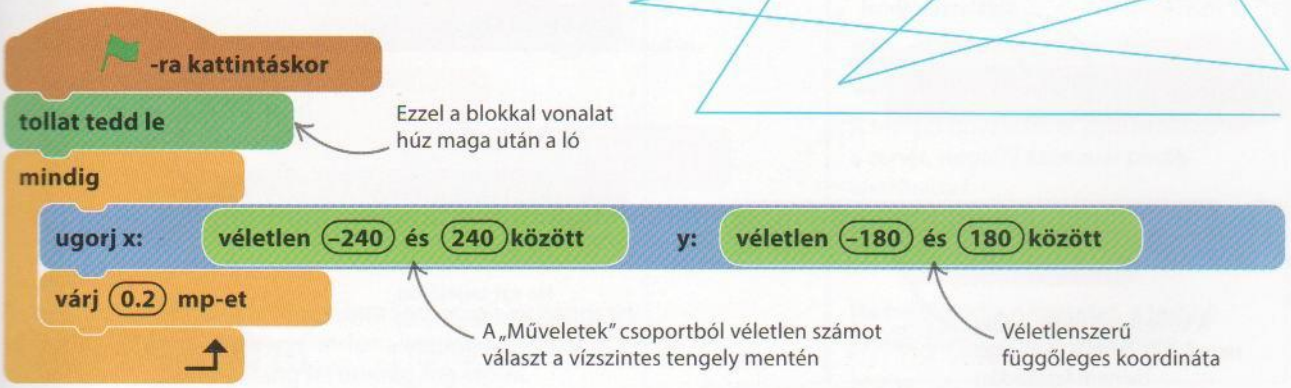


△ **Az x és az y külön változik**
Ezekkel a blokkokkal az x koordinátát módosíthatod az y megváltoztatása nélkül, és fordítva.

Őrült lóugrások

Próbáld ki ezt a mókás utasítássorozatot, hogy gyakorold a koordinátákat. Válaszd ki a „Horse1" szereplőt, és add hozzá az alábbi blokkokat. A program az „ugorj x: 0 y: 0" blokkot használja, hogy véletlen helyekre mozgassa a lovat, vonalat húzva maga után, amerre jár.





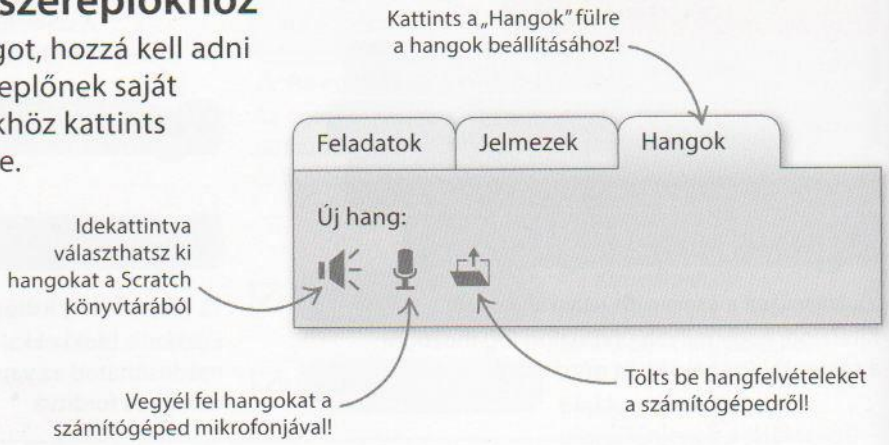
Hangok

LÁSD MÉG
Érzékelés 66–67 ›
Mókás maki 74–81 ›

A Scratch-programoknak nem kell csendben futniuk. A „Hangok" blokkjaival választhatsz a hangok közül és szerkeszthetsz zenét. Használhatod a számítógépeden tárolt zenéket, vagy felvehetsz akár teljesen újakat is.

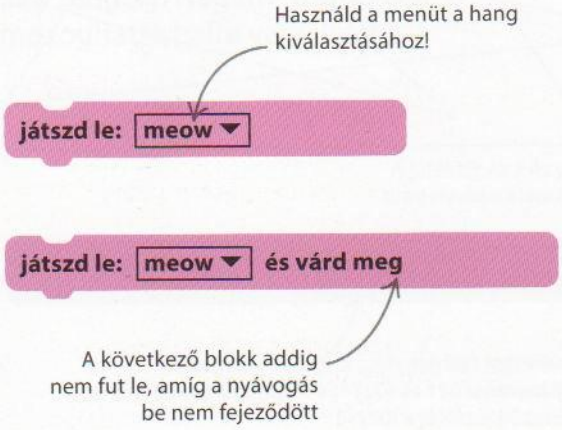
Hangok hozzáadása szereplőkhöz

Ahhoz, hogy lejátssz egy hangot, hozzá kell adni egy szereplőhöz. Minden szereplőnek saját hangkészlete van. A kezelésükhöz kattints a blokkok feletti „Hangok" fülre.



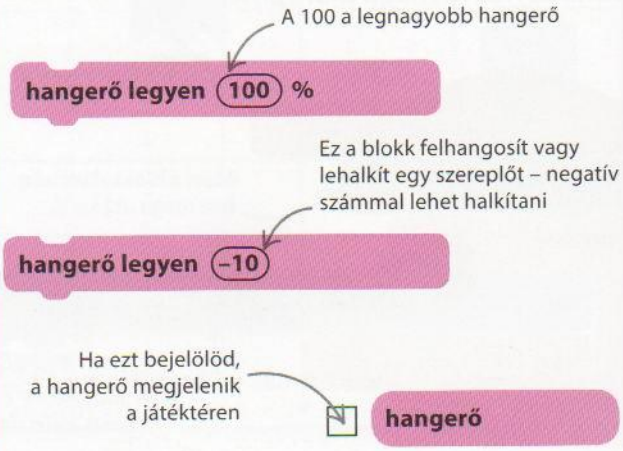
Hang lejátszása

Két blokk használható hang lejátszására: a „játszd le" és a „játszd le és várd meg". A „várd meg" hatására nem fut tovább a program, amíg a hang véget nem ér.



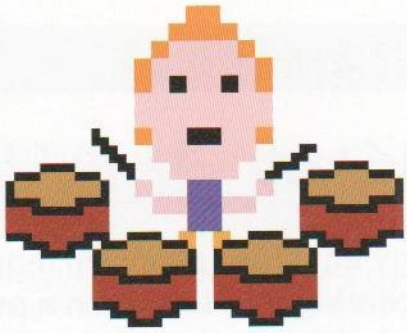
Hangosítsd fel!

A szereplők hangerejét külön-külön lehet szabályozni. A 0 a néma, a 100 pedig a leghangosabb.

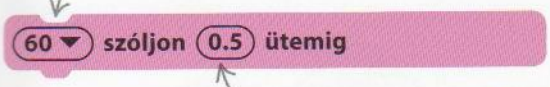


Saját zene készítése

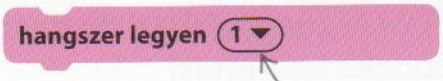
Bizonyos blokkokkal saját zenét készíthetsz. Egy zenekarnyi hangszer áll a rendelkezésedre, egy dobkészletet is beleértve. A hangok hosszát ütemben mérjük.



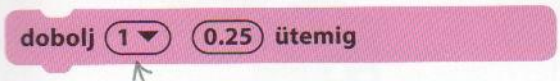
A hangmagasság beállítása



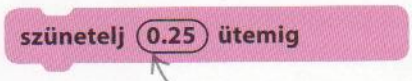
Magasabb érték hosszabb hangot eredményez. Egy hang lehet rövidebb is egy ütemnél



A legördülő menüből lehet hangszert választani



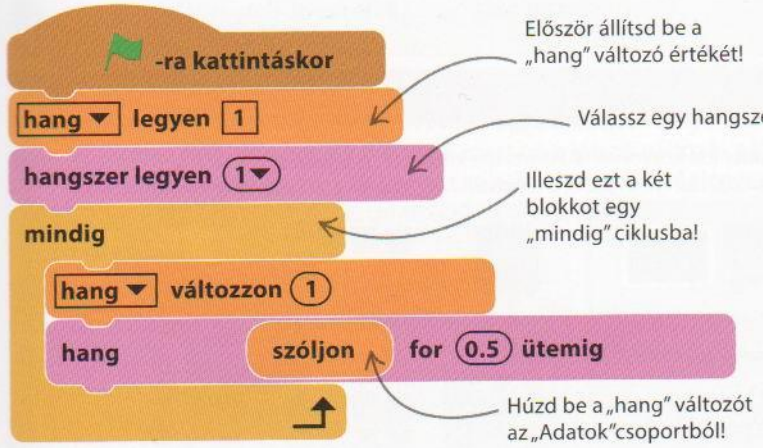
Használd ezt a menüt, és választhatsz a különféle dobok közül



Ez a blokk szünetet iktat be a zenébe. Magasabb érték hosszabb szünetet eredményez

Zene lejátszása

A hangok egymásutánja alkotja a dallamot. Hozz létre egy „hang" nevű változót (*lásd* 50–51. o.), és az alábbi utasítássorozatot add hozzá valamelyik szereplőhöz!



Először állítsd be a „hang" változó értékét!

Válassz egy hangszert!

Illeszd ezt a két blokkot egy „mindig" ciklusba!

Húzd be a „hang" változót az „Adatok" csoportból!

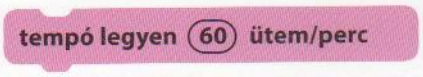
△ **Skálázás**

Ez az utasítássorozat a zöld zászlóra kattintva egy hangsort játszik le. A hangmagasság minden alkalommal fél hangot emelkedik, és minden hang fél ütemig fog szólni.

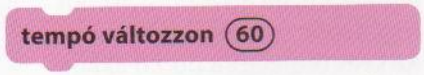
TANÁCSOK

Tempó

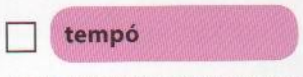
A zene sebességét tempónak nevezzük. Ez határozza meg, hogy egy ütem egy zenedarabon belül mennyi ideig tartson. Három blokk kezeli a tempót.



A tempó mértékegysége ütem/perc.



A tempó növelésével gyorsíthatod a zenét, negatív számmal pedig lassíthatod.



Ha bejelölöd a négyzetet, a tempó értéke megjelenik a játéktéren.

2. PROJEKT

Játék dobókockával

Egy egyszerű program lehet hasznos és egyben szórakoztató is. Ebben a programban egy dobókockával dobhatsz. Lehet olyat játszani, hogy ki dobja a legnagyobbat, de használhatod valódi dobókocka helyett is.

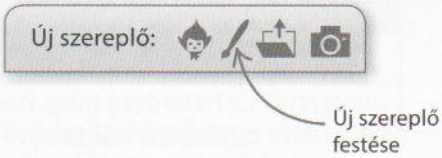
LÁSD MÉG

- **40–41** Jelmezek
- **46–47** Egyszerű ciklusok
- **50–51** Változók
- **52–53** Matematika

Hogyan készíts dobókockát?

A dobókockának a programban hat jelmeze van. Minden jelmez a kocka egy oldalát mutatja, rajta a pöttyökkel: egytől hatig.

1 Válaszd az ecsetet egy új szereplőhöz.

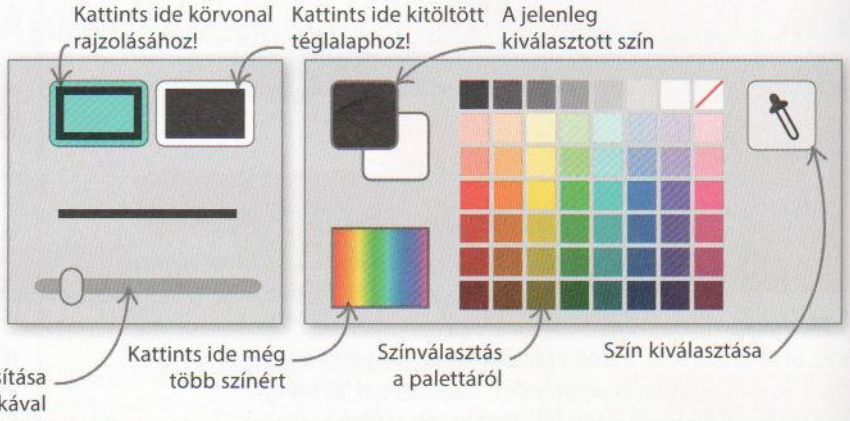


2 Kattints a téglalap eszközre a rajzterületen. Hogy szép színes legyen a kockád, válassz egy színt a palettáról és a kitöltött téglalapot (ahogy az alsó ábra mutatja). Utána a rajzterületen tartsd lenyomva a „Shift" billentyűt, nyomd le a bal egérgombot, és az egeret húzva rajzolj középre egy színes négyzetet.

TANÁCSOK

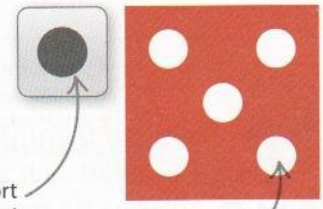
Színválasztás

A rajzterület alatt található a színválasztó. Kattints a kitöltött téglalapra, ha egyszínű, „tömör" téglalapot szeretnél rajzolni. Az üres téglalapra pedig akkor, ha egy körvonalat. Használd a csúszkát, ha a körvonal vastagságát akarod megváltoztatni. Szín választásához egyszerűen csak kattints a kívánt színre.

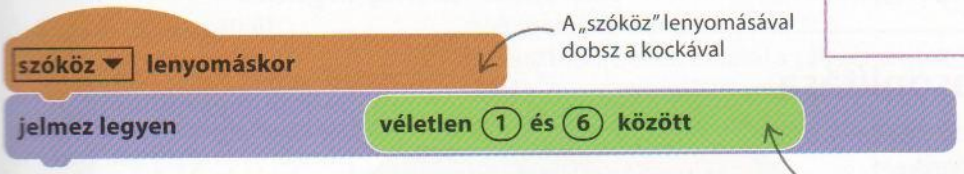


3 Kattints a jelmezre jobb egérgombbal a rajzterület bal oldalán, és válaszd a „duplázás" menüpontot. Ismételd addig, amíg hat jelmezed nem lesz.

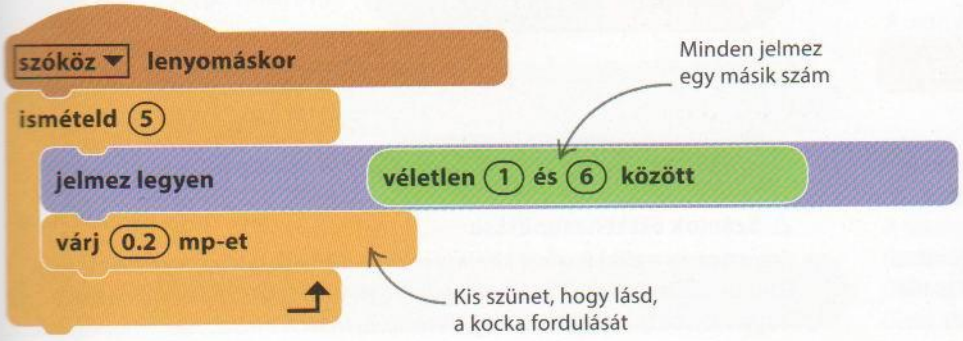
4 Válassz ki egy jelmezt. Kattints a körrajzoló eszközre, majd alul a kitöltött körre. Adj hozzá fehér köröket, és készítsd el a kocka oldalait.



5 Add az alábbi utasítássorozatot a dobókocka szereplőhöz. A „szóköz" billentyű leütésével dobhatsz a kockával. Próbáld ki néhányszor, hogy lásd, jól sikerültek-e a jelmezek.



6 Előfordulhat, hogy kétszer egymás után ugyanazt dobod. Ilyenkor úgy néz ki, mintha nem működne a program, mert nem látszik változás a képen. Ez az utasítássorozat ötször vált jelmezt, mielőtt mutatná a végleges számot. Ha leütöd a „szóközt", olyan, mintha valóban kockát dobtál volna.



TANÁCSOK

Forgatóeszköz

Ahhoz, hogy a kocka forogjon a program futása során, elforgathatod az egyes jelmezeket más és más szöggel. Kattints a jobb alsó sarokban a „Vektoros képpé alakítás" gombra! Ha visszatérsz a rajzterületre, megjelenik a forgatóeszköz.

Mentsd el a munkád!

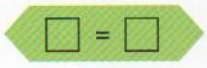
Igaz vagy hamis?

A számítógép igaz-hamis kérdések sorozatával dönti el, hogyan fusson tovább a program. Azokat a kifejezéseket, amelyek csak igaz vagy hamis értékűek lehetnek, logikai kifejezéseknek nevezzük.

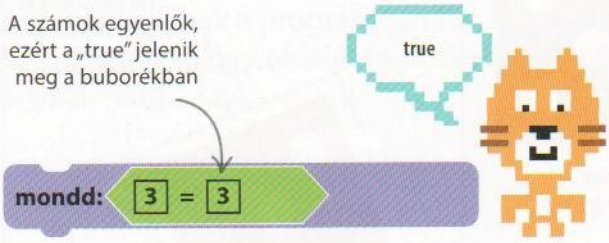
LÁSD MÉG
Feltételek és elágazások 64–65 ❯
Feltételek 118–119 ❯

Számok összehasonlítása

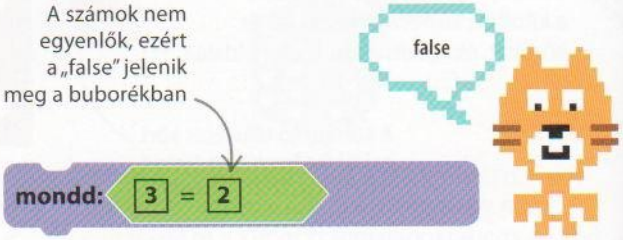
Összehasonlíthatsz számokat az „=" blokkal, a „Műveletek" csoportból.



◁ **Az „=" blokk**
Ennek a blokknak két értéke lehet: – „true" (igaz), ha a két szám egyenlő, és „false" (hamis), ha nem.



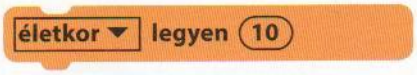
△ **Igaz állítás**
Ha az „=" blokkot egy beszédblokkba teszed, akkor vagy „true", vagy „false" fog megjelenni



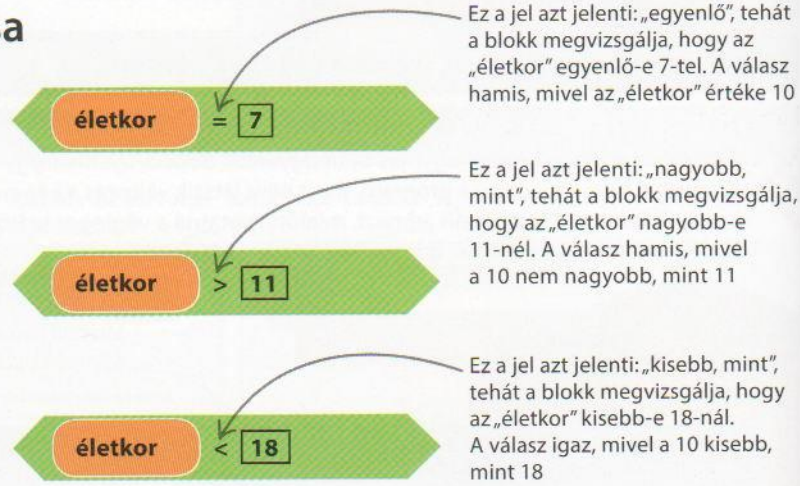
△ **Hamis állítás**
Ha a blokkon belüli számok eltérnek, akkor a „false" felirat fog megjelenni

Változók összehasonlítása

Változókat is használhatsz összehasonlító blokkokban. Konkrét számokat nincs értelme összehasonlítani, hiszen az eredmény mindig ugyanaz, a változók értéke viszont változhat.



△ **Változó létrehozása**
Az „Adatok" csoportban hozz létre új változót „életkor" néven. Az értéke legyen 10 (a blokkra kattintva ellenőrizheted, hogy megváltozott az értéke). Húzd az „életkor" változót a logikai kifejezésekbe.



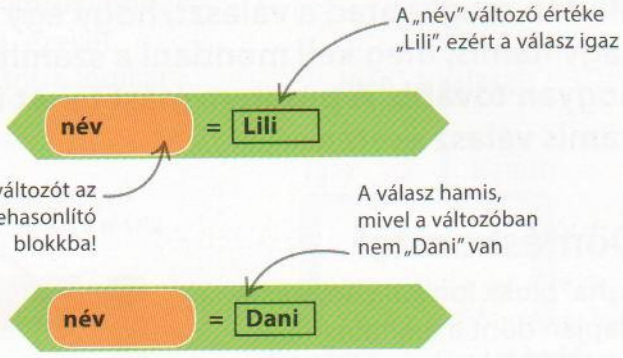
△ **Számok összehasonlítása**
Keresd meg a zöld logikai kifejezéseket a „Műveletek" csoportban. Ellenőrizheted, hogy két szám egyenlő-e, de ugyanígy megnézheted, hogy valamelyik nagyobb vagy kisebb-e, mint a másik.

TANÁCSOK

Szavak összehasonlítása

Az „=" blokkot nem csak számok esetében lehet alkalmazni – arra is választ ad, hogy két karakterlánc azonos-e. Összehasonlításkor nem veszi figyelembe a nagybetűk és kisbetűk közötti különbséget.

△ **Változó létrehozása**
Ahhoz, hogy ezt kipróbáld, hozz létre egy „név" változót, és add neki a „Lili" értéket!



Nem!

A „nem" blokk egyszerűsítheti a dolgunkat a logikai kifejezésre adott válasz megfordításával. Például könnyebb ellenőrizni, hogy valaki nem 10 éves-e, mint végigpróbálni az összes többi lehetőséget.

◁ **A „nem" blokk**
A „nem" blokk megváltoztatja a választ igazról hamisra, hamisról pedig igazra.

△ **A „nem" blokk nélkül**
A 7 nem egyenlő 10-zel, ezért a válasz hamis

△ **A „nem" blokkal**
A „nem" blokk hozzáadásával a válasz megváltozik. Mivel a 7 nem egyenlő 10-zel, a válasz igaz.

Logikai kifejezések összekapcsolása

Bonyolultabb logikai kifejezések esetén összakapcsolhatunk több összehasonlító blokkot is, ezzel egyszerre több kérdést is feltéve.

△ **Összehasonlító blokkok**
A „vagy" és az „és" blokkokkal egyszerre több logikai kifejezés is megvizsgálható.

◁ **A gyakorlatban**
A felső példa azt ellenőrzi, hogy valaki fiatalabb-e, mint 18, vagy idősebb, mint 65. Az alsó pedig azt, hogy az illető életkora 11, 12, 13 vagy 14 év-e.

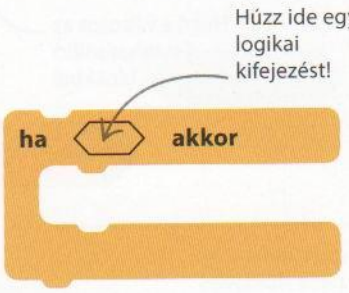
Feltételek és elágazások

Miután megkaptad a választ, hogy egy kifejezés igaz-e vagy hamis, meg kell mondani a számítógépnek, hogyan tovább. A program mást tehet igaz és mást hamis válasz esetén.

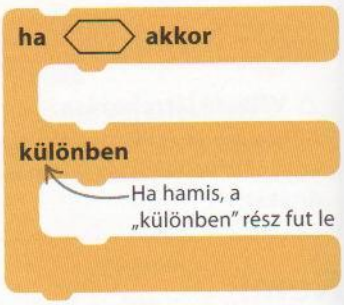
LÁSD MÉG
❮ 62–63 Igaz vagy hamis?
Érzékelés 66–67 ❯

Döntéshozatal

A „ha" blokk logikai kifejezések alapján dönt a folytatásról. A belső blokkok csak akkor futnak le, ha a logikai kifejezés igaz.



△ **„ha-akkor"**
Ha a logikai kifejezés igaz, lefutnak a belső blokkok



△ **„ha-akkor-különben"**
Ha a logikai kifejezés igaz, az első rész blokkjai futnak le, ha hamis, akkor a másodiké.

A „ha-akkor" blokk használata

A „ha-akkor" blokk segítségével választhatsz, hogy a program egy része lefusson-e a logikai kifejezés értékétől függően. Add ezt az utasítássorozatot a macska szereplőhöz, és próbáld ki!

△ **Nyávogó macska**
A program megvizsgálja a logikai kifejezést, és a belső blokkok csak akkor futnak le, ha a kifejezés igaz. Vagyis a macska csak akkor nyávog, ha arra kérjük.

Húzd az „=" blokkot a „Műveletek" csoportból a „ha-akkor" blokkba! Utána építsd bele a „válasz" változót (az „Érzékelés" csoportból)!

Ez a blokk a „ha-akkor" blokkon belül van, így a macska csak akkor nyávog, ha a válasz „igen"

Itt a vége a „ha-akkor" blokknak

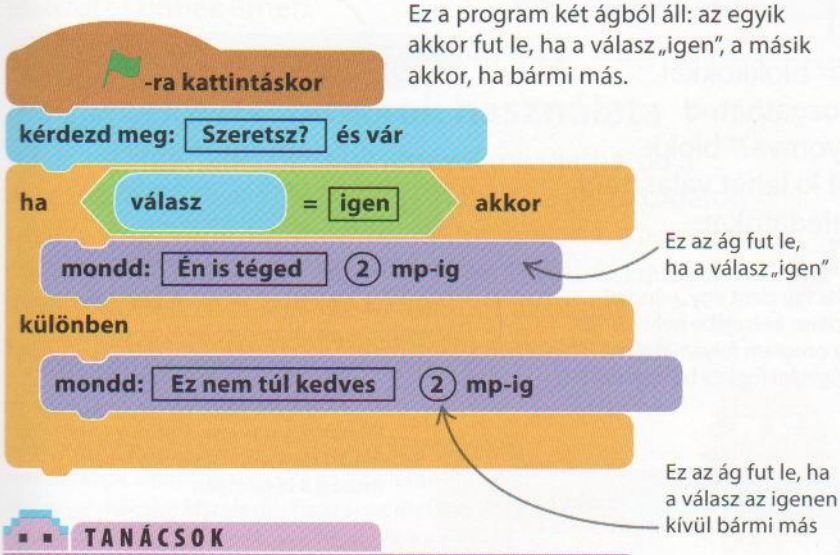
A „gondold" blokkok a „ha-akkor" blokkon kívül vannak, ezért választól függetlenül le fognak futni

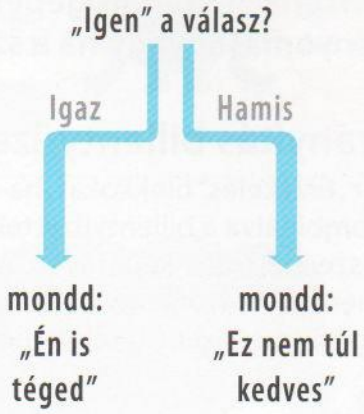
△ **Hogyan működik?**
A program megvizsgálja, hogy a logikai kifejezés igaz-e. Ha igen, lefuttatja a „ha-akkor" belsejében lévő blokkokat.

Elágazó utasítások

Gyakran szeretnénk, hogy valamit csináljon a program, ha egy kifejezés igaz, és valami mást, ha hamis. A „ha-akkor-különben" blokk segítségével két utat adunk a programnak, ezeket elágazásnak nevezzük. Csak az egyik ág fog lefutni, a logikai kifejezéstől függ, hogy melyik.

▽ **Elágazó program**
Ez a program két ágból áll: az egyik akkor fut le, ha a válasz „igen", a másik akkor, ha bármi más.





△ **Hogyan működik?**
A program megvizsgálja, hogy „igen" választ írtál-e be. Ha igen, akkor az első üzenet jelenik meg, ha nem, akkor a második.

TANÁCSOK

Logikai blokkok

A logikai kifejezések blokkjai a Scratchben csúcsosak. Ennek ellenére beleteheted őket nem szögletes lyukakba is.

△ **„Érzékelés" blokkok**
Ezek a blokkok érzékelik, hogy egy szereplő érint-e egy másikat, vagy egy billentyű le van-e nyomva.

△ **„Vezérlés" blokkok**
Több „Vezérlés" blokkban is vannak csúcsos lyukak logikai kifejezések számára.

▷ **Elágazások**
Ahogy egy fa ágai, a program is sok irányba ágazhat el futása során.

Érzékelés

Az „Érzékelés" csoport blokkjai észlelik, hogy mi történik a számítógépen. Érzékelik a billentyűk lenyomását vagy ha a szereplők összeérnek.

LÁSD MÉG
❮ 40–41 Jelmezek
❮ 56–57 Koordináták

Ez a blokk egy adott billentyű lenyomását érzékeli. A menüből kiválaszthatod, melyiket

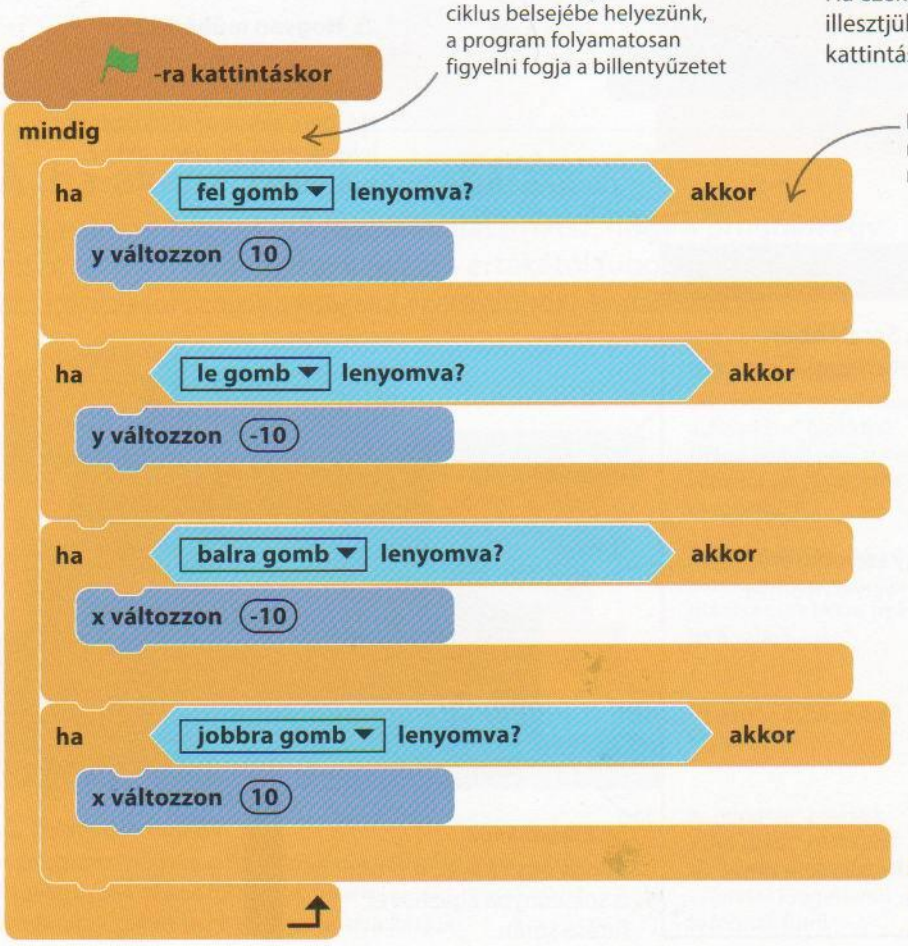
Ez a blokk az egér gomb lenyomását érzékeli

△ **„Érzékelés" blokkok**
Ha ezeket a blokkokat „ha-akkor" blokkba illesztjük, a program érzékeli az egér-kattintásokat és a billentyűlenyomásokat.

Irányítás billentyűzettel

Az „Érzékelés" blokkokat „ha-akkor" blokkokkal kombinálva a billentyűzettel is mozgathatod a szereplőket a képernyőn. A „lenyomva?" blokk menüjéből a billentyűk többségét ki lehet választani. Egérkattintáshoz is rendelhetsz feladatokat.

Ha mindent egy „mindig" ciklus belsejébe helyezünk, a program folyamatosan figyelni fogja a billentyűzetet



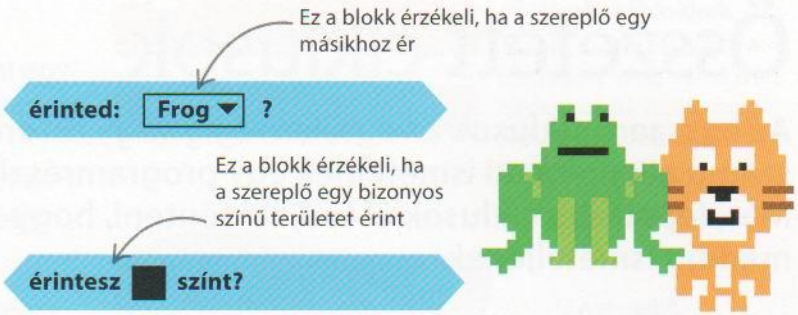
Ellenőrzi, hogy lenyomtuk-e a „fel" nyilat. Ha igen, a szereplő felfelé mozog a képernyőn

△ **A szereplők mozgatása**
A billentyűkkel pontosan irányíthatod a szereplőket, különösen játékoknál hasznos.

◁ **Mozgató utasítássorozat**
Ez az utasítássorozat lehetővé teszi, hogy a szereplőket nyílbillentyűkkel mozgasd fel, le, jobbra és balra.

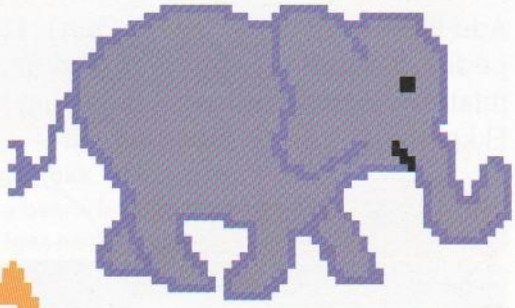
Szereplők ütközése

Jó tudni, ha egy szereplő hozzáér egy másikhoz – például játékokban. Az „Érzékelés" blokkok használatával a program reagálhat arra, ha a szereplők egymáshoz vagy egy meghatározott színhez érnek.



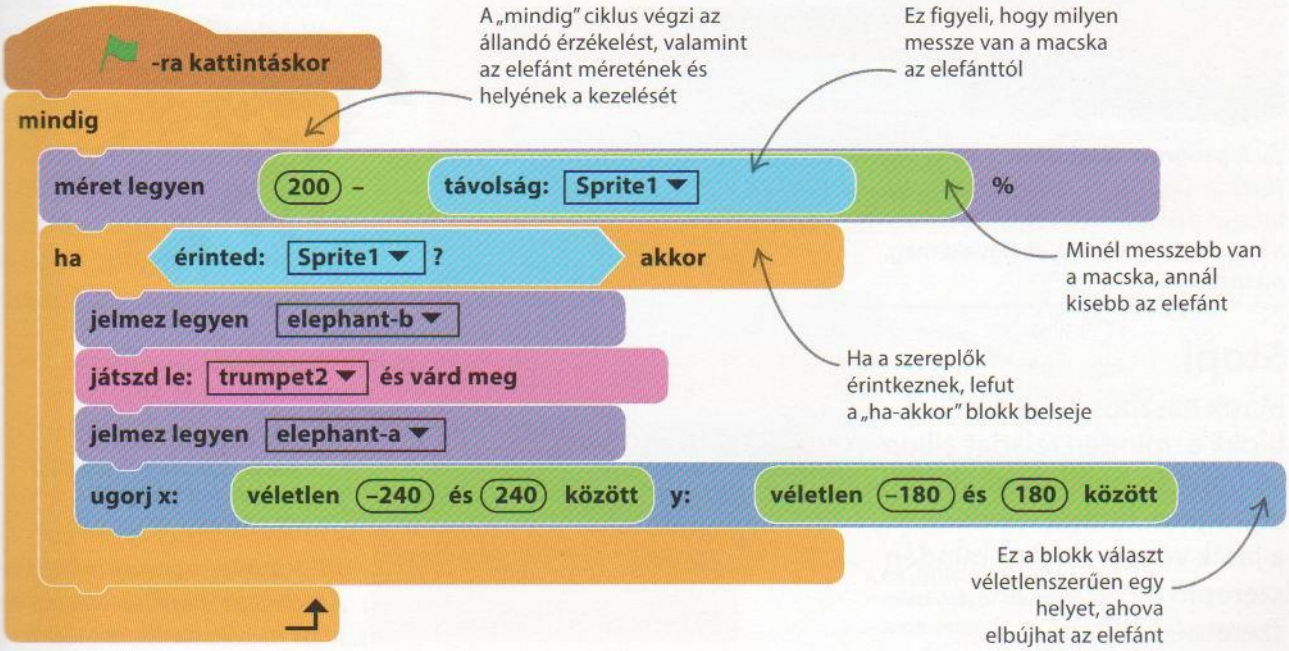
Az „Érzékelés" blokkok használata

Az „Érzékelés" blokkok segítségével mozgathatod a macskát a játékban. Add a macskához az előző oldalon található mozgató utasítássorozatot, állítsd be a „room1" hátteret, és add hozzá az elefánt szereplőt! A „Hangok" fülről add hozzá a „trumpet2" hangot az elefánt szereplőhöz, és írd be az alábbi utasítássorozatot!



▽ **Találd meg az elefántot!**

Ez a program az „Érzékelés" blokkok segítségével teremt kapcsolatot a macska és az elefánt között. Ahogy a macska közeledik hozzá, az elefánthoz elkezd növekedni. Amikor a macska hozzáér, az elefánt jelmezt vált, hangot ad ki, és elugrik.



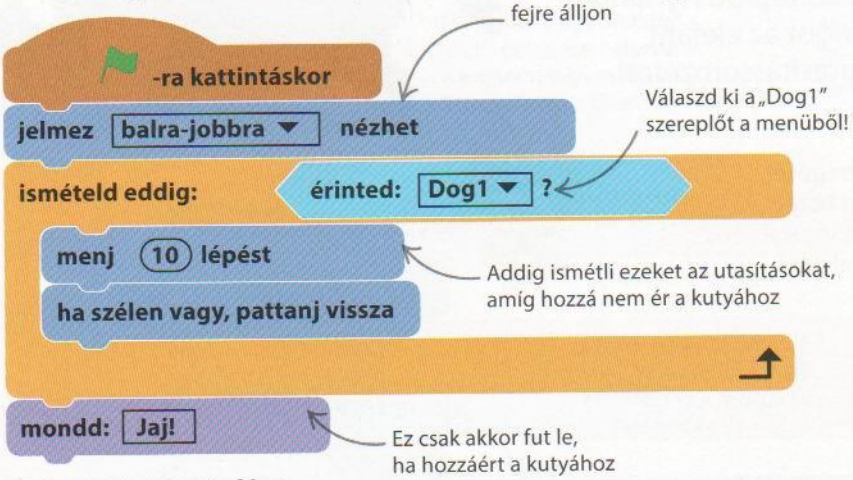
Összetett ciklusok

Az egyszerű ciklusok a végtelenségig vagy bizonyos számú alkalommal ismételnek egy programrészletet. Más, fejlettebb ciklusok el tudják dönteni, hogy meddig ismételjenek.

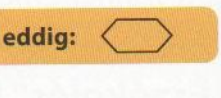
LÁSD MÉG
❮ 46–47
Egyszerű ciklusok
❮ 62–63
Igaz vagy hamis?

Ismétlés valamilyen esemény bekövetkeztéig

Add hozzá egy projekthez a „Dog1" szereplőt, a macskához pedig az alábbi utasítássorozatot! Az „ismételd eddig" ciklus miatt mindaddig mozogni fog, amíg hozzá nem ér a kutyához. Ekkor megáll, és azt mondja: „Jaj!"



△ **A program tesztelése**
Húzd el a kutyát a macska útjából, és futtasd a programot! Ezután húzd vissza a kutyát a macska elé, és figyeld meg, mi történik!



△ **Az „ismételd eddig" blokk**
A blokkon belül lévő utasításokat addig ismétli, amíg a megadott feltétel nem teljesül (a macska hozzáér a kutyához).

Stop!

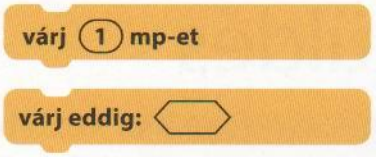
Másik hasznos „Vezérlés" blokk a „minden feladat álljon le" utasítás, amely megállítja a program futását. Például a játék végén, amikor minden szereplő mozgását le szeretnéd állítani.

◁ **Utasítássorozatok leállítása**
A legördülő menüből válaszd ki, hogy melyik utasítássorozatot szeretnéd leállítani.

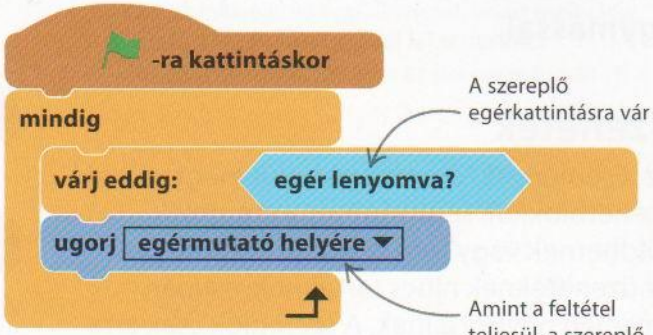
Várakozás

Könnyebb úgy játszani vagy megnézni egy program működését, ha egy pillanatra meg lehet állítani a futást. Vannak blokkok, amelyekkel adott idejű szünetet tudunk elérni, és vannak olyanok is, amelyek egy feltétel teljesülésére várnak.

△ **A „várj … mp-et" blokk**
A „várj … mp-et" blokkba beírhatod, hány másodpercig várjon a szereplő.



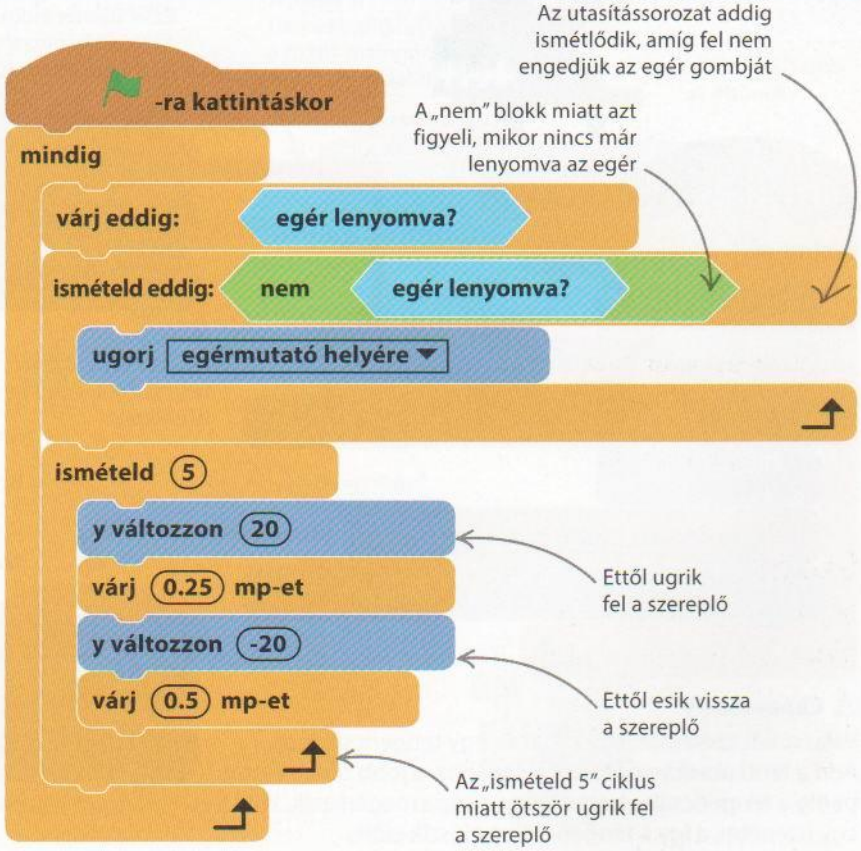
◁ **Várakozóblokkok**
A „várj … mp-et" blokk a megadott ideig vár. A „várj eddig:" blokk pedig reagál arra, ami a programban történik.



△ **A „várj eddig" blokk**
Addig vár, amíg a benne lévő logikai kifejezés igaz nem lesz.

Mágneses egér

Egy programban különböző ciklusokat egyszerre is használhatunk. Ez a program akkor indul, ha kattintunk az egérrel. A szereplő addig követi az egérmutatót, amíg lenyomva tartjuk a gombot. Amikor elengedjük, a szereplő ötször felugrik. Ezután az egész kezdődik elölről, mivel egy „mindig" ciklusban van.



▷ **Egymásba ágyazott ciklusok**
Figyeld meg, hogyan ágyazódnak egymásba a ciklusok a „mindig" blokkon belül!

Üzenetküldés

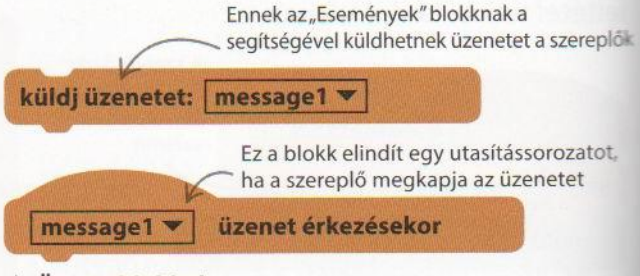
A szereplők üzenetek segítségével közölhetik a többi szereplővel, hogy mit csináljanak. A Scratch azt is lehetővé teszi, hogy a szereplők beszélgessenek egymással.

LÁSD MÉG

- ❮ **38–39** Mozgásban
- ❮ **40–41** Jelmezek
- ❮ **44–45** Események

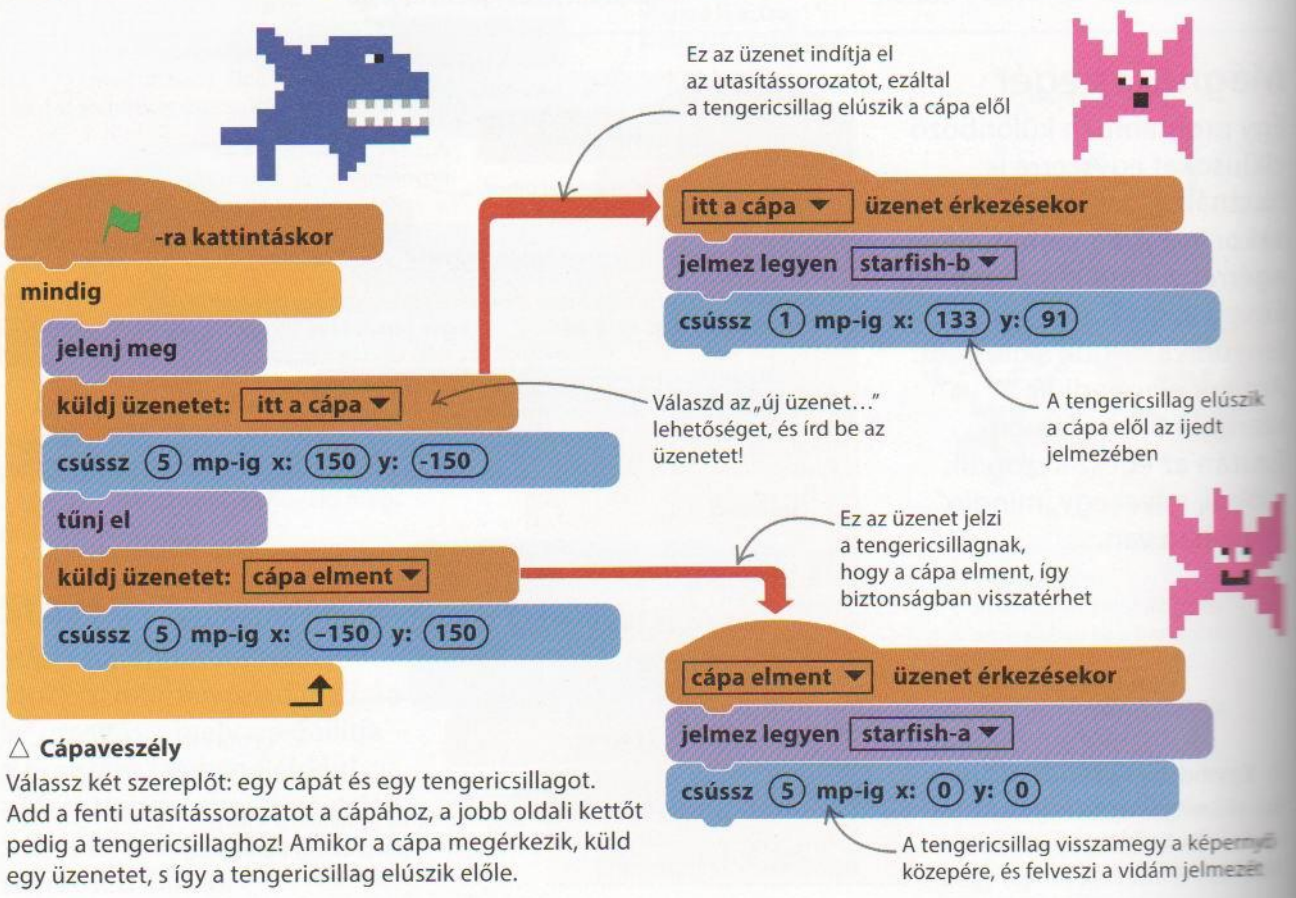
Üzenetek

Az „Események" csoporton belül megtalálható üzenetblokkok segítségével a szereplők küldhetnek vagy fogadhatnak üzeneteket. Az üzeneteknek nincs tartalmuk, csupán egyetlen névből állnak. A szereplők csak arra az üzenetre reagálnak, amelyekre be vannak programozva, a többit figyelmen kívül hagyják.



△ **Üzenetblokkok**

Az üzenetblokkok egyik fajtájával a szereplő üzenetet küldhet, egy másikkal pedig fogadhat. Választhatsz a meglévő üzenetek közül, de készíthetsz újat is.



△ **Cápaveszély**

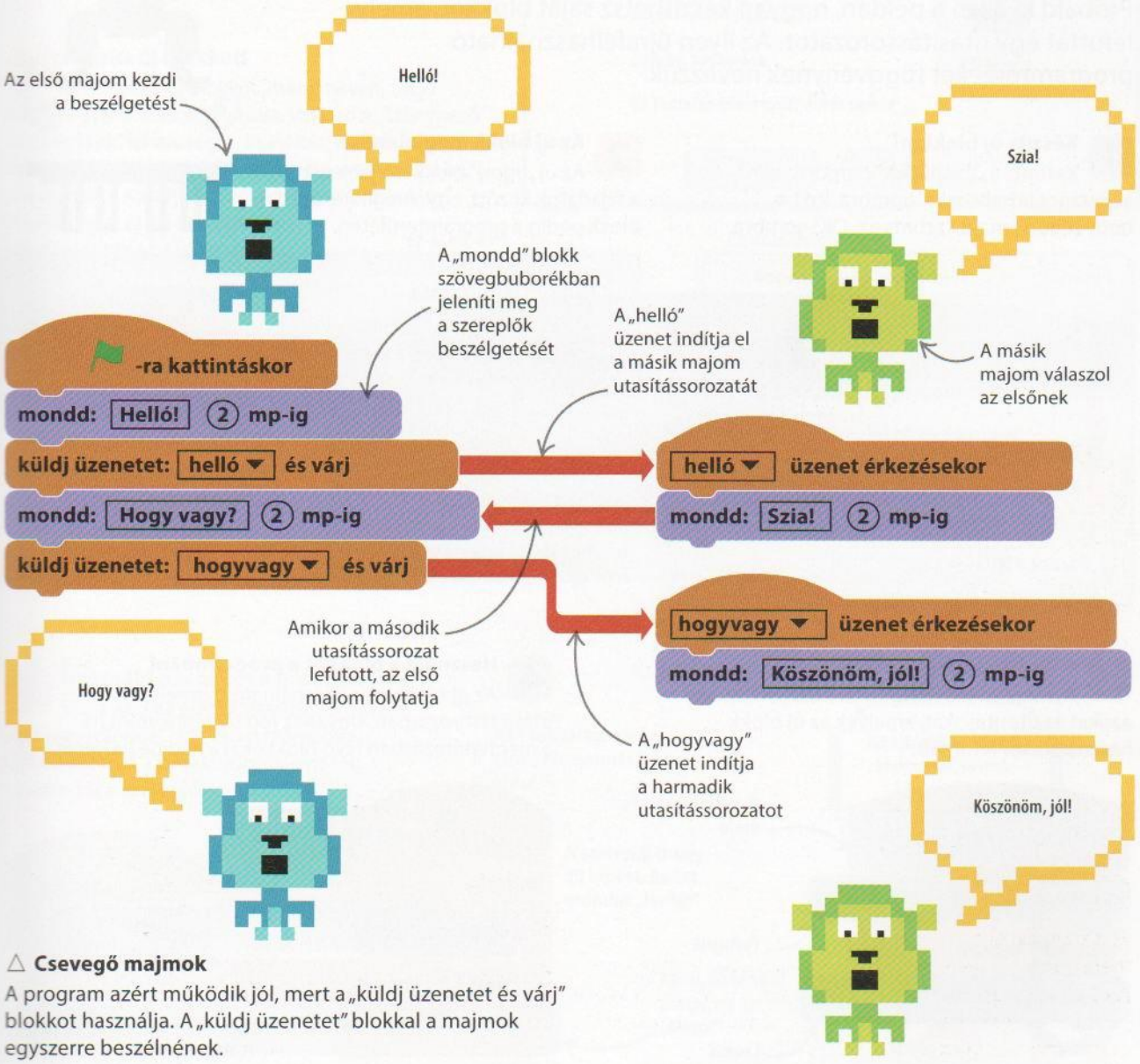
Válassz két szereplőt: egy cápát és egy tengericsillagot. Add a fenti utasítássorozatot a cápához, a jobb oldali kettőt pedig a tengericsillaghoz! Amikor a cápa megérkezik, küld egy üzenetet, s így a tengericsillag elúszik előle.

Beszélgetések

A szereplők közötti beszélgetésekhez használd a „küldj üzenetet és várj", valamint a „mondd" blokkokat. Kezdj új projektet, és adj hozzá két majom (Monkey1, Monkey2) szereplőt. A bal oldali utasítássorozatot add az egyik, a jobb oldali kettőt pedig a másik majomhoz.

küldj üzenetet: message1 és várj

△ **Várakozó blokkok**
Ez a blokk elküld egy üzenetet, majd megvárja, amíg a többi szereplőnél lefut minden utasítássorozat, ami erre az üzenetre indult el.



△ **Csevegő majmok**
A program azért működik jól, mert a „küldj üzenetet és várj" blokkot használja. A „küldj üzenetet" blokkal a majmok egyszerre beszélnének.

Blokkok készítése

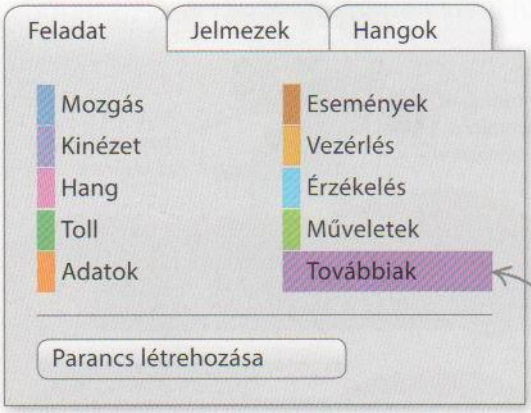
LÁDS MÉG
〈 50–51 Változók
Kísérletezz!
82–83 〉

Ahelyett hogy blokkok ugyanolyan sorát újra és újra beírnánk, készíthetünk belőlük egy saját blokkot. Minden ilyen új blokk több különböző utasítást tartalmazhat.

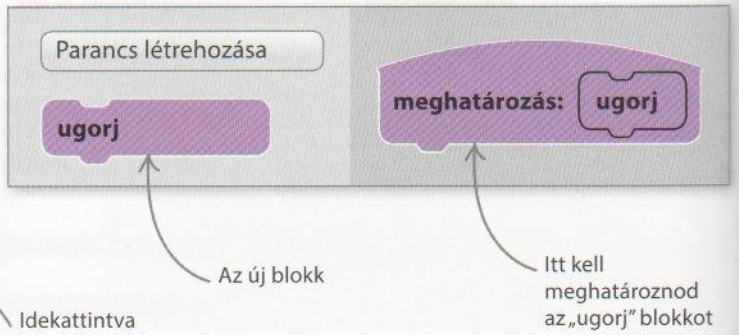
Saját blokk készítése

Próbáld ki ezen a példán, hogyan készíthetsz saját blokkot, amely lefuttat egy utasítássorozatot. Az ilyen újrafelhasználható programrészeket függvénynek nevezzük.

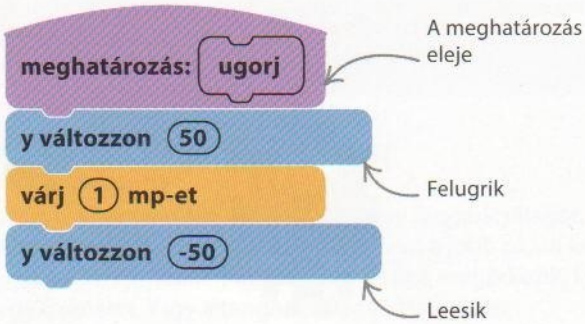
1 Készíts új blokkot!
Kattints a „Továbbiak" csoportban a „Parancs létrehozása" gombra. Írd be, hogy „ugorj", majd kattints az „OK" gombra.



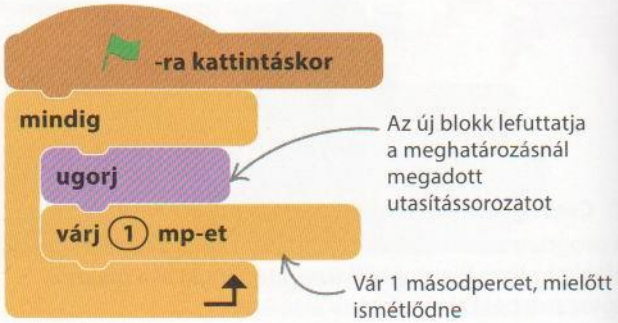
2 Az új blokk megjelenése
Az új „ugorj" blokk megjelenik a feladatok között, egy „meghatározás" blokk pedig a programterületen.



3 A blokk meghatározása
A „meghatározás" blokkal hozzáadhatod azokat az utasításokat, amelyek az új blokk használatakor lefutnak.

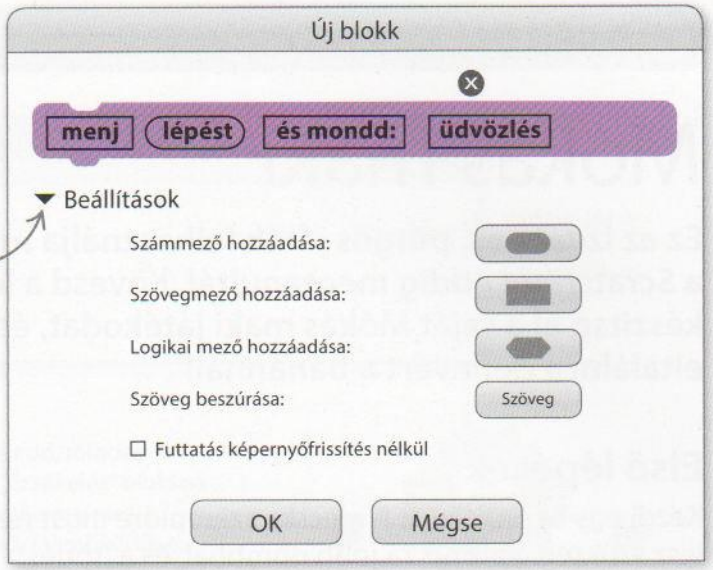


4 Használd a blokkot a programban!
Az új blokk már használható bármelyik utasítássorozatban. Ugyanaz fog történni, mint ha a meghatározásban lévő blokkokat írnád be helyette.



Blokkok bemenettel

Egy új blokknak ablakok segítségével át lehet adni számokat vagy szavakat, hogy azokkal dolgozzon. Ezekkel az ablakokkal megadható például, hogy mennyit lépjen a szereplő.



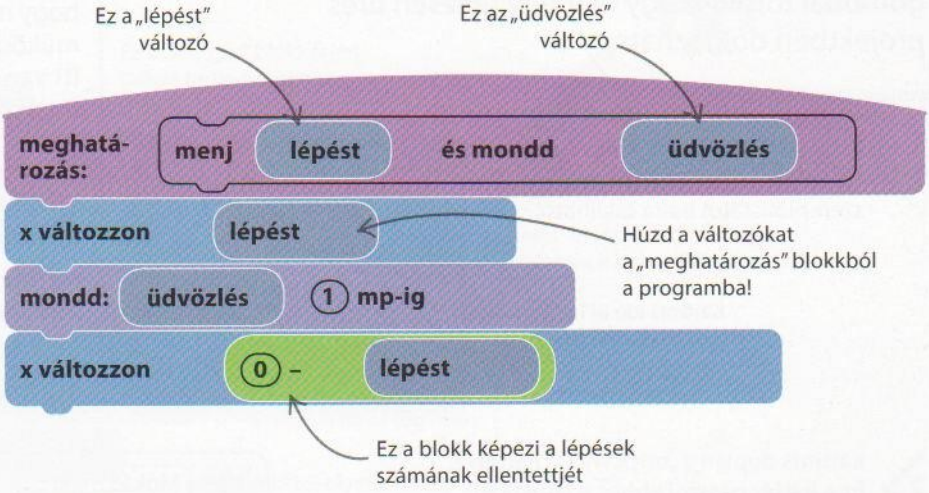
Kattints ide a beállítások megjelenítéséhez!

1 Készíts új blokkot!

Készíts egy új blokkot „menj" néven, majd kattints a „Beállítások" gombra. Válaszd a „Számmező hozzáadása:" lehetőséget, és írd be: „lépést". Válaszd a „Szöveg beszúrása:" lehetőséget, és írd be: „és mondd:". Kattints a „Szövegmező hozzáadása" lehetőségre, és írd be: „üdvözlés". Végül kattints az „OK" gombra.

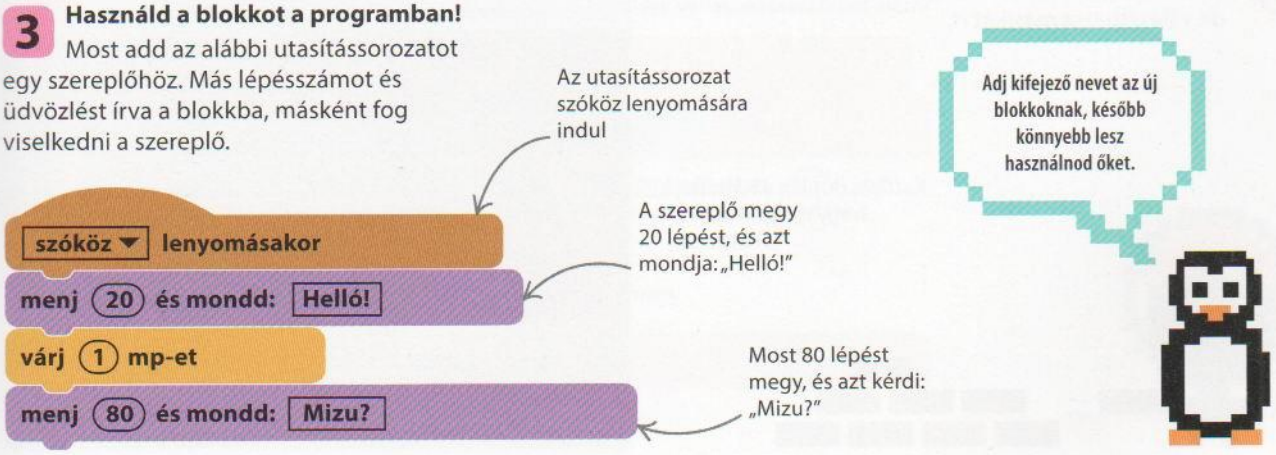
2 Blokk definiálása

A „meghatározás" blokkban az ablakok a „lépést" és az „üdvözlés" változókra cserélődnek. Húzd ezeket a változókat az utasítássorozat bármely részére, ahol szükség van rájuk. Add ezt az utasítássorozatot egy szereplőhöz.



3 Használd a blokkot a programban!

Most add az alábbi utasítássorozatot egy szereplőhöz. Más lépésszámot és üdvözlést írva a blokkba, másként fog viselkedni a szereplő.



3. PROJEKT

Mókás maki

Ez az izgalmas, pörgős játék felhasználja mindazt, amit a Scratchből eddig megtanultál. Kövesd a lépéseket, készítsd el a saját Mókás maki játékodat, és próbáld eltalálni a denevért a banánnal!

LÁSD MÉG

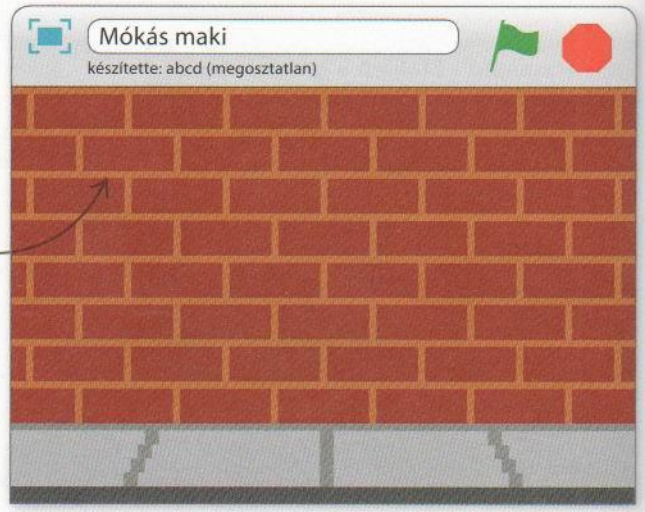
- ❮ **38–39** Mozgásban
- ❮ **40–41** Jelmezek
- ❮ **66–67** Érzékelés

Első lépések

Kezdj egy új projektet! A macska szereplőre most nem lesz szükség. Kattints rá jobb gombbal, és a „törlés" gombbal töröld ki. Így már egy teljesen üres projektben dolgozhatsz.

1 Válassz új hátteret a könyvtárból. Az ehhez szükséges gomb a szereplőlistától balra található.

2 Kattints duplán a „brick wall1" háttérre. Ez a háttér passzol ehhez a játékhoz, de választhatsz másikat is.



TANÁCSOK

A hibák elkerülése

Ez az eddigi legnagyobb Scratch-projekted, ezért elképzelhető, hogy nem fog minden úgy működni elsőre, ahogy szeretnéd. Itt van pár tipp, amely megkönnyítheti a dolgod:

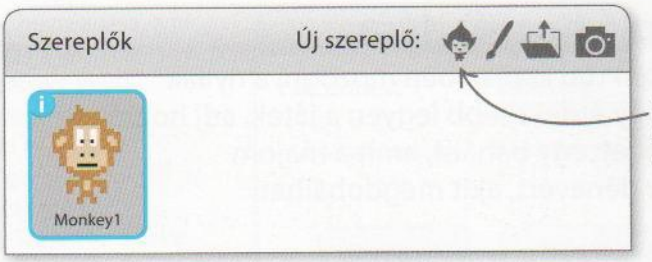
Mindig a megfelelő szereplőhöz add az utasítássorozatokat!

Kövesd az utasításokat pontosan!

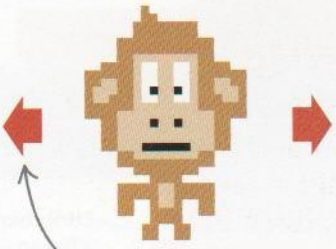
Ne felejtsd el létrehozni a változót, mielőtt használnád!

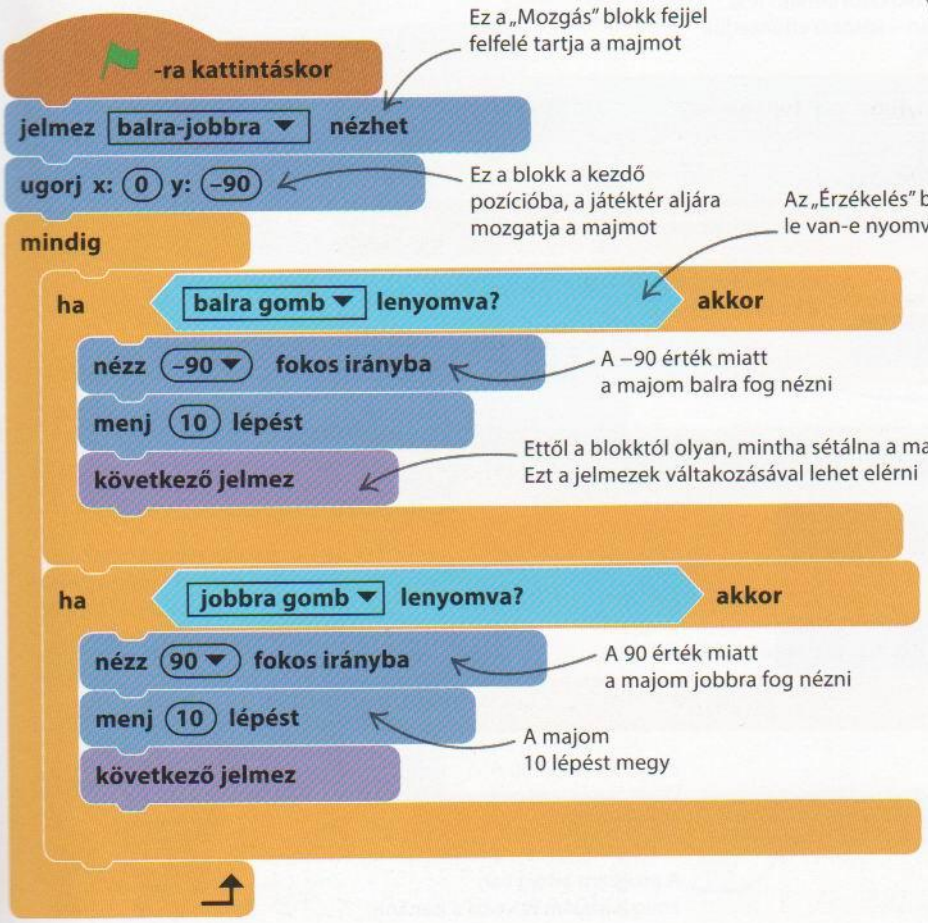
Ellenőrizd, hogy helyes értékeket tartalmaznak-e a blokkok!

3 Válassz egy új szereplőt a szereplők könyvtárból. Kattints az „Állatok" kategóriában a majom (Monkey1) szereplőre. A játékos ezt a szereplőt fogja irányítani.



4 Add a majomhoz a lenti utasítássorozatot. Ne feledd, minden blokkot megtalálsz, színek szerint rendezve. Az „Érzékelés" blokkok lehetővé teszik, hogy a nyílbillentyűkkel mozgasd a majmot a játéktéren. Miután beírtad, futtasd le, hogy lásd, jól működik-e.



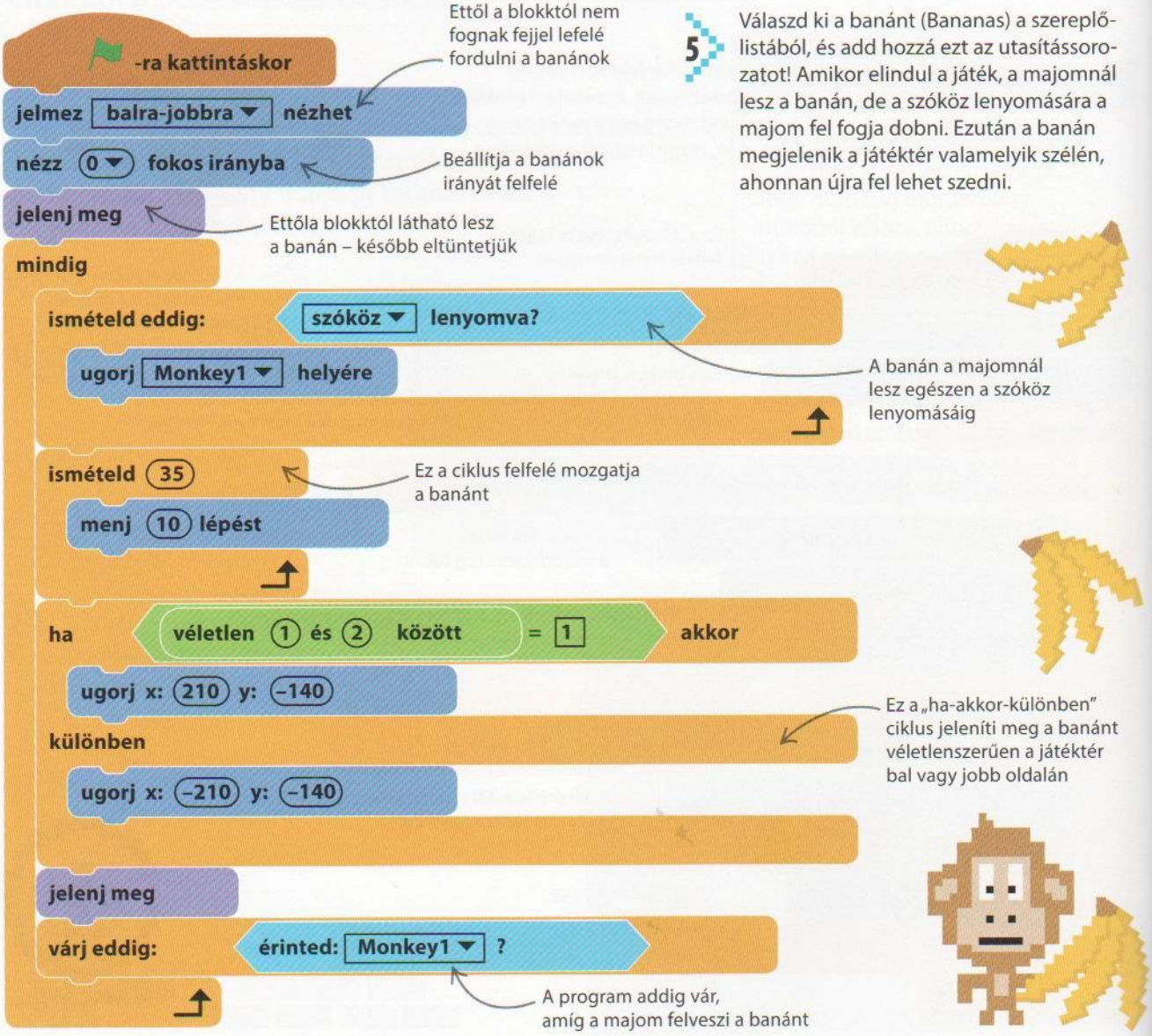


Mentsd el a munkád!

MÓKÁS MAKI

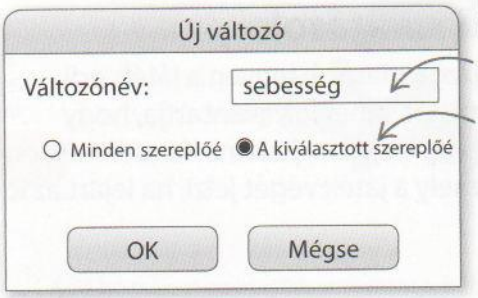
Több szereplő hozzáadása

Most már a majom tud keresztben mozogni a nyilak segítségével. Hogy érdekesebb legyen a játék, adj hozzá további szereplőket: egy banánt, amit a majom dobálhat, és egy denevért, akit megdobálhat.



5 Válaszd ki a banánt (Bananas) a szereplőlistából, és add hozzá ezt az utasítássorozatot! Amikor elindul a játék, a majomnál lesz a banán, de a szóköz lenyomására a majom fel fogja dobni. Ezután a banán megjelenik a játéktér valamelyik szélén, ahonnan újra fel lehet szedni.

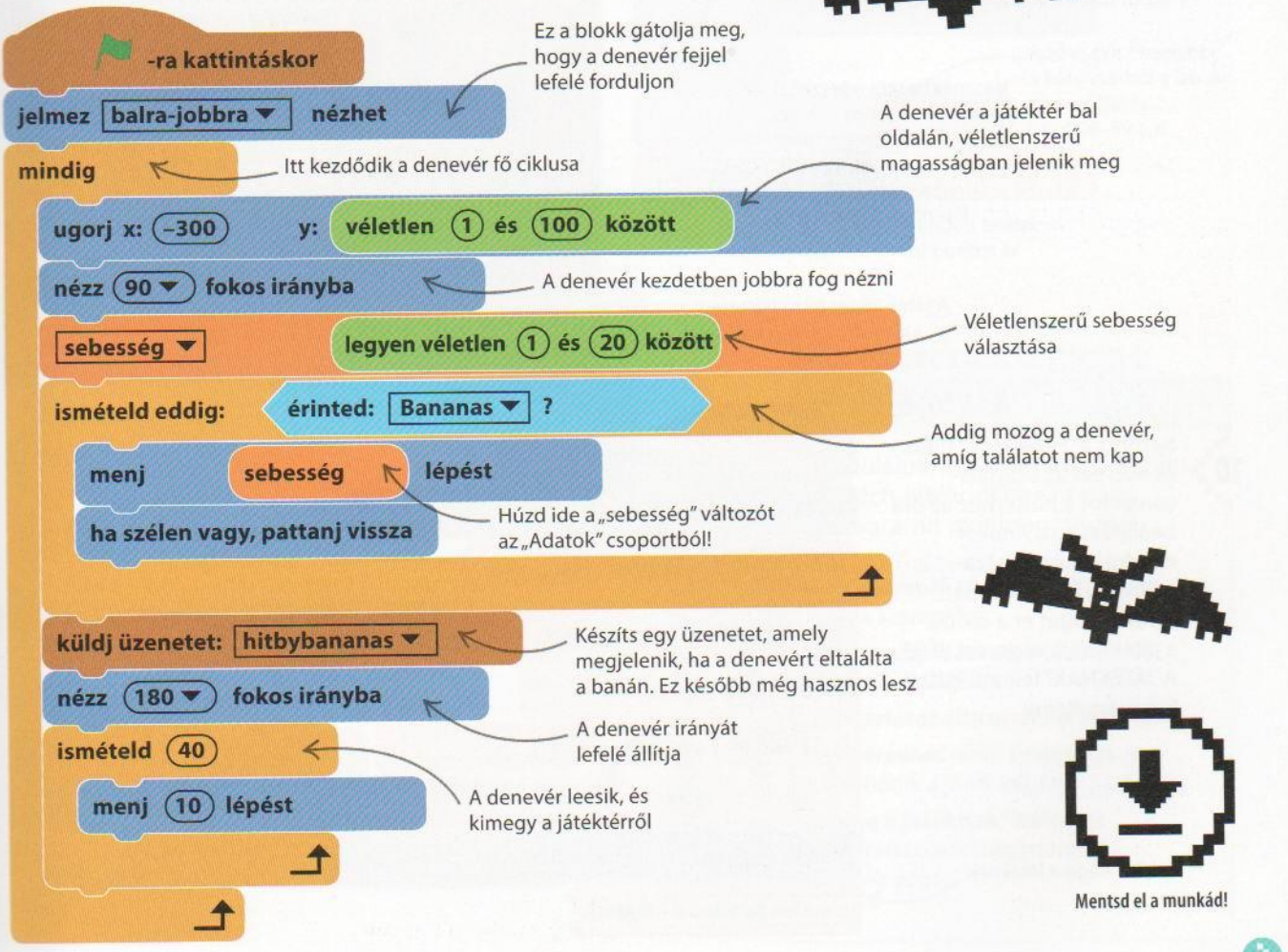
6 A következő lépés a repülő denevér hozzáadása, ami leesik, ha eltalálja a banán. Add hozzá a denevér (Bat2) szereplőt, és készíts egy „sebesség" nevű változót (csak a denevér szereplőé legyen). Új változó létrehozásához válaszd ki az „Adatok" csoportban a „Változó létrehozása" gombot. A „sebesség" blokk előtti négyzetből szedd ki a pipát, hogy ne jelenjen meg a játéktéren az értéke.



Nevezd el az új változót „sebesség"-nek!

Ezt a változót csak a denevér fogja használni

7 Add a denevérhez az alábbi utasítássorozatot! A „mindig" ciklusban a denevér véletlenszerűen elhelyezkedik a játéktér bal oldalán, véletlenszerű sebességgel. Utána elkezd repkedni jobbra-balra a játéktéren. Amikor eltalálja egy banán, leesik a földre.



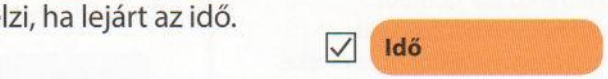
Mentsd el a munkád!

MÓKÁS MAKI

A végső simítások

Hogy még izgalmasabb legyen a játék, adj hozzá számlálót, amely nyilvántartja, hogy hány denevért sikerült eltalálni. Készíts még egy hátteret, amely a játék végét jelzi, ha lejárt az idő.

8 Hozd létre az „Idő" nevű változót. Figyelj, hogy ez a változó minden szereplőnek elérhető legyen, vagyis válaszd a „Minden szereplőé" lehetőséget. Jelöld be mellette a négyzetet, hogy játék közben lásd, mennyi idő van még hátra.

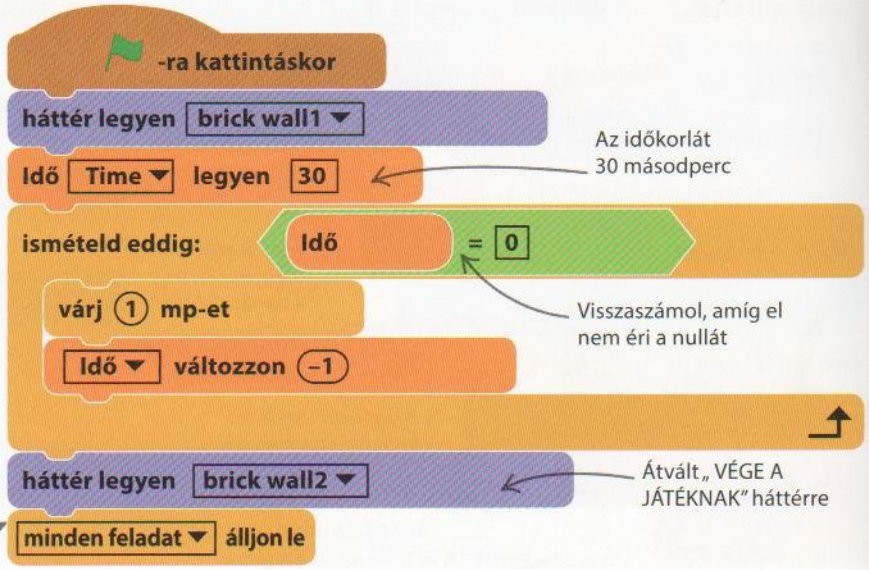


9 Kattints a háttereknél a háttér kis képére, majd a „Hátterek" fülre. Jobb egérgombbal kattints a jelenlegi háttérre, és duplázd. Írd az új háttérre: „VÉGE A JÁTÉKNAK".

Használd a szövegeszközt a háttérre íráshoz – mivel a Scratchben nincs magyar betűkészlet, az ékezeteket külön kell kitenned a betűk fölé

A játék végét jelző háttered valahogy így fog kinézni

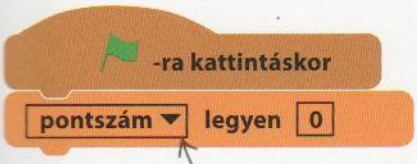
10 Kattints a „Feladatok" fülre, és add ezt az utasítássorozatot a háttérhez az óra beállításához. Amikor elindul, elkezd visszaszámolni egy ciklusban. Amikor véget ér a ciklus, a játéknak is vége, és „VÉGE A JÁTÉKNAK" feliratú háttér fog megjelenni.



11 Kattints a banán szereplőre. Hozz létre egy „Pontszám" nevű változót, és tedd elérhetővé minden szereplő számára. Húzd a pontszámot a játéktér jobb felső sarkába.

Jelöld be a négyzetet, hogy megjelenjen a játéktéren!

12 Add ezt a rövid utasítássorozatot a banán szereplőhöz. A játék elején beállítja a pontszám értékét 0-ra.

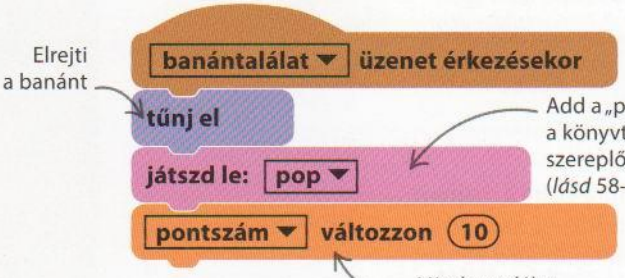


Lenullázza a pontszámot

13 Add hozzá ezt az utasítássorozatot a banán szereplőhöz. Amikor a banán eltalálja a denevért, a program hangot ad ki, megnöveli a pontszámot tízzel, és elrejti a banánt.

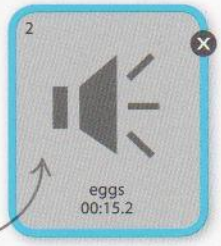
Elrejti a banánt

Add a „pop" hangot a könyvtárból a banán szereplőhöz (*lásd* 58–59 o.)!



Minden találat 10 pontot ér

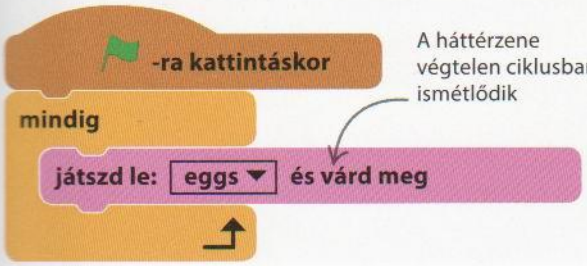
14 Adj zenét a játékhoz. Kattints a háttérre, majd a „Hangok" fülre. Töltsd be az „eggs" zenét a könyvtárból.



Add hozzá az „eggs" zenét!

15 Add a lenti utasítássorozatot a háttérhez. Folyamatosan játszani fogja az „eggs" zenét, de amint a „minden feladat álljon le" blokk lefut, leáll.

A háttérzene végtelen ciklusban ismétlődik



Mentsd el a munkád!

JEGYEZD MEG!

Eredmények

Gratulálok, elkészült az első teljes Scratch-játékod! Íme néhány dolog, amit már megtanultál:

Egy szereplő tárgyakat dobál egy másik szereplőre.

Egy szereplő leesik a játéktérről, amikor eltalálják.

Időkorlátot állítottál be a játékban.

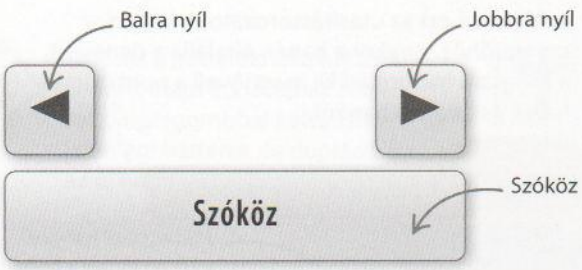
Háttérzenét adtál a játékhoz, ami ismétlődik a játék végéig.

„Vége a játéknak" hátteret készítettél, ami megjelenik a játék végén.

MÓKÁS MAKI

Ideje játszani

A program játékra kész. Kattints a zöld zászlóra! Hányszor találod el a denevért, mielőtt lejár a játékidő?



△ **Irányítás**

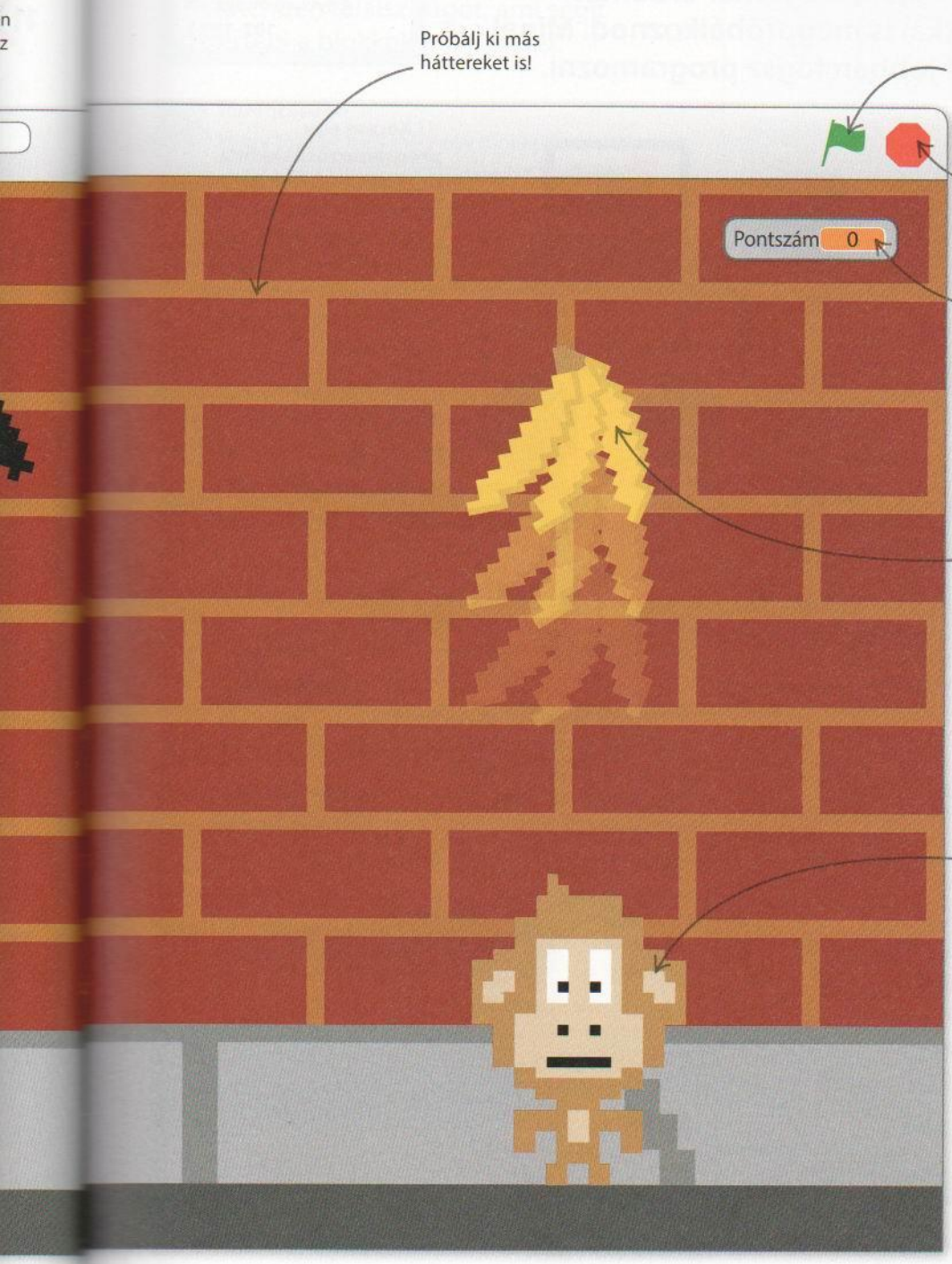
Mozgasd jobbra-balra a majmot a nyilak segítségével. Szóköz lenyomásakor feldobja a banánt a denevér felé.

TANÁCSOK

Több szereplő hozzáadása

További denevérek hozzáadásához kattints a jobb gombbal a denevérre, majd a „duplázás" menüpontra! Próbálj néhány új repülő szereplőt is hozzáadni:

1. **Adj hozzá egy szereplőt** a szereplőlistából! A repülő víziló (Hippo1) remek választás lehet.
2. **Kattints a denevér** szereplőre!
3. **Kattints az utasítássorozatára,** és tartsd lenyomva az egérgombot!
4. **Húzd az utasítássorozatot** az új szereplőre!
5. **Az utasítássorozat megjelenik** az új szereplőnél is.



◁ **Kész őrület!**
Rengeteg módon átalakíthatod a Mókás maki játékot. A sebességek, pontszámok, hangok vagy a szereplők megváltoztatásával saját, egyedi változatot hozhatsz létre.

Kísérletezz!

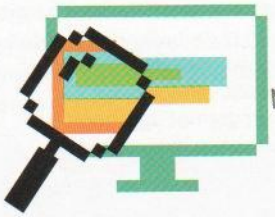
Megtanultad a Scratch alapjait, most már érdemes az összetettebb funkciókkal is megpróbálkoznod. Minél többet gyakorolsz, annál jobban fogsz programozni.

LÁSD MÉG
Mi a Python? **86–87 ❯**
Egyszerű utasítások **102–103 ❯**

Mivel próbálkozz?

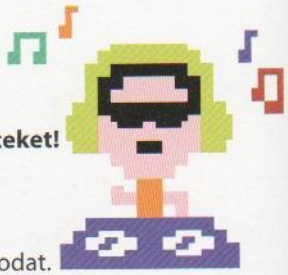
Nem tudod, hogy mit próbálj ki Scratchben? Íme néhány ötlet. Ha úgy érzed, egyedül még nem tudnál egy teljes programot megírni, akkor kezdheted azzal, hogy a meglévőket módosítod.

◁ **Keress egy programozószakkört!**
Nézz körül, van-e a lakóhelyed környékén olyan hely, ahol programozást tanulhatsz? Ilyen helyeken találkozhatsz más Scratch-használókkal, és megbeszélhetitek az ötleteiteket.



A Scratch weboldalán megnézheted az összes megosztott projekt programját.

△ **Nézz programokat!**
Mások programjait nézegetni remek lehetőség a tanulásra. Nézd meg a megosztott Scratch-projekteket! Mit tudsz belőlük megtanulni?

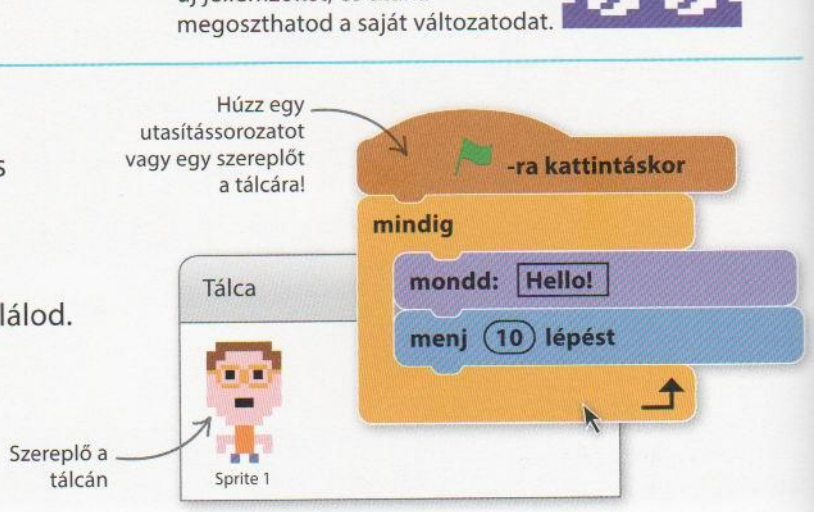


▷ **Remixelj meglévő projekteket!**
A Scratchben a megosztott projektekben hozzáadhatsz új jellemzőket, és utána megoszthatod a saját változatodat.

Tálca

A tálca lehetővé teszi, hogy hasznos utasítássorozatokat, szereplőket, hangokat, jelmezeket tárolj, és egy másik projektben használd őket. A tálcát a Scratch-képernyő alján találod.

▷ **Húzd és dobd!**
Lehúzhatsz szereplőket és utasítássorozatokat a tálcára, aztán később használhatod őket más projektekben.



Segítség!

Nehéz megírni egy programot, ha nem tudsz mindent a blokkok használatáról. A Scratchben találsz súgót, ami segít megérteni a blokkok működését.

1 Parancssúgó
Hogy többet tudj meg egy blokkról, kattints a „Parancssúgó" gombra a képernyő tetején.

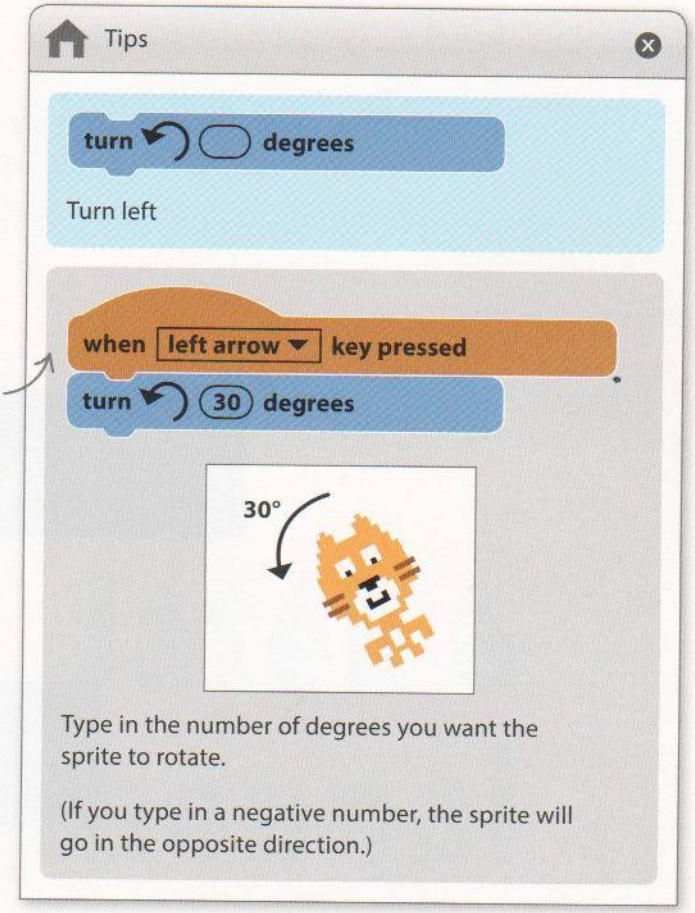
Ez a „Parancssúgó" gomb

A súgóablak bemutatja az összes blokkot

2 Kérdezz!
Az egérmutató kérdőjellé változik. Kattints vele arra a blokkra, amelyről többet szeretnél megtudni.

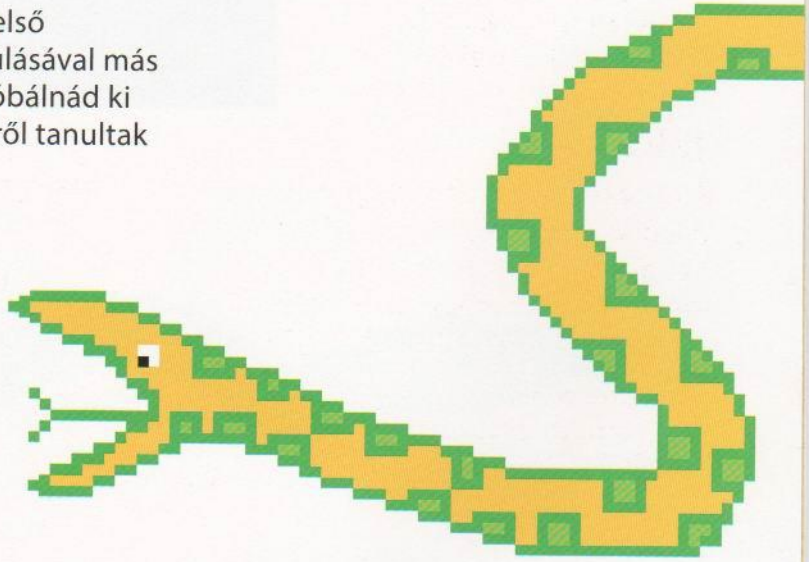
Az egérmutató kérdőjellé változik

3 Súgóablak
A súgóablak megnyitásával mindent megtudhatsz (angol nyelven) az adott blokkról.



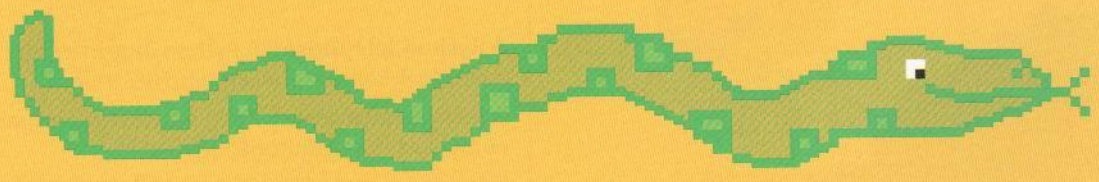
Tanulj másik programnyelvet!

Most már rajtad áll, hogy tökéletesítsd az első programozási nyelvedet. Más nyelvek tanulásával más típusú programokat is írhatsz. Miért ne próbálnád ki következő nyelvként a Pythont? A Scratchről tanultak segíteni fognak az elsajátításában.



▷ **Hasonlít a Scratchre**
A Pythonban is vannak ciklusok, változók és elágazások. Hasznosítsd a Scratch-tudásod a Python tanulása során!

Játék a Pythonnal



Mi a Python?

A Python egy szöveges programozási nyelv. Tovább tart megtanulni, mint a Scratchet, de sokkal több dologra használható.

LÁSD MÉG

A Python telepítése **88–91 ›**

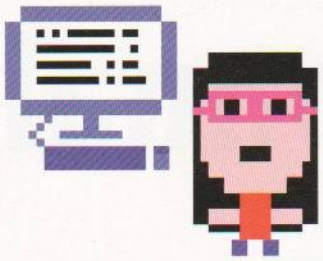
Egyszerű utasítások **102–103 ›**

Bonyolultabb utasítások **104–105 ›**

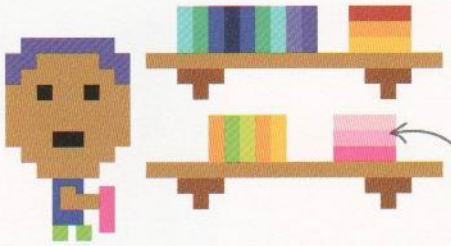
Hasznos nyelv

A Python sokoldalú nyelv. Használható szövegek kezelésére vagy internetes alkalmazások készítésére is. Néhány példa, hogy miért érdemes Pythont tanulni.

1 Könnyű elsajátítani és használni
A Python-programok egyszerű nyelven íródnak. Más programnyelvekhez képest a kódok írása és olvasása meglehetősen egyszerű.

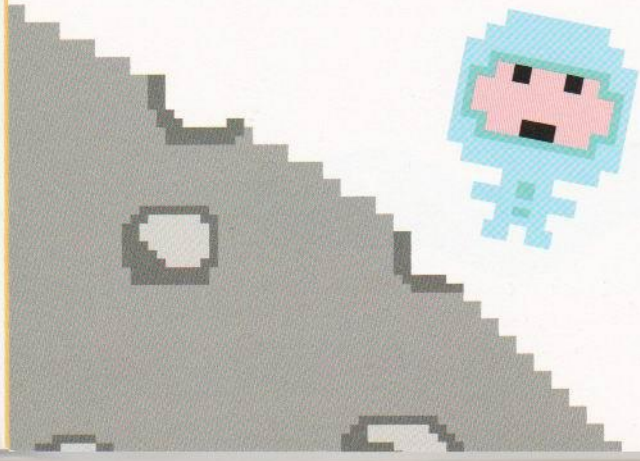


2 Kész kódokat tartalmaz
A Python beépített könyvtárai előre megírt programokat tartalmaznak – ezeket használhatod a saját programjaidhoz is. Megkönnyítik összetettebb programok írását.



A Python sok olyan programot tartalmaz, amelyre építhetsz

3 Nagy cégek is használják
A Python a valós életben is nagyon hatékony. Többek között a Google, a NASA és a Pixar is használja.



TANÁCSOK

Első lépések

Mielőtt elkezdenél Pythonban programozni, érdemes megismerni, hogyan is működik. A következő oldalakon megtanulhatod:

A Python telepítését: A Python ingyenes, de magadnak kell telepítened (*lásd* 88–91. o.).

A felület használatát: Készíts egyszerű programot, és mentsd el a számítógépeden!

Hogyan szerezhetsz tapasztalatot: Próbálj ki több egyszerű programot, hogy lásd, hogyan működnek!

A Scratch és a Python

Sok elem, amely megtalálható a Scratchben, létezik a Pythonban is, csak másképp néz ki. Íme néhány hasonlóság a két nyelv között.

△ **Kiíratás a Scratchben**
Scratchben a „mondd" blokk megjelenít egy szöveget a képernyőn.

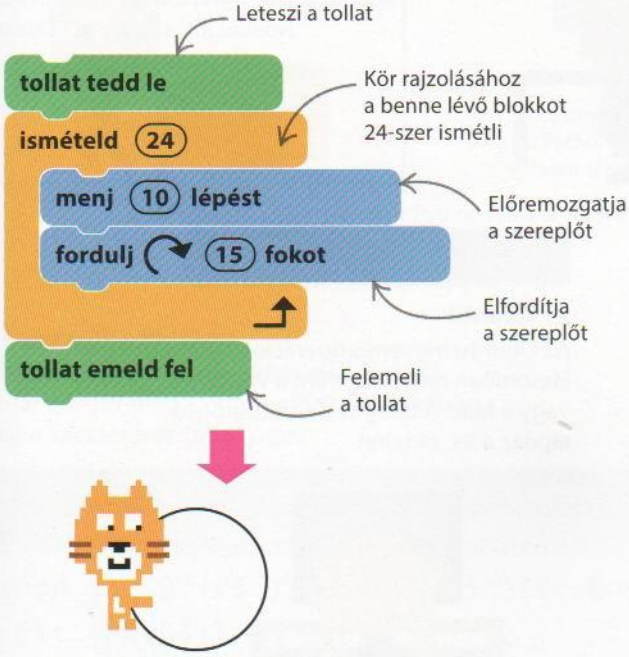
```
print('Helló, világ!')
```

Ez a kiírandó szöveg helye

Az üzenet így fog megjelenni a képernyőn

```
Helló, világ!
```

△ **Kiíratás a Pythonban**
Pythonban a „print" utasítás segítségével írathatsz ki szöveget a képernyőre.



△ **Teknőcgrafika a Scratchben**
A fenti program a „tollat tedd le" blokk segítségével rajzol kört.

```
from turtle import *
pendown()
for n in range(24):
    forward(10)
    right(15)
penup()
```

Ez kezdi a ciklust

Az óramutató járásával megegyezően fordítja a teknőst 15°-kal

△ **Teknőcgrafika a Pythonban**
A Pythonban is van teknőc. A fenti kód használható kör rajzolására.

A Python telepítése

Mielőtt elkezdenél Pythonban programozni, le kell töltened, és telepítened kell a számítógépedre. A Python 3 ingyenes, könnyen telepíthető, és működik Windows, Mac és Linux operációs rendszerek (pl. Ubuntu) alatt is.

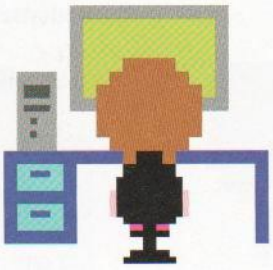
Mi az IDLE?

A Python 3 telepítéséhez egy ingyenes program is tartozik, az IDLE (Integrated DeveLopment Environment, integrált fejlesztési környezet). Ezt kezdő programozók számára tervezték. Egy egyszerű szövegszerkesztőt tartalmaz, amelyben a Python-kódot írhatod és szerkesztheted.

WINDOWS

△ **Windows**

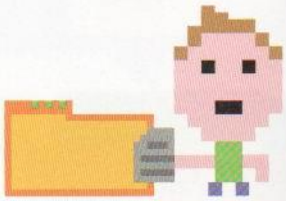
Először is meg kell tudnod, 32 vagy 64 bites-e az operációs rendszered. Kattints a Start gombra, aztán Windows 7 alatt a „Vezérlőpult", majd a „Rendszer", Windows 10 alatt a „Gépház", majd a „Rendszer", végül a „Névjegy" menüpontra. Itt találod a „Rendszer típusa" bejegyzést.



TANÁCSOK

A kód mentése

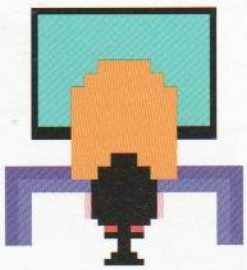
Pythonban a „File > Save As..." menüpont használatával mentheted és nevezheted el a munkáidat. Először hozz létre egy mappát a kódoknak! Adj a mappának beszédes nevet, például „PythonKódok", és mentsd el!



MAC

△ **Mac**

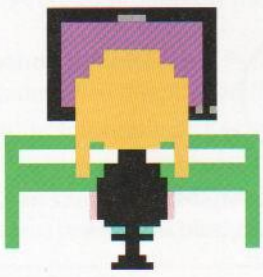
Ha Apple Macen dolgozol, mielőtt letöltöd a Pythont, meg kell nézni, hogy milyen operációs rendszert használsz. Kattints a bal felső alma ikonra, és válaszd „A Mac névjegye" menüpontot!



UBUNTU

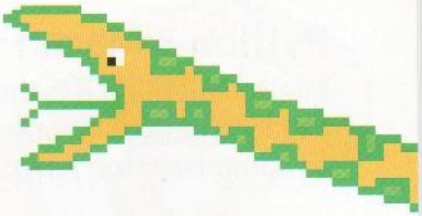
△ **Ubuntu**

Az Ubuntu ingyenes operációs rendszer. Hasonlóan működik, mint a Windows vagy a Mac. A telepítési útmutatóhoz lapozz a 91. oldalra!



Python 3 a Windowson

Mielőtt telepíted a Python 3-at a számítógépre, kérj engedélyt a számítógép tulajdonosától! A telepítéshez szükséged lehet az adminisztrátori jelszóra.



1 Menj a Python weboldalára!
Írd be a böngészőbe az alábbi címet, ez a Python weboldala. Kattints a „Downloads" gombra, hogy a letöltési oldalra juss.

http://www.python.org

Ez a Pythonhoz tartozó URL (internetcím)

2 Töltsd le a Pythont!
Kattints a Windowshoz tartozó legújabb, 3-as számmal kezdődő Python-verzióra. A lista elején lesz.

- Python 3.3.3 Windows x86 MSI Installer
- Python 3.3.3 Windows x86-64 MSI Installer

Ezt válaszd, ha 32 bites Windowst használsz!

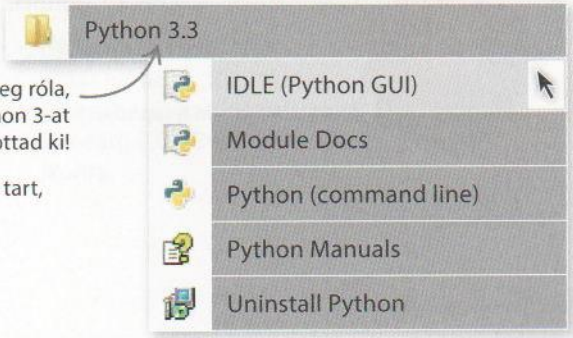
Bármelyik verzió jó, csak az a fontos, hogy 3-assal kezdődjön

Ezt válaszd, ha 64 bites Windowst használsz!

3 Telepítsd!
A telepítő automatikusan letöltődik. Ha lejött, kattints rá kétszer a telepítéshez. Válaszd az „install for all users" lehetőséget, utána kattints a „next" gombra minden alkalommal, anélkül hogy változtatnál az alapbeállításokon.

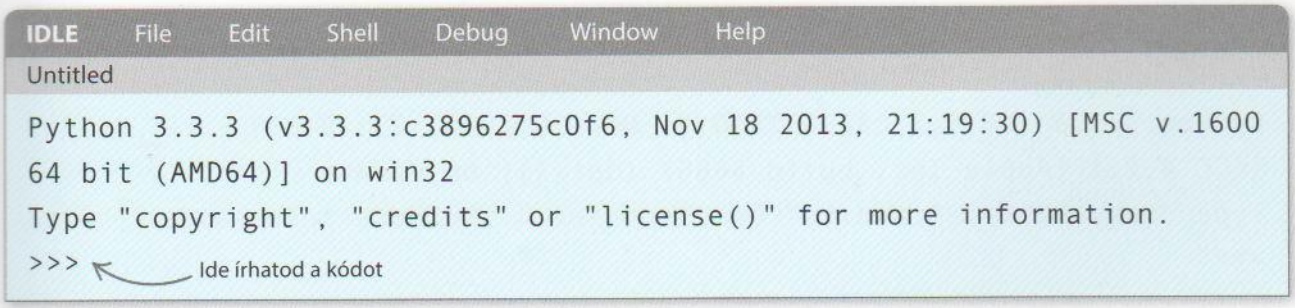
Amíg a Python telepítése tart, a Windows-telepítőikon lesz látható

4 Az IDLE futtatása
Most ellenőrizd, hogy jól sikerült-e a telepítés. Nyisd meg a „Start" menüt, kattints a „Minden program" gombra, azon belül keresd meg a Pythont, és válaszd az „IDLE" lehetőséget.



Bizonyosodj meg róla, hogy a Python 3-at választottad ki!

5 Megnyílik a Python
Az alábbihoz hasonló ablak jelenik meg. Most már elkezdheted a programozást – csak írj az ablakba a kacsacsőrök (>>>) után.



Python 3 Macen

Mielőtt telepíted a Python 3-at a számítógépre, kérj engedélyt a számítógép tulajdonosától! A telepítéshez szükséged lehet az adminisztrátori jelszóra.

1 Menj a Python-linkre!
Írd be a böngészőbe az alábbi címet, ez a Python weboldala. Kattints a „Downloads" gombra, hogy a letöltési oldalra juss.

Q http://www.python.org

2 A Python letöltése
Nézd meg, milyen operációs rendszert használsz (*lásd* 88. o.), és kattints a megfelelő verziójú Python 3-ra. Kéri majd a számítógép, hogy tölts le egy .dmg fájlt. Mentsd az asztalra.

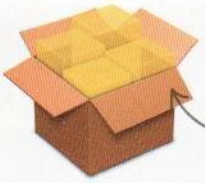
Az újabb Mac-verziókhoz jó

- Python 3.3.3 Mac OS X 64-bit... (for Mac OS X 10.6 and later)
- Python 3.3.3 Mac OS X 32-bit... (for Mac OS X 10.5 and later)

Bármelyik verzió jó, csak az a fontos, hogy 3-assal kezdődjön

A legtöbb Mac-verzióhoz jó

3 Telepítsd!
Kattints kétszer a .dmg fájlra! Megjelenik egy ablak, amelyben sok különböző fájl található. Ezek között lesz a telepítő „Python.mpkg" fájl is. Kattints rá kétszer a telepítéshez.



A Python telepítője

Python.mpkg

4 Az IDLE futtatása
A telepítés folyamán minden lehetőségnél kattints a „next" gombra az alapbeállítások elfogadásához. Ha a telepítés befejeződött, nyisd meg az „Applications" mappát, azon belül pedig a „Python" mappát (figyelj, hogy biztosan a Python 3-at válaszd ki, ne a Python 2-t). A telepítés ellenőrzéséhez kattints duplán az „IDLE" ikonra.

Az IDLE ikonja

5 Megnyílik a Python
Az alábbihoz hasonló ablak jelenik meg. Most már elkezdheted a programozást – csak írj az ablakba a kacsacsőrök után.

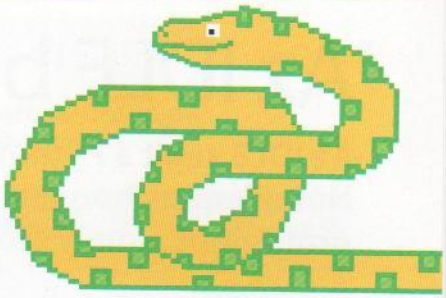
IDLE File Edit Shell Debug Window Help

Untitled

```
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 16 2013, 23:39:35)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
```

Python 3 Ubuntun

Ha Linux Ubuntu operációs rendszert használsz, akkor telepítheted a Python 3-at böngésző használata nélkül – csak kövesd az alábbi lépéseket! Ha más típusú Linuxot használsz, kérd meg a számítógép kezelőjét, hogy segítsen a telepítésben!

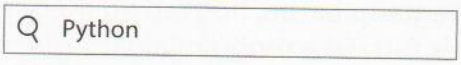


1 Nyisd meg az Ubuntu szoftverközpontot!

Keresd meg az Ubuntu szoftverközpontot az Indítóban vagy a Keresőben, és kattints kétszer az ikonjára.

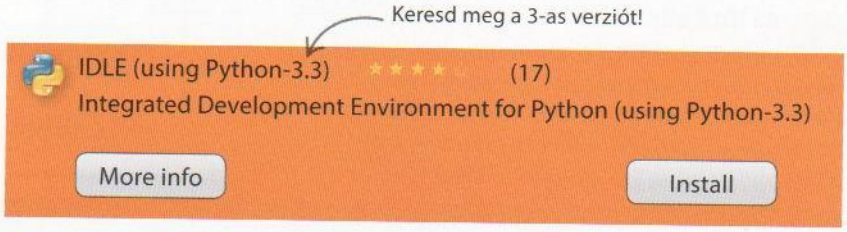
2 Írd be a keresőbe: „Python"!

Látni fogsz egy keresőablakot a jobb felső sarokban. Ide írd be a „Python" szót, majd nyomd meg az „Enter" billentyűt.



3 Válaszd az „IDLE"-t, és kattints az „Install" gombra!

Keresd meg az „IDLE (using Python)" lehetőséget. Jelöld ki a 3-assal kezdődő verziót, és kattints a „Install" gombra.



4 Menj a Keresőbe!

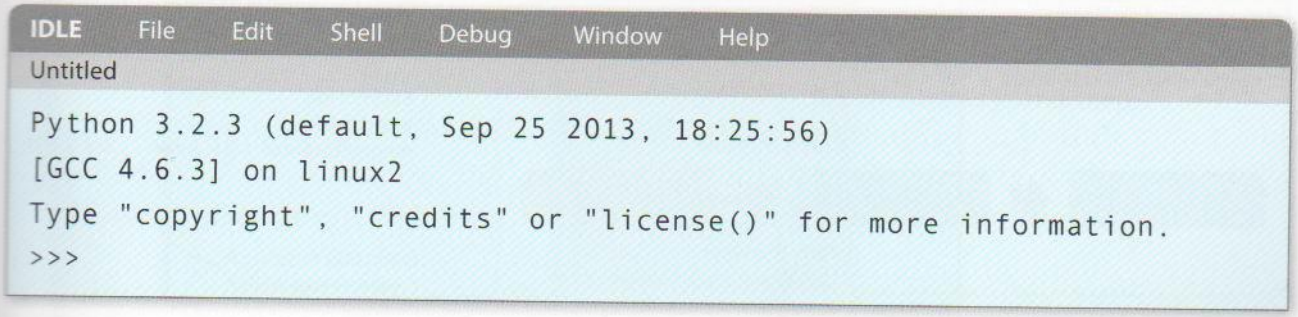
Miután a telepítés befejeződött, ellenőrizd a program működését. Először válaszd a legfelső, Kereső ikont az Indítón.

5 Az IDLE futtatása

Írd be az „IDLE" szót a keresőbe, és kattints kétszer a kék-sárga „IDLE (using Python 3)" ikonra.

6 Megnyílik a Python

Az alábbihoz hasonló ablak jelenik meg. Most már elkezdheted a programozást – csak írj az ablakba a kacsacsőrök után.



Az IDLE bemutatása

Az IDLE segít a Python-kódok írásában és futtatásában. Nézzük meg, hogyan működik! Írj egy egyszerű programot, amely szöveget jelenít meg a képernyőn.

LÁSD MÉG
‹ 88–91
A Python telepítése
Melyik ablak?
106–107 ›

Munka az IDLE-ben

Kövesd az alábbi pontokat, hogy az IDLE használatával Python-programot hozz létre! Így megtanulhatod, hogyan írd meg, mentsd el és futtasd a programjaidat.

1 Az IDLE elindítása
Indítsd el az IDLE-t, a számítógéped operációs rendszerének megfelelően (*lásd* 88–91. o.). Megnyitáskor a terminálablak fog megjelenni. Itt látható a program kimenete (minden, amit a program ír ki) és az összes hibaüzenet, ha van.

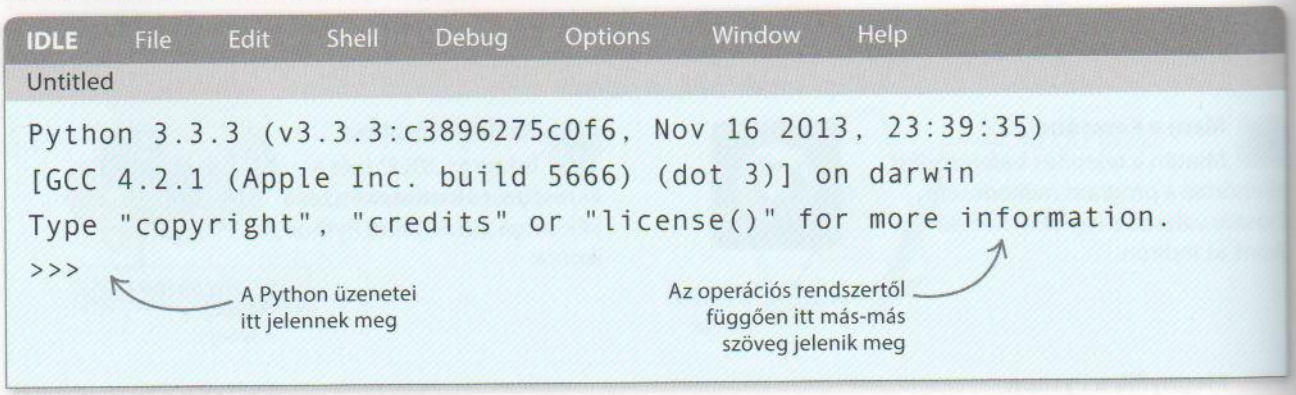
TANÁCSOK

Különböző ablakok

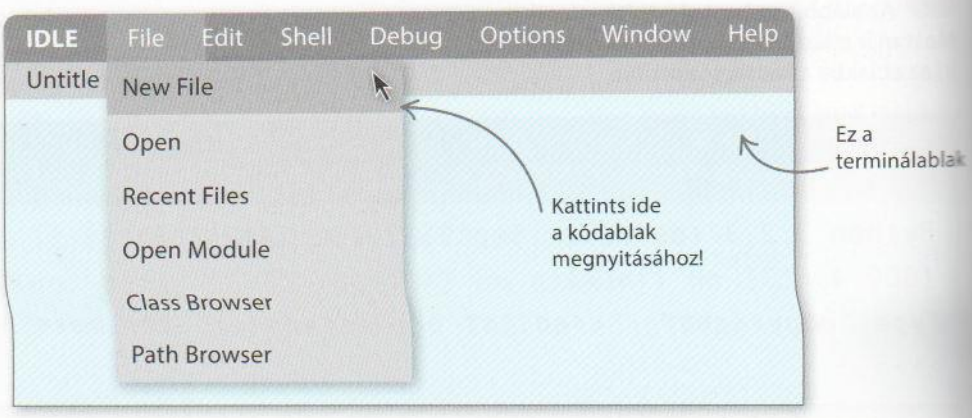
A Python két különböző ablakot használ – a terminálablakot (shell) és a kódablakot (code) (*lásd* 106–107. o.). Eltérő színnel jelezzük, hogy meg lehessen különböztetni őket.

Terminálablak

Kódablak



2 Nyiss új ablakot!
Lépj be a „File" menübe a terminálablak felső részén, és válaszd a „New File" menüpontot. Ez nyitja meg a kódablakot.

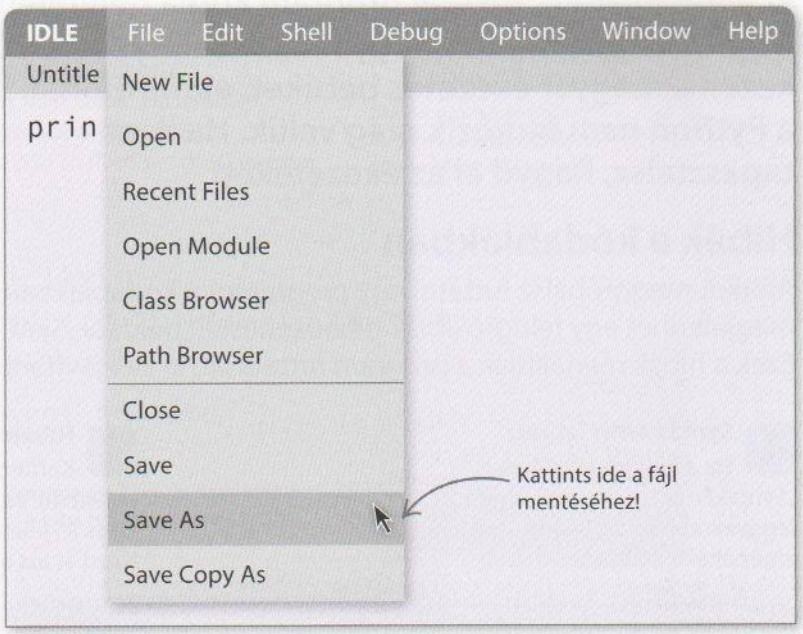


3 Írd be a kódot!
Az új kódablakba írd be ezt a szöveget. Ez az utasítás kiíratja a „Helló, világ!" szavakat.

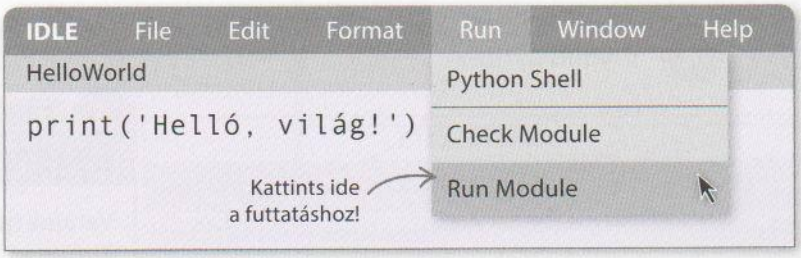
```
print('Helló, világ!')
```

Használj félidézőjelet (aposztrófot)!

4 Mentsd el a kódablak tartalmát!
Kattints a „File" menüre, és válaszd a „Save As..." menüpontot. Írd be fájlnévnek, hogy „HelloWorld", és kattints a „Mentés" gombra.



5 Futtasd a programot!
A kódablakban kattints a „Run" menüre, majd a „Run Module" menüpontra. Ez futtatja a programot a terminálablakban.



6 Kimenet a terminálablakban
Nézz a terminálablakra! A „Helló, világ!" üzenet jelenik meg, amikor a program lefut. Elkészítetted az első Python-programodat.

```
>>>
Helló, világ!
>>>
```

Az üzenet idézőjelek nélkül jelenik meg

JEGYEZD MEG!

Az IDLE működése

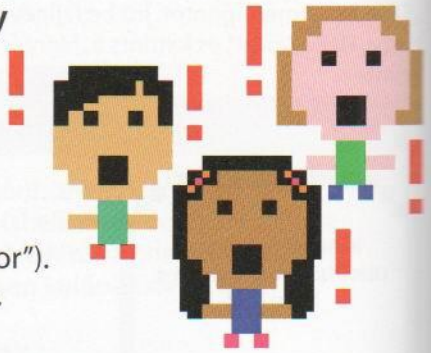
Mindig kövesd ezt a három lépést: írd be a kódot, mentsd el és futtasd! Ha nem mented el a programot, nem fog lefutni. Ha mégis megpróbálod, figyelmeztetés jelenik meg.

Írd be! ➡ **Mentsd el!** ➡ **Futtasd!**

Hibák

LÁSD MÉG
Hibák és hibakeresés 148–149 ❯
Hogyan tovább? 176–177 ❯

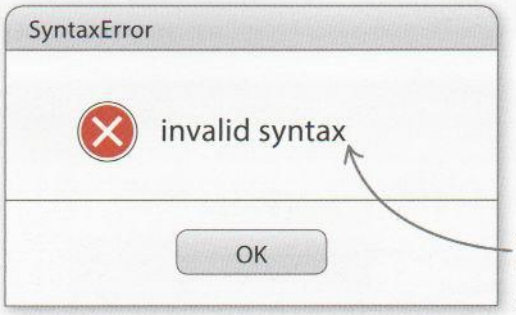
Előfordul, hogy a program nem működik elsőre, de mindig ki lehet javítani. Ha egy program kódja helytelenül van beírva, a Python jelzi, hogy mi a hiba. Habár az IDLE jól kezeli a magyar ékezetes betűket, előfordulhat, hogy a Python nem birkózik meg velük. Ha ilyet tapasztalsz, hagyd el az ékezeteket!

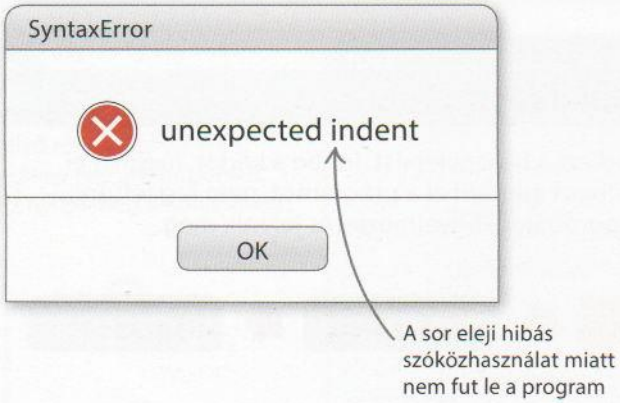


Hibák a kódablakban

Amikor megpróbálsz futtatni egy programot a kódablakban, megjelenhet egy felugró ablak hibaüzenettel (például „SyntaxError"). Ezek a hibák megállítják a program futtatását, ki kell javítani őket.

1 Syntax error (elírás)
Ha a felugró ablakban a „SyntaxError" üzenet olvasható, az leggyakrabban azt jelenti, hogy valamit elgépeltél a kódban.





2 Hibajelzés
Kattints az „OK" gombra a felugró ablakban, és visszatérsz a programhoz. Piros jel jelenik meg a hiba mellett. Nézd át azt a sort figyelmesen!

```
print('Helló, világ!)
```

Hiányzik az idézőjel

A hiba jele

TANÁCSOK

Gyakori hibák

Vannak hibák, amelyeket könnyű elkövetni. Kerüld el az alábbi gyakori hibákat:

Kis- és nagybetűk: Figyelj a betűállásra! Ha „print" helyett „Print"-et írsz, akkor a Python nem fogja érteni az utasítást.

Félidézőjelek (') és idézőjelek ("): Ne keverd a különféle idézőjeleket! A nyitó és záró idézőjelnek egyformának kell lennie.

Kötőjel és aláhúzásjel: Ne keverd össze a kötőjelet (-) az aláhúzásjellel (_)!

Különböző zárójelek: A különböző zárójeleket – mint a (), {}, [] – más és más utasításoknál kell használni. Mindig a megfelelőt válaszd, és ne felejtsd el bezárni őket!

Hibák a terminálablakban

Előfordulhat, hogy a terminálablakban jelenik meg piros hibaüzenet. Ez is megállítja a program futását.

1 Névhiba

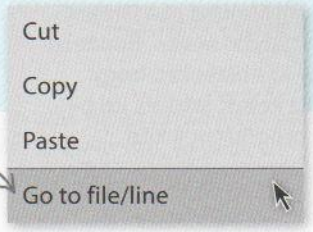
Ha a „NameError" hibaüzenet jelenik meg, az azt jelenti, hogy a Python nem ismer fel egy szót, amelyet használtál. Ha a hiba a kódablakban beírt kódban van, akkor jobb gombbal kattints a hibaüzenetre a terminálablakban, és válaszd a „Go to file/line" menüpontot.

Annak a programsornak a száma (a kódablakban), amelyikben a hiba van

```
>>>
Traceback (most recent call last):
    File "C:\PythonCode\errors.py", line 1, in <module>
        pront('Helló, világ!')
NameError: name 'pront' is not defined
```

A szó, amelyet a Python nem ért

Kattints ide, hogy megjelenítsd a hiba helyét a kódablakban!



2 Javítsd ki a hibát!

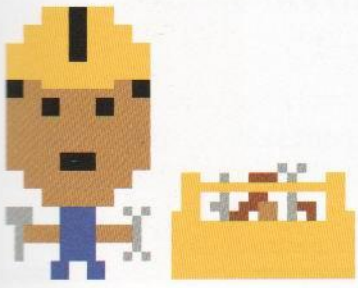
A kódablakban a hibás sort hibaüzenet jelzi. A „pront" szó lett beírva a „print" helyett. Javítsd a kódot.

```
pront('Helló, világ!')
```

Javítsd „print"-re!

A hibák felderítése

Használd a tippeket erről a két oldalról, hogy megtaláld a hibát, és nézd át újra a hibás sort! Menj végig a jobb oldali listán! Segít a hibakeresésben.



▷ **Ha rosszul mennek a dolgok**

Vannak módszerek, amelyekkel könnyen megtalálhatod a hibákat. Íme egy ellenőrzőlista.

HIBAJAVÍTÁS	
Nézd át a kódodat a következők alapján!	✓
Pontosan lemásoltad, amit be kellett írnod?	✓
Nem gépeltél el valamit?	✓
Két idézőjel (') közé írtad, amit ki akarsz írni a programmal?	✓
Nem tettél felesleges szóközöket a sor elejére? Ezek nagyon fontosak a Pythonban.	✓
Megnézted a megjelölt előtti és utáni programsort is? Előfordulhat, hogy azokban van a hiba.	✓
Megkértél valakit, hogy hasonlítsa össze a kódodat a könyvben találhatóval? Talán más észreveheti, ami neked elkerülte a figyelmedet.	✓
Python 3-at vagy Python 2-t használsz? A Python 3-ra írt programok nem biztos, hogy működnek Python 2-n.	✓

4. PROJEKT

Kísértetház

Ez az egyszerű játék rávilágít, mire kell figyelni, ha Pythonban programozol. Ha beírtad a kódot, futtasd a programot, és már játszhatsz is. Sikerül kijutnod a kísértetházból?

LÁSD MÉG

A Kísértetház elemzése **98–99 ›**

A program folyamata **100–101 ›**

1 Indítsd el az IDLE-t, és a „File" menüben nyiss egy új ablakot! Mentsd el a játékot „Kísértetház" néven. Rendezd el úgy az ablakokat, hogy mind a kettőt lásd, és írd be ezt a programot a kód ablakba.

```
# Kísértetház
from random import randint
print('Kísértetház')
bátor_vagyok = True
pontszám = 0
while bátor_vagyok:
    szellem_ajtó = randint(1, 3)
    print('Három ajtó van előtted…')
    print('Az egyik mögött szellem van.')
    print('Melyiket nyitod ki?')
    ajtó = input('1, 2 vagy 3?')
    ajtó_szám = int(ajtó)
    if ajtó_szám == szellem_ajtó:
        print('SZELLEM!')
        bátor_vagyok = False
    else:
        print('Nincs szellem!')
        print('Lépj be a következő szobába!')
        pontszám = pontszám + 1
print('Menekülj!')
print('Vége a játéknak! Az elért pontszám:', pontszám)
```

Használj félidézőjeleket!

Csak ott használj nagybetűt, ahol a példában is látod!

Ne feledkezz meg a kettőspontról!

Ez aláhúzásjel legyen, ne kötőjel!

Ezt a részt négy szóközzel beljebb kell kezdeni. Ha ez nem történik meg automatikusan, ellenőrizd, hogy a „bátor_vagyok" után írtál-e kettőspontot!

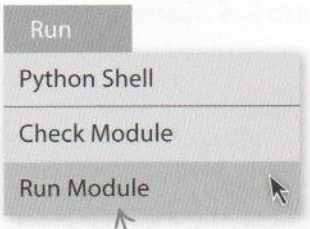
Két egyenlőségjelet használj!

Ez nyolc szóköznyi behúzással fog megjelenni, de négyre kell csökkenteni

Töröld az összes behúzást!

Itt nem kell a „pontszám" köré idézőjelet rakni

2 Ha pontosan beírtad a kódot, futtasd a programot a „Run" menüben a „Run Module" menüpontot választva. De előbb még el kell mentened a programot.



Válaszd a „Run" menüből a „Run Module" menüpontot a kódablakban!

3 A játék a terminálablakban indul. A szellem a három ajtó egyike mögött rejtőzik. Melyiket választod? Írd be az 1-es, 2-es vagy 3-as számot, majd nyomd meg az „Enter" billentyűt.

```
Kísértetház
Három ajtó van előtted…
Az egyik mögött szellem van.
Melyiket nyitod ki?
1, 2 vagy 3?
```

Írd be a tipped!

4 A játék célja, hogy olyan ajtót válassz, amelyik mögött nincs szellem. Ha sikerül, akkor továbbmehetsz a következő ajtóhoz.

```
Kísértetház
Három ajtó van előtted…
Az egyik mögött szellem van.
Melyiket nyitod ki?
1, 2 vagy 3?3
Nincs szellem!
```

Itt jelenik meg a beírt szám

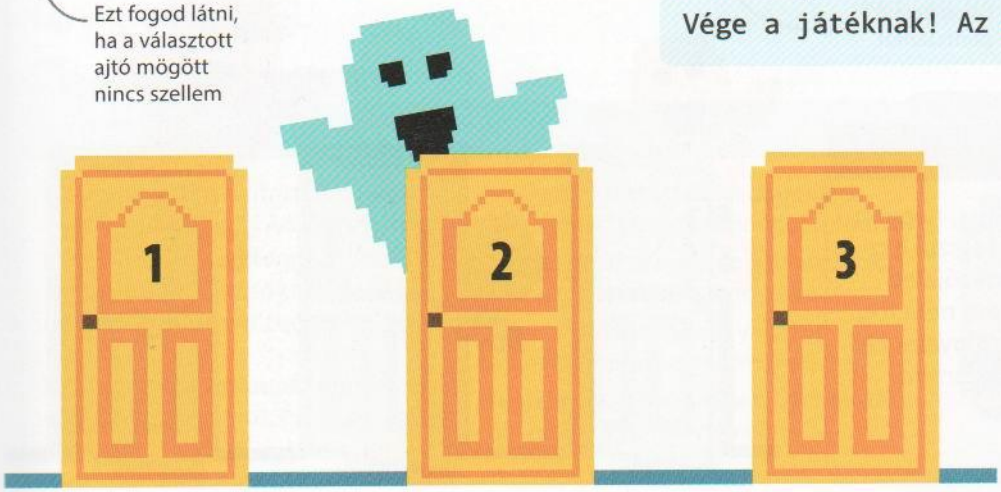
Ezt fogod látni, ha a választott ajtó mögött nincs szellem

5 Ha nincs szerencséd, akkor olyan ajtót választottál, amelyik mögött szellem van, és vége a játéknak. Futtasd újra a programot, és gyűjtsd a pontokat!

```
Kísértetház
Három ajtó van előtted…
Az egyik mögött szellem van.
Melyiket nyitod ki?
1, 2 vagy 3?2
SZELLEM!
Meneküljl!
Vége a játéknak! Az elért pontszám: 0
```

Ez jelenik meg, ha a szellem az ajtó mögött van

A pontszám azt mutatja, hogy hány szobát éltél túl



A Kísértetház elemzése

Ebben a játékban a Python több kulcsfontosságú elemét megismerheted. Elemeire bonthatod a kódot, hogy lásd, miként épül fel a program, és a különböző részeknek mi a feladatuk.

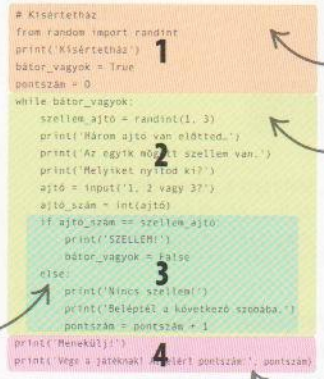
LÁSD MÉG

〈 **96–97** Kísértetház

A program folyamata **100–101** 〉

A kód szerkezete

A Python a sorok elején szóközökkel jelöli, hogy mely utasítások tartoznak össze. Ezeket behúzásnak nevezzük. Például a „while bátor_vagyok:" után következő sorok négy szóközzel beljebb kezdődnek. Ez mutatja, hogy a cikluson belül vannak.



◁ **Kódkulcs**
Ez az ábra mutatja a Kísértetház játék felépítését. A számozott részeket külön-külön megvizsgáljuk az alábbiakban.

Ez egy megjegyzés. Nem fog látszani, amikor lefut a program

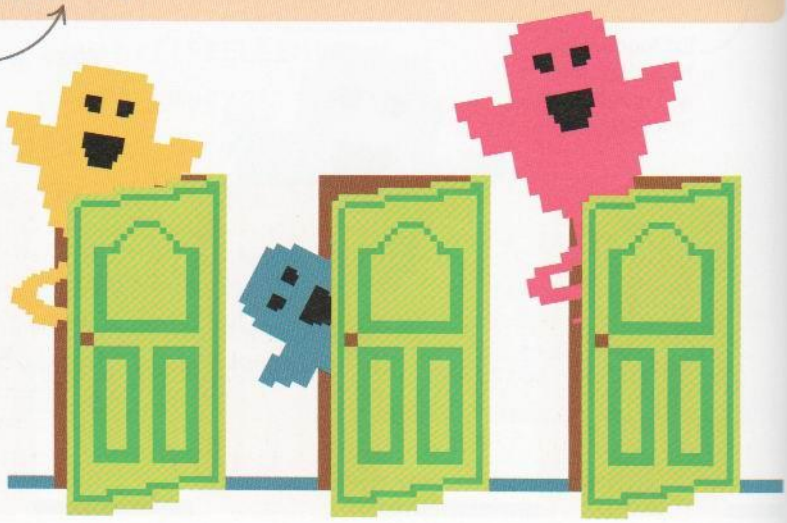
1 A kezdeti beállítások
Ezek az utasítások csak egyszer futnak le – a játék elején. Beállítják a címet, a változókat és a „randint" utasítást.

```
# Kísértetház
from random import randint
print('Kísértetház')
bátor_vagyok = True
pontszám = 0
```

Betölti a „randint" utasítást, amely véletlen számokat generál

A „print" utasítás szöveget jelenít meg a program futása során

Lenullázza a pontszámot



TANÁCSOK

Figyelmesen gépelj!

Amikor Pythont használsz, nagyon figyelj a gépelésre! Ha rosszul írsz be egy kettőspontot, idézőjelet vagy zárójelet, a program nem fog megfelelően működni. Figyelni kell a kis- és nagybetűkre, valamint a szóközökre is.

2 A fő ciklus

Ez a ciklus mondja el a történetet, és fogadja a játékos tippjeit. Egészen addig fut, amíg a választott ajtó mögött nincs szellem. Amikor szellemet találunk, a „bátor_vagyok" változó értéke hamisra (False) változik, és a ciklus véget ér.

3 Elágazás

A program különbözőképpen futhat tovább attól függően, hogy van-e szellem a választott ajtó mögött. Ha volt szellem, akkor a „bátor_vagyok" változó értéke hamisra változik. Ha nem volt, akkor a játékos pontszáma nő eggyel.

```
while bátor_vagyok:
    szellem_ajtó = randint(1, 3)
    print('Három ajtó van előtted...')
    print('Az egyik mögött szellem van.')
    print('Melyiket nyitod ki?')
    ajtó = input('1, 2 vagy 3?')
    ajtó_szám = int(ajtó)
    if ajtó_szám == szellem_ajtó:
        print('SZELLEM!')
        bátor_vagyok = False
    else:
        print('Nincs szellem!')
        print('Lépj be a következő szobába!')
        pontszám = pontszám + 1
```

Ez választ egy véletlen számot 1 és 3 között

A „print" utasítás szöveget jelenít meg a képernyőn

Ez a sor vár választ a játékostól

Ez az ág fut le, ha a szellem a választott ajtó mögött van

Ha nincs szellem, akkor ez az üzenet jelenik meg

A pontszám 1-gyel nő minden alkalommal, ha nincs az ajtó mögött szellem

4 A játék befejezése

Ez a rész csak egyszer fut le: miután szellemmel találkozol, és véget ér a ciklus. A Python onnan tudja, hogy ez már nem része a ciklusnak, hogy nincs előtte behúzás.

```
print('Meneküljl!')
print('Vége a játéknak! Az elért pontszám:', pontszám)
```

Megjeleníti az üzenetet, hogy menekülni kell a szellem elől

Ez egy változó – azt mutatja, hogy hány szobán jutottál át a játék során

JEGYEZD MEG!

Eredmények

Gratulálok! Elkészítetted az első Python-játékodat! A könyvben később még sokat fogsz tanulni ezekről az utasításokról, de már így is sok mindent tudsz:

Beírtál egy programot: Begépeltél egy teljes programot, és elmentetted.

Programot futtattál: Megtanultad, hogyan kell Python-programot futtatni.

Strukturáltál programot: Behúzásokat használtál a program tagolására.

Használtál változókat: Változót használtál a pontszám elmentésére.

Megjelenítettél szöveget: Üzeneteket jelenítettél meg a képernyőn.

A program folyamata

Mielőtt továbbmennénk, fontos megérteni, hogyan működnek Pythonban a programok. Amit a programozás alapjairól megtanultál a Scratch kapcsán, alkalmazható a Pythonra is.

LÁSD MÉG

❮ 30–31 Színes blokkok és utasítások

Egyszerű utasítások 102–103 ❯

Bonyolultabb utasítások 104–105 ❯

Bemenettől a kimenetig

A program bemenő adatokat kap, feldolgozza (vagy módosítja) őket, majd visszaadja a kimenő eredményeket. Ez kicsit olyan, mint amikor a cukrász megkapja a hozzávalókat, sütit készít belőlük, majd odaadja neked.

Bemenet

- Beviteli utasítás
- Billentyűzet
- Egér

Feldolgozás

- Változók
- Matek
- Ciklusok
- Elágazások
- Függvények

Kimenet

- Kiíró utasítás
- Képernyő
- Grafika

△ **A program folyamata a Pythonban**
A Pythonban a billentyűzettel és az egérrel adjuk meg a bemeneti információt a programnak, amely ciklusok, elágazások és változók segítségével dolgozza fel őket. Az eredmények a képernyőn jelennek meg.

Kísértetház – Scratch-szemmel

A program futása a legtöbb programozási nyelven egyforma. Itt láthatsz néhány példát a Kísértetház játékból a bemenetre, feldolgozásra és a kimenetre – és arra, ahogy ezek Scratchben kinéznének.

TANÁCSOK

Egyszerre egy utasítássorozat

Van egy fontos különbség a Scratch és a Python között. A Scratchben sok utasítássorozat futhat egy időben, a Pythonban azonban egyszerre csak egy.

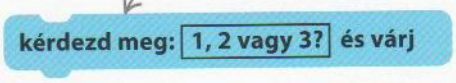
1 Bemenet

A Pythonban az „input()" függvénnyel vihetünk be adatot a billentyűzetről. Ez megegyezik a „kérdezd meg: … és várj" Scratch-blokkal.

```
ajtó = input('1, 2 vagy 3?')
```

A kérdés megjelenik a képernyőn

A kérdés a Scratch-blokkban



„kérdezd meg: … és várj" Scratch-blokk

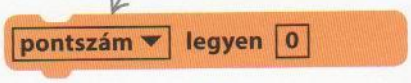
2 Feldolgozás

Változók segítségével tárolhatjuk a pontszámot, a „randint" függvény választ véletlenszerűen ajtót. Ezeknek megfelelő blokkok találhatók a Scratchben is.

```
pontszám = 0
```

Beállítja a „pontszám" változó értékét 0-ra

Ez a Scratch-blokk állítja be a „pontszám" változó értékét 0-ra



„pontszám legyen 0" Scratch-blokk

```
szellem_ajtó = randint(1, 3)
```

Választ egy véletlen egész számot 1 és 3 között

„véletlen" Scratch-blokk

Ez a Scratch-blokk választ véletlen számot

3 Kimenet

A „print()" függvény kezeli a kimenetet a Pythonban, ahogy a „mondd" blokk teszi a Scratchben.

```
print('Kísértetház')
```

Megjelenik a „Kísértetház" felirat a képernyőn

Egy szövegbuborékban megjelenik a „Kísértetház" felirat

„mondd" Scratch-blokk

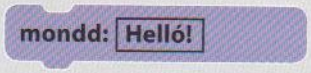
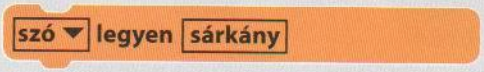
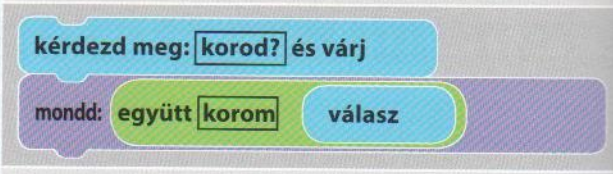
Egyszerű utasítások

Első pillantásra a Python kicsit bonyolultnak tűnhet, különösen a Scratchhez képest. Nem különbözik ugyanakkor annyira a két nyelv, mint elsőre gondolnád. Megmutatjuk a hasonlóságokat a két nyelv között.

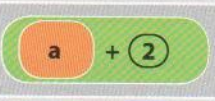
LÁSD MÉG

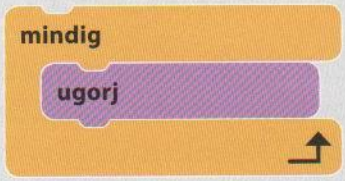
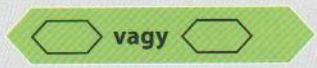
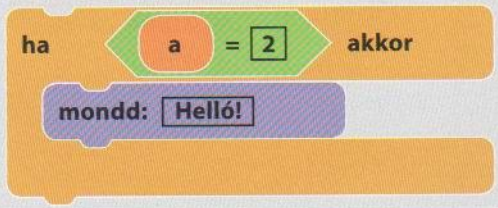
〈 86–87
Mi a Python?

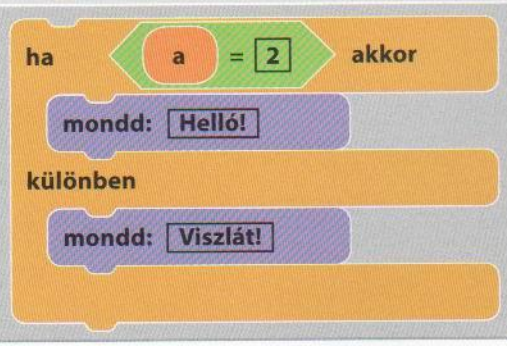
Bonyolultabb utasítások
104–105 〉

Utasítás	Python 3	Scratch 2
Futtatás	„Run" menü vagy F5 gomb (a kódablakban)	
Leállítás	„Ctrl+C" gombok (a terminálablakban)	
Szöveg kiírása	`print('Helló!')`	
Változó értékadása számmal	`mágikus_szám = 42`	
Változó értékadása szöveggel	`szó = 'sárkány'`	
Szöveg beolvasása billentyűzetről változóba	`kor = input('korod?')` `print('korom:' + kor)`	

Változó értékének növelése	`macskák = macskák + 1`
	vagy
	`macskák += 1`
		

Összeadás	`a + 2`	
Kivonás	`a - 2`	
Szorzás	`a * 2`	
Osztás	`a / 2`	

Utasítás	Python 3	Scratch 2
Mindig ciklus	`while True:` `    jump()`	
10-szer ismétlődő ciklus	`for i in range (10):` `    jump()`	
Egyenlő?	`a == 2`	
Kisebb?	`a < 2`	
Nagyobb?	`a > 2`	
Tagadás	`not`	
Vagy	`or`	
És	`and`	
Ha-akkor	`if a == 2:` `    print('Helló!')`	

Ha-akkor-különben	`if a == 2:`
	`    print('Helló!')`
	`else:`
	`    print('Viszlát!')`
		

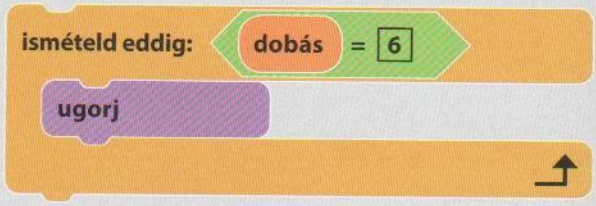
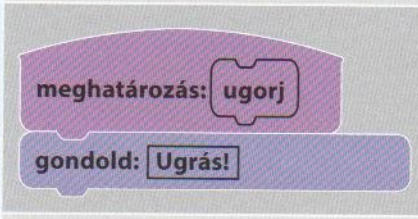
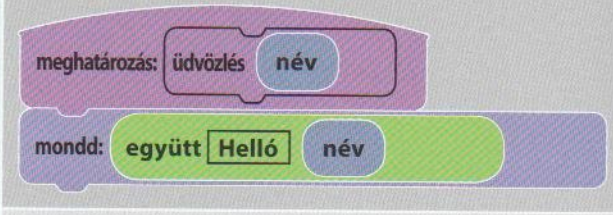
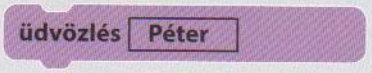
Bonyolultabb utasítások

A Pythonban használhatók a bonyolultabb Scratch-blokkokhoz hasonló utasítások: például csinálhatunk összetett ciklusokat, kezelhetünk szövegeket és listákat, rajzolhatunk ábrákat teknőccel.

LÁSD MÉG

‹ 86–87
Mi a Python?

‹ 102–103
Egyszerű utasítások

Utasítás	Python 3	Scratch 2
Elöl tesztelő ciklusok	`while dobás != 6:` `  ugorj()`	
Várakozás	`from time import sleep` `sleep(2)`	
Véletlen számok	`from random import randint` `dobás = randint(1, 6)`	
Függvény vagy szubrutin definiálása	`def ugorj():` `  print('Ugrás!')`	
Függvény vagy szubrutin hívása	`ugorj()`	
Paraméteres függvény vagy szubrutin definiálása	`def üdvözlés(név):` `  print('Helló, ' + név)`	
Függvény vagy szubrutin hívása	`üdvözlés('Péter')`	

Utasítás	Python 3	Scratch 2

Teknőcgrafika	`from turtle import *`
	`clear()`
	`pendown()`
	`forward(100)`
	`right(90)`
	`penup()`
		töröld a rajzokat
		tollat tedd le
		menj (100) lépést
		fordulj (90) fokot
		tollat emeld fel

Karakterláncok összefűzése	`print(üdvözlés + név)`	mondd: együtt üdvözlés név
Karakterlánc egy karakterének kérése	`név[0]`	letter (1) of név
Karakterlánc hossza	`len(név)`	név hossza
Üres lista készítése	`menü = list()`	Lista létrehozása
Elem beszúrása egy lista végére	`menü.append(valami)`	add valami to menü
Lista elemeinek száma	`len(menü)`	menü lista hossza
Lista 5. elemének értéke	`menü[4]`	mondd: item (5) of menü
Lista 2. elemének törlése	`del menü[1]`	delete (2) of menü
Lista eleme-e?	`if 'olíva' in menü:` `  print('Jaj, ne!')`	ha menü tartalmazza: olíva ? akkor mondd: Jaj, ne!

Melyik ablak?

Két ablak közül választhatunk az IDLE-ben. A kódablakba írhatjuk és menthetjük a programunkat, a terminálablakban pedig közvetlenül futtathatunk Python-parancsokat.

LÁSD MÉG

❮ 92–93
Az IDLE bemutatása

❮ 96–97
A Kísértetház

A kódablak

Eddig a kódablakot használtuk programírásra. Begépeljük, elmentjük, futtatjuk a programot, a kimenet pedig a terminálablakban jelenik meg.

▽ **Programok futtatása**
A Python programok futtatásának folyamata. Futtatás előtt mindig menteni kell a programot.

1 Írj be egy programot a kódablakba!
Írd be ezt a kódot a kódablakba, mentsd el, és a futtatáshoz kattints a „Run" menü „Run Module" menüpontjára.

```
a = 10
b = 4
print(a + b)
print(a - b)
```

Legyen „a" értéke 10

Legyen „b" értéke 4

A „print" utasítás kiírja a művelet eredményét

2 A kimenet a terminálablakban
Amikor fut a program, a kimenete a terminálablakban jelenik meg.

```
>>>
14
6
```

A műveletek eredménye megjelenik a terminálablakban

A terminálablak

A Python megérti a terminálablakba írt parancsokat is. Ezek nyomban a sor beírása után lefutnak, és az eredmény azonnal látszik.

```
>>> a = 10
>>> b = 4
>>> a + b
14
>>> a - b
6
```

Az első két parancsnak nincs kimenete, azok csak értékadások

Azonnali eredmény

◁ **Kód és kimenet egy helyen**
A terminálablakban láthatók a parancsok és az eredmények is. Könnyebb megmondani, hogy melyik eredmény melyik művelethez tartozik, ha a terminálablakba írjuk a parancsokat.

△ **Kísérletezz!**
A terminálablak azonnali eredményt ad, és ez megkönnyíti a parancsok kipróbálását.

A Python-játszótér

A terminálablak alkalmas a parancsok kipróbálásra, beleértve a rajzolást is. A teknőccel ugyanúgy lehet rajzolni a képernyőre, ahogy a tollal a Scratchben.

Betölti a teknőcöt vezérlő összes parancsot

```
>>> from turtle import *
>>> forward(100)
>>> right(120)
>>> forward(100)
```

Előremozgatja a teknőcöt

◁ **Vidd be a kódot!**
Írd ezeket a parancsokat a terminálablakba. Beírás után rögtön lefutnak. Amerre a teknőc mozog, húz egy vonalat.

◁ **A teknőcgrafika**
Kitalálod, hogyan kell más alakzatokat rajzolni, például négyzetet vagy ötszöget? Ha újra neki akarsz fogni, írd a „clear()" parancsot a terminálablakba.

Melyik ablakot használd?

A kódablakot vagy a terminálablakot érdemes használni? Az adott programtól függ, és attól, hogy szeretnéd-e később is felhasználni.

▷ **Kódablak**
Hosszabb kódokhoz a kódablak a legjobb, mert el tudod menteni, és később is szerkesztheted. Ez egyszerűbb, mint újra beírni parancsokat, ha újra le szeretnéd futtatni a programot, vagy ha valami hasonlóra van szükséged. Viszont minden alkalommal menteni és futtatni kell.

◁ **Terminálablak**
A terminálablak tökéletes gyors kísérletek elvégzésére, például egy parancs működésének a kipróbálására. Használható számológépként is. Viszont nem menti el a parancsokat, ezért ha valamit többször szeretnél megismételni, használd inkább a kódablakot.

TANÁCSOK

A kód színezése

Az IDLE kiszínezi a kód elemeit. A színek megsúgják neked, hogy a Python melyik beírt szót miként értelmezi.

◁ **Beépített függvények**
A „print"-hez hasonló utasítások lila színűek.

◁ **Karakterláncok**
A karakterláncokat zöld szín jelöli. Ha a zárójelek is zöldek, az arra utal, hogy kimaradt egy idézőjel.

◁ **Szimbólumok és nevek**
A kód túlnyomó része fekete.

◁ **Kimenet**
A terminálablakban megjelenő kimenet kék színű.

◁ **Kulcsszavak**
A kulcsszavak, mint az „if" vagy az „else", narancssárgák. A Python nem engedi meg, hogy kulcsszavakat változónévként használj.

◁ **Hibák**
Piros szín figyelmeztet a hibaüzenetekre a terminálablakban.

Változók a Pythonban

A változókat a programban adatok megőrzésére használjuk. Olyanok, mint a dobozok, amelyekbe belerakjuk az adatokat, és fel is címkézzük őket.

LÁSD MÉG

Adattípusok **110–111** ❯

Matek a Pythonban **112–113** ❯

Karakterláncok a Pythonban **114–115** ❯

A bemenet és a kimenet **116–117** ❯

Függvények **130–131** ❯

Változó létrehozása

Amikor egy számot vagy karakterláncot beleteszünk egy változóba, értékadásnak nevezzük. Ezt az „=" jellel fejezzük ki. Próbáld ki ezt a kódot a terminálablakban.

A változó neve

A változóhoz rendelt érték

```
>>> csontok = 3
```

△ **Értékadás számmal**

Számmal történő értékadáskor írd be a változó nevét, az egyenlőségjelet és utána a számot.

A változó neve

A változóhoz rendelt karakterlánc

```
>>> kutya_neve = 'Blöki'
```

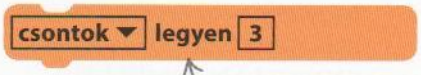
△ **Értékadás karakterlánccal**

Karakterlánccal történő értékadáskor írd be a változó nevét, az egyenlőségjelet és utána félidézőjelek közé a karakterláncot.

JEGYEZD MEG!

Változók a Scratchben

A Pythonban az értékadó utasítás ugyanazt végzi el, mint ez a Scratch-blokk. A Pythonban azonban nem kell sehova külön kattintani a változó létrehozásához. Itt rögtön létrejön a változó, amikor értéket adunk neki.



Scratch-blokk, amellyel értéket adunk a változónak

Változó kiírása

A „print" utasítással lehet írni a képernyőre. Ennek semmi köze a printerhez, azaz a nyomtatóhoz. Megnézhetjük vele a változó értékét.

```
>>> print(csontok)
3
```

A változó neve

△ **Numerikus kimenet**

A „csontok" változó a 3-as számot tárolja, így ezt írja ki a terminálablak.

```
>>> print(kutya_neve)
Blöki
```

Nem kell idézőjel

△ **Karakteres kimenet**

A „kutya_neve" változó karakterláncot tárol, és ez a karakterlánc kerül a terminálablakba. Nem jelenik meg idézőjel a karakterlánc kiírásakor.

Változó tartalmának megváltoztatása

Ahhoz, hogy megváltoztassuk egy változó értékét, csak új értéket kell neki adni. Az „ajándékok" változó eredeti értéke 2. Ez változik 3-ra, amikor újból értéket adunk neki.

```
>>> ajándékok = 2
>>> print(ajándékok)
2
>>> ajándékok = 3
>>> print(ajándékok)
3
```

Módosítja a változó értékét

A változók használata

Egy változó értéke használható egy másik változó értékadásakor az „=" segítségével. Például a „nyulak" változó értéke a nyulak számát tárolja. Ezt felhasználhatjuk arra, hogy a „cilinderek" változónak értéket adunk úgy, hogy minden nyúlnak jusson cilinder.

1 Változók értékadása

Add az 5 értéket a „nyulak" változónak, majd a „nyulak" által ugyanezt az értéket rendeld a „cilinderek" változóhoz.

A változó neve

A változóhoz rendelt érték

```
>>> nyulak = 5
>>> cilinderek = nyulak
```

A „cilinderek" változónak ugyanaz lesz az értéke, mint a „nyulak" változónak

2 A változók értékének kiírása

Két változó kiírásához írd őket ugyanabba a zárójelbe a „print" utasítás után, vesszővel elválasztva. A „cilinderek" és a „nyulak" változó értéke egyaránt 5.

```
>>> print(nyulak, cilinderek)
5 5
```

Írj szóközt a vessző után!

3 Változtasd meg a „nyulak" értékét!

Ha megváltoztatod a „nyulak" értékét, az nincs hatással a „cilinderek" értékére. A „cilinderek" változó értéke csak akkor változik, ha új értéket adsz neki.

```
>>> nyulak = 10
>>> print(nyulak, cilinderek)
10 5
```

Adj a „nyulak" változónak új értéket!

A „cilinderek" értéke változatlan

TANÁCSOK

A változó elnevezése

Van néhány szabály, amelyet be kell tartani a változók elnevezésekor:

Minden betű és szám használható.

Nem kezdődhet számmal változónév.

Szimbólumok (pl. -, /, # vagy @) nem használhatók.

Szóköz nem használható.

Aláhúzásjel (_) használható szóköz helyett.

A kis- és nagybetűk között van különbség, tehát a „Kutyák" és a „kutyák" két különböző változó.

Ne használj Python-utasításokat változónévnek (pl. „print")!

Adattípusok

A Pythonban többféle adattípus van. Legtöbbször a Python tudja, hogy milyen adatot használunk, de néha meg kell változtatnunk egy-egy adat típusát.

LÁSD MÉG

Matek a Pythonban **112–113 ❯**

Karakterláncok a Pythonban **114–115 ❯**

Feltételek **118–119 ❯**

Listák **128–129 ❯**

Számok

A Pythonban a számoknak két típusuk van: az „integer", azaz egész szám (tizedesvessző nélküli szám) és a „float", azaz tizedes tört (úgynevezett lebegőpontos szám). Egész szám használható dolgok (pl. bárányok) megszámlálására, míg a tizedes tört dolgok (pl. tömeg) mérésére.

```
>>> bárány = 1
>>> print(bárány)
1
```

Egész szám

△ **Egész számok**
Az egész számok tizedesvessző nélküli számok, mint az 1 a „bárány" változóban.

```
>>> bárány = 1.5
>>> print(bárány)
1.5
```

Az 1,5 tizedes tört

△ **Tizedes törtek**
A tizedes törtek tizedesvesszővel (angolul ponttal) tagolt számok, mint az 1,5. Általában nem használjuk dolgok megszámolására.

Karakterláncok

Ahogy a Scratchben, a Pythonban is egy szöveget hívunk karakterláncnak. A karakterláncok állhatnak betűkből, számokból, szóközökből és egyéb szimbólumokból (pl. pont, vessző). A karakterláncot általában félidézőjelek közé írjuk.

▷ **A karakterláncok használata**
Karakterlánc értékadásakor a szöveget félidézőjelek közé kell tenni.

Karakterlánc félidézőjelek között

```
>>> a = 'Programozni jó!'
>>> print(a)
Programozni jó!
```

Az „a" változó értékének kiírása

A logikai értékek

A Pythonban a logikai adattípusnak csak „True" (igaz) vagy „False" (hamis) értéke lehet.

▷ Igaz érték
Ha egy változónak a „True" értéket adjuk, a típusa logikai lesz.

```
>>> a = True
>>> print(a)
True
```

Nincs idézőjel

A kiírt logikai érték

▷ Hamis érték
Ha egy változónak a „False" értéket adjuk, a típusa akkor is logikai lesz.

```
>>> a = False
>>> print(a)
False
```

A kiírt logikai érték

TANÁCSOK

Adattípus azonosítása

A Pythonban sokféle adattípus van. Hogy megtudd, mi milyen típusú, használd a „type" utasítást.

A „type" utasítás

```
>>> type(24)
<class 'int'>
>>> type(24.3)
<class 'float'>
>>> type('24')
<class 'str'>
```

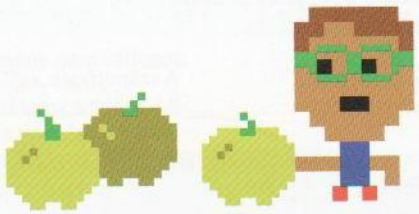
A 24 egész szám („int")

A 24,3 tizedes tört („float")

A ' 24 ' karakterlánc („str"), mert idézőjelek között van

Adattípusok átalakítása

A változók bármilyen adattípust tárolhatnak, de problémát okozhat, ha összekeverjük a különböző típusokat. Ilyenkor át kell alakítani az adat típusát, különben hibaüzenetet kapunk.



Ez a karakterlánc jelenik meg a képernyőn

▷ Típuskeveredés
Az „input" utasítás mindig karakterláncot olvas be, akkor is, ha számjegyeket írtunk. Ebben a példában az „alma" változó tartalma karakterlánc, így hibaüzenetet kapunk.

A változó neve

```
>>> alma = input('Írd be az almák számát: ')
Írd be az almák számát 2
>>> print(alma + 1)
TypeError
```

Megpróbál 1-et hozzáadni az „alma" változóhoz

Hibaüzenet jelenik meg, mert a Python nem tud számot hozzáadni egy karakterlánchoz

▷ Adattípus átalakítása
Karakterlánc számmá alakítására az „int()" függvény használható.

```
>>> print(int(alma) + 1)
3
```

A program most már működik, és kiírja az eredményt

A karakterlánc-változót egész számmá alakítjuk, így már hozzá lehet adni egy számot

Matek a Pythonban

A Python sokféle matematikai feladat megoldására használható, beleértve az összeadást, a kivonást, a szorzást és az osztást. A műveletekben változók is használhatók.

LÁSD MÉG

52–53 Matematika

108–109 Változók a Pythonban

Egyszerű számítások

A Pythonban az egyszerű számítások elvégezhetők a terminálablakban. A „print()" függvényt ilyenkor nem kell használni, a Python anélkül is kiírja az eredményt. Íme néhány példa:

Nem oszthatsz nullával, ha mégis megpróbálod, hibaüzenetet kapsz.

```
>>> 12 + 4
16
```

Használd a terminálablakot, és azonnal megkapod az eredményt!

△ **Összeadás**
Használd a „+" jelet számok összeadásához!

Az eredmény megjelenik, amint lenyomod az „Enter" billentyűt

```
>>> 12 - 4
8
```

△ **Kivonás**
Használd a „–" jelet, hogy kivond a második számot az elsőből!

A számítógép a „*" jelet használja a szorzáshoz, nem pedig az „x" vagy a „·" jelet

```
>>> 12 * 4
48
```

△ **Szorzás**
Használd a „*" jelet két szám összeszorzásához!

```
>>> 12 / 4
3.0
```

Az osztás eredménye Pythonban mindig tizedes tört

△ **Osztás**
Használd a „/" jelet, hogy eloszd az első számot a másodikkal.

Zárójelek használata

Zárójelekkel mondhatjuk meg a Pythonnak, hogy melyik műveletet végezze el elsőként. Mindig a zárójelben lévő műveletet végzi el legelőször.

Először kiszámolja, hogy 6 + 5 = 11, aztán a 11-et megszorozza 3-mal

```
>>> (6 + 5) * 3
33
```

△ **Először az összeadás**
Ebben a példában zárójelbe tettük az összeadást, hogy azt hajtsa végre először a Python.

Először kiszámolja, hogy 5 * 3 = 15, aztán a 15-höz hozzáadja a 6-ot

```
>>> 6 + (5 * 3)
21
```

Más eredmény

△ **Először a szorzás**
Itt a zárójel miatt a szorzást számolja először, így más lesz az eredmény.

Eredmények tárolása változóban

Ha néhány változónak számot adtunk értékül, használhatjuk őket képletekben. Ha egy változónak ilyen képletet adunk értékül, akkor nem maga a művelet, hanem az eredménye lesz a változó értéke.

1 Egyszerű összeadás

Ez a program összeadja a „hangyák" és a „pókok" változókat, majd az eredményt a „rovarok" változóban tárolja.

```
>>> hangyák = 22
>>> pókok = 35
>>> rovarok = hangyák + pókok
>>> print(rovarok)
57
```

Összeadja a két változó értékét

Kiírja a „rovarok" változó értékét

2 Módosítsd egy változó értékét!

Módosítsd a „hangyák" vagy a „pókok" változó értékét. Add össze őket újra, és az eredményt tedd a „rovarok" változóba.

```
>>> hangyák = 22
>>> pókok = 18
>>> rovarok = hangyák + pókok
>>> print(rovarok)
40
```

Módosítsd a „pókok" értékét!

Add össze őket újra!

Az eredmény megváltozik

3 Az értékadás átugrása

Ha a „rovarok" változónak nem adunk újra értéket, nem fog módosulni, még ha megváltoztattuk is a „hangyák" vagy a „pókok" értékét.

```
>>> hangyák = 11
>>> pókok = 17
>>> print(rovarok)
40
```

Kiírja a „rovarok" értékét

Az eredmény nem változott (még mindig 18 + 22)

Véletlen számok

Véletlen számokhoz először be kell tölteni a „randint" függvényt. Ehhez az „import" utasításra van szükség. A „randint()" egy előre megírt függvény, amely egy véletlen egész számot ad vissza.

```
>>> from random import randint
>>> randint(1, 6)
3
```

Betölti a „randint()" függvényt

Véletlen számot ad 1 és 6 között

A 3 lett a véletlen szám

△ **Dobj a kockával!**

A „randint()" függvény a zárójelbe írt két szám közötti véletlen számot ad vissza, ebben a programban 1 és 6 között.

JEGYEZD MEG!

A „véletlen" blokk

A „randint()" függvény úgy működik, mint a „véletlen" Scratch-blokk. A Scratchben is a legkisebb és a legnagyobb számot kell beírni a blokkba. A Pythonban a számok zárójelben állnak, vesszővel elválasztva.

△ **Egész számok**

A Python „randint()" függvénye és a Scratch-blokk is egész számot ad vissza – soha nem tizedes törtet.

Karakterláncok a Pythonban

Szavak és mondatok kezelésére tökéletes a Python. Különböző karakterláncokat (sztringeket) összefűzhetünk, vagy egyes részeiket külön is kezelhetjük.

LÁSD MÉG

❮ 54–55
Karakterláncok és listák

❮ 110–111
Adattípusok

Karakterlánc létrehozása

Egy karakterlánc tartalmazhat betűket, számokat, szimbólumokat és szóközöket is. Ezeket együttvéve karaktereknek nevezzük. A karakterláncokat változókban is tárolhatjuk.

▷ **Karakterláncok változókban**
Változók tárolhatnak karakterláncokat is. Írd ezt a két karakterláncot az „a" és a „b" változóba.

Az idézőjel jelzi, hogy a változó tartalma karakterlánc

```
>>> a = 'Fuss! '
>>> b = 'Jönnek az ufók.'
```

Karakterláncok összeadása

Két szám összeadásának eredménye egy újabb szám. Ugyanígy, ha két karakterláncot adunk össze, egy új karakterlánccá fűzzük őket.

Az „a" és a „b" változó együttes tartalma kerül a „c" változóba

```
>>> c = a + b
>>> print(c)
Fuss! Jönnek az ufók.
```

△ **Karakterláncok összefűzése**
A „+" jel fűzi össze a karakterláncokat, az eredmény a „c" változóba kerül.

Az új karakterláncot a „c" változóba tároljuk

```
>>> c = b + ' Vigyázz! ' + a
>>> print(c)
Jönnek az ufók. Vigyázz! Fuss!
```

△ **Karakterlánc beszúrása**
Egy karakterlánc beszúrható két másik közé. Próbáld ki a fenti példát!

Az új karakterlánc az üzenet közepén jelenik meg

TANÁCSOK

Karakterlánc hossza

A „len()" függvénnyel megkapjuk egy karakterlánc hosszát. A Python megszámolja a benne lévő összes karaktert a szóközökkel együtt, és ezt az értéket adja eredményül.

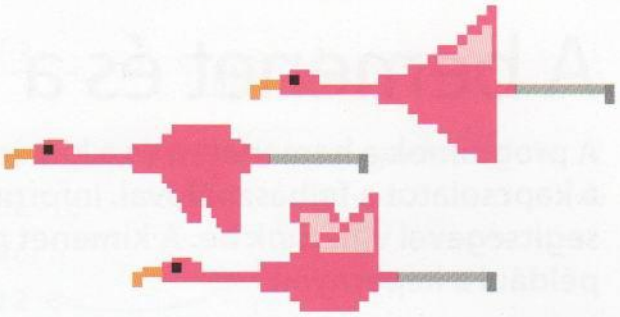
Kiszámolja az „a" változóban lévő karakterlánc („Fuss!") hosszát

```
>>> len(a)
5
>>> len(b)
15
```

A „b" változóban lévő karakterlánc („Jönnek az ufók.") hossza 15 karakter

A karakterek számozása

Egy karakterlánc minden karakteréhez tartozik egy szám, amely a helyére utal. Így önálló betűnként, szimbólumonként kezelhetjük őket.



1 A számozás nullától indul

A Python a sorszámozást a 0-tól kezdi. A második karakter sorszáma 1, a harmadiké 2 és így tovább.

```
>>> a = 'FLAMINGÓ'
```

2 A karakterek számolása

A helyre utaló sorszámot indexnek nevezzük. Segítségével kiemelhetünk egy karaktert a karakterláncból.

```
>>> a[3]
'M'
```

Az index köré szögletes zárójel kerül

Az „a" változó 3-as sorszámú karaktere

3 Szeletelés

Két index segítségével kivághatunk egy részt a karakterláncból. A beírt tartomány utolsó karaktere már nem kerül bele az eredménybe.

```
>>> a[1:7]
'LAMING'
```

Kettőspont jelöli a karaktertartományt

Az „a" változónak az 1-estől a 6-os indexig tartó szelete

4 Az elejétől vagy a végétől

Ha elhagyjuk a kezdő- vagy a záróindexet, akkor a Python automatikusan a legelejétől, illetve a legvégéig értelmezi a szeletet.

```
>>> a[:3]
'FLA'
>>> a[3:]
'MINGÓ'
```

A 0-s indextől kezdi

A 7-es indexszel végződik

Aposztrófok

A karakterláncokat írhatjuk félidézőjelbe vagy normál idézőjelbe is. Az viszont fontos, hogy egyforma legyen az elején és a végén. A könyvben mindig félidézőjelet (aposztrófot) használunk. De mit tegyünk, ha aposztrófot írnánk a karakterláncba?

```
>>> print('Nézd má\', esik.')
Nézd má', esik.
```

Aposztróf használata a karakterláncban

△ **Aposztróf levédése**

A Python az aposztrófot nem tekinti a karakterlánc lezárásának, ha elé „\" jelet írunk. Ezt levédésnek nevezzük.

A bemenet és a kimenet

A programok a bemeneten és a kimeneten keresztül tartják a kapcsolatot a felhasználóval. Információt a billentyűzet segítségével vihetünk be. A kimenet pedig megjelenhet például a képernyőn.

LÁSD MÉG

❮ 100–101
A program folyamata

❮ 110–111
Adattípusok

Ciklusok a Pythonban
122–123 ❯

A bemenet

Az „input()" függvény segítségével fogad információt a program a billentyűzetről. Addig vár a program, amíg a felhasználó le nem üti az „Enter" billentyűt.

1 A bemenet használata

Egy program megmondhatja, hogy mit kell beírni. Ezt az üzenetet az „input()" függvény zárójelében kell megadni.

A kettőspont utáni szóköz szebbé teszi a kimenetet

```
név = input ('Írd be a neved: ')
print('Szia,', név)
```

A program kimenete attól függ, hogy milyen nevet írt be a felhasználó

2 Kimenet a terminálablakban

A program futása során az „Írd be a neved: " kérdés és a válasz jelenik meg a terminálablakban.

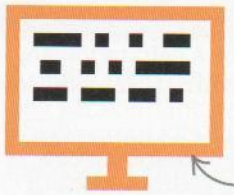
```
Írd be a neved: Juli
Szia, Juli
```

A program kimenete

Ide írja a felhasználó a választ

A kimenet

A „print()" függvény karaktereket jelenít meg a terminálablakban. Szöveg és változók kiírására is használható.



1 Változók létrehozása

Készíts három változót a példához. Kettő közülük legyen karakterlánc, egy pedig egész szám.

```
>>> a = 'Dávid'
>>> b = 'kora'
>>> c = 12
```

Az idézőjelek jelölik, hogy ezek karakterláncok

Nincs idézőjel, tehát ez egy egész szám

2 A „print()" függvény használata

Több elemet is írhatsz a „print()" függvény zárójelének a belsejébe. Lehetnek különböző típusú változók vagy lehet karakterlánc és változó vegyesen is.

```
>>> print(a, b, c)
Dávid kora 12
>>> print('Viszlát,', a)
Viszlát, Dávid
```

Vessző választja el az elemeket

Karakterláncok tagolása

Eddig a kimenet elemei egy sorban, szóközzel elválasztva jelentek meg. Íme két másik lehetőség a tagolásukra.

```
>>> print(a, b, c, sep='-')
Dávid-kora-12
```

A kimenetek közötti karakter

△ **Kötőjelezd a kimenetet!**
Tehető kötőjel a kiírandó változók közé. Más karakter is használható, például a „+" vagy a „*".

A határolójel

```
>>> print(a, b, c, sep='\n')
Dávid
kora
12
```

Minden változó új sorba kerül

△ **Kimenetek új sorban**
A szóközt vagy a kimenetek közötti karaktert határolójelnek vagy szeparátornak („sep") nevezzük. Ha „\n"-t használunk, új sorba kerül minden kimenet.

A kimenet háromféle lezárása

Többféleképpen is jelezhetjük a „print" függvény kimenetének a végét.

```
>>> print(a, '.')
Dávid .

>>> print(a, end='.')
Dávid.
```

Pont a karakterlánc végén

Pont hozzáadása záró („end") karakterként

△ **Mondatzáró pont hozzáadása a kimenethez**
Hozzá lehet adni egy pontot karakterláncként a kiírandókhoz, de akkor megjelenik előtte egy szóköz. Ezt elkerülheted, ha „end=' .' "-et használsz.

Ciklus a háromszoros kiíráshoz

Szóköz mint záró-karakter

```
>>> for n in range(3):
    print('Hurrá!', end=' ')
Hurrá! Hurrá! Hurrá!
```

A kimenet egy sorban jelenik meg

△ **Kimenet egy sorban**
A szóközt zárókarakterként használva az összes kimenet egy sorban jelenik meg.

TANÁCSOK

Lezárási lehetőségek

Az „end" és a „sep" kulcsszavak jelzik a Pythonnak, hogy a következő elem nem egy újabb karakterlánc. Jegyezd meg, különben hibás lehet a programod.


```
>>> print(a, end='\n\n\n\n')
Dávid


>>>
```

Mindegyik „\n" új sort kezd

Üres sorok a kimeneten

△ **Üres sorok a végén**
Az „\n" használatával minden kimenet új sorba kerül. Ha többet használsz belőle, üres sorokat tehetsz a program végére.

Feltételek

A programok a végrehajtás további menetéről döntéseket hoznak, változók, számok vagy karakterláncok összehasonlításával. Ezek a döntések „True" vagy „False" válaszokat eredményeznek.

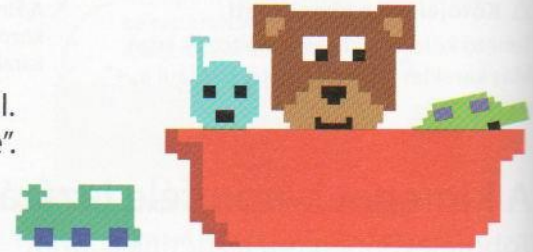
LÁSD MÉG

‹ 62–63
Igaz vagy hamis?

‹ 108–109
Változók a Pythonban

Logikai műveletek

Logikai műveletekkel összehasonlíthatunk változókat számokkal, karakterláncokkal vagy akár más változókkal. Az összehasonlítás eredménye mindig „True" vagy „False".



`==`	Egyenlő
`!=`	Nem egyenlő
`<`	Kisebb, mint
`>`	Nagyobb, mint
`<=`	Kisebb vagy egyenlő
`>=`	Nagyobb vagy egyenlő

△ **Az összehasonlító logikai műveletek típusai**
Hatféle összehasonlító művelet van. A Python két egyenlőségjelet használ az egyenlőség ellenőrzésére. (Egy egyenlőségjelet értékadásnál használunk.)

```
>>> játékok = 10
>>> játékok == 1
False
>>> játékok > 1
True
>>> játékok < 1
False
>>> játékok != 1
True
>>> játékok <= 10
True
>>> not játékok == 1
True
>>> játékok == 9 or játékok == 10
True
>>> játékok == 9 and játékok == 10
False
```

Megvizsgálja, hogy a „játékok" egyenlő-e 1-gyel

Megvizsgálja, hogy a „játékok" nagyobb-e 1-nél

Megvizsgálja, hogy a „játékok" kisebb-e 1-nél

Megvizsgálja, hogy a „játékok" különbözik-e 1-től

Megvizsgálja, hogy a „játékok" kisebb vagy egyenlő-e, mint 10

A „not" logikai művelet az ellentétére változtatja a választ (a példában hamisról igazra)

Az „or" logikai művelet megvizsgálja, hogy a „játékok" egyenlő-e 9-cel vagy 10-zel

Az „and" logikai művelet megvizsgálja, hogy a „játékok" egyenlő-e 9-cel és 10-zel is. Ez persze sohasem teljesülhet.

▷ **Ellenőrzés a terminálablakban**
A logikai műveletek működnek a terminálablakban is. Próbáld ki a példa alapján a következőkkel együtt: „not" (tagadás), „or" (vagy), „and" (és).

Ma van Ella szülinapja?

Ella szülinapja július 28-án van. Ez a program vesz egy „hónap" és egy „nap" változót, és logikai műveletekkel megvizsgálja, ma van-e Ella szülinapja.

1 Ellenőrizd a szülinapot!

Hozz létre változókat a hónap és a nap tárolására. Használd az „and" logikai műveletet a szülinap ellenőrzésére.

```
>>> nap = 28
>>> hónap = 7
>>> nap == 28 and hónap == 7
True
```

Az „and" logikai művelet megvizsgálja, hogy mind a két feltétel teljesül-e

Két egyenlőségjelet kell használni!

Ma van Ella szülinapja!

2 A „nem szülinap"-érzékelő

Megfordíthatod a választ a „not" logikai művelet használatával. Mindennap „True" választ fogsz kapni, Ella szülinapját kivéve.

```
>>> nap = 28
>>> hónap = 7
>>> not (nap == 28 and \
    hónap == 7)
False
```

Ezzel a karakterrel jelezhetjük a Pythonnak, hogy a következő sor valójában ennek a sornak a folytatása

Ella szülinapja van, tehát a válasz „False"

3 Szülinap vagy újév?

Derítsd ki az „or" logikai művelet használatával, hogy Ella szülinapja vagy újév napja van-e! A zárójelek segítségével együtt kezelhetők az összetartozó részek.

```
>>> day = 28
>>> hónap = 7
>>> (nap == 28 and hónap == 7) \
    or (nap == 1 and hónap == 1)
True
```

Július 28. ellenőrzése

A válasz „True" lesz, ha Ella szülinapja vagy újév napja van.

Karakterláncok

Két karakterlánc összehasonlítható a „==" vagy a „!=" logikai műveletekkel. A karakterláncoknak teljesen meg kell egyezniük ahhoz, hogy a válasz „True" legyen.

```
>>> kutya = 'Vau-vau'
>>> kutya == 'Vau-vau'
True
>>> kutya == 'vau-vau'
False
>>> kutya == 'Vau-vau '
False
```

A két karakterlánc teljesen megegyezik, ezért a válasz „True"

A karakterláncok nem egyformák, mert ez nem nagybetűvel kezdődik

A karakterláncok nem egyformák, mert szóköz van a záró idézőjel előtt

△ **Teljesen megegyezők**

Két karakterlánc akkor egyenlő, ha a nagybetűzés, a szóközök és a szimbólumok is teljesen megegyeznek bennük.

TANÁCSOK

Logikai műveletek

Az „in" logikai művelet segítségével megvizsgálhatjuk, hogy egy karakterlánc tartalmaz-e egy másikat. Ellenőrizhetjük, hogy egy karakterláncban szerepel-e egy bizonyos betű vagy betűcsoport.

Megvizsgálja, hogy az „a" betű benne van-e az „abc"-ben

```
>>> 'a' in 'abc'
True
>>> 'd' in 'abc'
False
```

A „d" nincs az „abc"-ben, ezért „False"

Elágazások

Logikai kifejezésekkel meghatározhatjuk, hogy a program egy adott feltétel igaz vagy hamis értékétől függően milyen ágon fusson tovább. Ezt elágazásnak nevezzük.

LÁSD MÉG

‹ 64–65
Feltételek és elágazások

‹ 118–119
Feltételek

Csináld vagy ne csináld!

Az „if" (ha) utasítás működésének a lényege, hogy ha a feltétel teljesül, vagyis a válasz „True", akkor a benne lévő utasítások lefutnak. Ha nem teljesül, akkor ezek után folytatódik a végrehajtás. Az „if"-ben lévő utasításokat négy szóközzel behúzva kell írni.

1 Az „if" feltétel
A program megkérdezi a felhasználót, hogy ma van-e a szülinapja. Ha a válasz „i", akkor megjelenik egy köszöntő üzenet.

```
válasz = input('Ma van a szülinapod? (i/n) ')
if válasz == 'i':
    print('Boldog szülinapot!')
```

Irányítja a felhasználót, hogy mit írjon be

Ez a része a programnak csak akkor fut le, ha a válasz „i"

Négy szóközzel behúzva

2 Kimenet, ha a feltétel teljesül
Futtasd a programot, és írj be „i"-t! Az üzenet megjelenik. Ha bármi mást írsz, akkor nem jelenik meg.

Írj be „i"-t!

```
Ma van a szülinapod? (i/n) i
Boldog szülinapot!
```

Az üzenet megjelenik

Csináld ezt vagy azt!

Az „if" kiegészíthető az „else" (egyébként) utasítással. Ez azt jelenti, hogy ha a feltétel teljesül, akkor valami történik, ha viszont nem teljesül, akkor valami más fut le.

1 Az „if–else" feltétel
A program újévi üdvözletet küld, ha „i" volt a válasz, és valami mást ír ki, ha egyéb választ adtunk.

```
válasz = input('Ma van újév napja? (i/n)')
if válasz == 'i':
    print('Boldog új évet!')
    print('Kezdődhet a tűzijáték!')
else:
    print('Még nem.')
```

Ne felejtsd el a kettőspontot!

Ez csak akkor jelenik meg, ha a válasz „i"

Csak akkor fut le, ha a válasz nem „i"

Ide is kettőspontot kell tenni

2 **Kimenet, ha a feltétel teljesül**
Futtasd a programot, és írj be „i"-t. A program elküldi az újévi köszöntőt. Most nem mutatja a másik üzenetet.

```
Ma van újév napja? (i/n) i
Boldog új évet!
Kezdődhet a tűzijáték!
```

Írj be „i"-t!

3 **Az „else" ág kimenete**
Írj be „n"-t vagy más karaktert, ekkor nem jelenik meg az újévi köszöntő, a „Még nem." üzenet viszont igen.

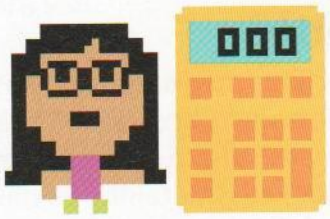
Írj be „n"-et!

```
Ma van újév napja? (i/n) n
Még nem.
```

Más üzenet jelenik meg

Csináld az egyiket!

Az „elif" utasítás az „else-if" rövidítése. Jelentése, hogy ha egy feltétel „True", akkor csinál valamit, egyébként pedig leellenőriz egy újabb feltételt. Ha az teljesül, akkor valami mást csinál. A következő számológép az „elif" utasítást használja.



1 **Az „if-elif-else" feltétel**
A program megvizsgálja, hogy mit írunk be. Ha „össz", „kiv", „szor" vagy „oszt", akkor kiírja a művelet eredményét.

Arra kéri a felhasználót, hogy írjon be egy számot

Ne felejtsd el az idézőjelet és a zárójelet!

```
a = int(input('a = '))
b = int(input('b = '))
op = input('össz/kiv/szor/oszt: ')
if op == 'össz':
    c = a + b
elif op == 'kiv':
    c = a - b
elif op == 'szor':
    c = a * b
elif op == 'oszt':
    c = a / b
else:
    c = 'Hiba'
print('Eredmény = ',c)
```

Írd az „össz" szót az összeadáshoz!

Írd az „oszt" szót az osztáshoz!

Hibaüzenet kerül a „c" változóba, ha bármi mást írsz be

Kiírja az eredményt vagy a hibaüzenetet

2 **Kimenet igaz érték esetén**
Teszteld a programot! Írj be két számot és a „kiv" műveletet. Az eredmény az első és a második szám különbsége lesz.

```
a = 7
b = 5
össz/kiv/szor/oszt: kiv
Eredmény = 2
```

Írj be két számot!

Írd be „kiv", hogy kivond a 7-ből az 5-öt!

Az eredmény a két változó különbsége lesz

3 **Az „else" ág kimenete**
Az „else" ág akkor fut le, ha nem a négy megadott művelet egyikét írod be. Ekkor hibaüzenet jelenik meg.

```
a = 7
b = 5
össz/kiv/szor/oszt: más
Eredmény = Hiba
```

Írj ide valami mást!

Hibaüzenet jelenik meg

Ciklusok a Pythonban

LÁSD MÉG

‹ 48–49
Tollak és teknőcök

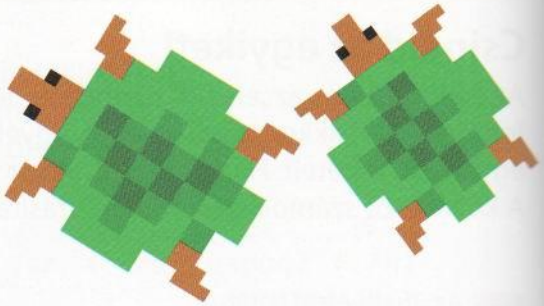
„While" ciklus **124–125 ›**

Kiugrás a ciklusból **126–127 ›**

Ha egy programban sok ismétlődő sor van, akkor annak beírása időrabló, olvasása, megértése pedig nehézkes lehet. Ilyen esetekben célszerűbb ciklusokat használni. A legegyszerűbb ciklusnál mi adjuk meg, hogy hányszor ismétlődjön a tartalma. Ez a „for" ciklus.

Ismétlés

A „for" ciklus megismétli az utasításokat anélkül, hogy többször be kellene írnunk. Akkor használható, ha pontosan tudjuk, hogy hányszor kell ismételnie. Például ha szeretnénk kiírni egy 30 fős osztály névsorát.



1 Programozd a teknőcöt!

A „for" ciklussal lerövidítheted a kódot. Ezzel a programmal irányíthatsz egy teknőcöt, ami nyomot hagy maga után a képernyőn, bármerre jár. Rajzolhatsz vele alakzatokat, például háromszöget.

120 fokkal jobbra fordítja a teknőcöt

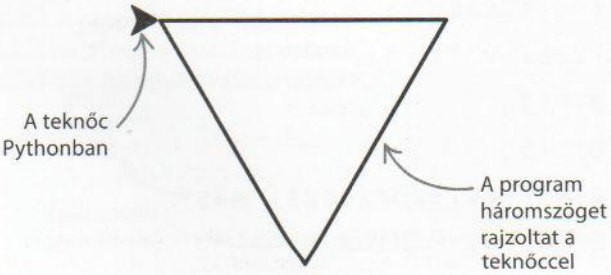
```
from turtle import *
forward(100)
right(120)
forward(100)
right(120)
forward(100)
right(120)
```

Betölti a teknőc irányításához szükséges utasításokat

Előremozgatja a teknőcöt

2 Háromszög rajzolása a teknőccel

A program irányítja a teknőcöt, hogyan rajzoljon háromszöget az oldalak és a szögek megadásával. Amikor futtatod a programot, a teknőc külön ablakban fog megjelenni.



3 A „for" ciklus használata

A fenti program ugyanazt a két utasítást ismétli háromszor – minden oldalhoz egyszer. Rövidebb lesz ez a program, ha a két utasítást „for" ciklusba írod. Próbálj a lenti kód használatával háromszöget rajzolni.

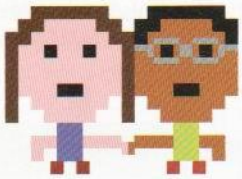
```
for i in range(3):
    forward(100)
    right(120)
```

A „for" cikluss hatására háromszor ismétlődnek az utasítások

A cikluson belüli utasításokat négy szóközzel beljebb kell írni

Ciklusváltozók

A ciklusváltozó az ismétléseket számolja. Értéke a „range" függvénybe írt első számtól (vagy 0-tól) kezdődik, és megáll a megadott legnagyobb szám előtt eggyel.



Ciklusváltozó

A ciklus tartalma tízszer fut le

```
for i in range(10):
    print(i, end=' ')
```

A Python megáll eggyel a megadott szám előtt

```
>>> 0 1 2 3 4 5 6 7 8 9
```

△ **Egyszerű ciklusváltozó**

Itt a „range" függvényben nincs megadva a ciklusváltozó kezdőértéke. Ilyenkor a Python 0-tól kezd számolni, ahogy a karakterláncoknál is teszi.

Ennek hatására kettesével számol

```
for i in range(2, 11, 2):
    print(i, end=' ')
```

```
>>> 2 4 6 8 10
```

A kimeneten kettesével növekednek a számok

△ **Számolás kettesével**

A ciklusnak van egy harmadik paramétere is. Ez mondja meg, hogy kettesével számoljon. A ciklus 10-nél áll meg, mert az eggyel kevesebb, mint a megadott 11.

Ennek hatására visszafelé számol

```
for i in range(10, 0, -1):
    print(i, end=' ')
```

```
>>> 10 9 8 7 6 5 4 3 2 1
```

△ **Számolás visszafelé**

A program visszafelé számol 10-től, ahogy rakétakilövéskor szokás. A ciklusváltozó 10-zel indul, és addig csökken, amíg el nem éri az 1-et.

Egymásba ágyazott ciklusok

Egymásba ágyazott ciklusnak nevezzük, ha egy cikluson belül van egy másik ciklus. Ebben az esetben a külső ciklus tartalma akkor ismétlődik, ha a belső ciklus a végére ért.

Ha n-szer szeretnénk lefuttatni a ciklust, akkor a „range" függvénybe n+1-et kell írni

Külső ciklus

```
n = 3
for a in range(1, n + 1):
    for b in range(1, n + 1):
        print(b, 'x', a, '=', b * a)
```

Belső ciklus

Ezt a műveletet kilencszer írja ki

△ **Ciklusok a ciklusban**

Ebben a példában, mialatt a külső ciklus egyszer fut le, a belső háromszor. Így a külső ciklus összesen háromszor, a belső pedig kilencszer fut le.

```
>>>
1 x 1 = 1
2 x 1 = 2
3 x 1 = 3
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
```

Az „a" értéke

A „b" értéke

Első alkalommal fut le a külső ciklus (közben a belső háromszor)

Második alkalommal fut le a külső ciklus

Harmadik alkalommal fut le a külső ciklus

△ **Mi történik?**

Az egymásba ágyazott ciklusok kiírják az 1-es, a 2-es és a 3-as szorzótábla első három sorát. Az „a" változó akkor változik, amikor a külső ciklus újra lefut. A „b" az „a" minden értékénél elszámol 1-től 3-ig.

„While" ciklusok

Akkor használunk „for" ciklust, ha előre tudjuk, hányszor kell a tartalmát megismételni. Előfordul azonban az is, hogy egy feltétel teljesüléséig kell valamit ismételnünk. A „while" ciklus addig ismétlődik, amíg szükséges.

LÁSD MÉG

‹ 118–119
Feltételek

‹ 122–123
Ciklusok a Pythonban

Kiugrás a ciklusból
126–127 ›

A „while" ciklus

A „while" ciklus addig ismétlődik, amíg a megadott feltétel teljesül. Ezt a feltételt nevezzük ciklusfeltételnek, értéke igaz vagy hamis lehet.

1 A „while" ciklus létrehozása
Állítsd be a „válasz" változó kezdőértékét. A ciklusfeltételnek kezdetben teljesülnie kell, különben a program egyszer sem lép be a ciklusba.

▷ Hogyan működik?
A ciklus megvizsgálja, hogy a feltétel teljesül-e. Ha igen, akkor újra lefut a tartalma. Ha nem, akkor kihagyja a ciklust.

A cikluson belüli utasításokat be kell húzni négy szóközzel

```
válasz = 'i'
while válasz == 'i':
    print('Maradj nyugodt!')
    válasz = input('Barátságos a szörny? (i/n) ')
print('Menekülj!')
```

A „válasz" változó legyen „i"

A „while" ciklus csak akkor fut le, ha a feltétel teljesül

Ha a feltétel nem teljesül, akkor a ciklus után következő, behúzás nélküli utasítás fut le, és más üzenet jelenik meg

2 Hogyan néz ki a program?
A beírt értéket elmentjük a „válasz" változóba. A ciklusfeltétel a „válasz == 'i'". Ha „i"-t írsz be, akkor a ciklus újra meg újra lefut. Ha „n"-t, akkor megáll.

```
>>>
Maradj nyugodt!
Barátságos a szörny? (i/n) i
Maradj nyugodt!
Barátságos a szörny? (i/n) i
Maradj nyugodt!
Barátságos a szörny? (i/n) n
Menekülj!
```

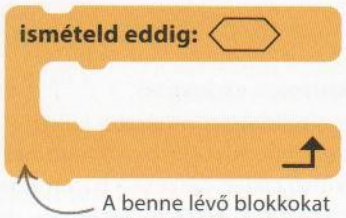
A válasz „i", a ciklus fut tovább

A válasz „n", így a ciklus véget ér, és új üzenet jelenik meg

JEGYEZD MEG!

Az „ismételd eddig" blokk

A Python „while" ciklusa olyan, mint a Scratchben az „ismételd eddig" blokk. Mindkettő addig ismétlődik, amíg valami meg nem változik a programban.



A benne lévő blokkokat a feltétel teljesüléséig ismétli

Végtelen ciklusok

Bizonyos ciklusok örökké futnak. Ha a „while" ciklus feltétele „True", sosem lehet hamis, így a ciklus soha nem ér véget. Ez gyakran hasznos, de lehet nagyon bosszantó is.

△ **Körbe-körbe**
A mindig teljesülő feltételen alapuló ciklust végtelen ciklusnak nevezzük. Sohasem ér véget.

1 **Végtelen ciklus létrehozása**
A ciklusfeltételt „True" értékre kell állítani. A cikluson belül nincs semmi, ami ezt meg tudná változtatni, így a ciklus folyamatosan fog futni.

A beírt szót a „válasz" változóban tároljuk

Mindig „True", így a ciklus nem ér véget

```
while True:
    válasz = input('Írj be egy szót, majd nyomj entert: ')
    print('Kérlek, ne írd be többször a „' + válasz + '" szót.')
```

2 **Hogyan néz ki a program?**
Az előző oldalon lévő program a ciklus feltételeként mindig megvizsgálja a felhasználó válaszát. Ha a válasz nem „i", akkor a ciklus leáll. A fenti program nem vizsgálja meg a választ, így a használója nem tudja leállítani.

Nem számít, mit írunk, a ciklus tovább fut

```
>>>
Írj be egy szót, majd nyomj entert: fa
Kérlek, ne írd be többször a „fa" szót.
Írj be egy szót, majd nyomj entert: víziló
Kérlek, ne írd be többször a „víziló" szót.
Írj be egy szót, majd nyomj entert: víz
Kérlek, ne írd be többször a „víz" szót.
Írj be egy szót, majd nyomj entert:
```

TANÁCSOK

Ciklus leállítása

Ha a program végtelen ciklusba kerül, az IDLE-ban le tudod állítani. Kattints a terminálablakba, és a „Ctrl" billentyűt lenyomva tartva nyomd le a „C" billentyűt. Ez utasítja az IDLE-t, hogy állítsa le a programot. Ugyanez történt a Scratchben a piros Stop gombra való kattintáskor.

JEGYEZD MEG!

A „mindig" blokk

Emlékszel a Scratchből a „mindig" blokkra? Ez mindaddig ismétli a benne lévő utasításokat, amíg a piros Stop gombra nem kattintasz. A „while True" ciklus ugyanígy működik. Ezzel készíthetsz folyamatosan kérdező vagy számokat kiíró programot.

A „mindig" blokk folyamatosan mozgatja a szereplőt

„While" ciklusok

Akkor használunk „for" ciklust, ha előre tudjuk, hányszor kell a tartalmát megismételni. Előfordul azonban az is, hogy egy feltétel teljesüléséig kell valamit ismételnünk. A „while" ciklus addig ismétlődik, amíg szükséges.

LÁSD MÉG

❮ 118–119
Feltételek

❮ 122–123
Ciklusok a Pythonban

Kiugrás a ciklusból
126–127 ❯

A „while" ciklus

A „while" ciklus addig ismétlődik, amíg a megadott feltétel teljesül. Ezt a feltételt nevezzük ciklusfeltételnek, értéke igaz vagy hamis lehet.

▷ **Hogyan működik?**
A ciklus megvizsgálja, hogy a feltétel teljesül-e. Ha igen, akkor újra lefut a tartalma. Ha nem, akkor kihagyja a ciklust.

1 A „while" ciklus létrehozása
Állítsd be a „válasz" változó kezdőértékét. A ciklusfeltételnek kezdetben teljesülnie kell, különben a program egyszer sem lép be a ciklusba.

A cikluson belüli utasításokat be kell húzni négy szóközzel

```
válasz = 'i'
while válasz == 'i':
    print('Maradj nyugodt!')
    válasz = input('Barátságos a szörny? (i/n) ')
print('Menekülj!')
```

A „válasz" változó legyen „i"

A „while" ciklus csak akkor fut le, ha a feltétel teljesül

Ha a feltétel nem teljesül, akkor a ciklus után következő, behúzás nélküli utasítás fut le, és más üzenet jelenik meg

2 Hogyan néz ki a program?
A beírt értéket elmentjük a „válasz" változóba. A ciklusfeltétel a „válasz == 'i'". Ha „i"-t írsz be, akkor a ciklus újra meg újra lefut. Ha „n"-t, akkor megáll.

```
>>>
Maradj nyugodt!
Barátságos a szörny? (i/n) i
Maradj nyugodt!
Barátságos a szörny? (i/n) i
Maradj nyugodt!
Barátságos a szörny? (i/n) n
Menekülj!
```

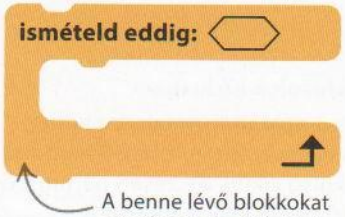
A válasz „i", a ciklus fut tovább

A válasz „n", így a ciklus véget ér, és új üzenet jelenik meg

JEGYEZD MEG!

Az „ismételd eddig" blokk

A Python „while" ciklusa olyan, mint a Scratchben az „ismételd eddig" blokk. Mindkettő addig ismétlődik, amíg valami meg nem változik a programban.



A benne lévő blokkokat a feltétel teljesüléséig ismétli

Végtelen ciklusok

Bizonyos ciklusok örökké futnak. Ha a „while" ciklus feltétele „True", sosem lehet hamis, így a ciklus soha nem ér véget. Ez gyakran hasznos, de lehet nagyon bosszantó is.

△ **Körbe-körbe**
A mindig teljesülő feltételen alapuló ciklust végtelen ciklusnak nevezzük. Sohasem ér véget.

1 **Végtelen ciklus létrehozása**
A ciklusfeltételt „True" értékre kell állítani. A cikluson belül nincs semmi, ami ezt meg tudná változtatni, így a ciklus folyamatosan fog futni.

A beírt szót a „válasz" változóban tároljuk

Mindig „True", így a ciklus nem ér véget

```
while True:
    válasz = input('Írj be egy szót, majd nyomj entert: ')
    print('Kérlek, ne írd be többször a „' + válasz + '" szót.')
```

2 **Hogyan néz ki a program?**
Az előző oldalon lévő program a ciklus feltételeként mindig megvizsgálja a felhasználó válaszát. Ha a válasz nem „i", akkor a ciklus leáll. A fenti program nem vizsgálja meg a választ, így a használója nem tudja leállítani.

Nem számít, mit írunk, a ciklus tovább fut

```
>>>
Írj be egy szót, majd nyomj entert: fa
Kérlek, ne írd be többször a „fa" szót.
Írj be egy szót, majd nyomj entert: víziló
Kérlek, ne írd be többször a „víziló" szót.
Írj be egy szót, majd nyomj entert: víz
Kérlek, ne írd be többször a „víz" szót.
Írj be egy szót, majd nyomj entert:
```

TANÁCSOK

Ciklus leállítása

Ha a program végtelen ciklusba kerül, az IDLE-ban le tudod állítani. Kattints a terminálablakba, és a „Ctrl" billentyűt lenyomva tartva nyomd le a „C" billentyűt. Ez utasítja az IDLE-t, hogy állítsa le a programot. Ugyanez történt a Scratchben a piros Stop gombra való kattintáskor.

JEGYEZD MEG!

A „mindig" blokk

Emlékszel a Scratchből a „mindig" blokkra? Ez mindaddig ismétli a benne lévő utasításokat, amíg a piros Stop gombra nem kattintasz. A „while True" ciklus ugyanígy működik. Ezzel készíthetsz folyamatosan kérdező vagy számokat kiíró programot.

A „mindig" blokk folyamatosan mozgatja a szereplőt

Kiugrás a ciklusból

A program beragadhat egy ciklusba, de van mód arra, hogy kiugorjunk belőle. A „break" utasítás elhagyja a ciklust (a végtelent is), a „continue" utasítás pedig a ciklus elejére ugrik a ciklusváltozó új értékével.

LÁSD MÉG

‹ 122–123
Ciklusok Pythonban

‹ 124–125
While ciklusok

A „break" használata

A „break" utasítás lehetővé teszi, hogy kiugorjunk a ciklusból, a ciklusfeltételtől függetlenül. A cikluson belül ezután következő utasítások nem fognak lefutni.

1 Írj egy egyszerű programot!
Ez a program a hetes szorzótáblát kérdezi ki. A ciklus addig ismétlődik, amíg mind a 12 kérdésre nem kapott választ. Írd a programot a kódablakba, hogy később módosíthasd.

```
tábla = 7
for i in range(1, 13):
 print('Mennyi ', i, 'x', tábla, '?', sep='')
 tipp = input()
 megoldás = i * tábla
 if int(tipp) == megoldás:
     print('Helyes!')
 else:
     print('Nem, a helyes válasz:', megoldás)
print('Vége')
```

Az „i" változó 1-től 12-ig számol

Az „i" a ciklusváltozó

2 Írj bele „break" utasítást!
Egy „break" használatával ki lehet ugrani a ciklusból. Ha a „stop" szöveget írod be, a program kiugrik a ciklusból.

```
tábla = 7
for i in range(1, 13):
    print('Mennyi ', i, 'x', tábla, '?', sep='')
    tipp = input()
    if tipp == 'stop':
        break
    megoldás = i * tábla
    if int(tipp) == megoldás:
        print('Helyes!')
    else:
        print('Nem, a helyes válasz:', megoldás)
print('Vége')
```

Ha a „tipp" egyenlő „stop"-pal, a program kihagyja a ciklus maradékát, és kiírja, hogy „Vége"

A „megoldás" változóban van a helyes válasz

```
>>>
Mennyi 1 x 7 ?
1
Nem, a helyes válasz: 7
Mennyi 2 x 7 ?
14
Helyes!
Mennyi 3 x 7 ?
stop
Vége
```

Első futáskor az „i" értéke 1

Az „i" értéke a következő alkalommal már 2

Ez indítja a „break" utasítást, és kiugrik a ciklusból

3 Hogyan működik?
Ha a felhasználó nem szeretné folytatni, és beírja, hogy „stop", lefut a „break" utasítás, és a program kiugrik a ciklusból.

Kihagyás

A „continue" utasítást akkor használjuk, ha ki szeretnénk hagyni egy kérdést a ciklusból való kiugrás nélkül. Arra utasítja a programot, hogy hagyja ki a ciklus hátralévő utasításait, és térjen át a ciklusváltozó következő értékére.

```
tábla = 7
for i in range(1,13):
    print('Mennyi ', i, 'x', tábla, '?', sep='')
    tipp = input()
    if tipp == 'stop':
        break
    if tipp == 'kihagy':
        print('Kihagyás')
        continue
    megoldás = i * tábla
    if int(tipp) == megoldás:
        print('Helyes!')
    else:
        print('Nem, a helyes válasz:', megoldás)
print('Vége')
```

A ciklus első futásakor azt a kérdést teszi fel, hogy: „Mennyi 1x7?"

A következő kérdésre ugrik

4 A „continue" használata
Írj egy „if" elágazást a ciklusba, hogy lásd, mikor írja be a felhasználó azt, hogy: „kihagy". Ilyenkor a program kiírja, hogy „Kihagyás", majd lefut a „continue" utasítás, és a ciklus elejére ugrik a ciklusváltozó új értékével.

5 Mi történik?
Ha a felhasználó nem akar válaszolni egy kérdésre, akkor a „kihagy" szó beírásával a következő kérdésre ugorhat.

```
>>>
Mennyi 1 x 7?
kihagy
Kihagyás
Mennyi 2 x 7?
14
Helyes!
Mennyi 3 x 7?
```

Írd be, hogy „kihagy", és a következő kérdésre ugrik

A ciklus folytatódik, ha jó választ adunk

Listák

Ha sok adatot szeretnénk eltárolni egy helyen, használhatunk listákat. A listák tartalmazhatnak számokat, karakterláncokat, más listákat vagy akár ezek kombinációit.

LÁSD MÉG

〈 54–55
Karakterláncok és listák

Vicces mondatok
132–133 〉

Mi a lista?

A lista egy olyan Python-adatszerkezet, amely sorrendben tárolja az adatokat. Minden hozzáadott elemhez tartozik egy sorszám, ennek segítségével hivatkozhatunk rá. A lista elemeit bővíthetjük, törölhetjük vagy módosíthatjuk.

▽ **Pillantás a listákra**
Minden listaelem félidézőjelek között áll, vesszővel elválasztva. Az egész listát szögletes zárójelek határolják.

A lista a „polcok" változóban van elmentve

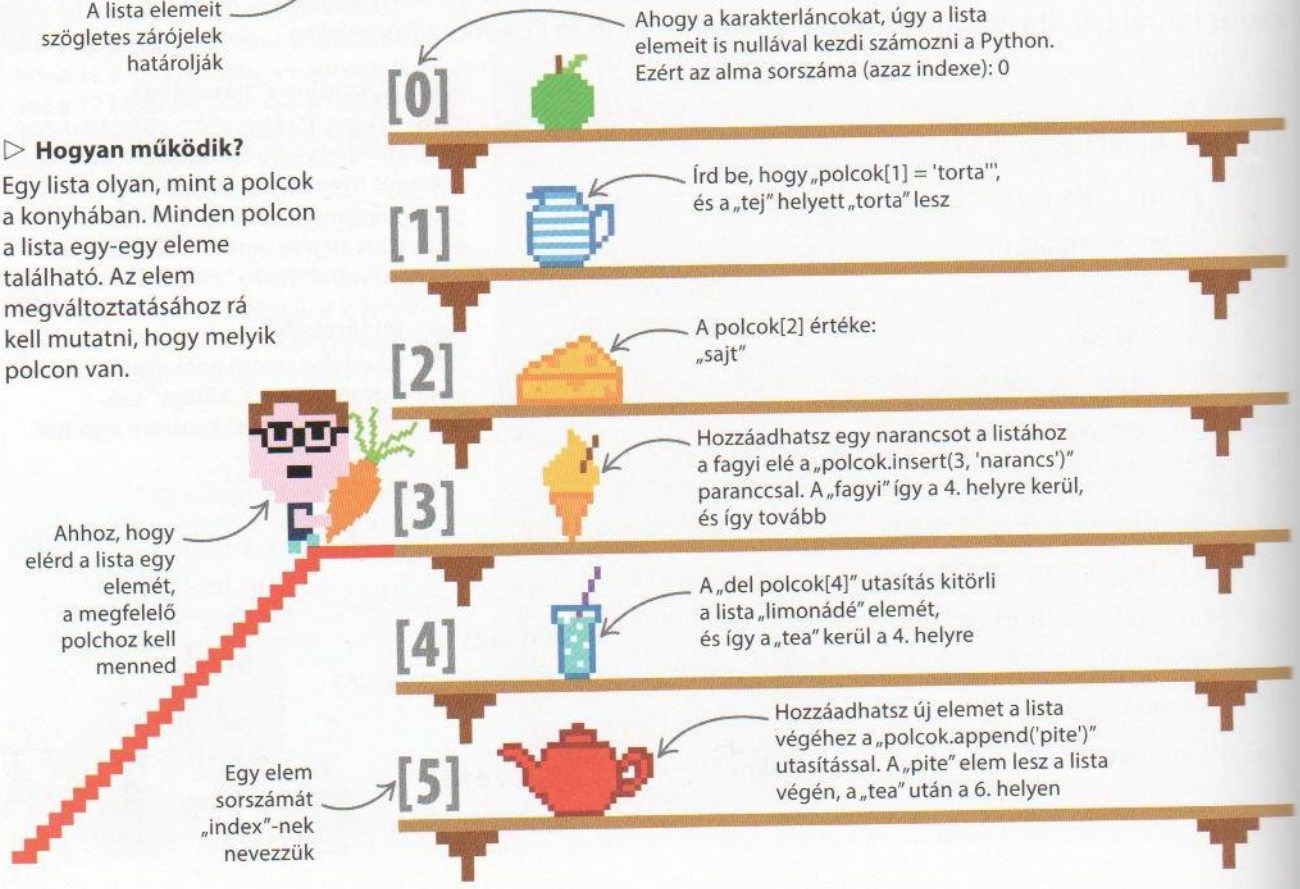
```
>>> polcok = ['alma', 'tej', 'sajt', 'fagyi', \
'limonádé', 'tea']
```

Az egyes elemeket vessző választja el

Ezt a jelet használjuk, ha a következő sorban akarjuk folytatni a programot

A lista elemeit szögletes zárójelek határolják

▷ **Hogyan működik?**
Egy lista olyan, mint a polcok a konyhában. Minden polcon a lista egy-egy eleme található. Az elem megváltoztatásához rá kell mutatni, hogy melyik polcon van.



A listák használata

Miután létrehoztunk egy listát, írhatunk programokat a benne lévő adatok kezeléséhez – például ciklussal. Akár kombinálhatjuk is a listákat egy új létrehozásához.

A lista a „nevek" változóban van elmentve

```
>>> nevek = ['Simon', 'Kata', 'Vica']
>>> for item in names:
        print('Szia,', item)

Szia, Simon
Szia, Kata
Szia, Vica
```

A ciklus belseje (ciklusmag) négy szóközzel beljebb kezdődik

A program futáskor kiírja, hogy „Szia,", és utána minden nevet a listából

FOGALOMTÁR

Módosítható objektumok

A Pythonban a lista módosítható, azaz megváltoztathatjuk a tartalmát. Törölhetjük, új elemeket adhatunk hozzá, vagy megváltoztathatjuk a sorrendjüket. Vannak olyan Python-elemek, ilyen pl. a „tuple" (*lásd* 134–135. o.), amelyek nem módosíthatók, miután létrehoztuk őket.

◁ Listák a ciklusokban

Használhatsz ciklusokat arra, hogy végigmenj a lista elemein. Ez a program köszön mindenkinek, aki a listában szerepel.

```
x = [1, 2, 3, 4]
y = [5, 6, 7, 8]
z = x + y
print(z)
z = [1, 2, 3, 4, 5, 6, 7, 8]
```

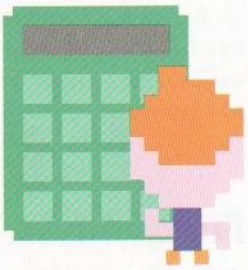
Ügyelj rá, hogy a listát szögletes zárójelek közé kell írni

Ez összefűzi a listákat

Az új listában benne van „x" minden eleme, utána pedig „y" minden eleme

◁ Listák összefűzése

Két listát össze lehet fűzni. Az új listában benne lesz mindkét lista minden eleme.



▽ Listák a listában

Egy lista elemei is lehetnek listák. A „bőrönd" lista két olyan listát tartalmaz, amelyben ruhák vannak felsorolva – mintha két ember közös bőröndbe csomagolta volna a ruháit.

A lista elemei szögletes zárójelen belül állnak, így a belső lista önálló eleme a „bőrönd" listának – „bőrönd[0]"

„bőrönd[1]"

```
>>> bőrönd = [['kalap', 'nyakkendő', 'zokni'], ['nadrág', 'cipő', 'póló']]
>>> print(bőrönd)
[['kalap', 'nyakkendő', 'zokni'], ['nadrág', 'cipő', 'póló']]
>>> print(bőrönd[1])
['nadrág', 'cipő', 'póló']
>>> print(bőrönd[1][2])
póló
```

Ez kiírja a teljes „bőrönd" listát

Ez kiírja a második lista, vagyis a „bőrönd[1]" elemeit

Ez a „bőrönd[1]" lista 2-es indexű elemét írja ki – ne feledd, hogy a Python 0-tól kezdi a számozást!

Függvények

A függvény olyan része a programnak, amely önálló részfeladatot lát el. Tagolja a kódot, neve van, és bármikor meghívható. A függvények használatával elkerülhetjük, hogy sokszor kelljen ugyanazt a kódot begépelni.

LÁSD MÉG
Vicces mondatok 132–133 ❯
Változók és függvények 138–139 ❯

Hasznos függvények

A Python sok hasznos, alapvető feladatokat ellátó függvényt tartalmaz. Amikor meghívunk egy függvényt, a Python megkeresi a hozzá tartozó kódot, és lefuttatja. Amint a függvény lefutott, a program onnan folytatódik, ahol a függvényt meghívtuk.

print()

△ **A „print()" függvény**
Ez a függvény tájékoztatást vagy eredményeket ír ki a képernyőre.

input()

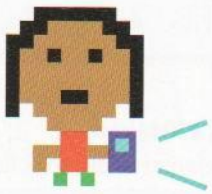
△ **Az „input()" függvény**
Ez a függvény a „print()" ellentéte. A felhasználó által beírt adatokat olvassa be.

randint()

△ **A „randint()" függvény**
Ez a függvény véletlen számot generál (mint a kockadobásnál). Véletlen események létrehozásánál használható.

Függvények létrehozása és meghívása

A Python beépített függvényein kívül más függvények is használhatók. Új függvény létrehozásához össze kell gyűjteni a benne használt utasításokat egy „csomagba", és el kell nevezni. Ezzel a névvel hívhatjuk meg később a függvényt.



1 Függvény létrehozása
Egy függvény definíciója mindig a „def" kulcsszóval és a függvény nevével kezdődik.

```
def üdvözlés():
    print('Szia!')
```

Kettőspont jelzi a függvény nevének a végét, és a benne lévő utasítások kezdetét

A függvényben lévő utasítás(ok)

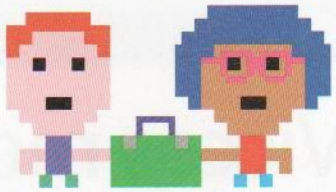
2 Függvény hívása
A függvény nevének a terminálablakba zárójelekkel történő beírása hívja meg a függvényt, és jeleníti meg a kimenetet.

```
>>> üdvözlés()
Szia!
```

Az „üdvözlés" függvény meghívása után megjelenik a kívánt kimenet

A zárójel jelzi, hogy ez függvény, nem pedig egy változó

Adatok átadása függvényeknek



Egy függvénynek meg kell mondani, hogy milyen adatokkal dolgozzon. Például a „print(a, b, c)" parancssor esetén a „print()" függvénynek az „a", „b" és „c" értékeket adjuk, a „magasság(1, 45)" kifejezésben pedig a „magasság()" függvénynek az 1 és a 45 számokat.

1 Paraméterek hozzáadása a függvényhez
A függvénynek átadott értékeket paramétereknek nevezzük. A paramétereket a definíció elején, a név utáni zárójelben kell megadni.

A paraméterek az „m" és a „cm"

```
def magasság(m, cm):
    teljes = (100 * m) + cm
    print(teljes, 'cm magas')
```

Kiírja a „teljes" változó értékét és a „cm magas" szöveget

A magasság centiméterben való kiszámításához a „m" értékét meg kell szorozni 100-zal (mivel 1 m = 100 cm)

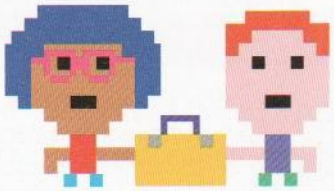
2 Átadott értékek
A függvénybe írt kód az átadott értékeket használja

Meghívja a függvényt, „m" = 1 és „cm" = 45 paraméterekkel

```
>>> magasság(1, 45)
145 cm magas
```

Kiírja, hogy 1 m 45 cm egyenlő 145 cm-rel

Adatok visszanyerése függvényből



A függvények hasznos tulajdonsága, hogy adatot küldhetnek vissza a programnak: ez a visszatérési érték. A „return" kulcsszóval és az utána írt értékkel határozható meg.

1 Számot visszaadó függvény definiálása
A Pythonban az „input()" függvény mindig karakterláncot ad vissza, akkor is, ha számot írunk be. Ez az új függvény számot ad vissza.

A szám egy „beírt" nevű karakterlánc-változóba kerül

```
def szám_be(prompt):
    beírt = input(prompt)
    szám = int(beírt)
    return szám

a = szám_be('Írd be a értékét: ')
b = szám_be('Írd be b értékét: ')
print('a + b =', a + b)
```

Visszaadja a változó értékét

Átalakítja a karakterláncot számmá, és tárolja a „szám" változóban

2 Szám mint kimenet
Ha a program az „input()" függvényt használná, a két karakterláncot, a „10"-et és a „7"-et adná össze. Az eredménye „107" lenne.

```
Írd be a értékét: 10
Írd be b értékét: 7
a + b = 17
```

Az „a + b" összeadás eredménye „17", mert a „szám_be()" függvény számot ad vissza

5. PROJEKT

Vicces mondatok

A ciklusok, a függvények és a listák magukban is sokféleképpen használhatók. Együtt viszont még összetettebb és érdekesebb programokat tudunk velük készíteni.

LÁSD MÉG

- ❮ **124–125** „While" ciklus
- ❮ **128–129** Listák
- ❮ **130–131** Függvények

Készíts vicces mondatokat!

Ez a program három lista szavaiból mondatokat készít. Mindegyikből véletlenszerűen választ egyet, és mondatokat állít össze belőlük.

1 Írd be a három listát egy új kódablakba. Ezzel definiáltad a listákat, amelyekből majd a mondatok készülnek.

Félidézőjel jelzi, hogy a lista elemei karakterláncok

A szögletes zárójel jelöli, hogy ezek listák

```
alany = ['Lili', 'Juli', 'Robi']
állítm = ['vesz', 'vezet', 'rág']
tárgy = ['lovat', 'biciklit', 'repülőt']
```

2 A mondatok úgy állnak össze, hogy véletlenszerűen választunk egy-egy elemet a listákból. Ehhez definiálj egy függvényt, mert többször fogjuk használni a programban.

Ez betölti a véletlen számot generáló függvényt („randint()")

Megadja, hogy hány szóból áll a lista (így a függvény bármilyen hosszú listával működni fog)

Választ egy véletlen számot, amely biztosan a lista egy elemére hivatkozik

```
from random import randint
def választ(szavak):
    szavak_száma = len(szavak)
    választott_szám = randint(0, szavak_száma - 1)
    választott_szó = szavak[választott_szám]
    return választott_szó
```

Tárolja a véletlen számhoz tartozó szót a „választott_szó" változóban

3 Írj ki egy mondatot úgy, hogy a „választ" függvényt lefuttatod minden listádra. Használd a „print" utasítást a mondat képernyőre írásához.

Add hozzá, hogy „egy", így értelmesebb lesz a mondat!

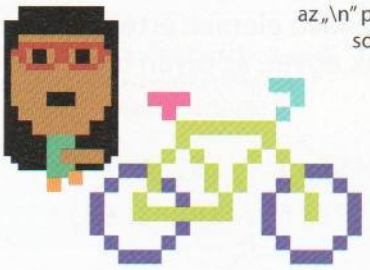
```
print(választ(alany), választ(állítm), 'egy', választ(tárgy), end='.\n')
```

Ez tesz pontot a mondat végére, az „\n" pedig új sort kezd

4 Mentsd el és futtasd a programot, a listák elemeiből álló vicces mondatokat fogsz kapni.

```
Lili rág egy biciklit.
```

A mondat véletlenszerűen jön létre a program futása során



Vicces mondatok mindörökké!

Végtelen ciklusba építve, a program egész addig mondatokat fog gyártani, amíg a „Ctrl+C" billentyűkkel le nem állítjuk.

1 A program végtelenítve írja a mondatokat, ha a „print" utasítást egy „while True" ciklusba tesszük.

```
while True:
    print(választ(alany), választ(állítm), 'egy', választ(tárgy), end='.\n')
    input()
```

A „print" utasítás ciklusba csomagolása

Mindig kiír egy új mondatot, ha lenyomjuk az „Enter"-t

2 Az „input()" függvény egy-egy újabb „Enter"-t vár. Enélkül a program túl gyorsan írná a mondatokat, és nem lehetne elolvasni őket.

A program ontja a mondatokat

```
Robi vezet egy repülőt.
Lili rág egy biciklit.
Juli vesz egy lovat.
```

TANÁCSOK

Olvasható kód

Nagyon fontos, hogy könnyen olvasható programot írj. Könnyebb a programon később változtatni, ha nem kell előbb mindig kibogozni a működését.

Tuple és szótár

LÁSD MÉG
‹ 110–111 Adattípusok
‹ 128–129 Listák

A Python az adatok sorrendben tárolásához listákat használ. Információ tárolására más adattípusok is vannak, ilyen például a „tuple"(elemek rendezett listája, más néven „rendezett n-es") és a „szótár". Az ilyen adattípusokat tárolónak vagy konténernek nevezzük.

Tuple

A tuple kicsit olyan, mint a lista, csak nem lehet megváltoztatni a benne lévő elemek értékét. Ha egyszer létrehoztunk egyet, az olyan is marad.

A tuple-t kerek zárójelbe írjuk

```
>>> sárkányA = ('Süsü', 15, 1.70)
>>> sárkányB = ('Paff', 16, 1.68)
```

A tuple elemeit vesszővel választjuk el

◁ **Miből áll egy tuple?**
Egy tuple vesszővel elválasztott, zárójelben írt elemeket tartalmaz. Jól tárolhatók benne összetartozó adatok, például egy sárkány neve, kora és magassága.

▷ **Egy elem kiragadása**
Egy elem használatához annak a tuple-ben elfoglalt pozícióját (indexét) kell megadni. A tuple is a nullától kezdi a számozást, ahogy a karakterláncok vagy a listák.

```
>>> sárkányB[2]
1.68
```

Ez kiválasztja a 2-es indexű elemet

```
>>> név, kor, magasság = sárkányA
>>> print(név, kor, magasság)
Süsü 15 1.7
```

A „sárkányA" tuple elemei külön-külön is használhatók

◁ **Tuple szétválasztása változókra**
Hozz létre három változót a „sárkányA" tuple-höz: „név", „kor", „magasság". A Python szétválasztja a tuple elemeit, és a változókba menti.

▷ **Tuple-ok listában**
A tuple-okat beletehetjük listába, mert a tárolók egymásba ágyazhatók. Használd ezt a programot, hogy több rendezett tuple-ból álló listát készíts.

Hozd létre a „sárkányok" nevű listát!

A listát szögletes zárójelbe kell írni

```
>>> sárkányok = [sárkányA, sárkányB]
>>> print(sárkányok)
[('Süsü', 15, 1.7), ('Paff', 16, 1.68)]
```

Minden tuple kerek zárójelben van, a lista szögletes zárójelein belül

A Python kiírja a lista összes elemét, nem csak a tuple-ok nevét

Szótárak

A szótárak olyanok, mint a listák, csak az elemeket index helyett egy „kulcs"-nak nevezett címke azonosítja. A szótár minden eleme egy kulcsból és egy értékből áll. Nem szükséges, hogy a szótár elemei a megadott sorrendben maradjanak, és az elemek értéke is változtatható.

▷ Szótár létrehozása

Ez a program létrehoz egy „kor" nevű szótárat. Minden elem kulcsa a személy neve, a hozzá tartozó érték pedig az életkora.

A szótárakat kapcsos zárójel jelöli

A szótár elemeit vesszők választják el

Használj kettőspontot a kulcs és az érték között!

```
>>> kor = {'Saci': 10, 'Szilvi': 8}
```

A kulcs úgy működik, mint az index

A szótárban tárolt érték (mindig kettőspont után következik)

```
>>> print(kor)
{'Saci': 10, 'Szilvi': 8}
```

A szótár neve

Ennek az elemnek a kulcsa: „Saci"

„Szilvi" értéke 8

◁ Szótár kiírása

A szótár esetében az elemek sorrendje megváltozhat.

▷ Új elem hozzáadása

Új értéket az új kulccsal megcímkézve adhatunk a szótárhoz.

A szótár neve

Új kulcs

```
>>> kor['Peti'] = 11
>>> print(kor)
{'Peti': 11, 'Saci': 10, 'Szilvi': 8}
```

Új elemet ad a szótárhoz

Az új elem bekerült a szótárba

A korábbi értékek is megmaradtak

```
>>> kor['Peti'] = 12
>>> print(kor)
{'Peti': 12, 'Saci': 10, 'Szilvi': 8}
```

Adj új értéket a „Peti" kulcsnak

„Peti" értéke megváltozott

◁ Érték módosítása

Adj új értéket egy létező kulcsnak, így megváltozik az értéke.

▷ Elemek törlése

Egy elemnek a szótárból való törlése nincs hatással a többi elemre, mert nem a sorszámuk, hanem a kulcsuk alapján azonosítjuk őket.

Ez törli az „Peti" kulcsú elemet

```
>>> del kor['Peti']
>>> print(kor)
{'Saci': 10, 'Szilvi': 8}
```

A „Peti" elem többé nem jelenik meg a szótárban

Listák a változókban

Első ránézésre kicsit fura lehet, ahogy a Python a listákat változókban tárolja. De nézzük meg, hogyan is működik ez a háttérben, és minden világos lesz.

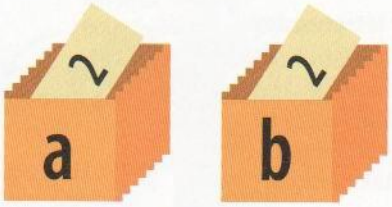
LÁSD MÉG

❮ **108–109** Változók a Pythonban

❮ **128–129** Listák

Emlékszel még, hogy a változók csupán értékeket tárolnak?

A változók olyanok, mint a dobozok. Egy változó értéke átmásolható egy másikba. Olyan, mintha lefotóznánk az „a" doboz tartalmát, és beletennénk a „b" dobozba.



△ **A változók működése**
Minden változó olyan, mint egy doboz, benne egy papírral, amelyen egy érték szerepel.

1 Érték mentése változóba
Add az „a" változónak a 2 értéket, majd az „a" változó értékét „b" változónak. A 2-t mint értéket átmásoljuk és mentjük „b"-be.

```
>>> a = 2
>>> b = a
>>> print('a =', a, 'b =', b)
a = 2 b = 2
```

Ez másolja „a" tartalmát „b"-be

Most „a"-nak és „b"-nek is 2 az értéke

Kiírja a változók nevét és értékét

2 Változó értékének módosítása
Ha egy változó értékét módosítjuk, akkor az nincs hatással a másik változóra. Ahogy attól sem változik meg a „b" dobozban lévő papír, hogy megváltoztatjuk az „a" dobozban lévő papíron a feliratot.

```
>>> a = 100
>>> print('a =', a, 'b =', b)
a = 100 b = 2
```

Módosítsd „a" értékét 100-ra!

Most „a" értéke 100, de „b" értéke még mindig 2

3 Egy másik érték módosítása
Ha megváltoztatod „b" értékét 22-re, „a" értéke továbbra is 100 marad. Hiába lett „b" értéke „a"-ból másolva, most már önálló változók: a „b" módosítása nincs hatással az „a"-ra.

```
>>> b = 22
>>> print('a =', a, 'b =', b)
a = 100 b = 22
```

A „b" értéke 22, „a" értéke pedig 100

Mi történik, ha listát teszünk változóba?

Egy érték másolása másik változóba az érték két különálló másolatát hozza létre. Ez jól működik, ha az érték szám, de mi történik más adattípusokkal? Ha a változó tartalma lista, akkor lesz egy kis különbség.

1 Lista másolása
Mentsd el az [1, 2, 3] listát a „listA" változóba. Aztán a „listA" változó értékét mentsd el a „listB" változóba. Most mindkettő tartalma [1, 2, 3].

Lista létrehozásához használj szögletes zárójelet!

Kiírja a változó nevét és tartalmát, hogy lássuk, mi van benne

```
>>> listA = [1, 2, 3]
>>> listB = listA
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 2, 3] listB = [1, 2, 3]
```

A „listA" és a „listB" változók tartalma most ugyanaz

2 A „listA" módosítása
Ha megváltoztatod a „listA[1]" elem értéket 1000-re, a „listB[1]" értéke is 1000 lesz. Az eredeti lista módosítása megváltoztatta a másolatot is.

Ez a lista második elemét módosítja, mert nullától kezdődik a számozás

```
>>> listA[1] = 1000
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 1000, 3] listB = [1, 1000, 3]
```

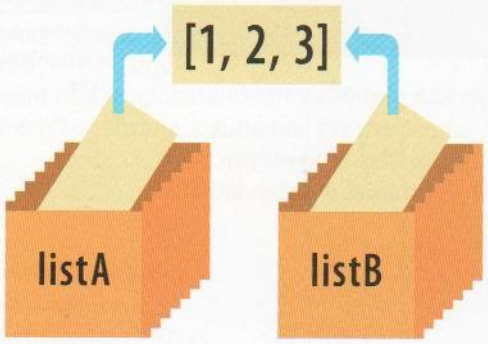
A „listA" és a „listB" második eleme is megváltozott

3 A „listB" módosítása
Ha megváltoztatod a „listB[2]" elem értékét 75-re, a „listA[2]" értéke is megváltozik. A lista másolatának módosítása megváltoztatja az eredeti listát is.

Ez a harmadik eleme a listának

```
>>> listB[2] = 75
>>> print('listA =', listA, 'listB =', listB)
listA = [1, 1000, 75] listB = [1, 1000, 75]
```

A „listA" és a „listB" harmadik eleme is megváltozott



△ **Mi történik?**
Egy listát tartalmazó változó nem magát a listát tartalmazza, csak egy rá mutató hivatkozást. A „listA" változó értékének lemásolása ezt a hivatkozást másolja le. Így a két változó ugyanazt a hivatkozást tartalmazza.

TANÁCSOK

Listák másolása

Független másolat létrehozásához a „copy" függvényt használjuk. A „listC" tartalma egy másik listára mutató hivatkozás lesz, és annak a listának az elemei a „listA" elemeinek a másolatai. A „listC" megváltoztatása nem lesz hatással a „listA"-ra, és fordítva.

```
>>> listC = listA.copy()
```

Változók és függvények

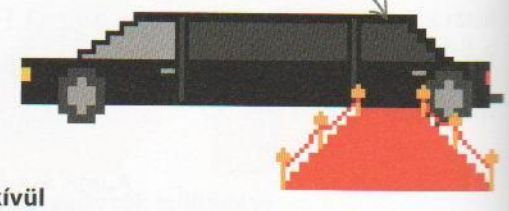
A függvényen belül létrehozott (lokális) változók és a főprogramban létrehozott (globális) változók különféleképpen működnek.

LÁSD MÉG
‹ 130–131 Függvények
Alakzatok rajzolása 158–159 ›

Lokális változók

A lokális változók csak a függvényen belül léteznek, ezért a főprogramban és más függvényeken belül nem használhatók. Ha megpróbálunk a függvényen kívül egy ilyen változóra hivatkozni, hibaüzenetet kapunk.

A lokális változók olyanok a függvényben, mint a filmsztárok a sötétített üvegű autóban – tudjuk, hogy ott vannak, de kívülről nem láthatjuk őket.



1 Változók a függvényen belül
Készíts lokális változót („a") az „fv1" függvényben, és írd ki az értékét a függvény meghívásával.

```
>>> def fv1():
        a = 10
        print(a)
>>> fv1()
10
```

Az „fv1" meghívása kiírja az „a" értékét

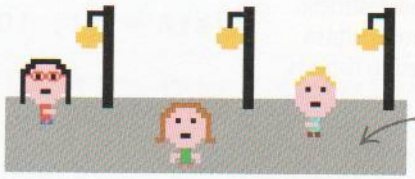
2 Változó a függvényen kívül
A főprogramban nem lehet kiírni az „a" változó értékét, mert az „a" csak az „fv1"-en belül létezik.

A főprogram nem ismeri az „a" változót, ezért hibaüzenet jelenik meg

```
>>> print(a)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
```

Globális változók

A főprogramban létrehozott változók neve globális változó. A függvények láthatják, de nem változtathatják meg az értékét.



A globális változók olyanok, mint az utcán sétáló emberek – mindenki láthatja őket

1 Változó a függvényen kívül
Készíts egy „b" nevű változót a főprogramban. Az új „fv2" függvény látja ezt a változót, és ki tudja írni az értékét.

```
>>> b = 1000
>>> def fv2():
        print(b)
>>> fv2()
1000
```

Az „fv2" látja a „b" változót, mert globális

Az „fv2" meghívása kiírja a „b" értékét

2 Ugyanaz a globális változó
A „b" változó értékét kiírhatjuk a főprogramban is. Nem függvényen belül hoztuk létre, így mindenhonnan látható.

```
>>> print(b)
1000
```

A „b" globális változó bárhol használható a főprogramban

A változók mint függvények bemenetei

Amikor egy változót egy függvény bemeneteként használunk, az értéke egy új, lokális változóba másolódik. Ennek a megváltoztatása a függvényen belül nincs hatással az eredeti változóra.

1 A változó értékének módosítása

Az „fv3" függvény bemenete az „y" lokális változó. A függvény kiírja az értékét, majd megváltoztatja „kenyér"-re, és kiírja az új értéket.

```
>>> def fv3(y):
        print(y)
        y = 'kenyér'
        print(y)
>>> z = 'vaj'
>>> fv3(z)
vaj
kenyér
```

Az „y" tartalmazza a híváskor átadott értéket

Az „y" értéke itt „kenyér"

Létrehoz egy „z" nevű globális változót

Az „y" változó az „fv3" meghívásakor a „z" változó értékét veszi fel

2 Változó kiírása

Ha kiírjuk a „z" értékét, látjuk, hogy az nem változott, miután meghívtuk az „fv3" függvényt, csupán átmásolódott az „y" lokális változóba.

```
>>> print(z)
vaj
```

Kiírja a „z" globális változó értékét, miután meghívtuk az „fv3" függvényt

Az „y" lokális változó másolatot tartalmaz a „z" változó értékéről. Attól, hogy megváltoztatjuk az „y" értékét „kenyér"-re, a „z" globális változó értéke változatlanul „vaj" marad

Globális változó elfedése

Globális változó nem módosítható függvényen belülről. Amikor egy függvény ezt megpróbálja, akkor igazából létrehoz egy ugyanolyan nevű lokális változót. Helyettesíti, elfedi a globális változót egy lokális változóval.

1 Globális változó módosítása

A „c" globális változó értéke 12345. Az „fv4" új értéket (555) ad a „c"-nek. Úgy tűnik, mint ha a „c" globális változó értéke változna meg.

```
>>> c = 12345
>>> def fv4():
        c = 555
        print(c)
>>> fv4()
555
```

A „c" globális változó kezdeti értéke

Kiírja a „c" értékét a függvényen belül

2 Változó kiírása

Ha kiíratjuk a „c" értékét a függvényen kívül, akkor láthatjuk, hogy nem változott meg. Az „fv4" a saját lokális változójának az értékét írja ki – ennek „c" is a neve.

```
>>> print(c)
12345
```

A globális „c" változó értéke nem változott

SZAKTANÁCS

Függvények meghívása

Függvényt kétféleképpen hívhatunk.

függvény(a)

A Pythonban némely függvényt meghívhatunk egy úgynevezett objektum („a") átadásával.

a.függvény()

Más függvények esetében a híváshoz a függvény nevét az objektum neve után írjuk ponttal elválasztva.

6. PROJEKT

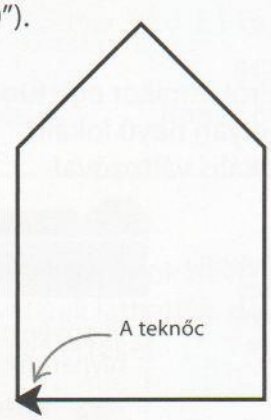
Rajzgép

LÁSD MÉG
❮ 122–123
Ciklusok a Pythonban
Könyvtárak 152–153 ❯

Ideje belevágni egy összetettebb projektbe. Ez a Rajzgép nevű program karakterláncokból állít elő utasításokat a teknőcnek, hogy különféle alakzatokat rajzolhassunk. A tervezésnél használt készségek fontosak minden programozási feladatnál.

Válassz tesztalakzatot!

Egy program megírásához, amely bármilyen alakzatot tud rajzolni, célszerű kiindulási alakzatot választani. Legyen ez a ház a példa, ezzel tesztelheted a programot. A projekt végére sokkal egyszerűbb kóddal tudod majd megrajzolni ezt a házat: egyetlen, néhány rövid rajzparancsot tartalmazó karakterlánc használatával (pl. „E100").



▷ **A teknőc házat rajzol**
A nyíl mutatja a teknőc végső irányát és helyzetét. A bal alsó sarokból indult, és az óramutató járásával megegyezően haladt körbe.

```
from turtle import *
reset()
left(90)
forward(100)
right(45)
forward(70)
right(90)
forward(70)
right(45)
forward(100)
right(90)
forward(100)
```

Betölti a teknőcöt vezérlő utasításokat

Kezdőhelyzetbe állítja a teknőcöt, és a rajzoláshoz leteszi a tollat

70 lépéssel előremozdítja a teknőcöt

90 fokkal jobbra fordítja a teknőcöt

△ **Programozd le, hogyan lehet házat rajzolni!**
Ez a program megmondja, hogyan rajzoljon házat a teknőc. Egyszerű feladat, mégis viszonylag sok utasítást igényel.

A program három része

A Rajzgép viszonylag nagy program lesz. Bontsuk három, különböző feladatot ellátó részprogramra.

1. függvény

△ **A Teknőcirányító**
Ez a függvény a felhasználótól kapott egyszerű parancsot alakítja át a teknőcnek szóló utasítássá. A beírandó parancs egy betű és egy szám lesz.

2. függvény

△ **Sztringkezelő**
Ebben a programban a felhasználó a vezérléshez egy karakterláncot ír be. Ez a függvény felbontja a karakterláncot kisebb részekre, és továbbítja őket a Teknőcirányítónak.

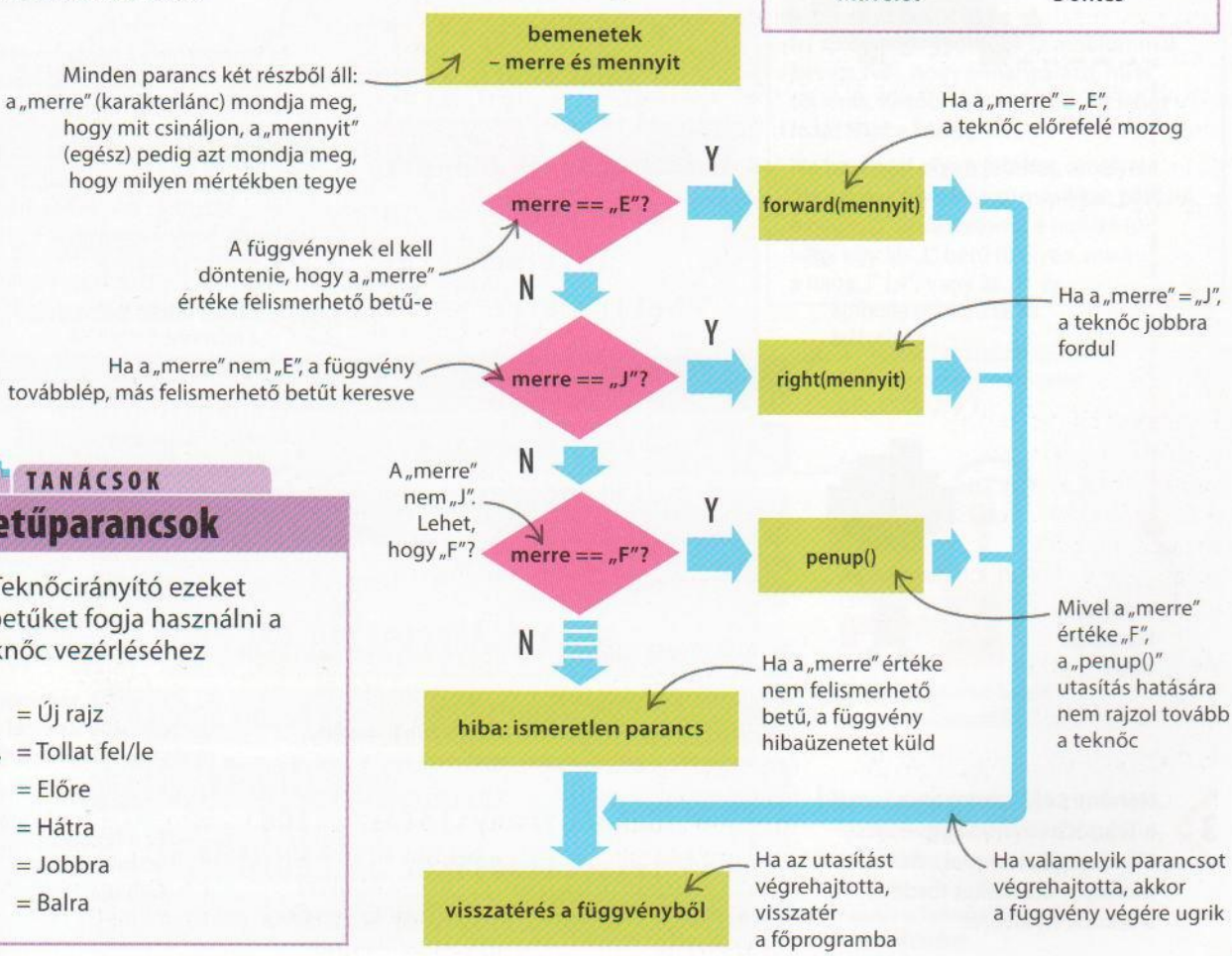
Főprogram

△ **Felhasználói felület**
A Sztringkezelőnek kell egy bemenetet kapnia. A Felhasználói felületen lehet beírni az utasításokat, amelyeket a Sztringkezelő feldolgoz.

Folyamatábra rajzolása

A programozók gyakran terveznek papíron, hogy később kevesebb hibával, jobb kódokat írjanak. A tervezés egyik módja a folyamatábra készítése, ezen ábrázoljuk a program lépéseit és elágazásait.

1 Ez a folyamatábra a Teknőcirányító függvényt mutatja. Ennek a bemenete egy betű („merre?") és egy szám („mennyit?"), a függvény ezeket alakítja teknőcutasításokká. Például az „E" és „100" átalakul „forward(100)" utasítássá. Ha a függvény nem ismeri fel a betűt, hibaüzenetet küld.



TANÁCSOK

Téglalapok és rombuszok

A folyamatábrák téglalapokból és rombuszokból állnak. A téglalapokban vannak a műveletek, a rombuszokban pedig a döntések.

Művelet Döntés

TANÁCSOK

Betűparancsok

A Teknőcirányító ezeket a betűket fogja használni a teknőc vezérléséhez

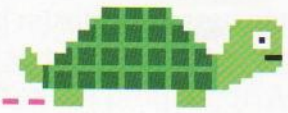
- **U** = Új rajz
- **F/L** = Tollat fel/le
- **E** = Előre
- **H** = Hátra
- **J** = Jobbra
- **B** = Balra

RAJZGÉP

A Teknőcirányító

A program első része egy függvény, amely a teknőcöt mozgatja, egyszerre egy paranccsal. A folyamatábrája az előző oldalon látható. Ez a kód lehetővé teszi, hogy a „merre" és a „mennyit" változókat teknőcutasításokká alakítsuk.

2 Ez a kód létrehozza a Teknőcirányító függvényt. A „merre" bemenetet iránnyá, a „mennyit" bemenetet pedig távolságra vagy a fordulás szögére fordítja le.



A teknőc kiindulási helye

```
from turtle import *
def teknőcirányító(merre, mennyit):
    merre = merre.upper()
    if merre == 'E':
        forward(mennyit)
    elif merre == 'H':
        backward(mennyit)
    elif merre == 'J':
        right(mennyit)
    elif merre == 'B':
        left(mennyit)
    elif merre == 'F':
        penup()
    elif merre == 'L':
        pendown()
    elif merre == 'U':
        reset()
    else:
        print('Ismeretlen parancs')
```

Betölti a teknőcutasításokat

A „merre" és „mennyit" változót definiálja mint a függvény paraméterét

Ez az utasítás a „merre" változó minden betűjét nagybetűvé alakítja

Ez mondja meg a függvénynek, hogy a „merre" változó „E" értékét fordítsa „forward" teknőcutasításra

Ahogy a folyamatábránál, itt is megnézzük az összes használható betűt, hogy megegyezik-e a „merre" változóval

Ez az utasítás elindítja a rajzolást

Ez az utasítás leállítja a rajzolást

Ez a parancs visszaállítja a teknőcöt a képernyő közepére

Ez az üzenet akkor jelenik meg, ha a függvény nem ismeri fel a „merre" változóban lévő betűt

3 Néhány példa, hogyan használd a Teknőcirányító függvényt. Minden alkalommal „merre, mennyit" értékeket fordít le a teknőc nyelvére.

```
>>> teknőcirányító('E', 100)
>>> teknőcirányító('J', 90)
>>> teknőcirányító('E', 50)
```

Meghívja a függvényt a nevével

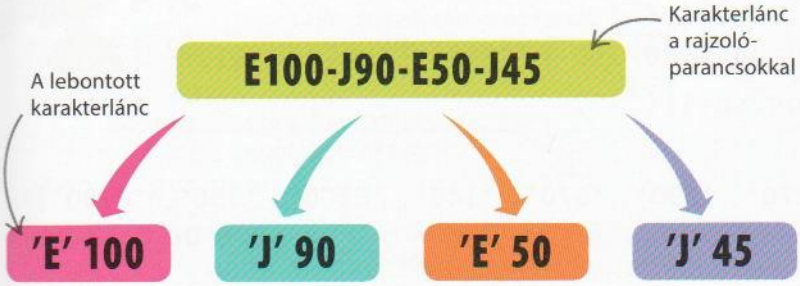
Ez a „merre" – „mennyit" bemenet 100 lépéssel mozgatja előre a teknőcöt

Ez a teknőcöt 90 fokkal jobbra forgatja

Írj pszeudokódot!

A tervezés másik módja, ha egy programot először pszeudokódban írunk meg. A „pszeudo" azt jelenti, hogy nem valódi, azaz álkód. Ez tehát csupán egy strukturált szöveg, amelybe ötleteket írunk.

4 Ideje megtervezni a Sztringkezelő függvényt. Ez a függvény több „merre" és „mennyit" bemenetet oszt szét betű-szám párokra, majd továbbítja őket a Teknőcirányítónak.



5 Ez a Sztringkezelő pszeudokódja. Segít elrendezni, megszerkeszteni az ötleteinket, anélkül hogy belebonyolódnánk a részletekbe.

```
függvény sztringkezelő(bemenet – a parancssorozat karakterláncként):
    a karakterlánc felosztása parancsokra
    minden parancsra:
        megvizsgáljuk, nem üres-e
            – ha üres, tovább a következő parancsra
        a parancs típusa az első betűből jön
        ha több karakter követi
            – alakítsuk őket számmá
        a teknőcirányító(parancs típusa, szám) függvény meghívása
```

A függvény karakterláncként kap egy parancssorozatot (pl. „E100-J90")

Felosztja a karakterláncot önálló parancsokra

Az üres parancs nem csinál semmit, ezért azt kihagyja

Felismeri az első betűt „merre" parancsként

Felismeri a következő karaktereket a „mennyit" értékeként

Átadja a Teknőcirányítónak a parancsokat

TANÁCSOK

Tiszta kódolás

Mivel nemcsak számítógépek olvassák a kódunkat, hanem mások is, fontos, hogy mindenki számára érthető, könnyen áttekinthető legyen:

Használj függvényeket a kód kisebb egységekre bontásához. Minden függvény egyetlen feladatért legyen felelős.

Adj olyan nevet a változóknak és a függvényeknek, amelyek tükrözik, hogy mire szolgálnak: a „kor_években" érthetőbb, mint a „ké".

Írj sok megjegyzést (# szimbólummal bevezetve), hogy elmagyarázd, mi is történik. Később könnyebben át lehet majd látni a kódot.

Ne használj olyan jeleket, amelyek összetéveszthetők egy másikkal: például a nagy „O" betű hasonlít a nullára (0), vagy egy kis „L" betű (l) olyan, mint a nagy „i" („I") vagy az „1"-es.

RAJZGÉP

A Sztringkezelő megírása

Az előző oldalon található pszeudokód a Sztringkezelő függvény terve, amelyik széttördeli a karakterláncot egyszerű parancsokká, és elküldi a Teknőcirányítónak. A következő lépés, hogy a pszeudokódot átírjuk valódi Python-kóddá, a „split()" függvény segítségével.

6 A „split()" függvény feloszt egy karakterláncot több kisebb karakterláncból álló listává. Minden töréspontot tagoló karakter jelöl (a mi programunkban „-").

Ez a karakterlánc tartalmazza a ház alakzatának parancsait

```
>>> parancsok = 'U-B90-E100-J45-E70-J90-E70-J45-E100-J90-E100'
>>> parancslista = parancsok.split('-')
>>> parancslista
['U', 'B90', 'E100', 'J45', 'E70', 'J90', 'E70', 'J45', 'E100', 'J90', 'E100']
```

A „split()" függvény egy lista elemeivé bontja a karakterláncot

7 Most írd át a pszeudokódot valódi Python-kóddá. Használd a „split()" függvényt a bemeneti karakterlánc felszeleteléséhez.

```
def sztringkezelő(parancsok):
    parancslista = parancsok.split('-')
    for parancs in parancslista:
        listahossz = len(parancs)
        if listahossz == 0:
            continue
        parancstípus = parancs[0]
        szám = 0
        if listahossz > 1:
            szám_szöveg = parancs[1:]
            szám = int(szám_szöveg)
        print(parancs, ':', parancstípus, szám)
        teknőcirányító(parancstípus, szám)
```

A karakterláncot a „-" jelek mentén felszeleteli

A ciklus végigmegy a lista minden elemén, és mindegyikből egy teknőcutasítás lesz

Megadja a parancs-karakter-lánc hosszát

Ha az utasítás hossza 0 (üres utasítás), akkor kihagyja, és a következőre ugrik

Az utasítás első karakterét (ne feledd, a számozás nullával kezdődik) beállítja a parancs típusának

Megvizsgálja, hogy több karakterből áll-e a parancs (szám része)

Veszi a megmaradt karaktereket úgy, hogy eldobja az elsőt

Átalakítja a karakt-tereket számokká

Kiírja az utasítást a képernyőre, így láthatod, hogy mit csinál a program

Továbbítja a parancsot a teknőcnek

8 Amikor a karakterlánc a ház alakzat rajzolását írja le, a Sztringkezelő ezt a kimenetet adja a terminálablakban:

```
>>> sztringkezelő('U-B90-E100-J45-E70-J90-E70-J45-E100-J90-E100')
U : U 0
B90 : B 90
E100 : E 100
J45 : J 45
E70 : E 70
J90 : J 90
E70 : E 70
J45 : J 45
E100 : E 100
J90 : J 90
E100 : E 100
```

Törli a képernyőt, és visszateszi a teknőcöt középre

A teknőc parancsait a „-" jelek tagolják

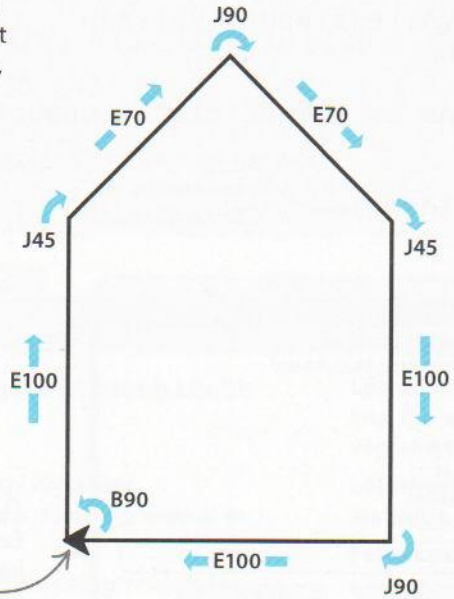
Az „E100" parancsból az „E" típus és „100" szám lett

Elfordítja a teknőcöt 45 fokkal, mielőtt megrajzolja a tetőt

Megrajzolja a tető jobb oldalát

A teknőc 90 fokkal jobbra fordul, hogy megrajzolja a ház alját

9 A Sztringkezelő függvény a karakterláncban érkezett összes utasítást beolvasta, azonosította és végrehajtotta. A házról rajzolt kép megjelenik az ablakban.



A program rajzoltat egy házat a teknőccel

JEGYEZD MEG!

Parancsok

Íme egy emlékeztető a program által felismert parancsokról. Néhány csak egy betűből áll, de van, amelyik számot is tartalmaz. Amikor meghívod a Sztringkezelő függvényt, mindaddig újra rajzol az ablakba, amíg nem kap „U" parancsot, és töröl mindent.

U	= Új rajz
F/L	= Tollat fel/le
E100	= Előre 100-at
H50	= Hátra 50-et
J90	= Jobbra 90 fokot
B45	= Balra 45 fokot

RAJZGÉP

Fejezd be a programot felhasználói felülettel!

A Rajzgéphez szükség van egy felületre, amely megkönnyíti a használatát. Ez lehetővé teszi a felhasználónak, hogy begépelje a karakterláncot.

10 Ez a kód létrehoz egy felugró ablakot, amelybe a felhasználó beírhatja a parancsokat. A „while True" ciklus segítségével mindig lehet újakat beírni.

A három félidézőjel azt jelzi a Pythonnak, hogy a lezáró három félidézőjelig minden egyetlen karakterláncnak számít, beleértve a sortöréseket is

```
tájékoztató = '''Írd be a parancsokat a teknőcnek:
pl. E100-J45-F-E100-B45-L-E100-J90-H50
U = Új rajz
F/L = Tollat fel/le
E100 = Előre 100-at
H50 = Hátra 50-et
J90 = Jobbra 90 fokot
B45 = Balra 45 fokot'''
képernyő = getscreen()
while True:
    teknőcfeladat = képernyő.textinput('Rajzgép', tájékoztató)
    print(teknőcfeladat)
    if teknőcfeladat == None or teknőcfeladat.upper() == 'VÉGE':
        break
    sztringkezelő(teknőcfeladat)
```

Segítség a különböző parancsok használatához

A karakterlánc vége

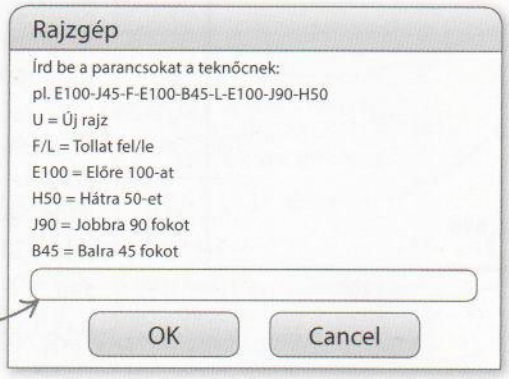
Megszerzi a felugró ablak létrehozásához szükséges adatokat

Ez a sor mondja meg, hogy mi legyen a felugró ablakban

Átadja a karakterláncot a Sztringkezelőnek

Leáll a program, ha a felhasználó beírja, hogy „VÉGE", vagy a „Cancel" gombra kattint

11 Ez az ablak fog megjelenni a teknőcablak előtt, ide kell beírni a parancsokat



Írd ide a parancsok sorozatát, majd kattints az „OK" gombra a futtatáshoz!

△ **A teknőc irányítása**

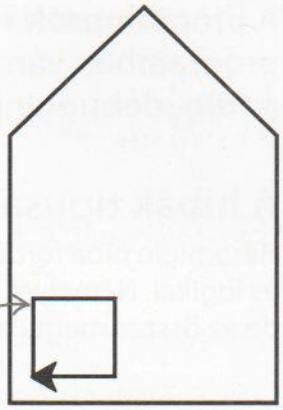
Ezzel a programmal könnyebben irányíthatod a teknőcöt, és nem kell újraindítani a programot, ha valami mást szeretnél rajzolni.

12 A Rajzgéppel nem csak alakzatok körvonalait lehet megrajzolni. Ha felemelt tollal mozgunk, az alakzaton belülre is tudunk rajzolni. Futtasd a programot, és írd be ezt a karakterláncot.

```
U-B90-E100-J45-E70-J90-E70-J45-E100-J90-E100-
H10-F-J90-E10-L-E30-J90-E30-J90-E30-J90-E30
```

Felemeli a tollat, így nem hagy nyomot mozgás közben

Leteszi a tollat, hogy megrajzolja az ablakot



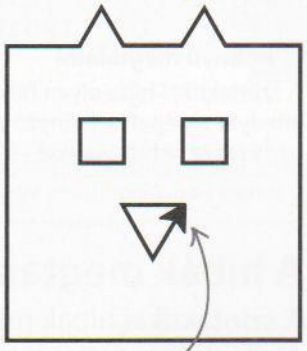
A háznak lett egy ablaka

Próbálj ki valami újat!

Már tudod, hogyan lehet részletgazdagabb rajzokat készíteni a Rajzgéppel. Próbáld megrajzolni ezt a bagolyfejet az alábbi utasítások segítségével!

```
U-E100-B90-E200-B90-E50-J60-E30-B120-E30-J60-E40-
J60-E30-B120-E30-J60-E50-B90-E200-B90-E100-B90-F-
E150-B90-E20-L-E30-B90-E30-B90-E30-B90-E30-J90-F-
E40-L-E30-J90-E30-J90-E30-J90-E30-B180-F-E60-J90-
L-E40-B120-E40-B120-E40
```

A szemek és a csőr megrajzolásához háromszor emeli fel a tollat



A nyíl mutatja, merre állt meg a teknőc. Ez azt jelenti, hogy a csőrt rajzolta meg utoljára

JEGYEZD MEG!

Amit megtanultál

Elkészítetted a Rajzgép programot, több kisebb cél elérésével:

Használtál folyamatábrát egy függvény megtervezéséhez, kidolgozva a döntési pontokat és a műveleteket.

Írtál pszeudokódot egy függvény megtervezéséhez, a valódi kód megírása előtt.

Létrehoztad a Teknőcirányító függvényt, amelyik kitalálja egy betűből és egy számból, hogy mit kell végrehajtania a teknőcnek.

Létrehoztad a Sztringkezelő függvényt, amely átalakítja a karakterláncot teknőcutasításokká.

Felhasználói felületet készítettél, hogy a felhasználó tudja hova beírni az utasításokat a program számára.

Hibák és hibakeresés

A programozók sem tökéletesek, elsőre a legtöbb programban van hiba. A hibát „bug"-nak, a hibakeresést pedig „debugging"-nak is nevezzük.

LÁSD MÉG
〈 94–95 Hibák
〈 122–123 Ciklusok a Pythonban
Hogyan tovább? **176–177 〉**

A hibák típusai

Háromféle hiba fordulhat elő a programokban: szintaktikai, futásidejű és logikai. Némelyeket könnyű észrevenni, másokat nehezebb, de az összes megtalálására és kijavítására vannak módszerek.

A Python-kulcsszó „for", nem pedig „fir"

```
fir i in range(5):
print(i)
```

△ **Könnyű megtalálni**
A szintaktikai hiba olyan hiba, amelyet elgépelés, hiányzó zárójel vagy rossz behúzás okoz.

Ez hibát fog okozni, mert nem lehet nullával osztani

```
a = 0
print(10 / a)
```

△ **Nehezebb megtalálni**
A futásidejű hibák csak a program futásakor jelennek meg. Ha számot adunk karakterlánchoz, vagy ha nullával osztunk.

A kor nem lehet egyszerre 5-nél kisebb és 8-nál nagyobb, ezért nem jár senkinek ingyenjegy

```
if kor < 5 and kor > 8:
    print('Ingyenjegy!')
```

△ **A legnehezebb megtalálni**
A logikai hibák a program gondolatmenetéből adódnak. Például „<" használata „>" helyett vagy kivonás helyett összeadás elvégzése.

A hibák megtalálása és javítása

A szintaktikai hibák megtalálása könnyű, mert az IDLE piros színnel jelöli őket, amikor futtatni kezdjük a programot. A futásidejű és a logikai hibák felfedezése több munkát igényel.

1 Problémás program
Ennek a programnak a célja, hogy összeadja a számokat 1-től a „max_szám" változó értékéig. A végén kiírja az összeget.

```
max_szám = 5
összeg = 0
for n in range(max_szám):
    összeg = összeg + n
print('A számok összege 1 és ', max_szám, ' között: ', összeg)
```

A legnagyobb szám, amelyet még hozzá kell adni

Ez a parancs kiír egy mondatot az eredménnyel

2 A kimenet
Az eredménynek 1+2+3+4+5-nek kellene lennie, de a program 10-et írt ki. Találd ki, hogy miért!

```
A számok összege 1 és 5 között: 10
```

Az összegnek 15-nek kellene lennie, nem 10-nek

3 A „print" és az „input()" hozzáadása

A program nem jelzi ki, mi történik az egyes lépések során. A „print" utasítással kiírjuk, hogy éppen mi történik. Az „input()" függvény pedig vár, amíg nem ütjük le az „Enter" billentyűt.

```
max_szám = 5
összeg = 0
for n in range(max_szám):
    összeg = összeg + n
    print('DEBUG: n = ', n, ' összeg = ', összeg)
    input()
print('A számok összege 1 és ', max_szám, ' között: ', összeg)
```

Ez a sor kiírja a ciklusváltozó pillanatnyi értékét és az aktuális összeget

4 Az új kimenet

A ciklus 0-tól 4-ig adja össze a számokat, nem pedig 1-től 5-ig. Ez azért van így, mert a „for" ciklus mindig 0-tól kezd számolni (hacsak nem utasítjuk másra), és leáll a megadott szám előtt eggyel.

```
DEBUG: n = 0 összeg = 0
DEBUG: n = 1 összeg = 1
DEBUG: n = 2 összeg = 3
DEBUG: n = 3 összeg = 6
DEBUG: n = 4 összeg = 10
A számok összege 1 és 5 között: 10
```

Ez a számok összege 0-tól 4-ig, nem pedig 1-től 5-ig

5 A hibás sor kijavítása

A ciklusnak 1-től „max_szám+1"-ig kell futnia, a ciklus így fogja összeadni a számokat 1-től 5-ig.

```
max_szám = 5
összeg = 0
for n in range(1, max_szám + 1):
    összeg = összeg + n
    print('DEBUG: n = ', n, ' összeg = ', összeg)
    input()
print('A számok összege 1 és ', max_szám, ' között: ', összeg)
```

Az új tartomány 1-től „max_szám"-ig fog terjedni (minthogy 1-gyel kisebb, mint a max_szám + 1)

6 A helyes kimenet

A „print" utasítás mutatja, hogy a program 1-től 5-ig adja össze a számokat, ezzel megadva a helyes választ. Kijavítottuk a hibát.

```
DEBUG: n = 1 összeg = 1
DEBUG: n = 2 összeg = 3
DEBUG: n = 3 összeg = 6
DEBUG: n = 4 összeg = 10
DEBUG: n = 5 összeg = 15
A számok összege 1 és 5 között: 15
```

„n = 3" esetén az összeg = 1+2+3

A helyes válasz van kiírva

Algoritmusok

Az algoritmus egy feladat elvégzésére szolgáló utasítások sorozata. Némely algoritmus hatékonyabb, kevesebb időráfordítást igényel, mint mások. Különféle algoritmusok vannak egyszerűbb feladatok elvégzésére (pl. számok rendezésére) is.

LÁSD MÉG

❮ **16–17** Gondolkodj úgy, mint egy számítógép!

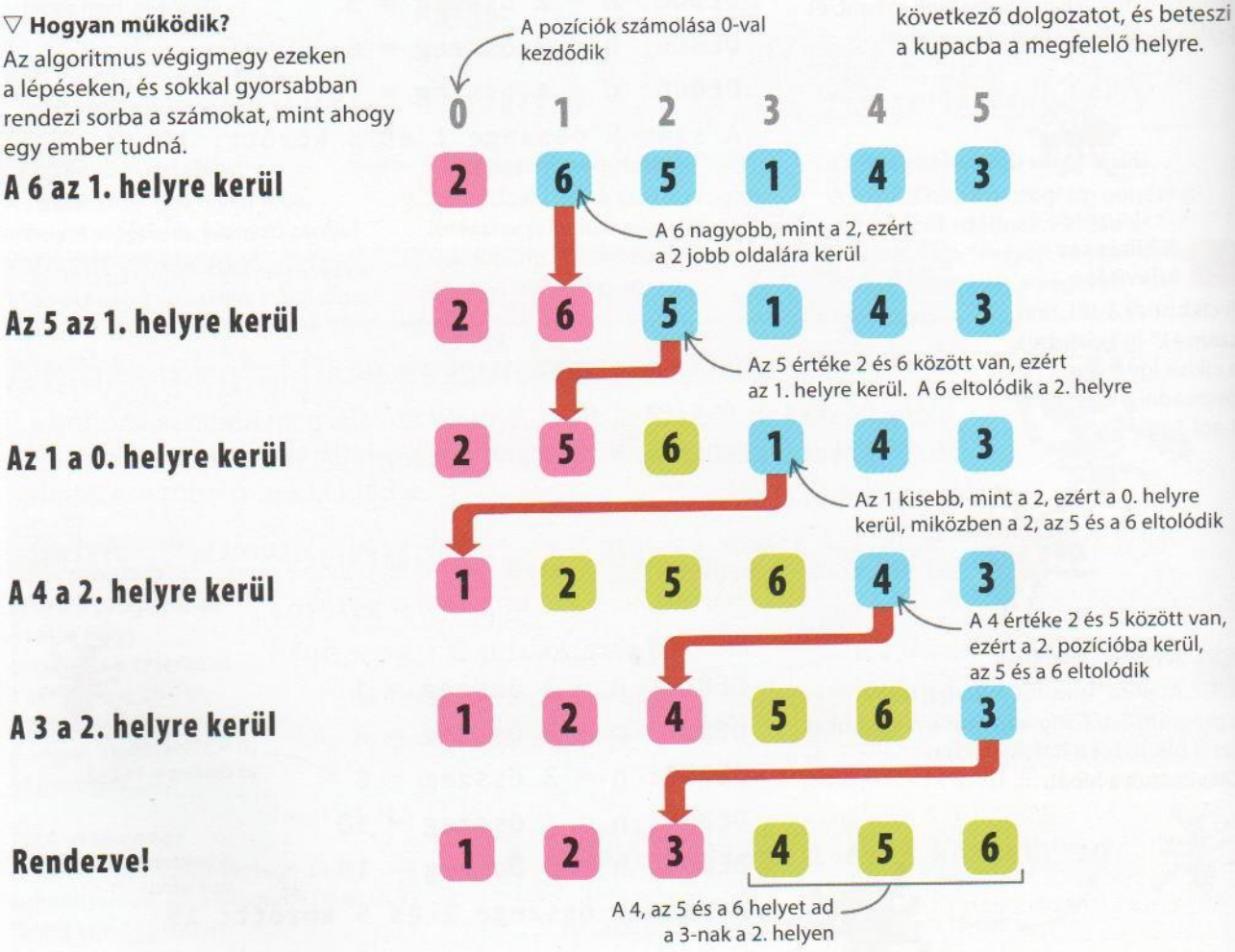
Könyvtárak **152–153** ❯

△ **Sorbarendezés**
A beszúrásos rendezés veszi a következő dolgozatot, és beteszi a kupacba a megfelelő helyre.

Beszúrásos rendezés

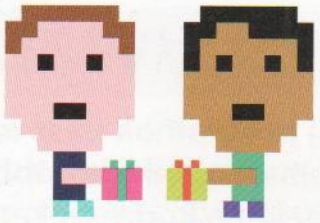
Képzeld el, hogy megkapod az osztályod dolgozatait, hogy rendezd őket pontszám szerint sorrendbe. A beszúrásos rendezésnél mindig a kupac tetején keletkezik egy rendezett rész, a még nem rendezett dolgozatokat egyenként helyezi a megfelelő helyre.

▽ **Hogyan működik?**
Az algoritmus végigmegy ezeken a lépéseken, és sokkal gyorsabban rendezi sorba a számokat, mint ahogy egy ember tudná.

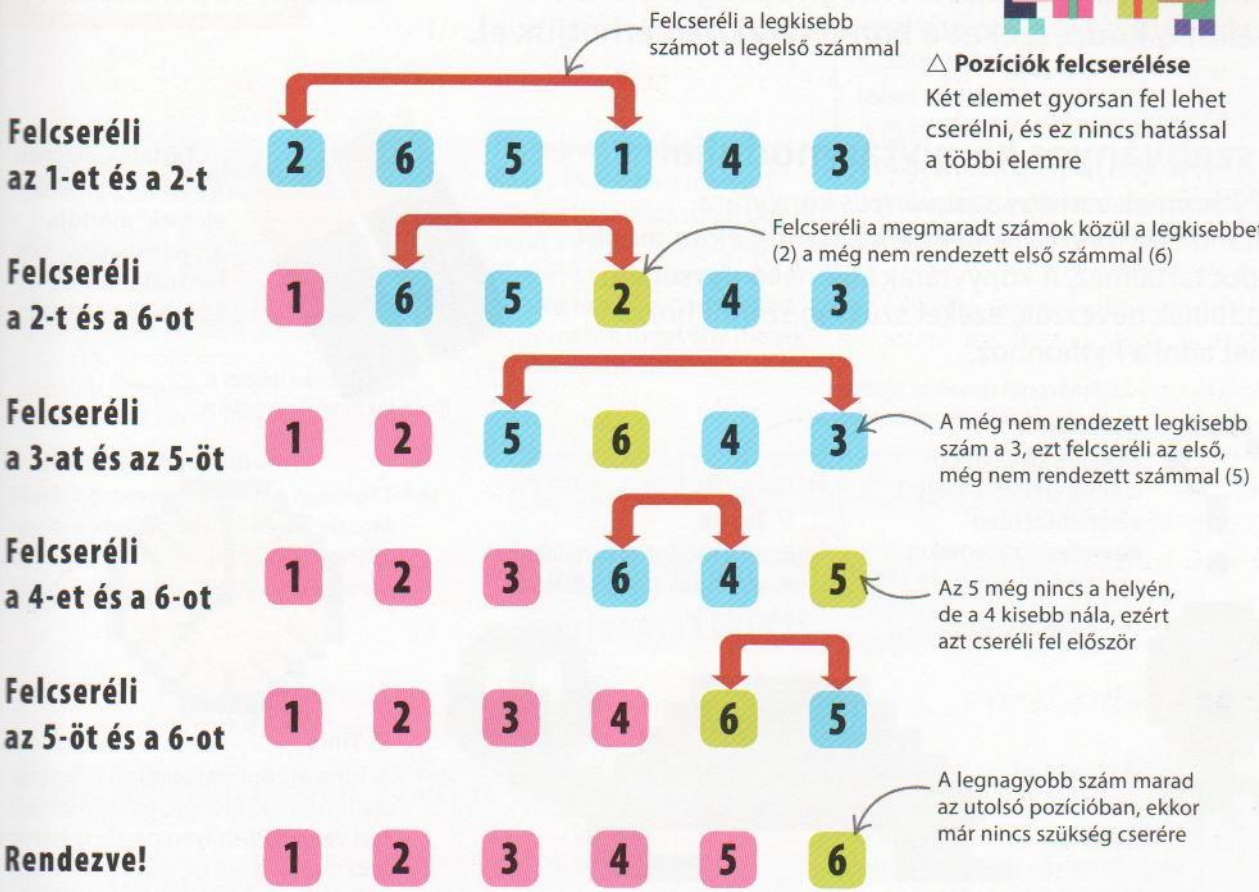


Kiválasztásos rendezés

A kiválasztásos rendezés másképp működik, mint a beszúrásos. Mindig csak két számot cserél fel, nem pedig eltolja a számokat. Egy lépésben mindig egy számot tesz a megfelelő helyre.



△ **Pozíciók felcserélése**
Két elemet gyorsan fel lehet cserélni, és ez nincs hatással a többi elemre



TANÁCSOK

Rendezés Pythonban

Sokféle rendezési algoritmus van, más-más előnyökkel és hátrányokkal. A Pythonban a „sort()" függvény a „Timsort" algoritmust használja – nevét tervezőjéről, Tim Petersről kapta. Ez két algoritmuson alapszik, a beszúrásos és az összefésüléses rendezésen. Írd be ezeket a sorokat, hogy lásd, miként működik!

Az „a" rendezetlen számok listája

```
>>> a = [4, 9, 3, 8, 2, 6, 1, 5, 7]
>>> a.sort()
>>> a
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Ez hívja meg a „sort()" függvényt

A számok az „a" listában már rendezettek

Könyvtárak

Új programok írása időbe telik, ezért célszerű felhasználni a korábban megírt programokat. Ezeket a kódrészeket a könyvtárakban érhetjük el.

LÁSD MÉG

Ablakok készítése **154–155 ❯**

Színek és koordináták **156–157 ❯**

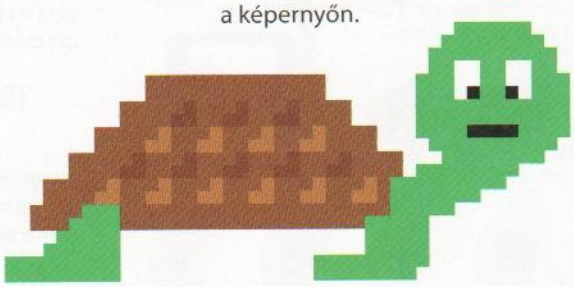
A szabványos könyvtár moduljai

A Pythonnak van egy szabványos könyvtára („Standard Library"), amely sok hasznos, előre megírt kódot tartalmaz. A könyvtárak különálló egységeit modulnak nevezzük, ezeket szükség szerint hozzá lehet adni a Pythonhoz.

◁ **Tartalék elemek**
A Python „tartalék elemek" mottója azt jelenti, hogy sok használatra kész kód tartozik hozzá.

◁ **Random**
Ez a modul véletlen számot generál, vagy véletlenszerűen átrendezi az elemeket.

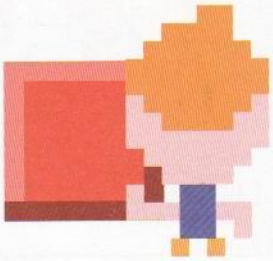
▽ **Turtle**
Ezzel a modullal vonalakat és alakzatokat rajzolhatunk a képernyőn.



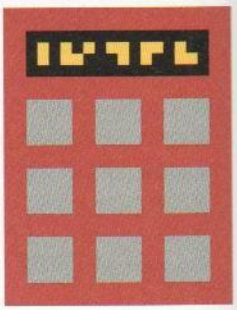
△ **Time**
A Time modul visszaadja a jelenlegi dátum és idő értékét, és számolni is tud velük. (Pl. milyen nap lesz három nap múlva?)

△ **Socket**
E modul segítségével kapcsolatot létesíthetünk számítógépek között hálózaton vagy az interneten.

▽ **Tkinter**
Ezzel a modullal gombokat, ablakokat és grafikákat készíthetünk interaktív programokhoz.



▷ **Math**
Ezzel a modullal bonyolultabb matematikai műveleteket lehet elvégezni.



Modulok betöltése

Használat előtt be kell tölteni az adott modult, ezzel új parancsok válnak elérhetővé. Modulok betöltésére az „import" utasítást kell használni. Ezt többféleképpen is megteheted a Pythonnal.

```
import random

random.randint(1, 6)
random.choice(lista)
```

A modul neve a függvény neve elé kerül

◁ **„import random"**
Ha így töltöd be a modult, akkor a használat során be kell írnod a modul nevét a függvény elé. Így könnyebben olvasható a kód, mert tudod, hogy melyik függvény melyik modulhoz tartozik.

▷ **„from random import *"**
Kisebb programok esetén hasznos lehet így betölteni, nagyobbaknál viszont zavaró, ha nem tudod, hogy melyik függvény melyik modulhoz tartozik.

Betölti az összes függvényt a Random modulból

```
from random import *

randint(1, 6)
choice(lista)
```

Ez a kód nem mutatja, hogy melyik modulhoz tartozik a függvény

Csak a „randint" függvény betöltése

```
from random import randint

randint(1, 6)
```

Csak a „randint" függvény érhető el

◁ **„from random import randint"**
Betölthetsz egyetlen függvényt is a modulból. Ha csak egy függvényt szeretnél használni egy modulból, akkor ez hatékonyabb megoldás az előzőnél.

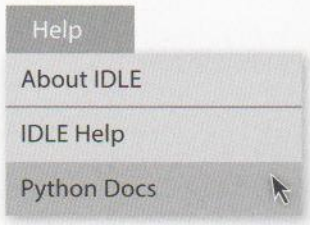
TANÁCSOK

Pygame

A Pygame nevű Python-könyvtár segítségével videojátékokat lehet készíteni. A Pygame-hez tartozik hang- és egy külön grafikai modul is. Jól használható, ha megértetted az alapokat ebből a könyvből.

Súgó és dokumentáció

Nem vagy biztos benne, hogy melyik modult használd, vagy milyen függvények elérhetők? A „Python Library Reference" tartalmaz minden lényeges információt. Kattints arra a könyvtárra, amelyikről többet szeretnél tudni.



◁ **Súgó**
Az IDLE-ablakok felső részén található a „Help" menü, amelynek a „Python Docs" menüpontja előhív egy ablakot, ahol rengeteg hasznos információt találsz (csak angolul).

Ablakok készítése

Sok program használ ablakokat és gombokat a vezérléshez. Ezek az elemek építik fel a grafikus felhasználói felületet, vagyis a GUI-t.

LÁSD MÉG
Színek és koordináták 156–157 ❯
Alakzatok rajzolása 158–159 ❯
Változtatások 160–161 ❯

Készíts egyszerű ablakot!

A GUI készítésének első lépése, hogy létrehozol egy ablakot, amelybe minden bekerül. Ehhez szükséged lesz a Tkinterre (a Python szabványos könyvtárából).

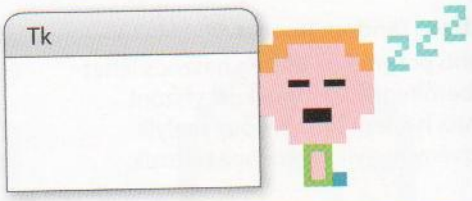
1 Írd be a kódot!
Ez tölti be a Tkinter modult, és készít egy új ablakot. A Tkintert be kell tölteni, mielőtt használnád.

```
from tkinter import *
ablak = Tk()
```

Ez tölti be a Tkinter modult

Ez készít egy Tkinter-ablakot

2 Megjelenik a Tkinter-ablak
Ha futtatod a kódot, megjelenik egy ablak. Elsőre kissé fakónak tűnhet, ám ez még csak az első lépése a GUI-nak.



Adj gombokat az ablakhoz!

Tedd a GUI-t interaktívabbá gombok hozzáadásával! Más-más üzenet jelenik meg attól függően, hogy melyik gombra kattintasz.

1 Készíts két gombot!
Írd be ezt a kódot, hogy hozzáadj két gombot az ablakhoz.

```
from tkinter import *
def Agomb():
    print('Köszönöm!')
def Bgomb():
    print('Jaj! Ez fájt!')
ablak = Tk()
gomb_A = Button(ablak, text='Nyomj meg!', command=Agomb)
gomb_B = Button(ablak, text='Ne nyomj meg!', command=Bgomb)
gomb_A.pack()
gomb_B.pack()
```

Ez az üzenet jelenik meg, ha az A gombra kattintunk

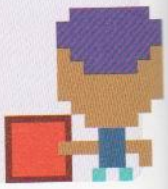
Ez az üzenet jelenik meg, ha a B gombra kattintunk

Ez a szöveg jelenik meg az A gombon

Ez mondja meg, hogy melyik gomb lenyomásakor melyik függvény fusson le

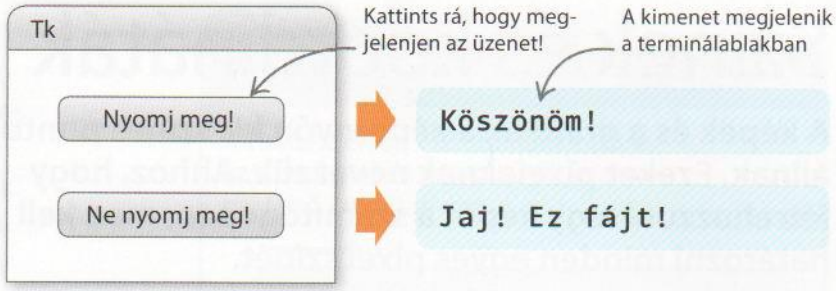
Ezek a sorok jelenítik meg a gombokat az ablakban

Ez a szöveg jelenik meg a B gombon



2 Kattints a gombokra, hogy kiírja az üzenetet!

A program futtatásakor megjelenik egy ablak két gombbal. A gombokra kattintva üzenetek jelennek meg a terminálablakban. Készítettél egy interaktív GUI-t, amely reagál a felhasználó parancsaira.



Dobj a dobókockával!

A Tkinterrel készíthetsz GUI-t egy egyszerű alkalmazáshoz. Az alábbi kód dobókockával való dobást szimulál.

1 Készíts dobókocka-szimulátort!

Ez a program létrehoz egy gombot, amelyre rákattintva a „dob()" függvény segítségével megjelenik egy 1 és 6 közötti szám.

```
from tkinter import *
from random import randint
def dob():
    szöveg.delete(0.0, END)
    szöveg.insert(END, str(randint(1,6)))
ablak = Tk()
szöveg = Text(ablak, width=1, height=1)
gomb_A = Button(ablak, text='Nyomd meg!', command=dob)
szöveg.pack()
gomb_A.pack()
```

Betölti a „randint" parancsot a Random modulból

Ez a kód kitörli a szöveget, és helyettesíti egy új, 1 és 6 közötti számmal

Készít egy szövegdobozt a véletlen számnak

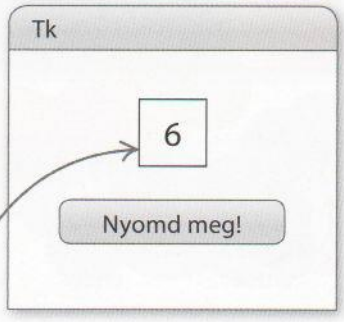
Ez határozza meg, hogy melyik függvény fusson a gombra kattintáskor

Ez teszi a szövegdobozt és a gombot az ablakba

Ez a szöveg jelenik meg a gombon

2 Nyomd meg a gombot a dobáshoz!

Futtasd a programot, majd kattints a gombra a kockadobáshoz, és nézd meg az eredményt. Könnyen átalakíthatod 12 oldalú kockával való dobásra, illetve fej vagy írás játékra.



TANÁCSOK

Világos és egyszerű

Amikor GUI-t tervezel, próbáld nem összezavarni a felhasználót túl sok gombbal. A gombokra írt szövegek pedig legyenek egyértelműek!

Színek és koordináták

A képek és a grafikák a képernyőn kis színes pontokból állnak. Ezeket pixeleknek nevezzük. Ahhoz, hogy létrehozzunk egy képet a számítógépen, meg kell határozni minden egyes pixel színét.

LÁSD MÉG

‹ 154–155
Ablakok készítése

Alakzatok rajzolása
158–159 ›

Változtatások
160–161 ›

Színek kiválasztása

Úgy kell meghatároznunk a színeket, hogy a számítógép megértse. A Tkinter tartalmaz egy hasznos eszközt, amely segít ebben.

1 Indítsd el a színválasztó eszközt!
Írd ezt a kódot a terminálablakba, hogy megnyisd a Tkinter színválasztó eszközét.

Ez betölti az összes Tkinter-függvényt

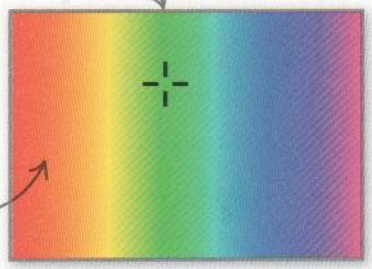
```
>>> from tkinter import *
>>> t = Tk()
>>> colorchooser.askcolor()
```

Betölti a színválasztó ablakot

2 Válassz egy színt!
A színválasztó ablak megjelenik. Válaszd ki a megfelelő színt, és kattints az „OK" gombra!

Kattints rá a kiválasztandó színre!

Ez az ablak megkönnyíti a színválasztást



3 A színértékek
Amikor kiválasztasz egy színt, számok egy listája jelenik meg a terminálalakban. Ezek a számok jelentik a piros, zöld és kék színek mennyiségét, amelyekből össze lehet keverni a választott színt.

```
((60.234, 190.742, 52.203), '#3cbe34')
```

A piros értéke — A zöld értéke — A kék értéke — A színkód hexadecimális (16-os számrendszerbeli) értéke (*lásd* 182–183. o.)

TANÁCSOK

Színkeverés

Minden pixel meghatározható piros, zöld és kék színekkel. Ezeket a színeket összekeverve bármilyen szín előállítható.

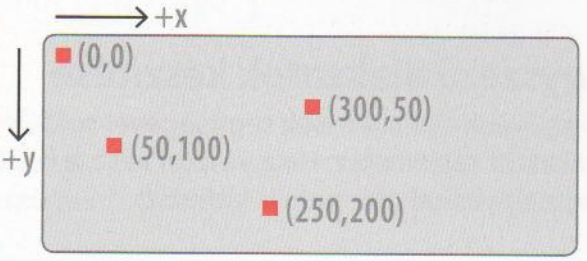
Rajzolás vászonra

Ahhoz, hogy grafikákat készítsünk Pythonban, létre kell hozni egy üres területet, ahova rajzolhatunk. Ezt a területet vászonnak nevezzük. Az x és y koordináták megadásával pontosan meghatározhatjuk, hogy hova szeretnénk rajzolni.

TANÁCSOK

Koordináták

A Tkinterben az x koordináta jobbra, az y pedig lefelé nő, az origó a vászon bal felső sarkában van



1 **Készíts grafikai programot!**

Ez a kód megnyit egy ablakot, benne egy vászonnal, majd véletlenszerűen elhelyezett köröket rajzol rá.

```
from random import *
from tkinter import *
méret = 500
ablak = Tk()
vászon = Canvas(ablak, height=méret, width=méret)
vászon.pack()
while True:
    szín = choice(['pink', 'orange', 'purple', 'yellow'])
    x0 = randint(0, méret)
    y0 = randint(0, méret)
    d = randint(0, méret/5)
    vászon.create_oval(x0, y0, x0 + d, y0 + d, fill=szín)
    ablak.update()
```

Betölti az összes függvényt a Random modulból

Betölti az összes Tkinter-függvényt

A „méret" változó beállítja a vászon méreteit

Készít egy vásznat az ablakon belül

Végtelen ciklus készíti a köröket

Választ egy véletlen színt a listából

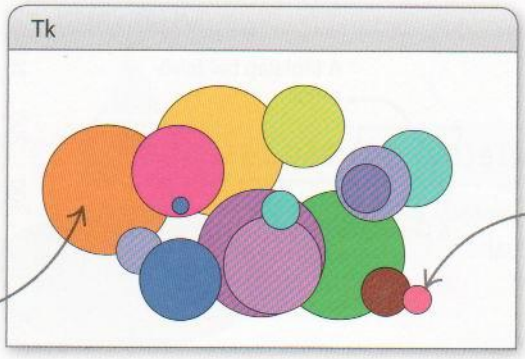
Készít egy kört véletlenszerű méretben a vászon véletlenszerű helyén

Ez a rész rajzolja meg a kört

Ez a rész színezi ki a kört a kiválasztott színnel

2 **Színezett vászon**

Futtasd a programot! Köröket fog rajzolni a vászonra.



A körök véletlenszerű helyre kerülnek

A körök mérete is véletlenszerű

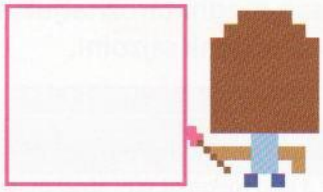
Alakzatok rajzolása

A Tkinterrel nemcsak ablakokat, gombokat és színes grafikákat lehet hozzáadni a programokhoz, hanem alakzatokat is lehet rajzolni.

LÁSD MÉG

‹ 160–161
Változtatások

Események kezelése
162–163 ›



Egyszerű alakzatok készítése

Téglalapok és ellipszisek segítségével sokféle alakzatot rajzolhatsz. Ha a vászon kész, a következő függvényekkel már neki is láthatsz.

```
>>> from tkinter import *
>>> ablak = Tk()
>>> rajz = Canvas(ablak, height=500, width=500)
>>> rajz.pack()
>>> téglalap1 = rajz.create_rectangle(100, 100, 300, 200)
>>> négyzet1 = rajz.create_rectangle(30, 30, 80, 80)
>>> ellip1 = rajz.create_oval(100, 100, 300, 200)
>>> kör1 = rajz.create_oval(30, 30, 80, 80)
```

Beállítja a vászon méretét

Készít egy vásznat, hogy rajzolhassunk rá

Beállítja a téglalap helyét és méretét koordináták segítségével (*lásd* alább)

Téglalapot rajzol

A négyzet olyan téglalap, amelynek egyenlő hosszúak az oldalai

Kört rajzol

Beállítja a kör helyét és méretét

Rajzolás koordinátákkal

Koordinátákkal mondjuk meg a számítógépnek, hogy pontosan hol legyenek az alakzatok. Az első szám (x) mutatja, hogy mennyit menjen jobbra, a második (y) pedig, hogy mennyit lefelé az origótól.

A vászon neve

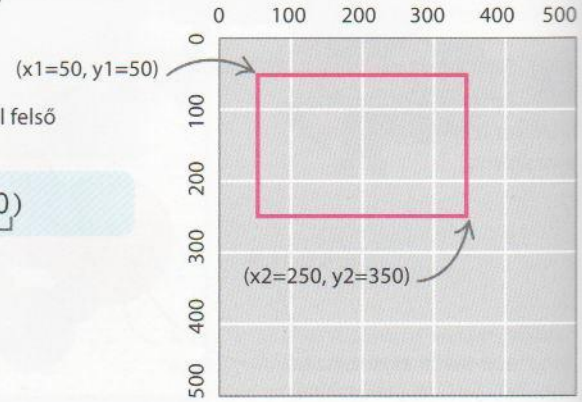
A téglalap bal felső sarka

```
>>> rajz.create_rectangle(50, 50, 250, 350)
```

A téglalap jobb alsó sarka

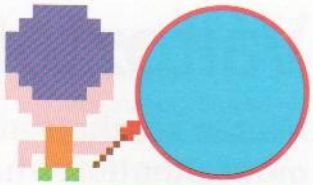
△ A koordináták beállítása
Az első két szám adja meg a téglalap bal felső sarkának koordinátáit, a második kettő pedig a jobb alsó sarokét.

▽ A koordináta-rendszer
A téglalap bal felső sarkának koordinátái (50, 50), a jobb alsó saroké pedig (250, 350).



Színek hozzáadása alakzatokhoz

Készíthetsz színes alakzatokat is. Ezeknél külön kell beállítani a körvonal („outline"), valamint a kitöltés („fill") színét.



Készít egy kék színnel kitöltött kört piros körvonallal!

```
>>> rajz.create_oval(30, 30, 80, 80, outline='red', fill='blue')
```

Földönkívüli rajzolása

Alakzatok kombinálásával szinte bármit megrajzolhatsz. Íme egy példa, hogy ellipszisek, vonalak és háromszögek segítségével hogyan rajzolhatsz földönkívülit.

1 Rajzolj földönkívülit!

A földönkívüli minden részéhez meg kell adni az alakzat formáját, méretét, helyzetét a vásznon, valamint a színét. Minden alakzat saját azonosítóval rendelkezik, ezeket változóban tárolhatjuk.

```
from tkinter import *
ablak = Tk()
ablak.title('A földönkívüli')
v = Canvas(ablak, height=300, width=400)
v.pack()
test = v.create_oval(100, 150, 300, 250, fill='green')
szem = v.create_oval(170, 70, 230, 130, fill='white')
szemgolyó = v.create_oval(190, 90, 210, 110, fill='black')
száj = v.create_oval(150, 220, 250, 240, fill='red')
nyak = v.create_line(200, 150, 200, 130)
kalap = v.create_polygon(185, 75, 215, 75, 200, 40, fill='blue')
```

Beállítja az ablak nevét „A földönkívüli"-re

Elkészíti a vásznat

Zöld ellipszist rajzol a testnek

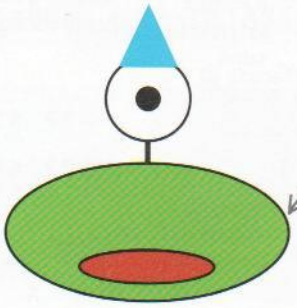
Fekete pöttyöt rajzol a szem közepére

Piros ellipszist rajzol a szájnak

Kék háromszöget rajzol a földönkívüli kalapjának

2 Találkozz a földönkívülivel

Futtasd a kódot a földönkívüli megrajzolásához. Zöld színű a teste, egy szeme van, piros a szája, és még egy szép kék kalapot is visel.



A kész földönkívüli

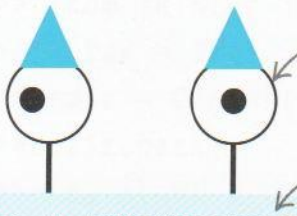
Változtatások

Miután készítettünk egy rajzot a vászonra, később még módosíthatjuk. Írhatunk kódot a rajz megváltoztatására vagy mozgatására.

LÁSD MÉG
❮ 158–159
Alakzatok rajzolása
Események kezelése
162–163 ❯

Alakzatok mozgatása

Egy alakzat mozgatásához meg kell mondani a számítógépnek (névvel vagy azonosítóval), hogy melyik alakzatot szeretnénk mozgatni, és hova.



A szemgolyó balra fordul, majd vissza

Az alakzat neve vagy azonosítója

```
>>> v.move(szemgolyó, -10, 0)
>>> v.move(szemgolyó, 10, 0)
```

Ez a függvény mozgatja az alakzatot

Beállítja a mozgás koordinátáit

◁ **A szemgolyó mozgatása**
Írd ezt a kódot a terminálablakba! Ettől a szemgolyó balra fordul, majd vissza.

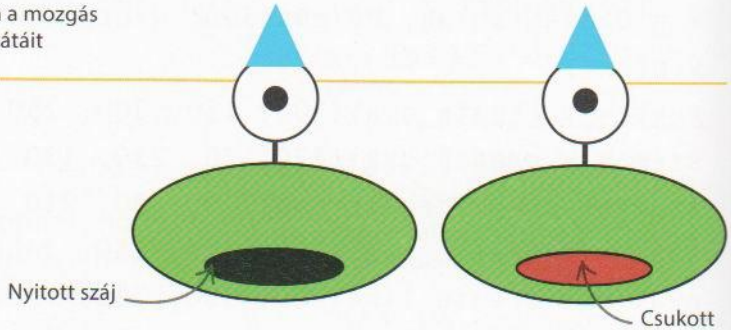
JEGYEZD MEG!

Beszédes nevek

Érdemes kifejező nevet adni az alakzatoknak, így később könnyebben azonosíthatod őket. Itt például a „szemgolyó" és a „száj" beszédes nevek.

Színek változtatása

A színek változtatásával kinyithatod és becsukhatod a földönkívüli száját.



Nyitott száj

Csukott száj

1 Írd be a kódot!
Írd be ezt a két függvényt! Ezek fogják kinyitni és becsukni a szájat.

Az „itemconfig()" függvény egy már létező alakzat tulajdonságait változtatja meg

A nyitott száj fekete színű

```
def szájat_nyit():
    v.itemconfig(száj, fill='black')
def szájat_csuk():
    v.itemconfig(száj, fill='red')
```

Az alakzat neve

A csukott száj piros színű

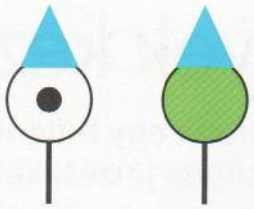
2 Nyitás és zárás
Írd ezt a kódot a terminálablakba, hogy mozgasd a száját!

```
>>> szájat_nyit()
>>> szájat_csuk()
```

Írd ezeket a parancsokat a terminálablakba, hogy kinyisd és becsukd a földönkívüli száját!

Bújócska

Alakzatokat el is lehet tüntetni az „itemconfig()" függvénnyel. Ha eltünteted a szemgolyót, majd kicsit később újra megjeleníted, olyan, mintha pislogna a földönkívüli.



◁ **Pislogó földönkívüli**
A földönkívüli pislogásához tüntesd el a szemgolyót, a szeme fehérjét pedig változtasd zöldre.

1 **Készíts pislogófüggvényeket!**
Ez a két függvény szükséges a pislogáshoz.

Zöldre változtatja a szem fehérjét

Az alakzat neve

```
def pislog_csuk():
    v.itemconfig(szem, fill='green')
    v.itemconfig(szemgolyó, state=HIDDEN)
def pislog_nyit():
    v.itemconfig(szem, fill='white')
    v.itemconfig(szemgolyó, state=NORMAL)
```

A szeme újra fehér lesz

Eltünteti a szemgolyót

A szemgolyó újra látható

2 **Pislogás**
Írd ezt a kódot a terminálablakba, és a földönkívüli pislogni fog.

```
>>> pislog_csuk()
>>> pislog_nyit()
```

A „pislog_nyit()" függvény nyitja ki újra a szemet

Beszéltetés

Szöveget is meg lehet jeleníteni a képernyőn, mintha beszélne a földönkívüli. A felhasználó parancsaira különböző válaszokat adhat.

1 **Szöveg hozzáadása**
Ez a kód szöveget ad a grafikához, és létrehozza a függvényt, amelyik ellopja a kalapját.

Elhelyezi a szöveget a vásznon

Írd az idézőjelbe, hogy mit mondjon a földönkívüli!

```
szavak = v.create_text(200, 280, text='Földönkívüli vagyok!')
def kalapot_lop():
    v.itemconfig(kalap, state=HIDDEN)
    v.itemconfig(szavak, text='Add vissza a kalapom!')
```

Eltünteti a kalapot

Amint eltűnik a kalapja, a földönkívüli vissza fogja kérni

Új üzenet jelenik meg, amikor eltűnik a kalap

2 **Lopd el a kalapot!**
Írd ezt a terminálablakba, és nézd meg, mi történik!

Írd ezt a kalap ellopásához!

```
>>> kalapot_lop()
```

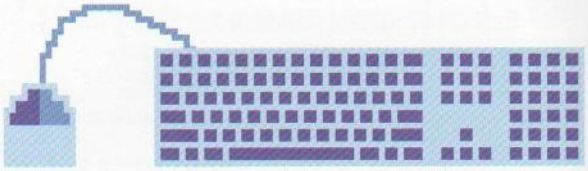
Események kezelése

Arról, hogy lenyomunk egy billentyűt vagy mozgatjuk az egeret, a számítógép jelzéseket kap. Ezt eseménynek nevezzük. A programok utasíthatják a számítógépet, hogy reagáljon ezekre az eseményekre.

LÁSD MÉG
❮ 158–159 Alakzatok rajzolása
❮ 160–161 Változtatások

Az események elnevezése

Számos eszközzel (egérrel, billentyűzettel stb.) kiválthatunk eseményeket. A Tkinterben az ilyen eseményeknek nevük van.



Az egér eseményei

- **<Button-1>** – Kattintás bal gombbal
- **<Button-3>** – Kattintás jobb gombbal

A billentyűzet eseményei

- **<Right>** – A jobbra nyíl lenyomása
- **<Left>** – A balra nyíl lenyomása
- **<space>** – A szóköz lenyomása
- **<Up>** – A fel nyíl lenyomása
- **<Down>** – A le nyíl lenyomása
- **<KeyPress-a>** – Az „A" billentyű lenyomása; Más betűket is írhatunk ide

Az egér eseményei

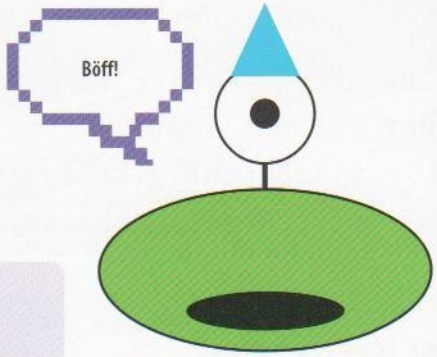
Ahhoz, hogy egy program válaszoljon egy eseményre, egyszerűen csak hozzá kell kapcsolni az eseményhez egy függvényt. Itt a „büfi" függvény van hozzárendelve a „<Button-1>" eseményhez.

```
ablak.attributes('-topmost', 1)
def büfi(event):
    szájat_nyit()
    v.itemconfig(szavak, text='Böff!')
v.bind_all('<Button-1>', büfi)
```

Ez hozza előre a Tkinter-ablakot a képernyőn

Létrehozza a „büfi" függvényt

Hozzákapcsolja a bal kattintást a „büfi" függvényhez



△ **Böfögő földönkívüli**
Kattints a bal gombbal, és a hozzárendelt függvény miatt a földönkívüli kienged egy büfit.

A billentyűzet eseményei

Függvények ugyanígy a billentyűkhöz is hozzárendelhetők. Írd be az alábbi kódot, hogy a pislogást az „A" és a „Z" billentyűk vezéreljék.

```
def pislog_csuk2(event):
    v.itemconfig(szem, fill='green')
    v.itemconfig(szemgolyó, state=HIDDEN)
def pislog_nyit2(event):
    v.itemconfig(szem, fill='white')
    v.itemconfig(szemgolyó, state=NORMAL)
v.bind_all('<KeyPress-a>', pislog_csuk2)
v.bind_all('<KeyPress-z>', pislog_nyit2)
```

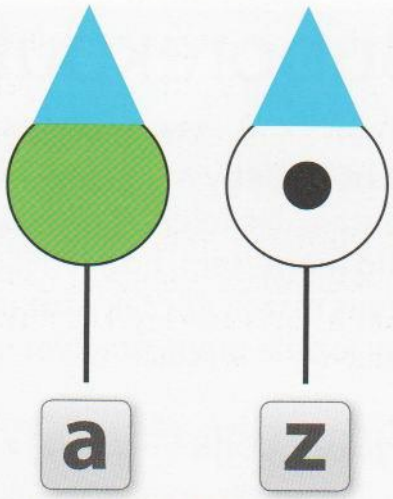
Zöld lesz a szeme (csukva)

Eltűnik a szemgolyó

Megjelenik a szemgolyó

Ez a parancs függvényt kapcsol eseményhez

Ez kapcsolja össze a „pislog_csuk2" függvényt a „Z" lenyomását



△ **Pislogó földönkívüli**
Amikor ez a kód fut, az „A" gomb lenyomására becsukja a szemét, a „Z" lenyomására pedig kinyitja.

Mozgatás nyilakkal

Billentyűk lenyomásával mozgatni is tudunk dolgokat. Ez a kód rendeli hozzá a nyilakhoz a földönkívüli szemgolyóját mozgató függvényeket.

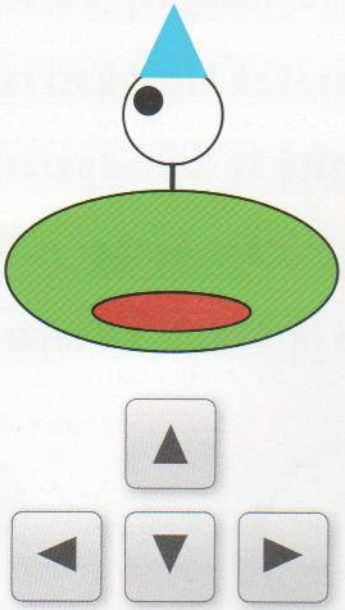
```
def szemet_irányít(event):
    bill = event.keysym
    if bill == "Up":
        v.move(szemgolyó, 0, -1)
    elif bill == "Down":
        v.move(szemgolyó, 0, 1)
    elif bill == "Left":
        v.move(szemgolyó, -1, 0)
    elif bill == "Right":
        v.move(szemgolyó, 1, 0)
v.bind_all('<Key>', szemet_irányít)
```

Ez a parancssor figyeli, hogy melyik billentyű van lenyomva

A szemgolyó felfelé mozog, ha a fel nyilat nyomjuk le

A szemgolyó balra mozog, ha a balra nyilat nyomjuk le

Bármelyik gomb megnyomása aktiválja a „szemet_irányít" függvényt



△ **A szemgolyó irányítása**
A szemgolyó abba az irányba mozog, amelyik nyilat lenyomjuk.

7. PROJEKT

Buborékpukkasztó

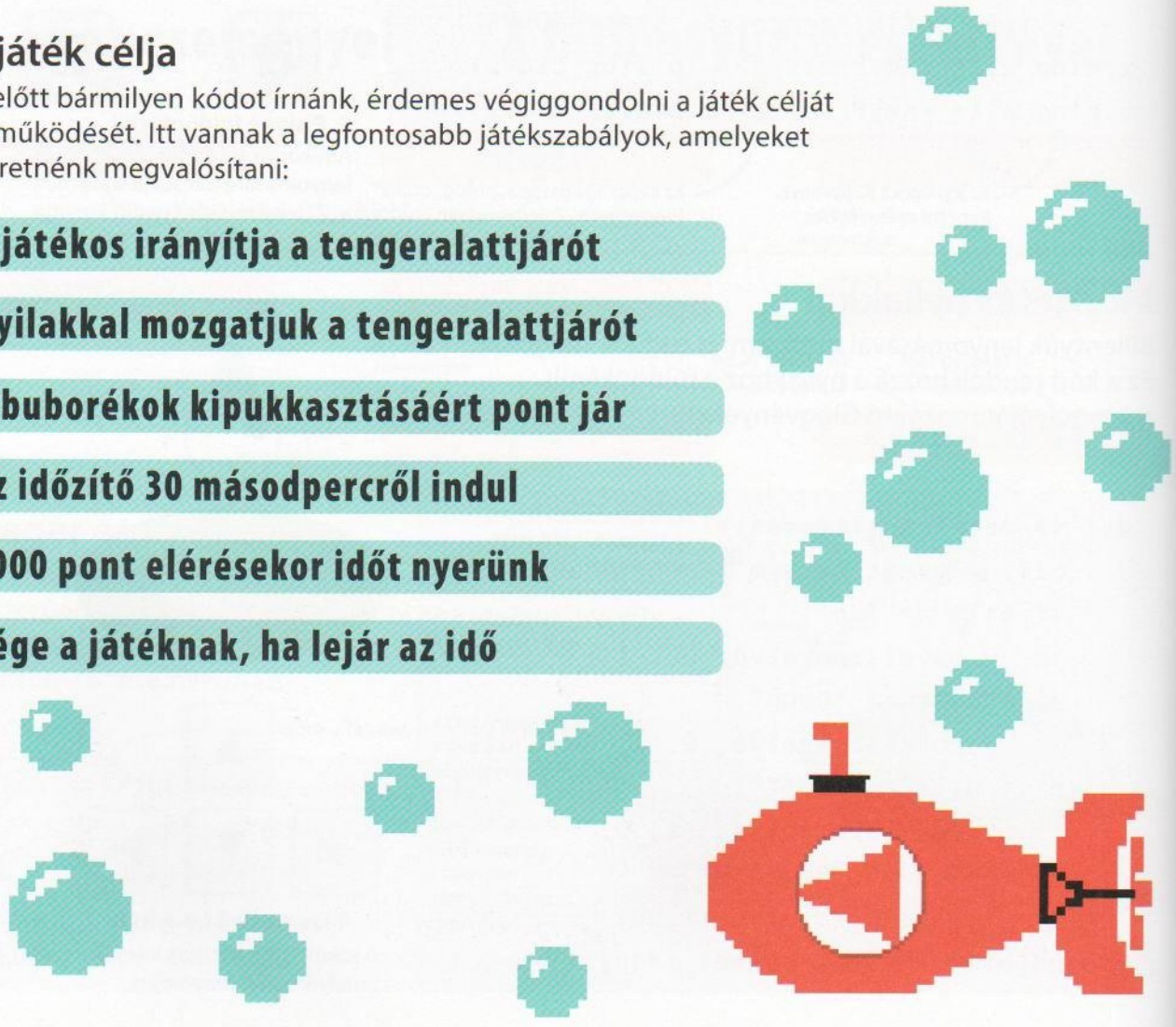
LÁSD MÉG
❮ 154–155 Ablakok készítése
❮ 156–157 Színek és koordináták
❮ 158–159 Alakzatok rajzolása

Ebben a projektben mindent felhasználunk az eddig tanultakból. Ez egy nagy projekt, ezért mindenképp részekre kell bontani. Ne felejtsd el gyakran elmenteni a munkád! Próbáld megérteni, hogyan illeszkednek egymáshoz a program egyes részei, és csak ezután lépj tovább! A végeredmény egy jópofa játék, amelyet megmutathatsz a barátaidnak is.

A játék célja

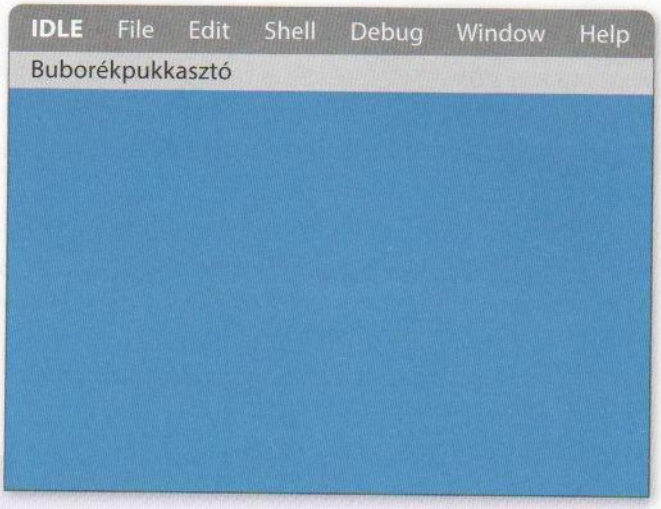
Mielőtt bármilyen kódot írnánk, érdemes végiggondolni a játék célját és működését. Itt vannak a legfontosabb játékszabályok, amelyeket szeretnénk megvalósítani:

- **A játékos irányítja a tengeralattjárót**
- **Nyilakkal mozgatjuk a tengeralattjárót**
- **A buborékok kipukkasztásáért pont jár**
- **Az időzítő 30 másodpercről indul**
- **1000 pont elérésekor időt nyerünk**
- **Vége a játéknak, ha lejár az idő**



Hozd létre a játékablakot és a tengeralattjárót!

Kezdd a játéktér elkészítésével! Nyiss egy új kódablakot az IDLE-ben. Írd be az alábbi kódot, amely elkészíti a játékablakot, valamint a tengeralattjárót.



1 Használd a Tkinter könyvtárat a grafikus felület (GUI) elkészítéséhez. Ez a parancssor készíti el a játék ablakát.

```
from tkinter import *
MAGASSÁG = 500
SZÉLESSÉG = 800
ablak = Tk()
ablak.title('Buborékpukkasztó')
v = Canvas(ablak, width=SZÉLESSÉG, height=MAGASSÁG, bg='darkblue')
v.pack()
```

Betölti az összes Tkinter-függvényt

Beállítja az ablak méretét

Menő címet ad a játéknak

Beállítja a háttér színét tengerkékre

Elkészíti a vásznat a rajzoláshoz

2 Egy egyszerű grafika fogja a tengeralattjárót ábrázolni – a Tkinter rajzfüggvényeivel könnyű elkészíteni. Írd be ezt a kódot, és futtasd le!

A tengeralattjáró egy körbe belerajzolt háromszög lesz

Megrajzolja a tengeralattjáró piros háromszögét

```
hajó_1 = v.create_polygon(7, 2, 7, 28, 30, 15, fill='red')
hajó_2 = v.create_oval(0, 0, 30, 30, outline='red')
HAJÓ_R = 15
KP_X = SZÉLESSÉG / 2
KP_Y = MAGASSÁG / 2
v.move(hajó_1, KP_X, KP_Y)
v.move(hajó_2, KP_X, KP_Y)
```

A tengeralattjáró sugara (mérete)

A „KP_X" és „KP_Y" változók adják meg a játéktér középpontjának a koordinátáit

Rajzol egy piros kört

A tengeralattjáró mindkét részét a középpontba mozgatja

Mentsd a munkád!

BUBORÉKPUKKASZTÓ

A tengeralattjáró irányítása

A program következő része a tengeralattjáró irányításáért felelős. Ez a kód létrehoz egy eseménykezelő függvényt, hogy figyelje, melyik nyílbillentyű lett lenyomva, és aszerint mozgatja a hajót.

3 Ez a kód hozza létre a „hajót_mozgat" függvényt. Ez mozdítja el a tengeralattjárót a lenyomott billentyű alapján. Futtasd le, hogy lásd, hogyan működik!

```
HAJÓ_SEB = 10
def hajót_mozgat(event):
    if event.keysym == 'Up':
        v.move(hajó_1, 0, -HAJÓ_SEB)
        v.move(hajó_2, 0, -HAJÓ_SEB)
    elif event.keysym == 'Down':
        v.move(hajó_1, 0, HAJÓ_SEB)
        v.move(hajó_2, 0, HAJÓ_SEB)
    elif event.keysym == 'Left':
        v.move(hajó_1, -HAJÓ_SEB, 0)
        v.move(hajó_2, -HAJÓ_SEB, 0)
    elif event.keysym == 'Right':
        v.move(hajó_1, HAJÓ_SEB, 0)
        v.move(hajó_2, HAJÓ_SEB, 0)
v.bind_all('<Key>', hajót_mozgat)
```

Ekkorát fog lépni a hajó egy gombnyomásra

Felfelé mozgatja mind a két részt, amikor a fel nyilat nyomjuk

Ez akkor fut le, ha a le nyilat nyomjuk, és lefelé mozgatja a hajót

Balra mozog, ha balra nyilat nyomunk

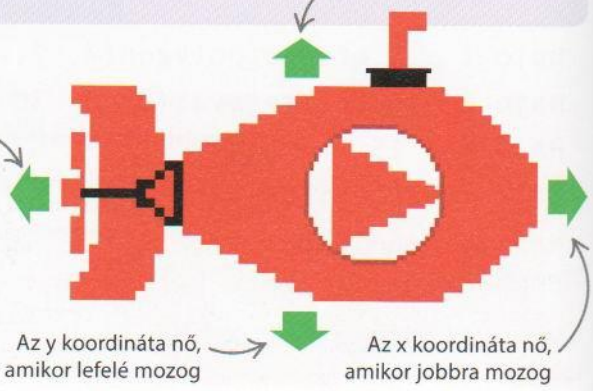
Jobbra mozog, ha jobbra nyilat nyomunk

Ha lenyomunk egy gombot, mindig lefut a „hajót_mozgat" függvény

Mentsd a munkád!

▷ **Hogyan működik?**

A „hajót_mozgat" függvény mozgatja különböző irányokba a hajót. A tengeralattjáró x és y koordinátájához hozzáadva egy számot jobbra vagy lefelé, azokból kivonva egy számot balra vagy felfelé mozog.



Készülj fel a buborékokra!

Most, hogy a tengeralattjáró már mozog, ideje elkészíteni a buborékokat. A buborékok különböző méretűek lesznek, és eltérő sebességgel fognak mozogni.

4 Minden buboréknak más lesz az azonosítója (a programnak tudnia kell megkülönböztetni őket), a mérete és a sebessége.

```
from random import randint
bub_id = list()
bub_r = list()
bub_seb = list()
MIN_BUB_R = 10
MAX_BUB_R = 30
MAX_BUB_SEB = 10
DIFF = 100
def buborékot_gyárt():
    x = SZÉLESSÉG + DIFF
    y = randint(0, MAGASSÁG)
    r = randint(MIN_BUB_R, MAX_BUB_R)
    id1 = v.create_oval(x - r, y - r, x + r, y + r, outline='white')
    bub_id.append(id1)
    bub_r.append(r)
    bub_seb.append(randint(1, MAX_BUB_SEB))
```

Ez létrehoz három üres listát az azonosítóknak, a méreteknek és a sebességeknek

Beállítja a méret alsó határát 10-re, a felsőt pedig 30-ra

Beállítja a buborék helyét a vásznon

Választ egy véletlen méretet a minimum és a maximum között

Ez a sor hozza létre a buborékot

Hozzáadja az azonosítót, a méretet és a sebességet a megfelelő listához

TANÁCSOK

Buboréklisták

Három listát kell használni, hogy a buborékok minden adatát eltároljuk. A listákat üresen hozzuk létre, és amint elkészül egy buborék, belekerülnek az adatok. Mindegyik lista más információt tartalmaz.

bub_id: a buborék azonosítóját (ID) tárolja, a program ennek alapján tudja mozgatni.

bub_r: a buborék sugarát (méretét) tárolja.

bub_seb: a buborék sebességét tárolja.

Mentsd a munkád!

BUBORÉKPUKKASZTÓ

Mozgasd a buborékokat!

Most már készen áll a három lista, a buborékok azonosítójával, véletlenszerűen választott méretével és sebességével. A következő lépés, hogy mozgásra bírjuk a buborékokat.

5 Ez a függvény végigmegy az összes buborékon, és elmozdítja mindet.

```
def buborékot_mozgat():
    for i in range(len(bub_id)):
        v.move(bub_id[i], -bub_seb[i], 0)
```

Végigmegy a lista összes buborékán

Elmozdítja a buborékot a képernyőn a saját sebességével

6 Ez a játék főciklusa. Mindaddig ismétlődik, amíg a játék tart. Próbáld ki!

Mentsd a munkád!

```
from time import sleep, time
BUB_VSZ = 10
#FŐCIKLUS
while True:
    if randint(1, BUB_VSZ) == 1:
        buborékot_gyárt()
    buborékot_mozgat()
    ablak.update()
    sleep(0.01)
```

Betölti a függvényeket a Time könyvtárból

Véletlen számot generál 1 és 10 között

Ha a véletlen szám 1, új buborékot készít (átlagosan 10 alkalomból 1-szer, így nem lesz túl sok buborék)

Meghívja a „buborékot_mozgat()" függvényt

Frissíti az ablakot, hogy újrarajzolja a buborékokat

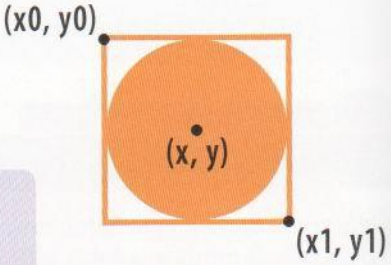
Lelassítja a játékot, hogy könnyebb legyen játszani

7 Most egy hasznos függvényt készítünk, amelyik megadja egy buborék pontos helyzetét az azonosítója alapján. Ezt érdemes az 5. pontban készített függvény után írni a kódban.

```
def koord_kér(azonosító):
    poz = v.coords(azonosító)
    x = (poz[0] + poz[2])/2
    y = (poz[1] + poz[3])/2
    return x, y
```

Kiszámolja a buborék közepének x koordinátáját

Kiszámolja a buborék közepének y koordinátáját



△ **Buborékok felderítése**
A függvény úgy határozza meg a buborék középpontját, hogy a buborék köré rajzolható négyzet sarkainak koordinátáit átlagolja.

Hogyan pukkasszunk buborékokat?

A játékos akkor kap pontot, ha kipukkaszt egy buborékot.
Ekkor a programnak el kell tüntetnie a buborékot a képernyőről.
Mindezt a következő függvények teszik meg.

8 Ez a függvény eltávolít egy buborékot a játékból. Kitörli a buborékot mindhárom listából, illetve a vászonról is. Ezt a függvényt közvetlenül a 7. pontban készített kódrész alá írd be.

```
def bub_töröl(i):
    del bub_r[i]
    del bub_seb[i]
    v.delete(bub_id[i])
    del bub_id[i]
```

Ez a függvény kitörli az „i" azonosítójú buborékot

Kitörli a buborékot a méretet és a sebességet nyilvántartó listákból

Törli a vászonról

Törli az azonosítók listájából

9 Ez a függvény azokat a buborékokat távolítja el, amelyek már kiúsztak a képernyőről. Ezt a kódot a 8. pontban írt függvény után írd be.

```
def bub_takarít():
    for i in range(len(bub_id)-1, -1, -1):
        x, y = koord_kér(bub_id[i])
        if x < -DIFF:
            bub_töröl(i)
```

Visszafelé megy végig a lista elemein, hogy akkor se legyen hiba, ha közben kitörlődött egy buborék

Meghatározza a buborék helyét

Ha a buborék már nincs a képernyőn, kitörli. Egyébként lelassulna a játék

10 Most frissítsd a 6. lépésben leírt főciklust ezekkel a függvényekkel. Futtasd le, hogy lásd, van-e bennük hiba.

```
# FŐCIKLUS
while True:
    if randint(1, BUB_VSZ) == 1:
        buborékot_gyárt()
    buborékot_mozgat()
    bub_takarít()
    ablak.update()
    sleep(0.01)
```

Új buborékot készít

Mozgatja a buborékokat

Kitörli azt a buborékot, amelyik nincs a képernyőn

Újrarajzolja az ablakot, hogy lássuk a változást

Mentsd a munkád!

BUBORÉKPUKKASZTÓ

Két pont távolságának kiszámítása

Ebben a játékban és sok másikban is fontos, hogy ismerjük két dolog távolságát. Ezt a Pitagorasz-tétel segítségével fogjuk kiszámítani.

11 Ez a függvény kiszámolja két objektum távolságát. Írd ezt a függvényt a 9. pontban írt függvény után!

```
from math import sqrt
def távolság(id1, id2):
    x1, y1 = koord_kér(id1)
    x2, y2 = koord_kér(id2)
    return sqrt((x2 - x1)**2 + (y2 - y1)**2)
```

Betölti az „sqrt" (négyzetgyök-) függvényt a Math könyvtárból

Megadja az első objektum koordinátáit

Megadja a második objektum koordinátáit

Visszaadja a távolságukat (a ** a hatványozás jele a Pythonban)

Pukkaszd ki a buborékokat!

A játékos pontot kap, ha kipukkasztja a buborékokat. A nagyobb és a gyorsabb buborékok több pontot érnek. A következő programrész kiszámolja, hogy mikor kell egy buboréknak kipukkadnia.

▷ **Az ütközés érzékelése**
Ha a hajó középpontjának és egy buborék középpontjának a távolsága kisebb, mint a sugaruk összege, akkor összeütköztek.

12 Amikor a tengeralattjáró és egy buborék összeütközik, a buboréknak el kell tűnnie, és a pontszámnak frissülnie kell. A kódnak ezt a részét a 11. pontban írt parancsok után írd be.

```
def ütközés():
    pontok = 0
    for bub in range(len(bub_id)-1, -1, -1):
        if távolság(hajó_2, bub_id[bub]) < (HAJÓ_R + bub_r[bub]):
            pontok += (bub_r[bub] + bub_seb[bub])
            bub_töröl(bub)
    return pontok
```

Ez a változó tárolja a pontszámot

A ciklus visszafelé haladva megy végig a buborékokon, hogy ne okozzanak gondot a már kitörölt buborékok

Megvizsgálja, hogy bármely buborék összeütközött-e a tengeralattjáróval

Kiszámolja, hogy hány pontot ér a buborék, és hozzáadja a „pont" változóhoz

Törli a buborékot

Visszaadja a pontszámot

13 Frissítsd a főciklust az új függvényekkel. Ügyelj a sorrendre, mindent a megfelelő helyre írj. Utána futtasd a programot. A buborékoknak el kell tűnniük, amikor érintik a tengeralattjárót. Nézd meg a terminálablakban, hány pontot szereztél.

```
pontszám = 0
#FŐCIKLUS
while True:
    if randint(1, BUB_VSZ) == 1:
        buborékot_gyárt()
    buborékot_mozgat()
    bub_takarít()
    pontszám += ütközés()
    print(pontszám)
    ablak.update()
    sleep(0.01)
```

A pontszám 0 legyen a játék kezdetén

Új buborékot készít

A buborék értékét hozzáadja a pontszámhoz

A pontszám a terminálablakban látható. Később majd ezt is megjelenítjük a játékban

Ez kis időre leállítja a folyamatot. Próbáld ki úgy is, hogy ezt törlöd!

Mentsd a munkád!

TANÁCSOK

A Python rövidítései

A „pontszám += ütközés()" a „pontszám = pontszám + ütközés()" kifejezés rövidítése. Hozzáadja az ütközés pontértékét a pontszámhoz, majd frissíti az értékét. Az ehhez hasonló kódok gyakoriak és nagyon hasznosak. Ugyanilyen módon a „–" jelet is használhatjuk. Például: „pontszám –= 10" jelentése ugyanaz, mint „pontszám = pontszám – 10".

BUBORÉKPUKKASZTÓ

Utolsó simítások

A játék főbb részei már működnek. A még hátralévő dolgok: megjeleníteni a pontszámot, illetve beállítani egy időmérőt.

14 Ezt a kódrészletet a 12. pontban írt kód után helyezd el. Ez jeleníti meg a pontszámot és a hátralévő játékidőt.

```
v.create_text(50, 30, text='IDŐ', fill='white' )
v.create_text(150, 30, text='PONTSZÁM', fill='white' )
időszöveg = v.create_text(50, 50, fill='white' )
pontszöveg = v.create_text(150, 50, fill='white' )
def pontot_mutat(pontszám):
    v.itemconfig(pontszöveg, text=str(pontszám))
def időt_mutat(maradék_idő):
    v.itemconfig(időszöveg, text=str(maradék_idő))
```

„IDŐ" és „PONTSZÁM" feliratokat készít, hogy a játékos tudja, melyik szám mit jelent

Beállítja a pontszámot és a hátralévő időt

Megjeleníti a pontszámot

Megjeleníti a hátralévő időt

15 A következő lépés a játék időtartamának és az extra időt érő ponthatárnak a beállítása, valamint hogy kiszámoljuk a játék végének időpontját. Ezt a részt közvetlenül a főciklus elé írd!

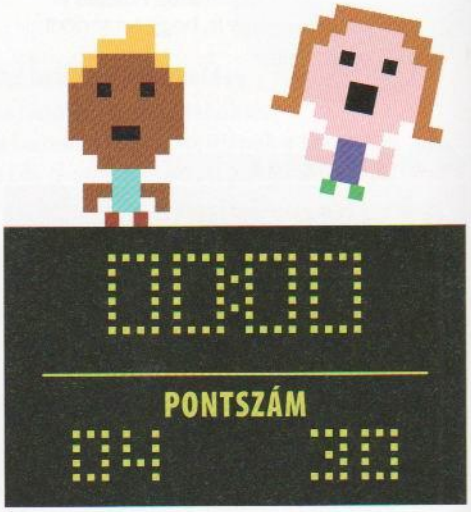
```
from time import sleep, time
BUB_VSZ = 10
IDŐLIMIT = 30
BÓNUSZPONT = 1000
pontszám = 0
bónusz = 0
vége = time() + IDŐLIMIT
```

Betölti a Time könyvtár függvényeit

30 másodperces játékidővel kezdődik a játék

Beállítja, hogy 1000 pont elérésekor járjon extra idő

A játék végének időpontja a „vége" változóban



△ **Eredményjelző**
Az eredményjelzőn követhető a játék pillanatnyi állása.

16 Frissítsd a főciklust az időt és a pontszámot kezelő új függvényekkel!

```
#FŐCIKLUS
while time() < vége:
    if randint(1, BUB_VSZ) == 1:
        buborékot_gyárt()
    buborékot_mozgat()
    bub_takarít()
    pontszám += ütközés()
    if (int(pontszám / BÓNUSZPONT)) > bónusz:
        bónusz += 1
        vége += IDŐLIMIT
    pontot_mutat(pontszám)
    időt_mutat(int(vége - time()))
    ablak.update()
    sleep(0.01)
```

A főciklust a játék végéig ismétli

Kiszámolja, hogy mikor kell extra időt adni

A „print(pontszám)"-ot a „pontot_mutat(pontszám)" függvénnyel helyettesítve a pontszám a játékablakban jelenik meg

Megjelenik a hátralévő idő

Mentsd a munkád!

17 Adj hozzá végül egy „VÉGE A JÁTÉKNAK" feliratot. Ez akkor jelenik meg, ha lejárt az idő. Írd ezt a program legaljára.

```
v.create_text(KP_X, KP_Y, \
    text='VÉGE A JÁTÉKNAK', fill='white', font=('Helvetica',30))
v.create_text(KP_X, KP_Y + 30, \
    text='PONTSZÁM: '+ str(pontszám), fill='white')
v.create_text(KP_X, KP_Y + 45, \
    text='Bónuszidő: '+ str(bónusz*IDŐLIMIT), fill='white')
```

A képernyő közepére helyezi a szöveget

Beállítja a betűtípust. Ilyen nagy betűkhöz a „Helvetica" jó választás

Kiírja, hány pontot értél el

Fehér színűre állítja a feliratot

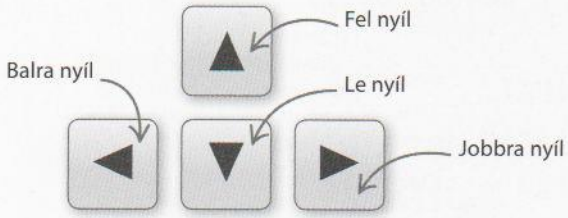
Kiírja, mennyi extra időt szereztél

Mentsd a munkád!

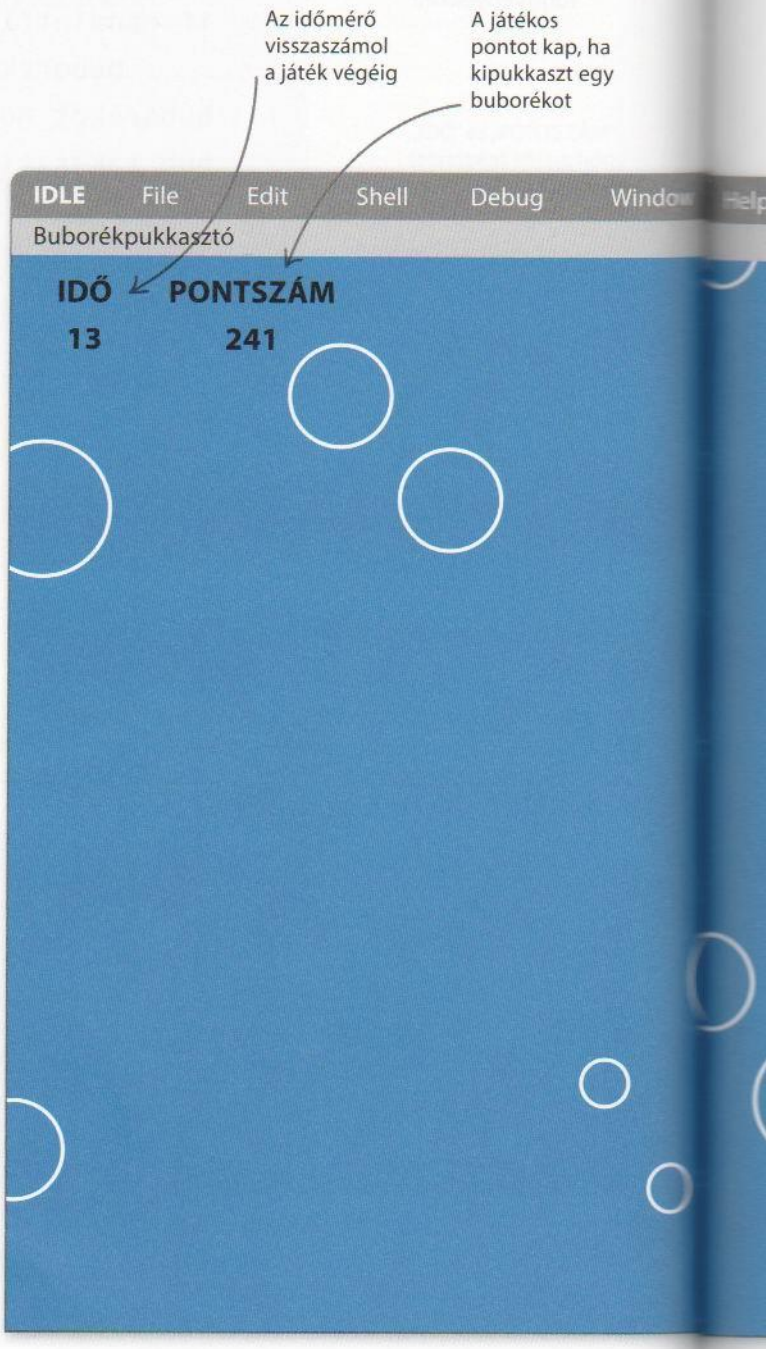
BUBORÉKPUKKASZTÓ

Ideje játszani

Remek munka! Befejezted a Buborékpukkasztó játék készítését, itt az idő játszani. Futtasd a programot, és próbáld ki! Ha valami nem működik, jusson eszedbe a hibakeresésről tanultak. Nézd végig a kódot az előző oldalakon, hogy mindent jól írtál-e be.



△ **Irányítás**
A tengeralattjárót a nyilakkal tudod irányítani. Ezt a program módosításával beállíthatod másképp is.



TANÁCSOK

Fejleszd a játékot!

Minden számítógépes játék egy alapötletből indul. Játszanak vele, tesztelik, módosítják, és továbbfejlesztik. Gondolj erre úgy, mintha a játékod első változata lenne. Itt van néhány javaslat, hogyan változtathatod vagy fejlesztheted a játékot új kód hozzáadásával:

Nehezítsd a játékot az időkorlát megváltoztatásával vagy a bónuszidő ponthatárának növelésével!

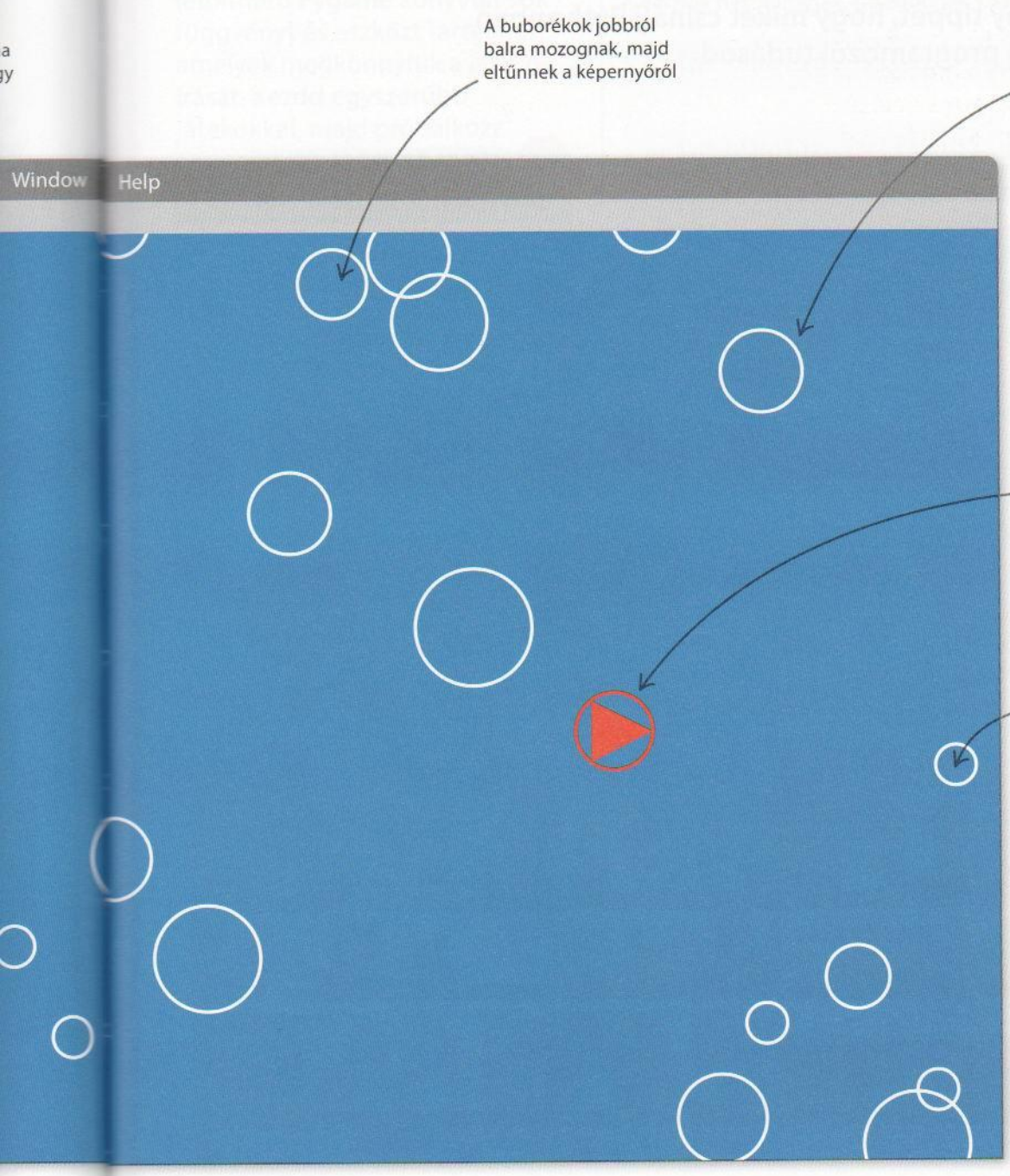
Válassz másik színt a tengeralattjárónak!

Készíts részletesebb rajzot a tengeralattjárónak!

Készíts különleges buborékot, amely megnöveli a tengeralattjáró sebességét!

Készíts intelligens bombát, amely a szóköz lenyomására felrobban, és kipukkasztja az összes buborékot!

Készíts ranglistát, hogy megőrizd a legjobb eredményeket!



◁ **Szuper-tengeralattjáró**
Most már játszhatsz a barátaiddal, és versenyezhettek, hogy ki a legügyesebb. Utána mutasd meg nekik a kódot, és magyarázd el, hogyan működik!

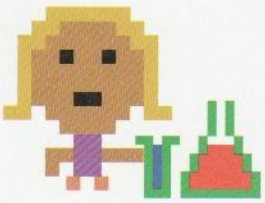
Hogyan tovább?

LÁSD MÉG
‹ 152–153 Könyvtárak
Számítógépes játékok 204–205 ›

Most, hogy megcsináltad a könyv összes Python-projektjét, a saját utadra léphetsz, hogy nagyszerű programozó legyél. Itt olvashatsz néhány tippet, hogy miket csinálj Pythonban, és hogyan fejleszd a programozói tudásod.

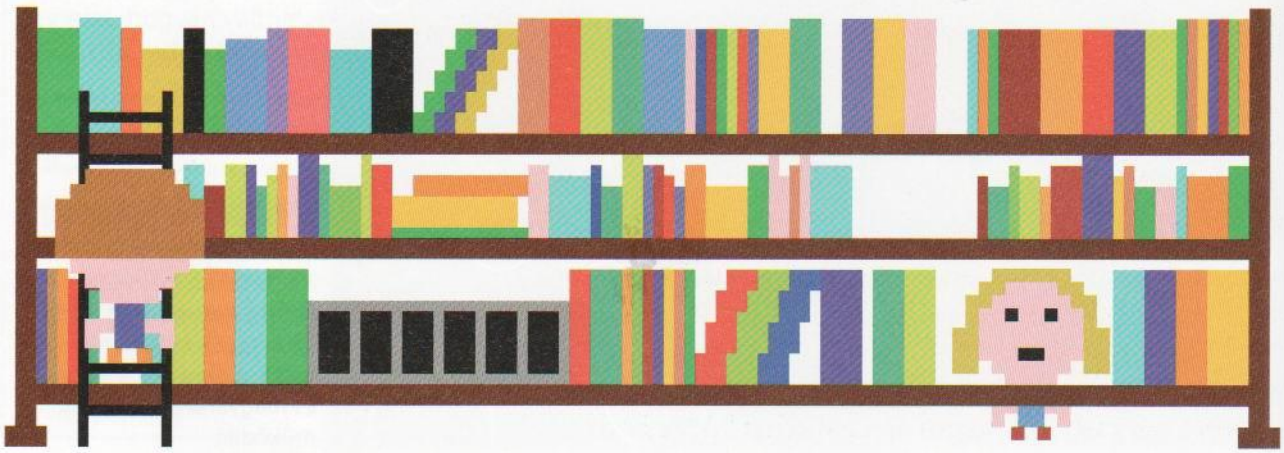
Kísérletezz!

Játssz a könyvben található példakódokkal! Találd ki, hogyan módosíthatod, javíthatod őket. Ne félj átírni a programokat. Ez remek módja, hogy kísérletezz a Pythonnal. Ne felejtsd el, hogy ez egy profi programozási nyelv, rengeteg lehetőséggel – szinte bármit meg lehet vele oldani.



Építs saját könyvtárakat!

A programozók szeretik újra felhasználni a kódjaikat, és megosztani másokkal. Készíts saját könyvtárat a hasznos függvényeidből, és oszd meg te is! Nagyszerű érzés, amikor más használja fel a te függvényeidet. Akár olyanokat is készíthetsz, mint a Tkinter vagy a Turtle.



JEGYEZD MEG!

Olvass sok kódot!

Keress érdekes programokat és könyvtárakat, amelyeket mások írtak, és nézd végig a kódjukat. Próbáld megérteni a működésüket, és hogy miért épp úgy készítették el őket. Így sok ötletet kaphatsz, illetve sokat megtudhatsz a beépített könyvtárakról is. Mindezt később fel is használhatod.

Készíts játékokat a Pythonnal!

Saját játékokat is készíthetsz Python nyelven. Az internetről letölthető Pygame könyvtár sok függvényt és eszközt tartalmaz, amelyek megkönnyítik a játékok írását. Kezdd egyszerűbb játékokkal, majd próbálkozz összetettebbekkel is!

TANÁCSOK

A Python különböző verziói

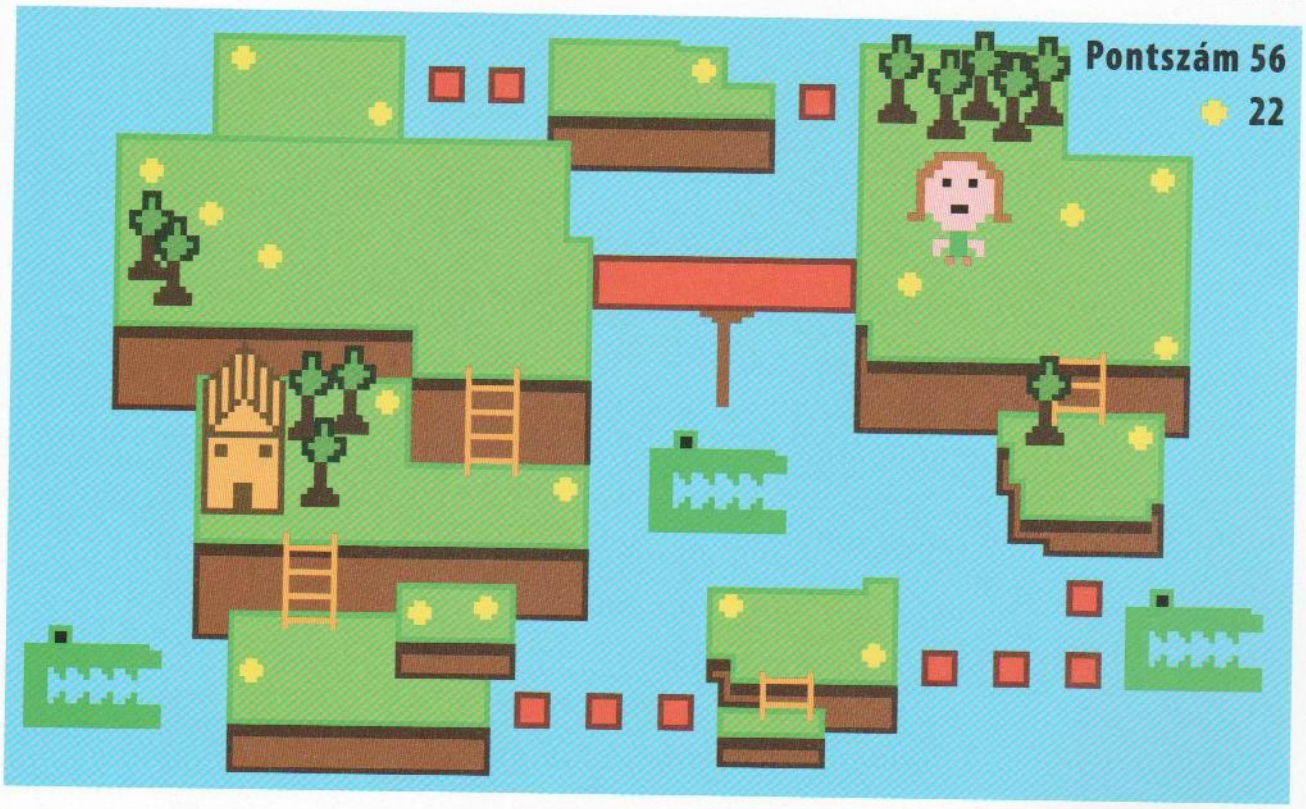
Amikor találsz egy programot valahol (az interneten vagy egy másik könyvben), akkor lehet, hogy a Python más verziójára írták. A verziók hasonlítanak, de adódnak köztük apró különbségek.

```
print 'Helló, világ!'
```

Python 2

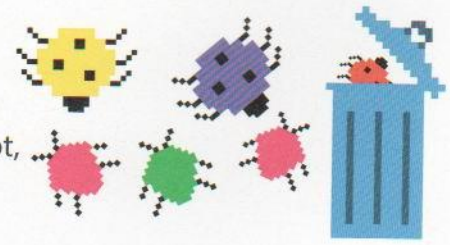
```
print('Helló, világ!')
```

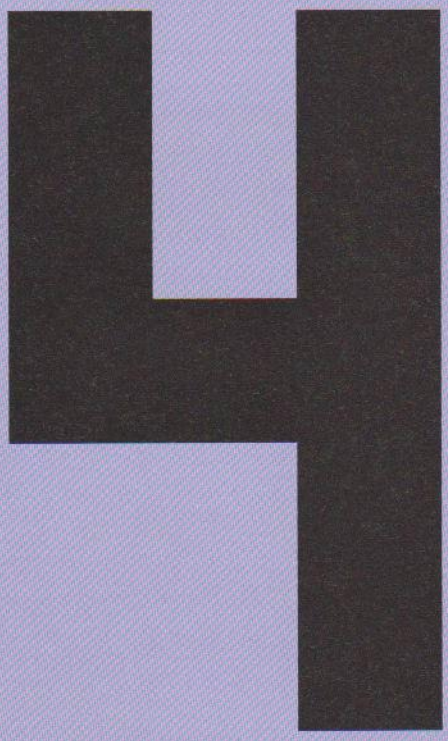
Python 3



Keresd a hibákat!

A hibakeresés nagyon fontos része a programozásnak. Ne add fel, ha valami nem működik! Tudod, a számítógép csak azt hajtja végre, amire utasítod, ezért mindig nézd végig a kódot, hogy miért nem működik jól. Gyakran másik programozó segíthet gyorsabban megtalálni a hibát a kódodban.





A számítógép belülről

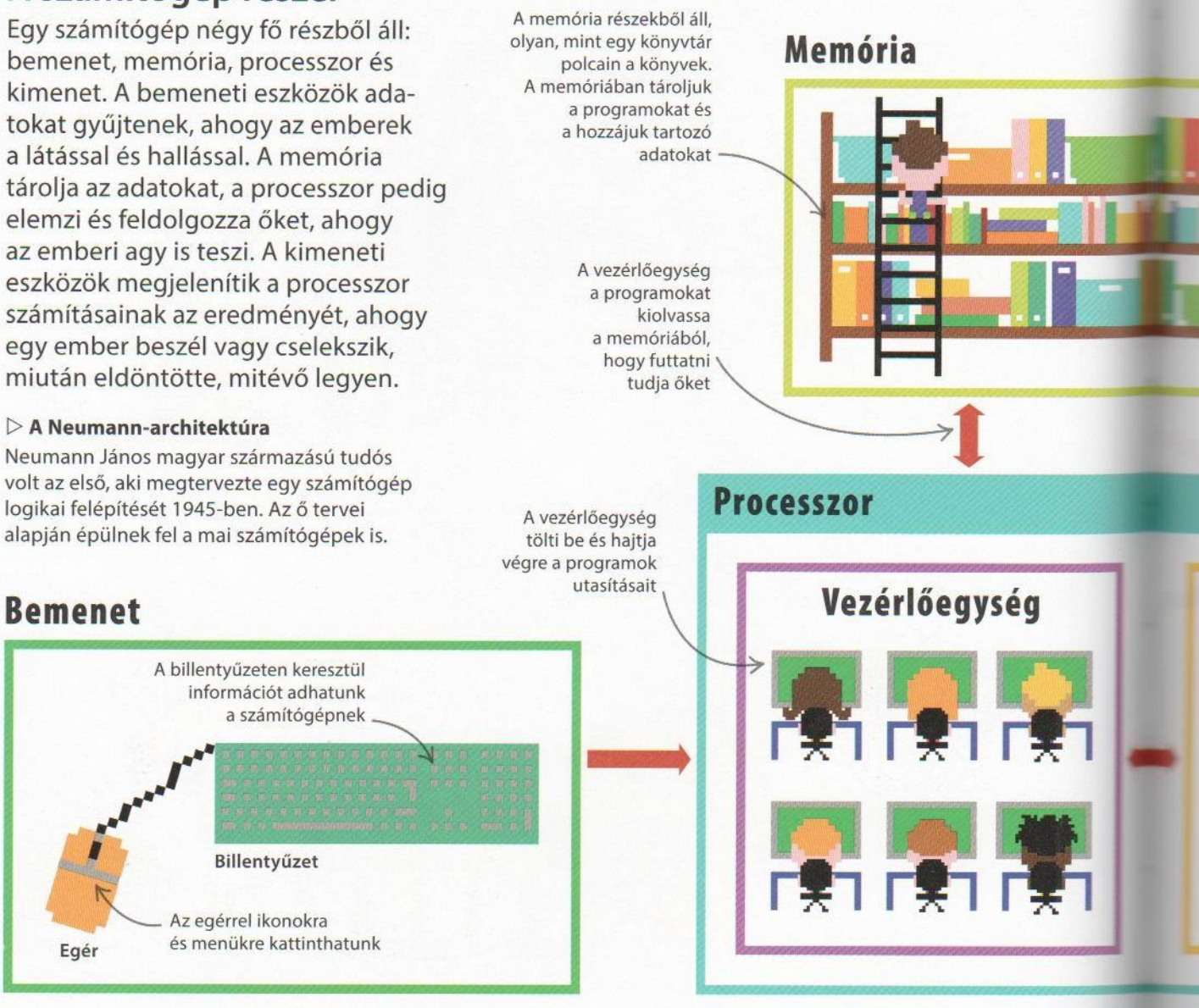
Az első számítógépek egyszerű számológépek voltak. Bizonyos szempontból nem sokat változtak azóta. Adatot kapnak (bemenet), számításokat végeznek, majd megadják a választ (kimenet).

LÁSD MÉG
Adattárolás fájlokban **192–193** ›
Az internet **194–195** ›
Miniszámítógépek **214–215** ›

A számítógép részei

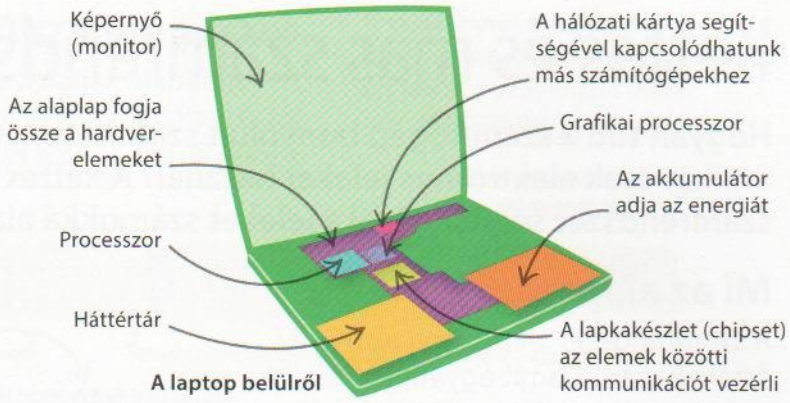
Egy számítógép négy fő részből áll: bemenet, memória, processzor és kimenet. A bemeneti eszközök adatokat gyűjtenek, ahogy az emberek a látással és hallással. A memória tárolja az adatokat, a processzor pedig elemzi és feldolgozza őket, ahogy az emberi agy is teszi. A kimeneti eszközök megjelenítik a processzor számításainak az eredményét, ahogy egy ember beszél vagy cselekszik, miután eldöntötte, mitévő legyen.

▷ **A Neumann-architektúra**
Neumann János magyar származású tudós volt az első, aki megtervezte egy számítógép logikai felépítését 1945-ben. Az ő tervei alapján épülnek fel a mai számítógépek is.



A számítógép hardvere

A hardver a számítógép összes megfogható része. Egy számítógép sok hardverelemet tartalmaz. Ahogy a számítógépek készítői egyre több funkciót építenek a gépekbe, a hardverelemeknek egyre kisebbeknek kell lenniük, kevesebb hőt kell termelniük, és kevesebbet kell fogyasztaniuk.

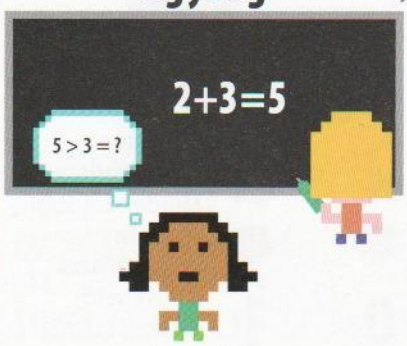


Az aritmetikai-logikai egység a memóriából nyer adatokat a számításokhoz

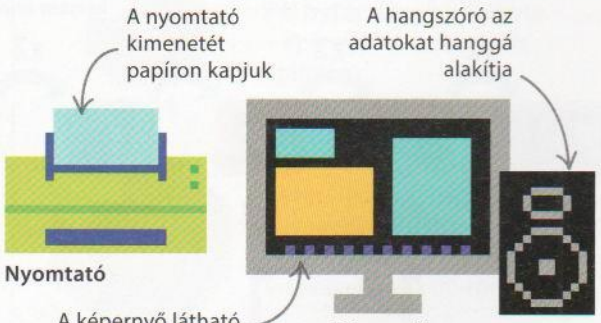
A processzor két részből áll, az egyik parancsokat hajt végre, a másik számításokat végez

Az aritmetikai-logikai egység (ALU, az angol elnevezés alapján) minden szükséges számítást elvégez

Aritmetikai-logikai egység



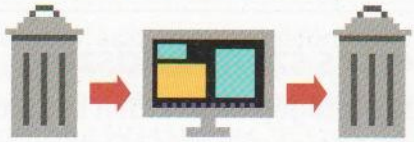
Kimenet



FOGALOMTÁR

GIGO

Szemétből csak szemét lesz – az angol GIGO (garbage in, garbage out) rövidítés arra utal, hogy a legjobb programok is használhatatlan eredményt adnak, ha rossz bemeneti adatot kapnak.



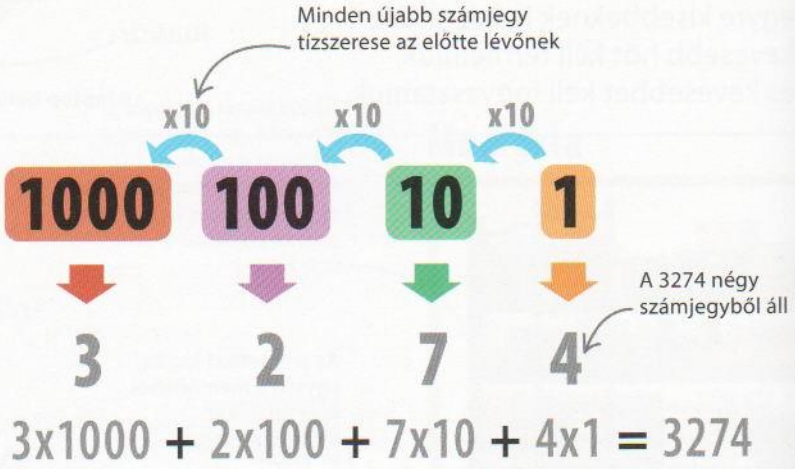
Kettes és más számrendszerek

LÁSD MÉG
Szimbólumok és kódok **184–185 ›**
Logikai kapuk **186–187 ›**

Hogyan tud a számítógép bonyolult számításokat végezni, amikor csak elektromos jeleket használ? A kettes (bináris) számrendszer segítségével a jeleket számokká alakítja.

Mi az alapszám?

A számrendszer alapszáma azon értékek száma, amit egyetlen számjegy ábrázolni tud. Minden további számjegy az alap hatványával növeli az ábrázolható értéket.



▷ **Tízes számrendszer**
A tízes (decimális) a legismertebb számrendszer, alapja a 10. Egy számjegy 10 különböző értéket tud felvenni, két számjegy 100-at, három pedig 1000-et.

Kettes számrendszer

A számítógép alapvetően két értéket „ért meg": „be(kapcsolva)" vagy „ki(kapcsolva)" állapotban lévő elektromos jeleket. Mivel csupán két érték van, a számítógép kettes számrendszerbeli számokat használ. Minden számjegy 1 vagy 0 értéket vehet fel, az újabb számjegyek pedig az előző kétszeresei.

▷ **1 és 0**
A bekapcsolt áramkör jelenti az 1-est, a kikapcsolt pedig a 0-t.

◁ **Kettes számrendszer**
Minden további számjegy megduplázza a felvehető értékek számát.

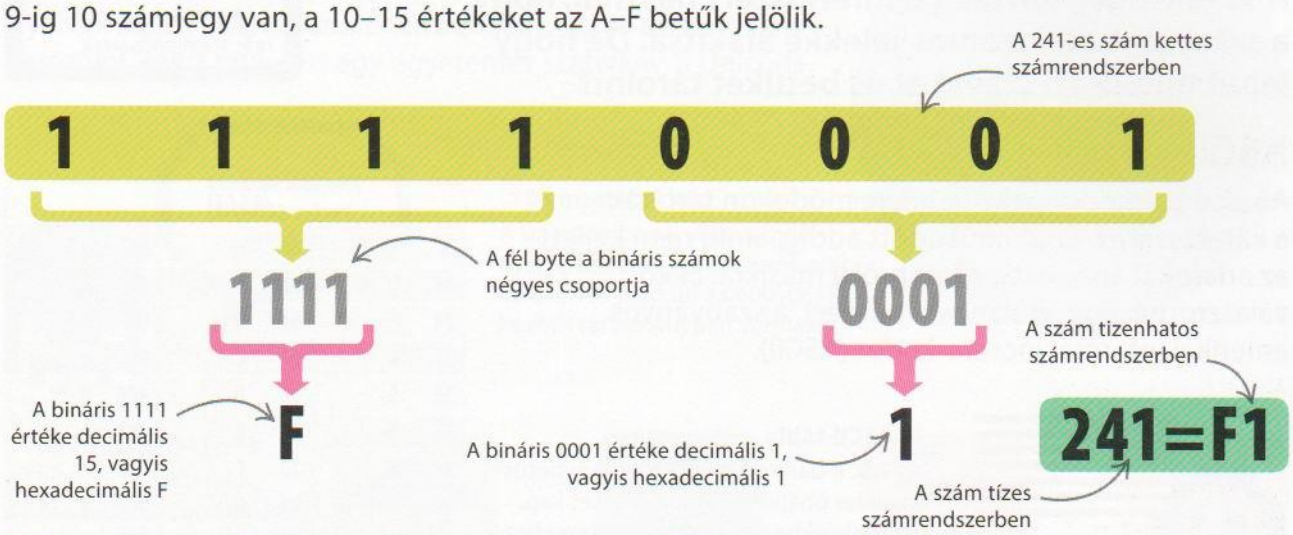
1x128 + 1x64 + 1x32 + 1x16 + 0x8 + 0x4 + 0x2 + 1x1 = 241

Tizenhatos (hexadecimális) számrendszer

A számítógépes programok sokszor használják alapként a 16-ot is, mert könnyű a kettesből erre átváltani. Mivel 0-tól 9-ig 10 számjegy van, a 10–15 értékeket az A–F betűk jelölik.

▽ A „fél byte" értelmezése

A fél byte négy bináris számjegyből áll, így leírható a tizenhatos (hexa) számrendszer egy számjegyével.



▽ A számrendszerek összehasonlítása

Ebben a táblázatban láthatod, hogy a számok tizenhatos számrendszerben való ábrázolása adja a legtöbb információt a legkevesebb számjegy használatával.

KÜLÖNBÖZŐ SZÁMRENDSZEREK

Decimális	Bináris	Hexadecimális
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

JEGYEZD MEG!

Bit, fél byte, byte

Egy bináris számjegy, azaz egy bit az információelmélet legkisebb egységének számít. Négy bit egy fél byte, nyolc bit pedig egy byte. Egy kilobit 1024 bit. Egy megabit 1024 kilobit.

Bit: Minden bit egy bináris adat: 1 vagy 0.

Fél byte: Négy bit fél byte – épp egy hexadecimális számjegy.

Byte: Nyolc bit vagy két hexadecimális számjegy egy byte. Ez 0-tól 255-ig vehet fel értékeket (azaz 00-tól FF-ig).

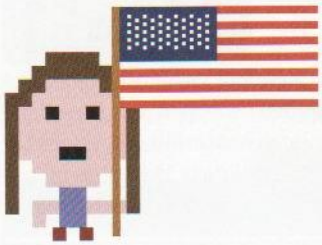
Szimbólumok és kódok

A számítógép kettes számrendszert használ, hogy a számokat elektromos jelekké alakítsa. De hogy lehet binárisan szavakat és betűket tárolni?

LÁSD MÉG
❮ 180–181 A számítógép belülről
❮ 182–183 A kettes és más számrendszerek

ASCII

Az első számítógépek különféle módokon tárolták a karaktereket. Ez jól működött addig, amíg nem kellett az adatokat átvinni egyik gépről a másikra. Ekkor választottak egy általános rendszert, a szabványos amerikai információcsere-kódot (ASCII).



▷ **ASCII-tábla**
Az ASCII-ban minden kis- és nagybetűs karakter önálló decimális értéket kap. Az írásjelekhez és egyéb karakterekhez, például a szóközhöz szintén tartoznak értékek.

▷ **ASCII binárisan**
A karakterekhez tartozó decimális számokat binárissá kell alakítani, hogy tárolhatók legyenek a számítógépen.

R = 82 = 1010010

r = 114 = 1110010

▽ **ASCII a Pythonban**
Az ASCII-kódok decimális és bináris értéke közötti átváltás a legtöbb nyelvben lehetséges, a Pythonban is.

Ez a parancs kinyomtatja a „Samu" név betűit, valamint a decimális és bináris ASCII-értékét

```
>>> név = 'Samu'
>>> for c in név:
        print(c, ord(c), bin(ord(c)))

S 83 0b1010011
a 97 0b1100001
m 109 0b1101101
u 117 0b1110101
```

Ezek az eredmények. Minden bináris értéket a „0b" kezdés jelez

ASCII

32	SPACE	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	DELETE

Unicode

Amikor az egész világon elkezdtek adatokat megosztani egymással a számítógépek, az ASCII-kód kevésnek bizonyult. Nyelvek százai által használt nemzeti karakterek ezreit kellett ábrázolni, ezért született egy egyetemes szabvány, a Unicode.

▷ **Nemzetközi kód**
A Unicode-ban a világ minden nyelve szerepel. Az alapvető arab karakterek például a 0600–06FF közötti tartományban vannak.

▷ **Unicode-karakterek**
A Unicode-karakterekre a hexadecimális értékükkel hivatkozunk, amelyek betűkből és számjegyekből állnak (*lásd* 182–183. o.). Minden karakternek saját kódja van. Folyamatosan bővül a karakterek listája – vannak köztük szokatlanok is, mint például az esernyő.


2602


2EC6


08A2


0036


0974


004D


2702


A147

JEGYEZD MEG!

A tizenhatos számrendszer

A tizenhatos számrendszer alapja a 16. A megszokott számjegyeket használják a 0–9 értékekre, a 10–15 értékeket az A–F betűk jelölik. Minden hexadecimális számnak van bináris megfelelője.

Az ë karakter Unicode-értéke a tizenhatos számrendszerben

Ugyanez az érték a kettes számrendszerben

ë = 00EB = 11100111

▽ **Unicode a Pythonban**
A Unicode-dal különleges karaktereket jeleníthetsz meg Pythonban. Egyszerűen írhatsz karakterláncot, amelyben Unicode-karakterkód is van.

A „\u" jelzés a hexadecimális érték előtt jelzi a számítógépnek, hogy ez Unicode

```
>>> 'm\u00EBg\u00EBszlek'
'mëgëszlek'
```

A kódot az „ë" karakterre fordítja a program

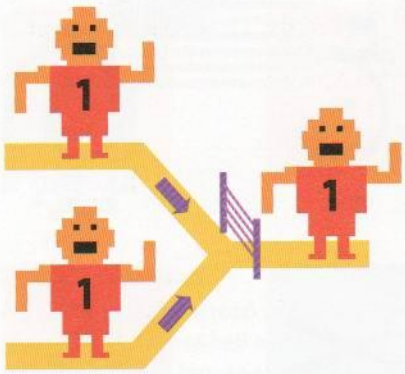
Logikai kapuk

LÁSD MÉG
❮ 180–181 A számítógép belülről
❮ 182–183 A kettes és más számrendszerek

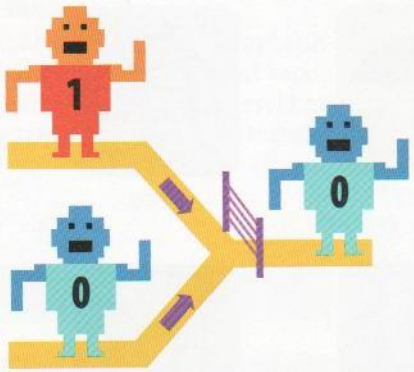
A számítógép elektromos jeleket használ nemcsak a számok és a betűk ábrázolására, hanem arra is, hogy logikai kapuk segítségével döntéseket hozzon. A logikai kapuknak négy fő típusuk van: „ÉS" (AND), „NEM" (NOT), „VAGY" (OR) és „KIZÁRÓ VAGY" (XOR).

ÉS kapu

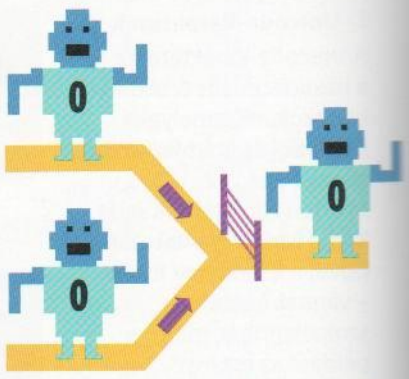
A logikai kapuk egy vagy több bemenő jelet használnak egy kimenő jel létrehozásához. Az ÉS kapunál csak akkor lesz a kimenő jel bekapcsolva (1), ha mindkét bemenő jel be van kapcsolva (1 és 1).



△ **Bemenet: 1 és 1 → kimenet: 1**
Mindkét bemenő jel be van kapcsolva, ezért az ÉS kapu kimenő jele is be van kapcsolva.



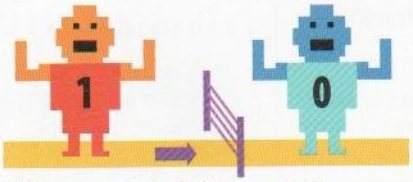
△ **Bemenet: 1 és 0 → kimenet: 0**
Ha az egyik bemeneti jel be van kapcsolva, de a másik nincs, akkor a kimeneti jel ki lesz kapcsolva.



△ **Bemenet: 0 és 0 → kimenet: 0**
Az ÉS kapu kimenete ki lesz kapcsolva, ha mindkét bemenete ki van kapcsolva.

NEM kapu

Ezek a kapuk minden bemenetet az ellenkezőjére fordítanak. Ha a bemenet be van kapcsolva, a kimenet ki lesz kapcsolva, és fordítva. A NEM kapuk ezért más néven „inverterek".



△ **Bemenet: 1 → kimenet: 0**
A NEM kapu a „be" értéket „ki" értékre alakítja, és fordítva.

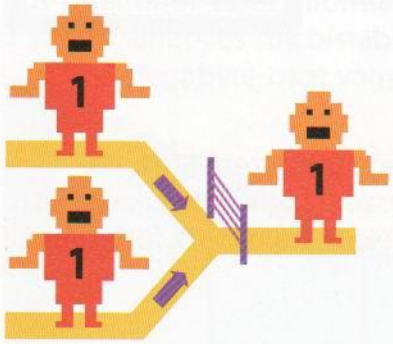
VALÓS VILÁG

George Boole (1815–1864)

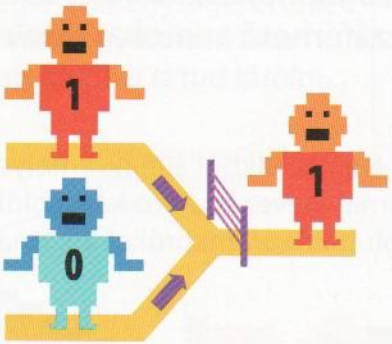
George Boole, a logikai kapuk kidolgozója, angol matematikus volt. Megalkotott egy rendszert, amellyel logikai problémákat lehet megoldani. A matematika ezen részét, amely igaz vagy hamis értékekkel dolgozik, az ő tiszteletére Boole-algebrának nevezték el.

VAGY kapu

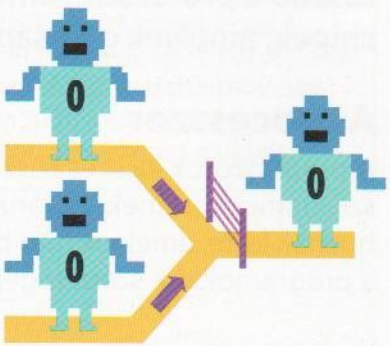
A VAGY kapu bekapcsolt kimenetet hoz létre, ha az egyik vagy mindkét bemenete be van kapcsolva.



△ **Bemenet: 1 és 1 → kimenet: 1**
Két bekapcsolt bemenet bekapcsolt kimenetet hoz létre.



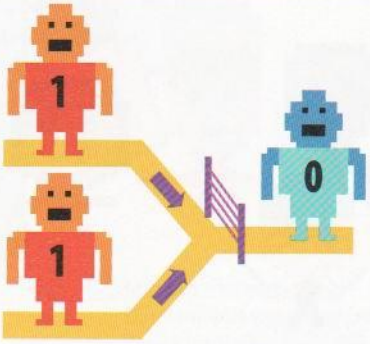
△ **Bemenet: 1 és 0 → kimenet: 1**
Egy bekapcsolt és egy kikapcsolt bemenet is bekapcsolt kimenetet hoz létre.



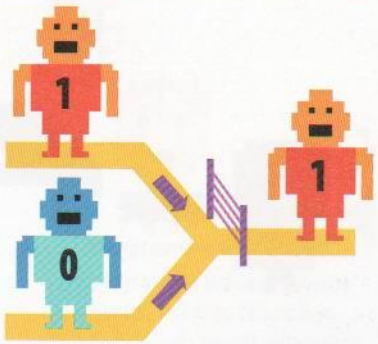
△ **Bemenet: 0 és 0 → kimenet: 0**
Kizárólag két kikapcsolt bemenet hoz létre kikapcsolt kimenetet a VAGY kapunál.

KIZÁRÓ VAGY kapu

Ez a fajta kapu csak akkor hoz létre bekapcsolt kimenetet, ha az egyik bemenet be, a másik ki van kapcsolva. Két be- vagy két kikapcsolt bemenet kikapcsolt kimenetet hoz létre.



△ **Bemenet: 1 és 1 → kimenet: 0**
Két bekapcsolt bemenet kikapcsolt kimenetet hoz létre.

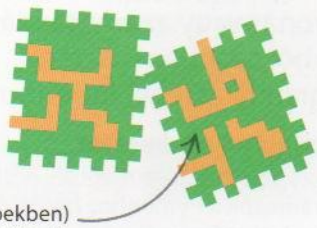


△ **Bemenet: 1 és 0 → kimenet: 1**
A kimenet csak akkor van bekapcsolva, ha a bemenetek különböznek.

TANÁCSOK

Számítógépes áramkörök építése

Ha ezt a négy alapvető logikai kaput kombinálod, számos feladatra alkalmas áramkört hozhatsz létre. Ha például az ÉS és a KIZÁRÓ VAGY kaput összekötöd, olyan áramkört hozhatsz létre, amely összead két bináris számjegyet (bitet). Ha két VAGY kaput összekötsz két NEM kapuval egy hurokba, olyan áramkört hozol létre, amely képes egy bit adat (egy 1 vagy egy 0) tárolására. A legerősebb számítógépek is apró logikai áramkörök milliárdjaiból épülnek fel.



A számítógépes lapkákban (chipekben) számos logikai áramkör található

Processzor és memória

A számítógépen belül sokféle elektronikus chip van. A legfontosabb a processzor, amely programokat futtat, és a memóriachipek, amelyek gyorsan hozzáférhető adatokat tárolnak.

LÁSD MÉG
❮ 180–181
A számítógép belülről
❮ 186–187
Logikai kapuk

A processzor

A processzorok kicsi és összetett áramkörökből állnak, amelyeket egy üvegszerű anyagra, szilíciumra kerülnek. A tranzisztornak nevezett apró kapcsolók egyszerű logikai kapukat hoznak létre, amelyek már bonyolultabb áramköröket alkotnak. Ezek az áramkörök futtatják a programokat a számítógépen.

◁ **Áramkörök a processzorban**
Az áramköröket egy órajel hangolja össze, ahogy a karmester a zenekart.

Gépi (bináris) kód

A processzor csak egyféle programnyelv parancsait érti, ezt gépi kódnak nevezzük. Ezek egyszerű utasítások olyan műveletekre, mint az összeadás, a kivonás vagy az adattárolás. Ezekből azonban összetett programokat lehet létrehozni.

```
Mentés a memóriába
83 e4 f0
83 ec 20
c7 44 24 1c 00 00 00
00
eb 11
Egy másik kódrészlet előhívása
c7 04 24 b0 84 04 08
e8 1d ff ff ff
Két érték összehasonlítása
83 44 24 1c 01
83 7c 24 1c 09
7e e8
```

▷ **A gépi kód értelmezése**
A gépi kód csak számokból áll, ezért a programozók a Pythonhoz hasonló programnyelveket használnak, az ilyen kódot aztán programok alakítják gépi kóddá.

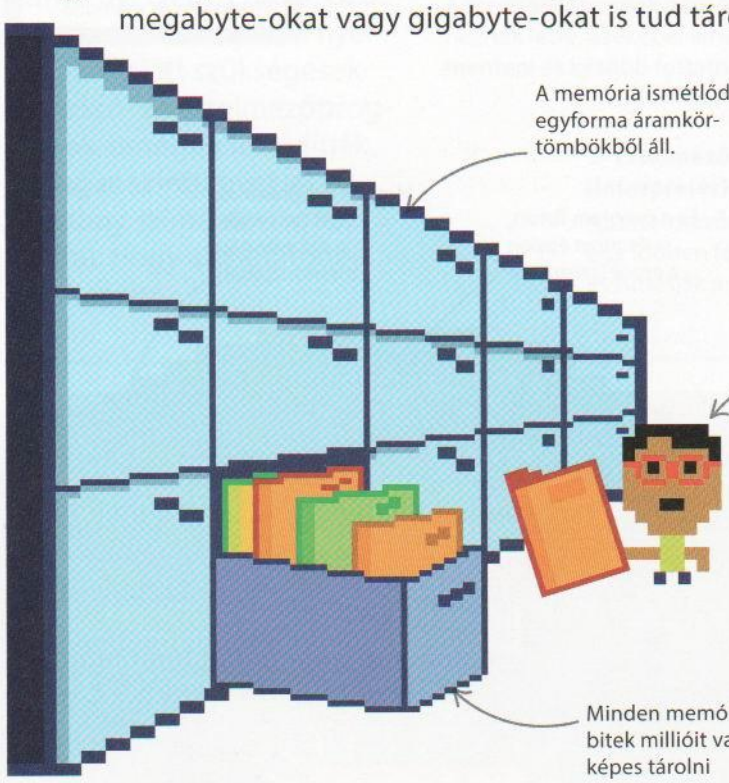
Memória

A processzorokhoz hasonlóan a memóriachipek is szilíciumalapra kerülnek. Logikai kapuk egyesítésével bistabil áramkör (flipflop) jön létre. Minden ilyen áramkör egy bitet tárol (ez az információ legkisebb egysége, értéke 1 vagy 0 lehet), és sok bistabil áramkör együttesen akár megabyte-okat vagy gigabyte-okat is tud tárolni.

FOGALOMTÁR

RAM

A memóriát sokszor RAM-nak (Random Access Memory) is nevezzük. Ez azt jelenti, hogy a memória bármely része közvetlenül elérhető. A korai adattárolók csak sorban tudták elérni az adatokat, így aztán nehézkesen működtek.

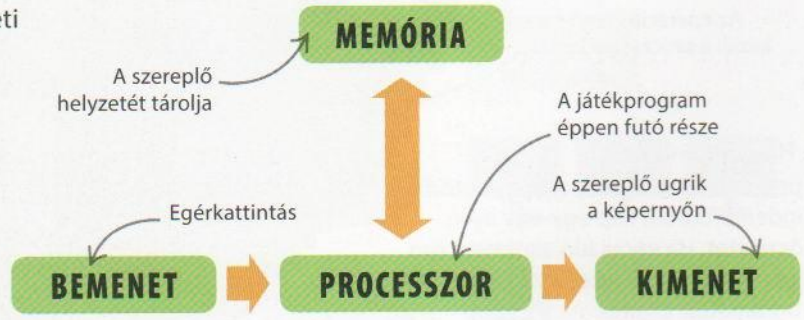


◁ **Programok és adatok**
A programok folyamatosan olvassák, írják és frissítik a memóriában tárolt adatokat.

JEGYEZD MEG!

Az információ feldolgozása

A processzor és a memória a bemeneti és a kimeneti eszközökkel együtt alkotják a számítógép lényegét. Egy játékprogramban például a felhasználó bemeneti adatot küld az egérrel, a processzor elvégzi a számításokat, olvassa és írja a memóriát, és végül kimenetet hoz létre: például egy szereplő ugrik egyet a képernyőn.



Alapvető programok

Vannak programok, amelyekre minden számítógépnek szüksége van a működéshez. A legfontosabbak az operációs rendszerek, a fordító- és az értelmező-programok.

LÁSD MÉG

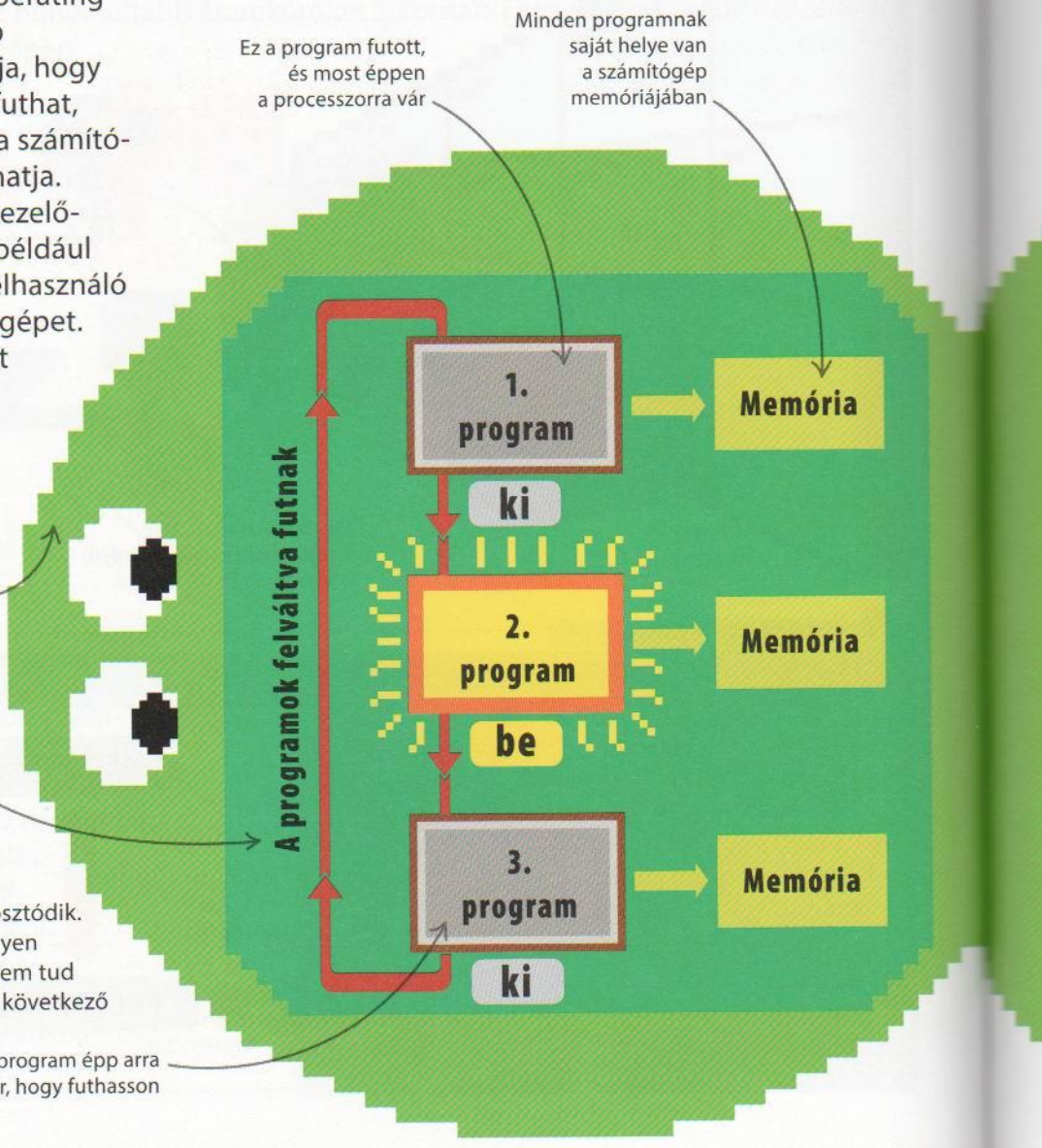
❮ 180–181
A számítógép belülről

❮ 182–183 A kettes és más számrendszerek

❮ 188–189
Processzor és memória

Operációs rendszer

Az operációs rendszer (Operating System, OS) a számítógép erőforrásait kezeli. Irányítja, hogy melyik program meddig futhat, illetve hogy futása során a számítógép mely részeit használhatja. Az operációs rendszer a kezelőfelületet is biztosítja, így például a fájlkezelőt, amellyel a felhasználó irányítani tudja a számítógépet. A leggyakrabban használt operációs rendszerek a Microsoft Windows, az Ubuntu Linux és az Apple Mac OS X.

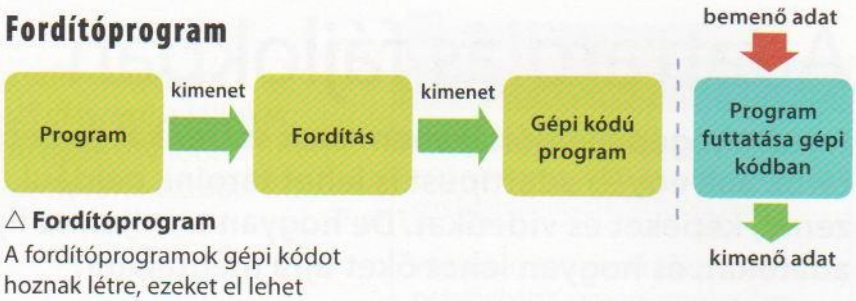


▷ **Hogyan működik?**

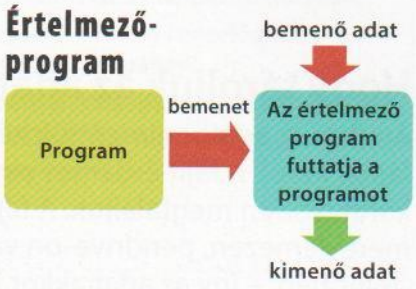
A processzoridő kis szeletekre osztódik. Minden program kap egy-egy ilyen időszeletet. Ha ennyi idő alatt nem tud végezni, akkor szünetel, amíg a következő program fut.

Fordító- és értelmezőprogramok

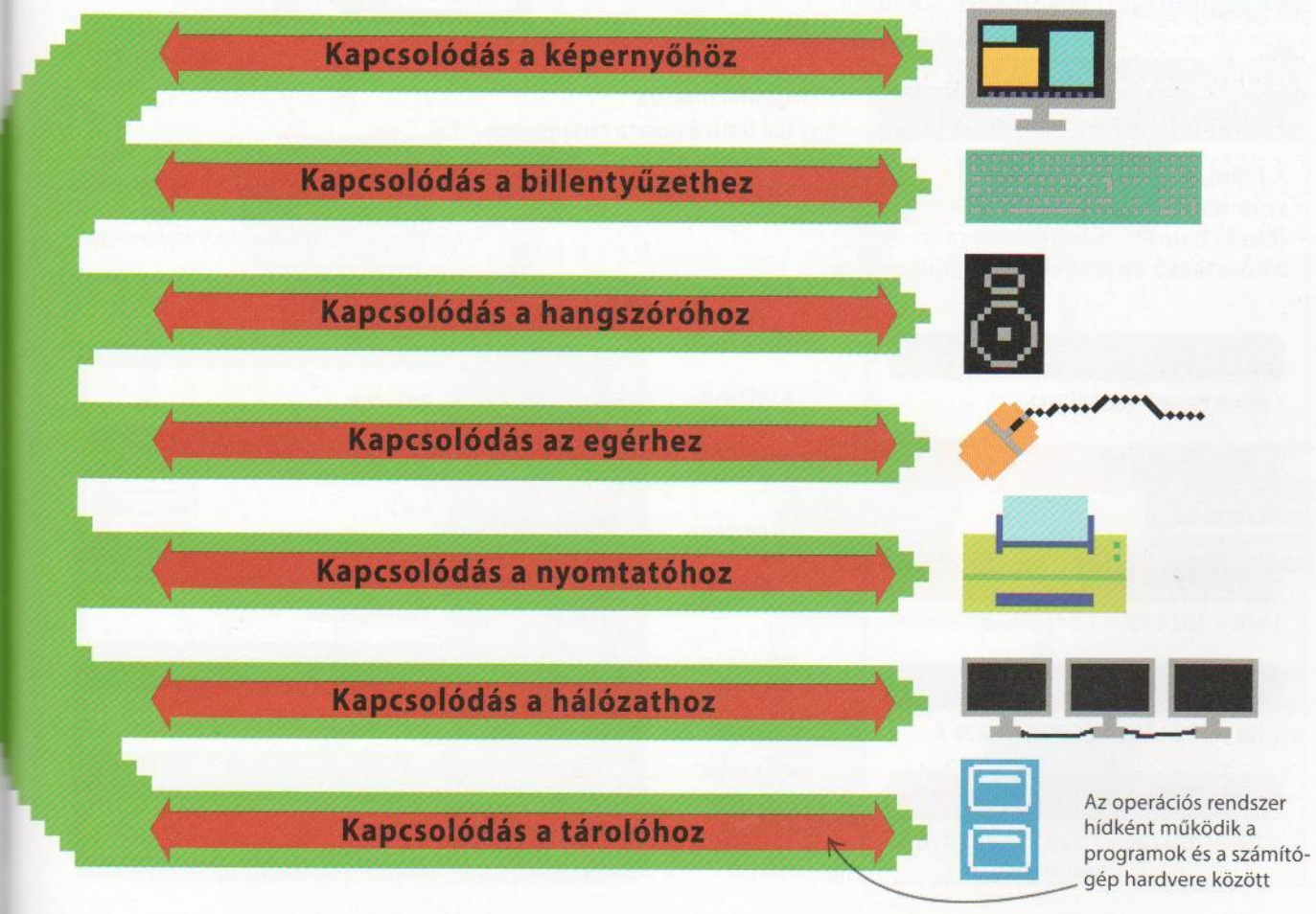
A programok írásához használt nyelvek, mint például a Python, úgynevezett magas szintű nyelvek. A processzorok nem értik ezeket a nyelveket, emiatt szükségesek a fordító- és értelmezőprogramok, amelyek lefordítják a magas szintű nyelveket alacsony szintűekre (gépi kódra), hogy a számítógép is megértse.



△ **Fordítóprogram**
A fordítóprogramok gépi kódot hoznak létre, ezeket el lehet menteni és később futtatni.



▷ **Értelmezőprogram (interpreter)**
Az értelmezőprogramok egy időben fordítják le és futtatják a programot.



Adattárolás fájlokban

A számítógép memóriája nem csak számokat és betűket tárol. Sok egyéb adattípust is lehet tárolni, például zenét, képeket és videókat. De hogyan tároljuk az ilyen adatokat, és hogyan lehet őket újra megtalálni?

LÁSD MÉG

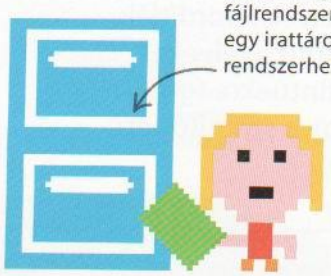
‹ 182–183 A kettes és más számrendszerek

‹ 188–189 Processzor és memória

‹ 190–191 Alapvető programok

Hogy tároljuk az adatokat?

Amikor az adatot elmentjük későbbi használatra, fájlba kerül. A fájlt érdemes elnevezni, hogy könnyebben megtaláljuk. A fájlok tárolhatók merevlemezen, pendrive-on vagy akár online „felhőben" – így az adat akkor is biztonságban van, ha a számítógép ki van kapcsolva.



A számítógép fájlrendszere hasonlít egy irattároló rendszerhez

TANÁCSOK

Fájlméretek

A fájlok lényegében kettes számrendszerbeli számjegyek (bitek) halmazai. A fájlok méretét a következő egységekben mérjük:

Byte (B)

1 B = 8 bit (pl. 10011001)

Kilobyte (KB)

1 KB = 1024 B

Megabyte (MB)

1 MB = 1024 KB = 1 048 576 B

Gigabyte (GB)

1 GB = 1024 MB = 1 073 741 824 B

Terrabyte (TB)

1 TB = 1024 GB = 1 099 511 627 776 B

▽ **Fájlinformációk**

Egy fájl több a puszta tartalmánál. A fájl tulajdonságai mindent tartalmaznak, amit a rendszernek tudnia kell róla.

Kattints jobb egérgombbal a fájl nevére, hogy megnézd a tulajdonságait

FÁJL TULAJDONSÁGAI

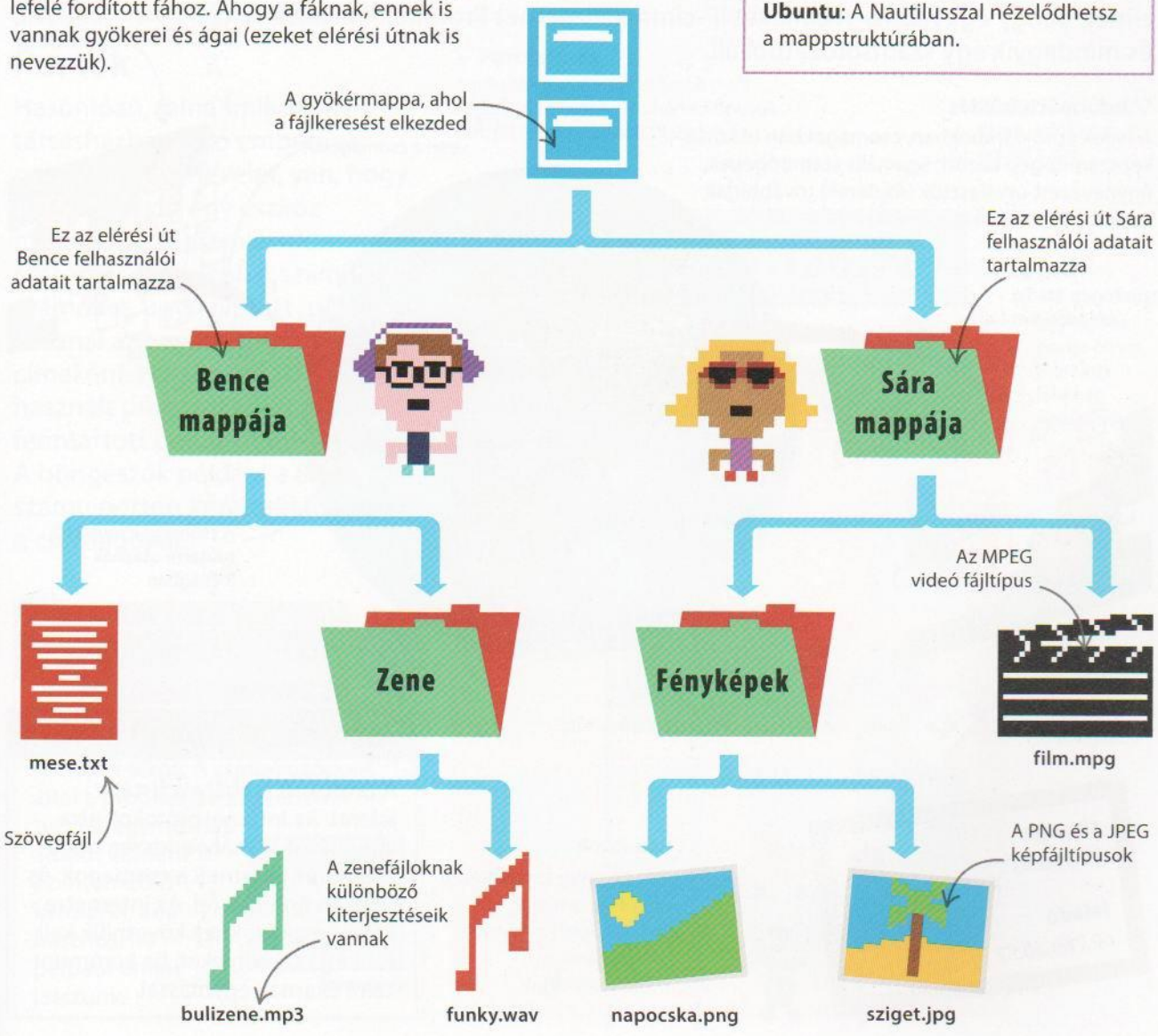
A fájlnév legyen könnyen megjegyezhető → **név**	bulizene
A fájl típusa, jellemzően három karakterben → **fájl-kiterjesztés**	mp3
A program, amely a fájl adatait kezelni tudja → **társítás**	Zenelejátszó
A fájl helye a számítógépen → **teljes elérési út**	/Felhasználók/Bence/Zene
A fájl mérete (*lásd* balra) → **méret**	50 MB

Mappák

Könnyebb megtalálni a fájlokat a számítógépen, ha jól vannak rendszerezve. Emiatt célszerű a fájlokat mappákba rendezni. Gyakran a mappák további mappákat tartalmaznak, úgynevezett fastruktúrát alkotnak.

▽ Mappastruktúra

Amikor a mappákat másik mappába mentjük, egy olyan struktúra jön létre, amely hasonlít egy fejjel lefelé fordított fához. Ahogy a fáknak, ennek is vannak gyökerei és ágai (ezeket elérési útnak is nevezzük).



TANÁCSOK

Fájlkezelés

A fájlkezelő program segít a fájlok és mappák keresésében. Minden operációs rendszernek más fájlkezelője van:

Windows: A Windows Explorerrel nézelődhetsz a mappastruktúrában.

Apple: A Finderrel nézelődhetsz a mappstruktúrában.

Ubuntu: A Nautilusszal nézelődhetsz a mappstruktúrában.

Adathordozás

Mielőtt a csomagokat elküldik az eszközök, nagy távolságok megtételére képes bináris jelekké (1 és 0) kell őket alakítani. Az interneten minden eszköznek van egy hálózati adaptere, amely elvégzi ezt a feladatot. Az egyes eszközök más-más módon küldenek adatokat.

△ **Elektromos jelek**
A rézdrótok az egyeseket és a nullákat különböző erősségű elektromos jelekként hordozzák.

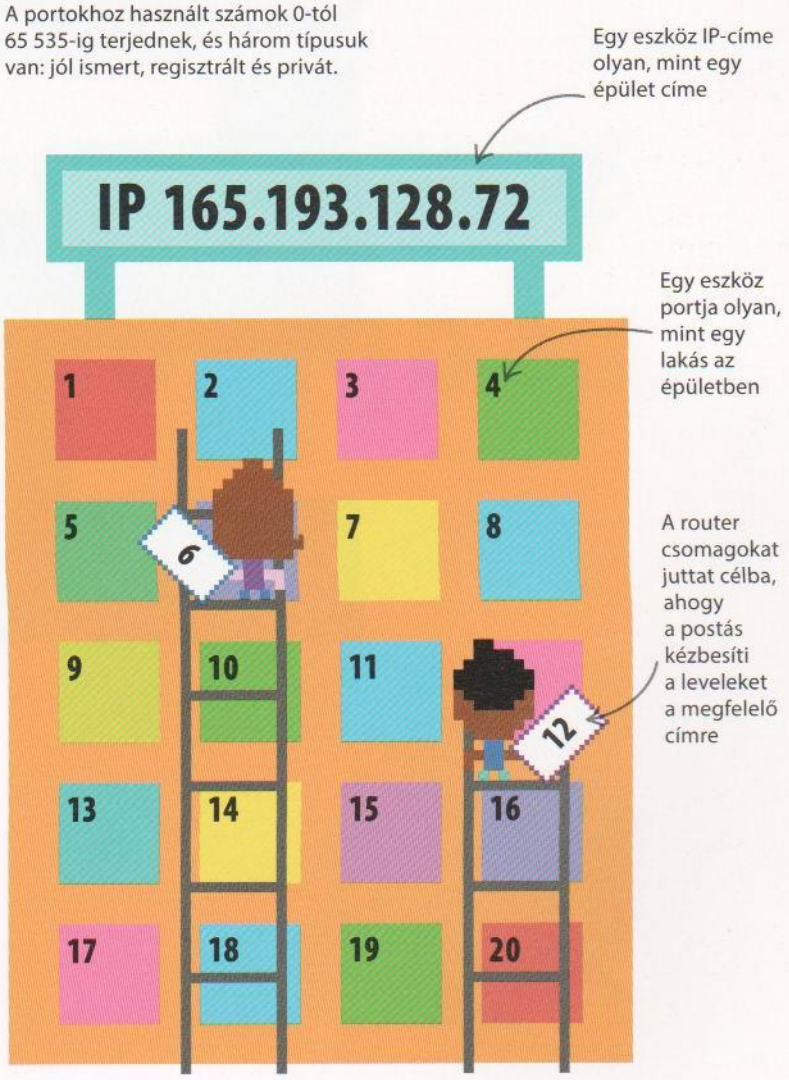
△ **Fény**
A különleges üvegszálas, úgynevezett optikai kábelek fényimpulzusokkal továbbítják az adatokat.

△ **Rádióhullámok**
A különböző típusú rádióhullámok kábel nélkül is tudják továbbítani az egyeseket és a nullákat.

Portok

Hasonlóan, mint amikor egy társasházban lakó embernek postázunk egy levelet, van, hogy a csomagokat egy eszköz megadott programjának szeretnénk küldeni. A számítógép számokat, úgynevezett „portokat" használ az egyes programok címeként. Néhány gyakran használt program külön számára fenntartott porttal rendelkezik. A böngészők például a 80-as számú porton keresztül fogadják a csomagokat.

▽ **Portszámok**
A portokhoz használt számok 0-tól 65 535-ig terjednek, és három típusuk van: jól ismert, regisztrált és privát.



TANÁCSOK

Csatlakozópontok (socketek)

Egy IP-cím és egy port együttesen egy csatlakozópontot, más néven socketet alkot. A csatlakozópont által a programok közvetlenül tudnak egymásnak adatot küldeni az interneten keresztül, ami hasznos, ha például online játszunk.

Programozás a valóságban

Programnyelvek

Több ezer különféle programozási nyelv létezik. Az, hogy adott esetben melyiket célszerű használni, sok tényezőtől függ, például a feladattól, illetve a felhasználási környezettől.

LÁSD MÉG
Számítógépes játékok 204–205 ›
Appok készítése 206–207 ›

Népszerű programozási nyelvek

Néhány programnyelv népszerűvé vált bizonyos feladattípusok és adott számítógépes környezet esetén. Itt láthatod az egyszerű „Hello World!" szöveget kiíró programot néhány népszerű nyelven megírva.

```
#include <stdio.h>
main()
{
    printf("Hello World!");
}
```

△ **C**
Minden idők egyik legnépszerűbb programnyelve. Gyakran használják hardverprogramozásra.

```
#include <iostream>
int main()
{
    std::cout << "Hello World!" << std::endl;
}
```

△ **C++**
A C nyelv továbbfejlesztett változata. Olyankor használják, ha a program sebessége fontos, például játékkonzolok esetén.

```
#import <stdio.h>
int main(void)
{
    printf("Hello World!");
}
```

△ **Objective-C**
A C nyelv továbbfejlesztett változata. Népszerű, mert Apple Mac és iOS eszközökön használható.

```
alert('Hello World!');
```

△ **JavaScript**
Internetes böngésző-alkalmazások (pl. játékok vagy levelezőprogramok) készítésére használják.

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

△ **Java**
Sokoldalú nyelv, amely szinte minden számítógépen fut. Gyakran használják Android platformon.

```
<?php
echo "Hello World!";
?>
```

◁ **PHP**
Főleg interaktív weboldalak készítésére használják. A PHP a honlapokat kiszolgáló webszervereken fut.

Régi programnyelvek

Sok olyan programnyelv van, amely nagyon népszerű volt húsz vagy harminc évvel ezelőtt, de ma már egyáltalán nem vagy csak ritkán, használják. Mai szemmel ezek a programnyelvek bonyolultnak, nehézkesnek tűnnek.

BASIC Az amerikai Dartmouth College-ban 1964-ben kifejlesztett BASIC nyelv a személyi számítógépek elterjedésének idején volt nagyon népszerű.

Fortran Az IBM által 1954-ben, elsősorban a nagy (mainframe) számítógépek számára kifejlesztett nyelvet ma is használják, például az időjárás-előrejelzés területén.

COBOL A COBOL nyelvet egy szakértői csapat fejlesztette 1959-ben, és ma is használatos bizonyos üzleti és banki alkalmazásokban.

VALÓS VILÁG

A millennium bug

Régebbi programnyelveken, mint például a COBOL-ban megírt programok csak két számjegyet használtak az évszám tárolására, így például 1999 helyett csak 99-et írtak. A 2000. évi ezredforduló közeledtével súlyos hibákat jósoltak a szakemberek az utolsó két számjegy 00-ra való váltása miatt.

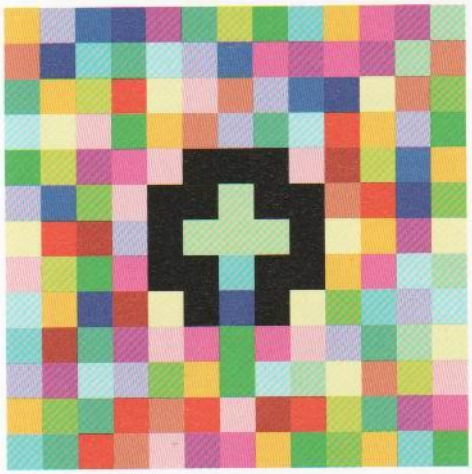
Világszerte jelentős összegekért frissítették a számítógépek rendszereit, hogy elkerüljék ezt a hibát

Fura programnyelvek

A sok ezer programnyelv között van néhány, amely igencsak különös, meglepő céllal készült.

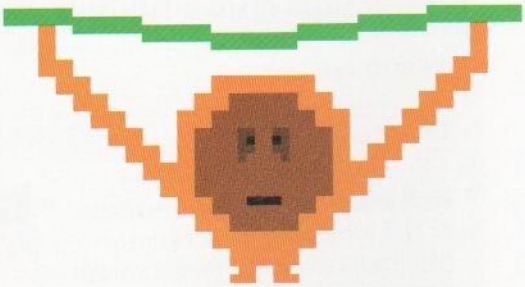
```
('&%:9]!~}|z2Vxwv-,POqponl$Hjig%eB@@>a=<M:9[p6tsl1TS/
QlOj)L(I&%$""Z~AA@UZ=RvttT`R5P3m0LEDh,T*?(b&`$#87[]{W
```

△ **Malbolge**
Arra tervezték, hogy gyakorlatilag lehetetlen legyen programot írni vele. Az első Malbolge-program csak két évvel a megjelenése után készült el, méghozzá egy másik program által.



△ **Piet**
A Piet nyelven készített programok úgy néznek ki, mint egy modern műalkotás. Így néz ki a „Hello World!" szöveget kiíró program.

△ **Chef**
A Chef nyelven írt programok ételrecepteknek néznek ki. A gyakorlatban azonban ritkán sül ki belőle finom ennivaló.



△ **Ook!**
Az orangutánok számára készített programnyelv mindössze három elemből áll: „Ook", „Ook!" és „Ook?". Ezekből lehet olyan parancsokat képezni, mint pl.: „Ook! Ook" vagy „Ook? Ook!"

Sztárprogramozók

Az informatika fejlődését világszerte programozók milliói alakítják, de időnként egy-egy különleges személyiség kitűnik a tömegből. Íme néhány a leghíresebb programozók közül.

LÁSD MÉG
‹ 18–19
Legyél te is programozó
Számítógépes játékok
204–205 ›

Ada Lovelace

Nemzetisége: brit

Élt: 1815–1852

Miről híres: Ada Lovelace-t minden idők első programozójaként tartják számon. 1843-ban készítette első programját Charles Babbage analitikus gépére – ezt tartjuk a mai számítógépek ősének. Kidolgozott egy módszert a karakterek számokkal való ábrázolására.

Alan Turing

Nemzetisége: brit

Élt: 1912–1954

Miről híres: Alan Turing brit matematikust tartják a modern programozás megalapítójának. Hatalmas jelentőségű volt az a munkája, amikor egy titkos német kódot fejtett meg a britek számára a II. világháborúban.

Grace Hopper

Nemzetisége: amerikai

Élt: 1906–1992

Miről híres: Grace Hopper készítette az első fordítóprogramot (amely gépi kódra fordít egy magas szintű programnyelvről). Programozói munkássága mellett az amerikai haditengerészet ellentengernagya volt.

Bill Gates és Paul Allen

Nemzetiségük: amerikai

Éltek: Gates 1955–, Allen 1953–

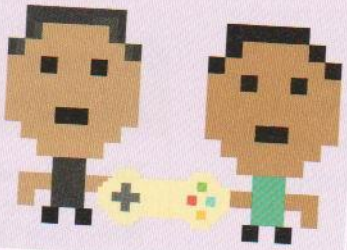
Miről híresek: Együtt alapították a Microsoftot az 1970-es években. Az általuk létrehozott Microsoft Windows és Office minden idők legnépszerűbb programjai közé tartoznak.

Jokoi Gunpei és Mijamoto Sigeru

Nemzetiségük: japán

Éltek: Jokoi 1941–1997, Mijamoto 1952–

Miről híresek: Mindketten a Nintendo játékgyárnak dolgoztak. Jokoi alkotta meg a Game Boyt, Mijamoto pedig olyan népszerű játékokat készített, mint például a Super Mario.



Tim Berners-Lee

Nemzetisége: brit

Élt: 1955–

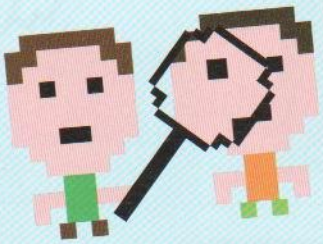
Miről híres: A híres svájci kutatóintézet, a CERN munkatársaként Tim Berners-Lee alkotta meg és tette mindenki számára elérhetővé a világhálót, azaz a World Wide Webet. II. Erzsébet 2004-ben lovaggá ütötte.

Larry Page és Sergey Brin

Nemzetiségük: amerikai

Élt: mindketten 1973–

Miről híresek: Page és Brin 1996-ban kezdtek dolgozni a Google első keresőjén. A különlegesen hatékony keresőmotor később forradalmasította az internetet.



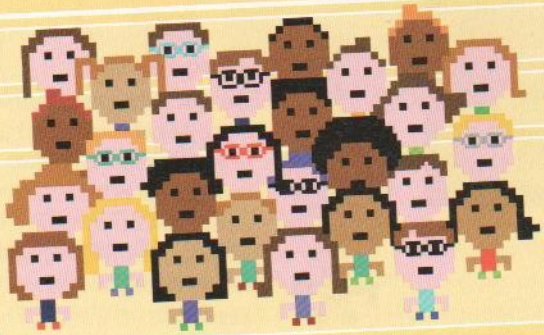
Mark Zuckerberg

Nemzetisége: amerikai

Élt: 1984–

Miről híres: Zuckerberg 2004-ben indította el a kollégiumi szobájából az azóta már több milliárd dollárt érő Facebookot. Zuckerberg jelenleg az egyik leggazdagabb ember a világon.

Mozgalom a nyílt forráskódért



Nemzetisége: nemzetközi

Működik: az 1970-es évek végétől napjainkig

Miről híres: A nyílt forráskódért küzdő mozgalom olyan programozókat tömörít világszerte, akik azt vallják, hogy a programoknak mindenki számára ingyenesen hozzáférhetőnek kell lenniük. Olyan jelentős programokat alkottak meg, mint a Linux operációs rendszer vagy a Wikipédia.

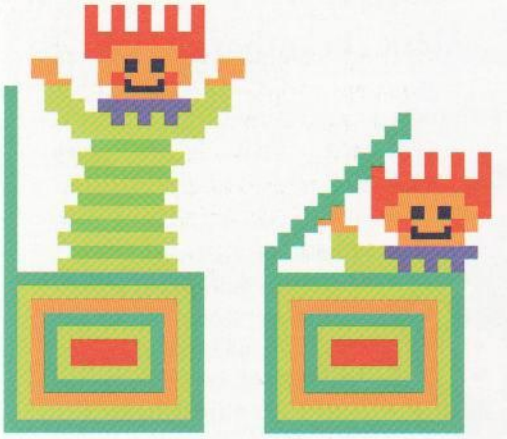
Fontos programok

A számítógépek és a programok a mindennapjaink szerves részévé váltak. Nap mint nap használunk összetett programokat, amelyeket komoly feladatok megoldására írtak.

LÁSD MÉG
❮ 180–181
A számítógép belülről
❮ 192–193
Adattárolás fájlokban

Fájlok tömörítése

Az interneten küldött szinte minden fájl valamilyen módon tömörítve van, hogy kisebb legyen a mérete. A tömörítés során a felesleges információkat eldobjuk, és csak a hasznosat tartjuk meg.

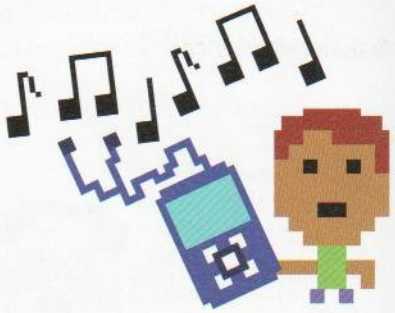


◁ **Adatok zsugorítása**
Egy fájl tömörítése olyan, mint amikor összenyomjuk a bohócot, hogy beleférjen a dobozba.

VALÓS VILÁG

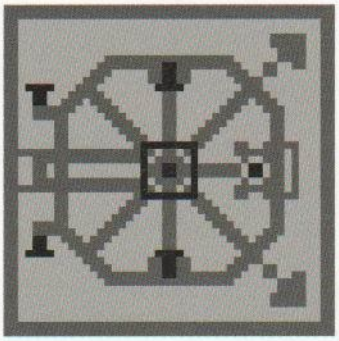
Zenefájlok

Tömörítés nélkül csupán néhány zeneszám férne a lejátszódra. Tömörített (pl. MP3) zenéből akár több ezer is elfér egy okostelefonon.



Titkos kódok

Ha bejelentkezünk egy weboldalra, vásárolunk az interneten, vagy e-mailt küldünk, speciális programok titkosítják a személyes adatainkat, nehogy illetéktelenek visszaéljenek vele. A nagy banki rendszerek is kivétel nélkül titkosított adatokkal dolgoznak.

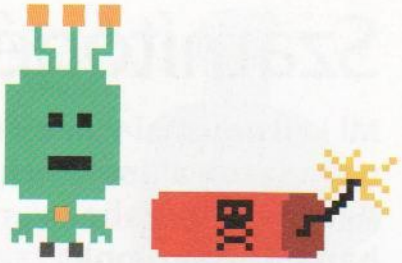


◁ **Kriptográfia**
A kriptográfia az informatika rejtjelekkel foglalkozó ága. Bonyolult matematikai módszereket alkalmaznak az adatok titkosítására és visszafejtésére, hogy azok ne kerüljenek rossz kezekbe.

Mesterséges intelligencia

A számítógép a játékok futtatásánál sokkal többre is képes. A mesterséges intelligencia hozzásegít a jobb orvosi ellátáshoz, vagy képes robotokat működtetni olyan helyeken, amelyek az emberek számára veszélyesek lennének (pl. háborús övezet, katasztrófa sújtotta vidék).

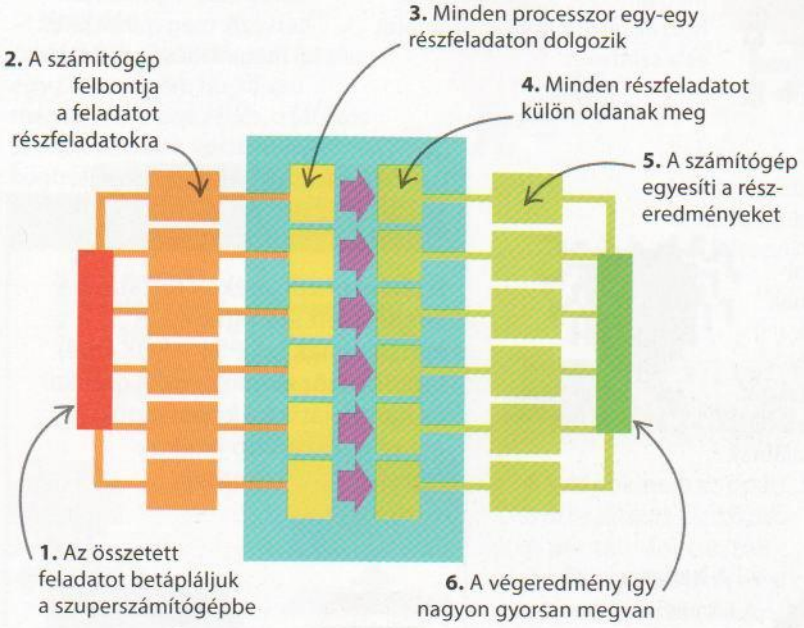
△ **Orvostudomány**
A fejlett rendszerek képesek hatalmas mennyiségű adatot tárolni és a páciens panaszai alapján részletes diagnózist felállítani.



△ **Bombák hatástalanítása**
Sok életet ment meg az az intelligens robot, amely képes hatástalanítani robbanótölteteket olyan területeken, ahol veszélyes lenne embernek tartózkodni.

Szuperszámítógépek

A szuperszámítógépek, kihasználva a processzorok közötti hatékony kommunikációt és gyors adatelérést, másodpercenként több milliárd számítási lépést tudnak elvégezni. Olyan tudományos-technológiai cégek használják őket, mint például a NASA, az amerikai űrkutatási hivatal.



△ **Hogyan működik?**
A megoldandó feladatot olyan részfeladatokra bontják, amelyeken egy időben tudnak dolgozni az egyes processzorok. A részeredmények összegzésével jön létre a megoldás.

VALÓS VILÁG

Időjárás-előrejelzés

Az időjárás pontos előrejelzéséhez hatalmas mennyiségű adatot kell feldolgozni. A szuperszámítógépek fel tudják osztani a feladatot, így minden processzor egy-egy kis földrajzi terület adatait számolja. Az eredmények egyesítésével létrejön a teljes előrejelzés.

Számítógépes játékok

Mi kell egy mai videojátékhoz? A számítógépes játékok ugyanazon alapelemek különféle keverékei. A legnépszerűbbeken nem csupán programozók, hanem egy komplett szoftverfejlesztő csapat dolgozik.

LÁSD MÉG
❮ 200–201
Sztárprogramozók
Appok készítése
206–207 ❯

Kik készítik a számítógépes játékokat?

A telefonodon lévő legegyszerűbb játékot is minden bizonnyal egy nagyobb fejlesztőcsapat készítette. Annak érdekében, hogy a játék sikeres legyen, minden apró részletre figyelni kell, ehhez különböző tudású emberek összehangolt munkája szükséges.

△ **A programozó**
A programozók írják a játék kódját, de ezt csak a csapat többi tagjától kapott információ alapján tudják megtenni.

◁ **A szinttervező**
A játék virtuális világának megálmodói, a szinttervezők hozzák létre a játékkörnyezetet és a szinteket.

△ **A tervezőgrafikus**
A játék minden szereplőjének és a háttereknek jól kell kinézniük. A grafikusok tervezik meg mindennek a megjelenését a játékban.

△ **A forgatókönyvíró**
Akárcsak egy könyv vagy film, egy népszerű játék sincs meg izgalmas történet nélkül. A forgatókönyvíró találja ki a szereplőket, és hogy miről szóljon a játék.

▷ **A tesztelő**
Egész nap játszani álomfoglalkozásnak tűnhet, de a játék tesztelése komoly feladat. Addig játsszák ugyanazt a szintet, amíg meg nem találnak minél több hibát.

◁ **A hangtervező**
A filmekhez hasonlóan, a játékok zenéje és hanghatásai is nagymértékben hozzájárulnak az adott játék hangulatához.

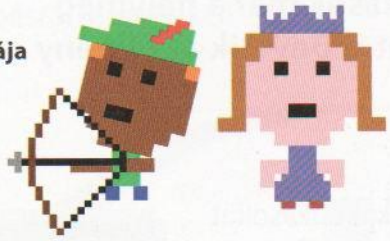
FOGALOMTÁR

Konzolok

A játékkonzol játékok futtatására kifejlesztett számítógép. A konzoloknak (pl. PS4, XBOX One) legtöbbször kiemelkedő a grafikai és hanglejátszási képességük, így egyre életszerűbb játékok futtatására alkalmasak.

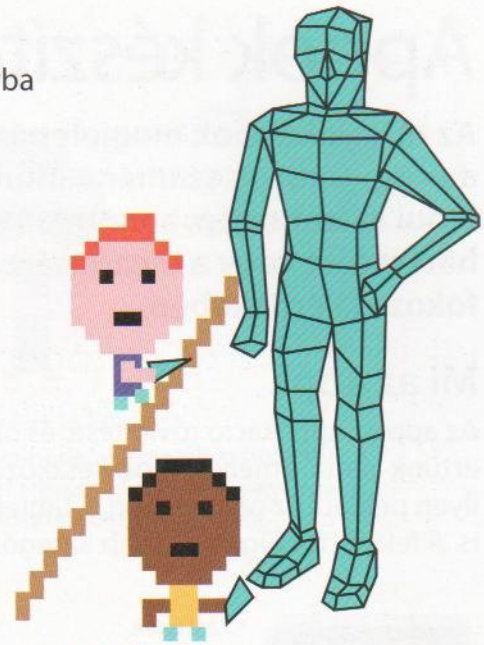
A játékok alapelemei

A játékok legfontosabb összetevőit gyakran egy játékmotorba (game engine) rakják össze. Az ilyen játékplatformok segítségével egyszerűbb új játékokat fejleszteni.



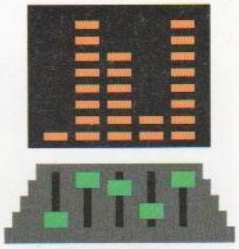
▷ A történet és a játék logikája
Minden játéknak kell hogy legyen célja (pl. a hercegnő megmentése) és sztorija. Jól kitalált játékszabályra nehezebb ráunni.

◁ A fizika szabályai
A virtuális világban is alkalmazni kell az olyan fizikai törvényszerűségeket, mint a gravitáció és az ütközések, hogy életszerű legyen a játék.



△ Grafika
Egyre összetettebb grafikára van szükség, hogy a játékok minél életszerűbbek legyenek. Az emberi mozdulatokat, a füstöt és a vizet különösen nehéz jól ábrázolni.

▷ Vezérlés
A kényelmes vezérlés már fél siker egy játék esetén. Ha jól van megtervezve a vezérlés, a játékos szinte észre sem veszi, hogy kontrollert használ.



▷ Hang
Minden párbeszédet, háttérzenét és hangeffektust jó minőségben kell felvenni.

△ A mesterséges intelligencia
Emberek gyakran játszanak számítógép-vezérelt szereplőkkel vagy ellenük. A mesterséges intelligencia révén életszerűbb válaszaik lesznek az ilyen szereplőknek.

VALÓS VILÁG

Komoly játékok

A játékok nem csupán a szórakozás eszközei. A pilóták, a sebészek és a katonák kiképzési céllal szimulátorokat használnak, és egyes vállalatok pedig stratégiai játékokkal fejlesztik a dolgozóik kreativitását.

Appok készítése

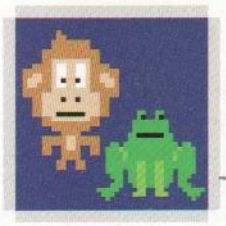
Az okostelefonok megjelenése új távlatokat nyitott a programozók számára. Miután mindenki zsebében lapul számítógép, a mobilalkalmazások már a helymeghatározást vagy a mozgásérzékelőt is bevetik az élmény fokozása érdekében.

LÁSD MÉG
〈 190–191 A legfontosabb programok
〈 198–199 Programnyelvek
〈 204–205 Számítógépes játékok

Mi az app?

Az app az applikáció rövidítése, és olyan alkalmazásokat értünk alatta, amelyek mobil eszközökre lettek kifejlesztve. Ilyen például az okostelefon, a tablet vagy akár az okosóra is. A feladattól függően több kategóriájuk is létezik.

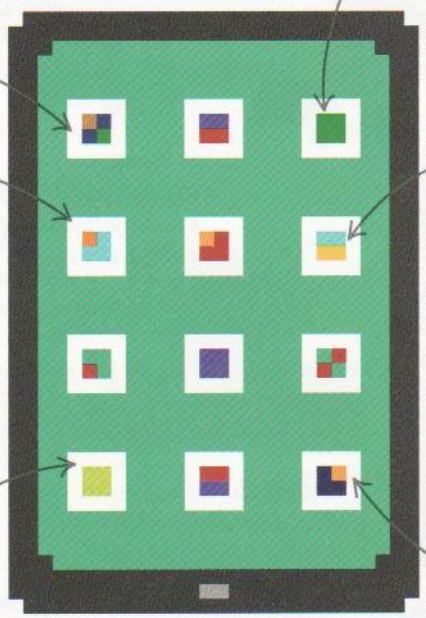
◁ **Közösségi háló**
A közösségi appok által közel vagy távol levő barátainkkal tudunk csevegni vagy képeket, zenéket, videókat megosztani.



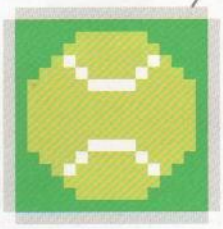
◁ **Játékok**
Az egyszerű rejtvényektől a gyors akciójátékig mindenféle játék elérhető mobil eszközökre.

△ **Utazás**
Az utazási app a helyzetünk és mások véleménye alapján ad ajánlásokat étteremre, szállodára vagy szabadidős programra.

△ **Időjárás**
Az ilyen appok segítségével pontos időjárás-előrejelzéseket kaphatunk jelenlegi tartózkodási helyünkre vagy a világ bármely más pontjára.



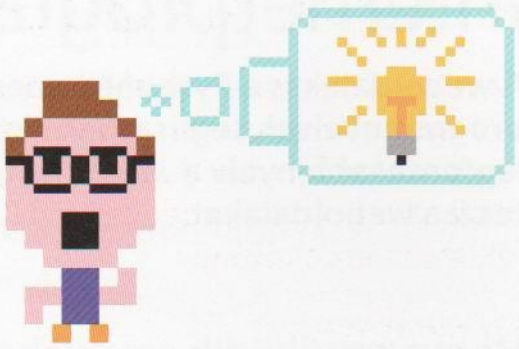
△ **Oktatás**
Az ismeretterjesztő appokból sok mindent megtudhatunk. A kisebb gyerekek számolni meg írni tanulhatnak, a nagyobbak pedig akár nyelveket is.



◁ **Sport**
Vannak, akik appokat használnak edzettségük nyomon követésére futás vagy bringázás során, mások a legfrissebb sporteredményeket figyelik.

Hogyan kell appot készíteni?

Sok kérdést kell megválaszolni, mielőtt hozzálátnánk egy app elkészítéséhez. Mi a cél? Milyen eszközökön fogják használni? Hogyan fogja kezelni a felhasználó? Az appot ezen válaszok ismeretében lépésről lépésre kell elkészíteni.



1 Van egy egy jó ötleted?
Minden appötletnek mobilbarátnak kell lennie. Ez lehet teljesen új gondolat vagy egy meglevő app továbbgondolt változata.

Mac

Android

Windows

2 Melyik operációs rendszer (platform)?
Csak bizonyosfajta mobil eszközön szeretnénk futtatni az appot? Léteznek olyan fejlesztőeszközök, amelyek az egyszer leprogramozott appot lefordítják az egyes platformok számára.

3 Tanulj meg appot készíteni!
Akármelyik platformról legyen szó, egy jó app elkészítéséhez alaposan meg kell tanulni az adott nyelv sajátosságait. Az online segédletek és a programozói fórumok sokat segíthetnek.

4 Írd meg a programot!
Egy jó app elkészítéséhez idő kell. Egy alapverzió akár hetek alatt összehozható, de egy igazán sikeres alkalmazás elkészítése hónapokig tart.

5 Teszteld!
Ha egy app hemzseg a hibáktól, a felhasználók hamar lemondanak róla. Már a megjelenés előtt megszabadulhatunk a hibáktól, ha teszteket is írunk a kódba, illetve ha megkérjük barátainkat az elsődleges tesztelésre.

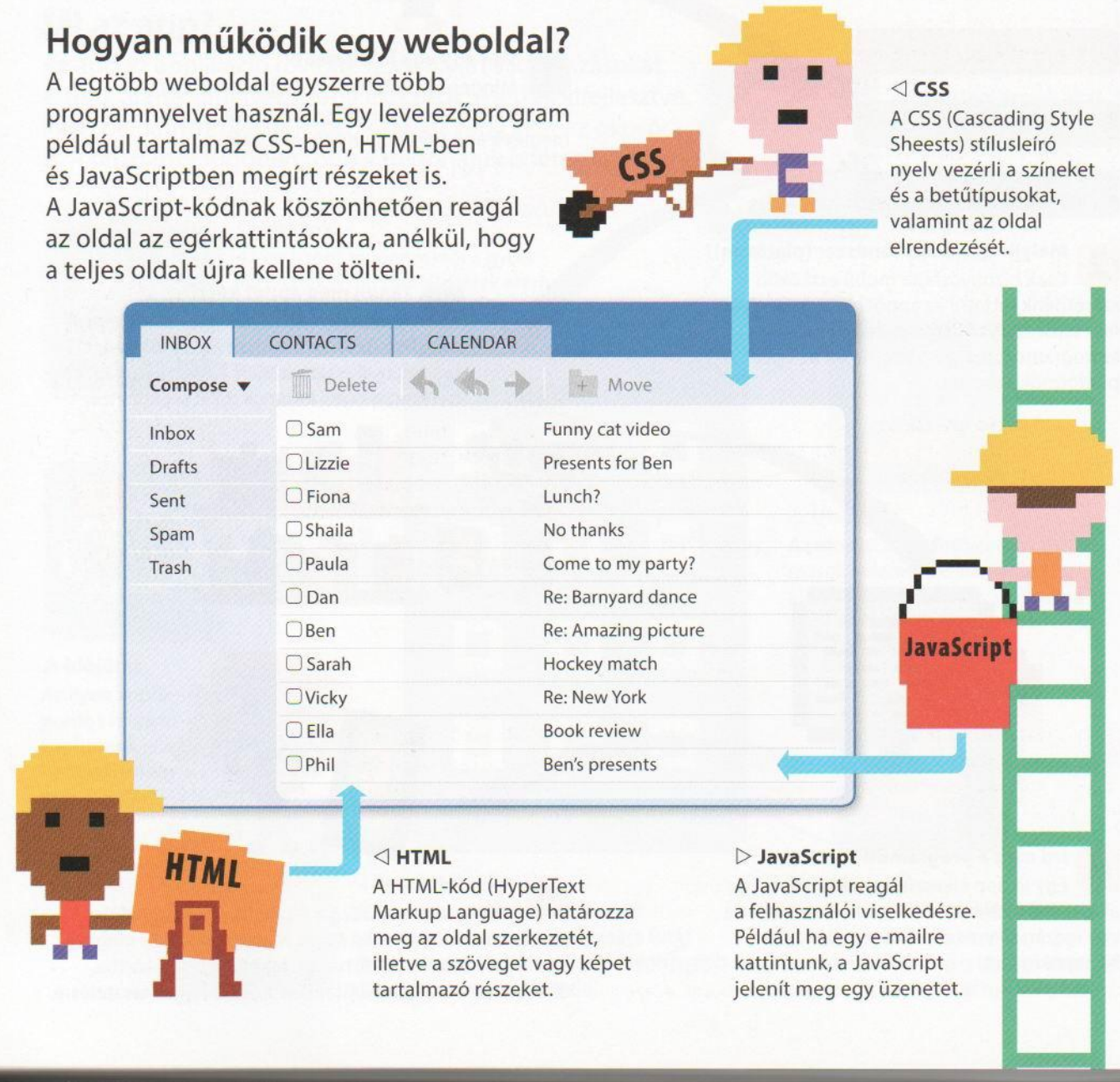
Internetprogramozás

A weboldalak is a Pythonhoz némileg hasonló programnyelvek segítségével készülnek. Az egyik legfontosabb nyelv a JavaScript, amely interaktívvá teszi a weboldalakat.

LÁSD MÉG
❮ 198–199
Programnyelvek
A JavaScript használata
210–211 ❯

Hogyan működik egy weboldal?

A legtöbb weboldal egyszerre több programnyelvet használ. Egy levelezőprogram például tartalmaz CSS-ben, HTML-ben és JavaScriptben megírt részeket is. A JavaScript-kódnak köszönhetően reagál az oldal az egérkattintásokra, anélkül, hogy a teljes oldalt újra kellene tölteni.



◁ **CSS**
A CSS (Cascading Style Sheests) stílusleíró nyelv vezérli a színeket és a betűtípusokat, valamint az oldal elrendezését.

◁ **HTML**
A HTML-kód (HyperText Markup Language) határozza meg az oldal szerkezetét, illetve a szöveget vagy képet tartalmazó részeket.

▷ **JavaScript**
A JavaScript reagál a felhasználói viselkedésre. Például ha egy e-mailre kattintunk, a JavaScript jelenít meg egy üzenetet.

HTML

Amikor megnyitunk egy weboldalt, a böngészőnk először egy HTML-fájlt tölt le. A benne lévő kód utasításai alapján építi fel az adott weboldalt. Megfigyelheted, hogyan működik mindez, ha beírod ezt a kódot az IDLE kódablakába (*lásd* 92-93. o.), és elmented „.html" kiterjesztéssel. Ha duplán rákattintasz, elindítja a böngésződet, benne a „Hello World!" üzenettel.

```
<html>
 <head>
  <title>A "Hello World!" ablaka</title>
 </head>
 <body>
  <h1>"Hello World!" HTML-ben</h1>
  <p>Hello World!</p>
 </body>
</html>
```

A szövegblokkokat a HTML-ben címkék, azaz „tag"-ek fogják közre. Ez a címke adja meg az ablak fejlécét.

A és címkék egy-egy bekezdést fognak közre.

Ez a címke jelzi a HTML-kód végét.

A JavaScript kipróbálása

A JavaScripttel egyszerű kísérletezni, mert minden mai böngésző értelmezni tudja. A JavaScript-kódot általában HTML-kódba helyezik, így a lenti példa két programnyelvet használ egyszerre. A JavaScript részt <script> címke határolja.

1 Írj JavaScript-kódot!

Nyiss új ablakot az IDLE-ben, és írd be az alábbi kódot. Gondosan ellenőrizd, amit beírtál, mert ha hiba marad benne, csak egy üres oldal fog megjelenni.

```
<script>
alert("Hello World!");
</script>
```

A <script> címke jelzi a JavaScript-kód elejét

Az „alert" utasítás hatására megjelenik egy figyelmeztető ablak

2 Mentsd el a fájlt!

Mentsd el a fájlt, és nevezd el például „teszt.html"-nek. Így HTML- és nem Python-fájlként lesz elmentve.

test.html

Ne felejts el „.html" kiterjesztést adni a fájlnak!

3 A felbukkanó ablak megjelenik

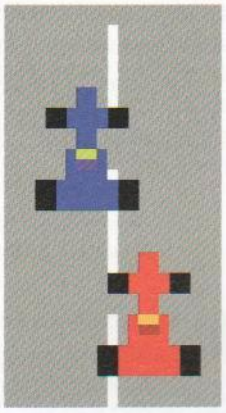
Megnyílik a böngésző egy felbukkanó ablakkal és „Hello World!" felirattal. Kattints az „OK" gombra az ablak bezárásához.

A JavaScript olyan interaktív elemeket is kezel, mint ez a gomb

TANÁCSOK

Játékok JavaScriptben

A JavaScript annyira jól kezeli az interaktivitást, hogy kiválóan használható a legkülönfélébb játékok készítésére, az egyszerű rejtvényektől az autóversenyig. Ezek nem igényelnek telepítést, mert a böngészőablakban futnak. A JavaScript használható levelezőprogram vagy interaktív naptár készítésére is.



A JavaScript használata

A JavaScripttel remekül lehet a HTML-en belül kisebb programokat létrehozni, amelyek életet lehelnek a web-oldalakba, és interaktívvá teszik őket. A JavaScript hasonlít a Pythonra, de a kód tömörebb, és nehezebb megtanulni.

LÁSD MÉG

‹ 162–163
Események kezelése

‹ 122–123
Ciklusok a Pythonban

‹ 208–209
Internetprogramozás

Kérdés-válasz

Ahogyan a Python, a JavaScript is alkalmas arra, hogy kérdezzen a felhasználótól. A JavaScript ezt felbukkanó ablakkal teszi meg. Az alábbi program megkérdezi a felhasználó nevét, majd köszönéssel válaszol.

1 Használj kérdezőablakot!

Ez a rövid program elmenti a felhasználó nevét egy változóban. Írd be a kódot az IDLE-kódablakba, és mentsd el „.html" kiterjesztéssel.

```
<script>
var név = prompt("Mi a neved?");
var üdvözlés = "Szia, " + név + "!";
document.write(üdvözlés);
</script>
```

Ez a sor jeleníti meg a kérdezőablakot, és menti a választ egy változóba

Az idézőjelek közé írt szöveg jelenik meg az ablakban

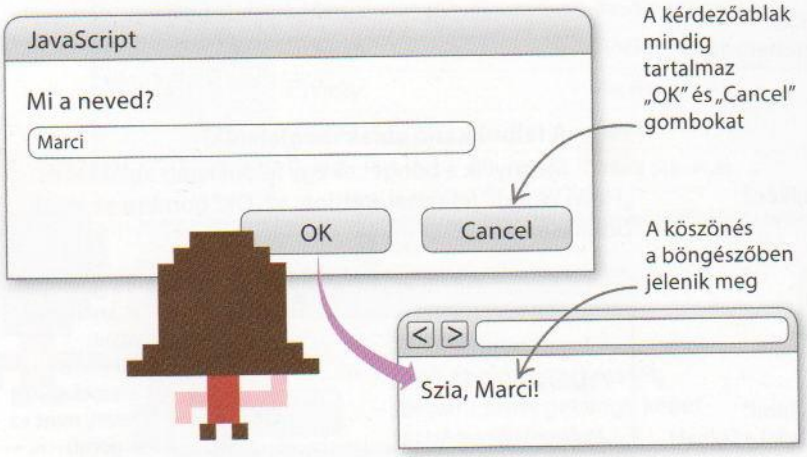
A JavaScript utasításait pontosvesszővel kell lezárni

A </script> tag jelzi a JavaScript-kód végét

Ez a sor jeleníti meg az üdvözlést

2 Megjelenik a kérdés

Kattints kétszer a HTML-fájlra a futtatásához. Írd be a neved a böngészőben megjelenő ablakba, és nyomd meg az „OK" gombot az üdvözlés megtekintéséhez.



A kérdezőablak mindig tartalmaz „OK" és „Cancel" gombokat

A köszönés a böngészőben jelenik meg

TANÁCSOK

Gépelj figyelmesen!

Ha JavaScriptben dolgozol, különösen ügyelj arra, hogy mindent pontosan írj! Ha hiba van a kódban, a böngésző üres oldalt fog megjeleníteni, anélkül hogy bármilyen üzenetben figyelmeztetne a hiba jellegére. Ha ilyen történik, figyelmesen nézd át a kódot!

Események

Esemény minden olyan történés, amelyet a számítógép érzékelni tud (pl. egérkattintás vagy billentyű lenyomása). Az eseménykezelő az a kódrészlet, amely reagál egy eseményre. A JavaScriptben gyakori az eseménykezelők használata, ettől érdekessé és interaktívvá válnak a weboldalaink.

1 Írd be a kódot!

Ebben a példában egy esemény (gombra kattintás) elindít egy egyszerű függvényt, amely kiír egy versikét. Írd be a kódot IDLE kódablakba, és mentsd el „.html" kiterjesztéssel.

```
<button onclick="versike()">Mondd ezt!</button>
<script>
 function versike()
 {
  document.write("Verebet fogtam, koppantottam");
 }
</script>
```

A függvény neve

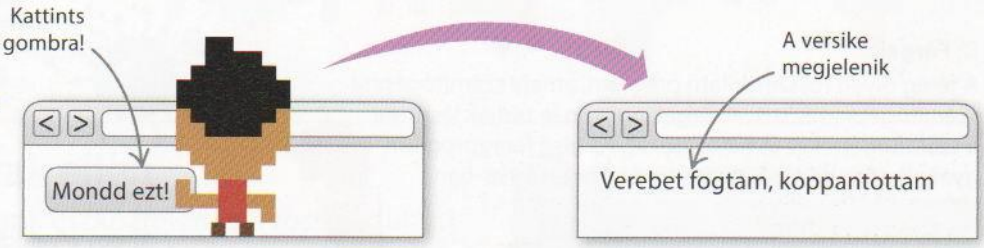
A HTML-kód hozzárendel egy függvényt egy gombhoz

A kód blokkjait kapcsos zárójelek veszik közre, a szerepük hasonló, mint a behúzásé a Pythonban

A JavaScript-kód definiálja a függvényt

2 Futtasd a programot!

Kattints duplán a HTML-fájlra, hogy a program elinduljon a böngészőben.



Ciklusok a JavaScriptben

A ciklus a program egy ismétlődő részlete. Ciklust alkalmazni sokkal egyszerűbb, mint újra meg újra begépelni ugyanazt a kódrészletet.

1 A ciklus kódja

A Pythonhoz hasonlóan a JavaScript is „for" utasítást használ a ciklus szervezésére. Kapcsos zárójelek közé kerül az ismétlendő kódrészlet. A ciklusváltozó számolja az ismétléseket.

```
<script>
for (var x=0; x<6; x++)
{
 document.write("Ciklus: "+x+"<br>");
}
</script>
```

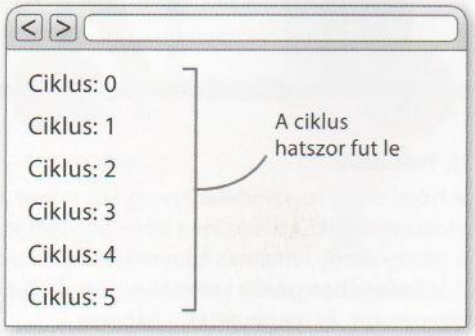
A <script> címke jelzi a JavaScript-kód elejét

Ez a sor hozza létre a 0-tól induló „x" ciklusváltozót, amely 1-gyel növekszik minden ismétléskor

Ez a sor kiírja azt, hogy „Ciklus:", és utána a ciklusváltozó értékét

2 A ciklus kimenete

Mentsd el a programot „.html" kiterjesztéssel, és futtasd le. A ciklus mindaddig fut, amíg x kisebb, mint 6 (a kódban ez az x < 6). Nagyobb számú ismétléshez növeld meg a „<" utáni számot a kódban.



Gonosz programok

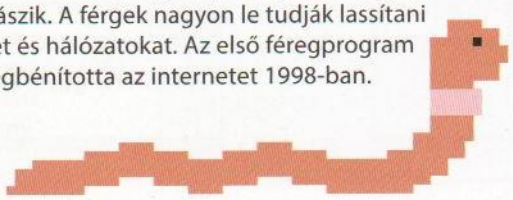
LÁSD MÉG
❮ 194–195
Az internet
❮ 202–203
Fontos programok

Nem minden program szórakoztató játék vagy hasznos alkalmazás. Vannak programok, amelyeket adatlopásra vagy számítógépek tönkretételére készítettek. Gyakran ártalmatlannak látszanak, és sokszor nehéz észrevenni, hogy áldozatul estünk.

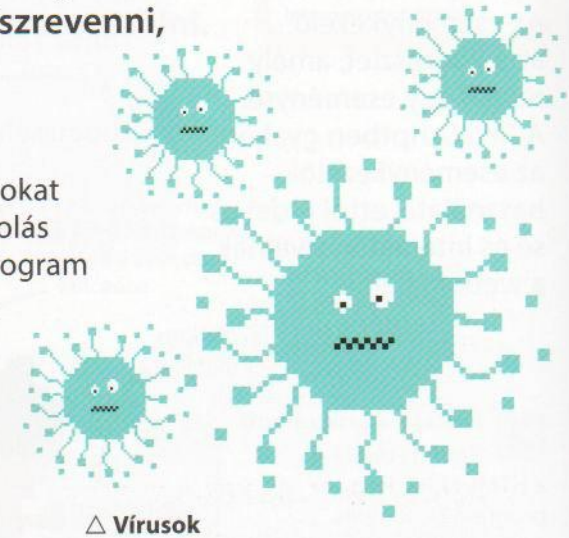
Rosszindulatú programok

A tudtunkon kívül a gépünkön tevékenykedő programokat rosszindulatúnak nevezzük. Az engedély nélküli behatolás egy számítógépbe bűncselekmény, mégis sok olyan program létezik, amely megpróbál betörni a számítógépünkbe.

▷ **Férgek**
A féreg olyan rosszindulatú program, amely számítógépről számítógépre mászik. A férgek nagyon le tudják lassítani a számítógépeket és hálózatokat. Az első féregprogram gyakorlatilag megbénította az internetet 1998-ban.



△ **Trójaiak**
A trójai olyan rosszindulatú program, amely ártalmatlannak álcázza magát. A kifejezés a trójai háborút idézi, amikor a görögök egy hatalmas falovat ajándékoztak a trójaiaknak. A ló belsejében görög katonák rejtőztek el, akik éjszaka előmásztak, és megnyerték a háborút.



△ **Vírusok**
Ahogy a valódi vírusok az emberi testben, a számítógépes vírusok is lemásoltatják önmagukat. Általában e-mail, pendrive vagy más fájlátviteli megoldások útján terjednek.

VALÓS VILÁG

Egy híres féreg

2000. május 5-én a Fülöp-szigeteki internet-felhasználók „ILOVEYOU" tárgyú e-mailt kaptak. A melléklet szerelmes levélnek tűnt, de valójában egy rosszindulatú program volt, amely kárt tett a számítógépen levő fájlokban.

◁ **ILOVEYOU**
A féreg hamar elterjedt a világban. Több mint 20 milliárd dollárra becsülik az általa okozott kárt.

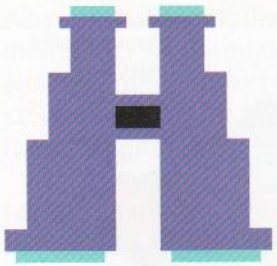
Mi a rosszindulatú programok célja?

A vírusok, a férgek és a trójaiak mind be akarnak hatolni a számítógépedbe. De mit tesznek, miután bejutottak? Törölhetnek vagy tönkretehetnek fájlokat, lophatnak bejelentkezési jelszavakat, vagy átvehetik az uralmat a géped felett egy nagyobb akció szervezett végrehajtása céljából.

▷ **Zombihálózat („botnet")**
A botnet megfertőzött gépek hálózata, amelyek felett valaki átvette az uralmat, hogy tömegesen küldjön kéretlen leveleket, vagy egy weboldalt elárasszon lekérésekkel, és így megbénítsa.

Védelem a rosszak ellen

Szerencsére nem vagyunk eszköztelenek a rosszindulatú programok ellen. A védelmet nyújtó programok kereskedelme nagy üzlet, számos szolgáltató verseng egymással. A leggyakoribbak a tűzfalak és a vírusirtók.



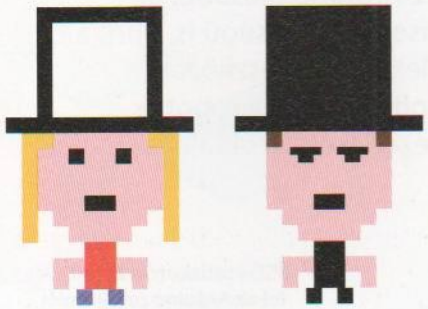
△ **Vírusirtó programok**
A vírusirtó program azonosítani próbálja az illetéktelen behatolót. Úgy szűri ki a rosszindulatú programokat, hogy megvizsgálja a gépen levő fájlokat, és összeveti őket vírusokról szóló adatbázisokkal.

△ **Tűzfalak**
A tűzfal célja, hogy kiszűrje az összes, a hálózatról érkező illetéktelen behatolót. Az internetről érkező minden adatot megvizsgál.

FOGALOMTÁR

Hackerek

A rosszindulatú programok készítői a hackerek. Aki bűnözési céllal állít elő ilyen programot, az a „fekete kalapos" hacker, a „fehér kalapos" pedig az, aki védelmet épít ki velük szemben.



Fehér kalapos hacker Fekete kalapos hacker

Miniszámítógépek

Egy számítógépnek nem kell feltétlenül nagynak és drágának lennie. Egyre több olyan kicsi és olcsó számítógép kapható, amelyik izgalmas, új utakat nyit a számítástechnikában.

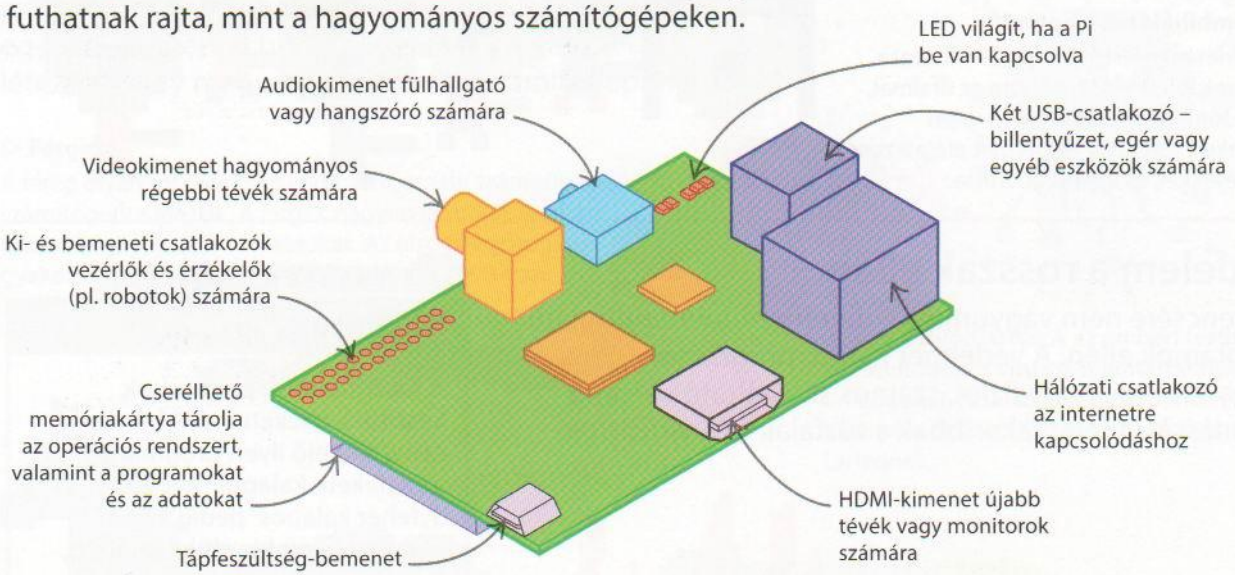
LÁSD MÉG

‹ 180–181
A számítógépek belülről

‹ 202–203
Fontos programok

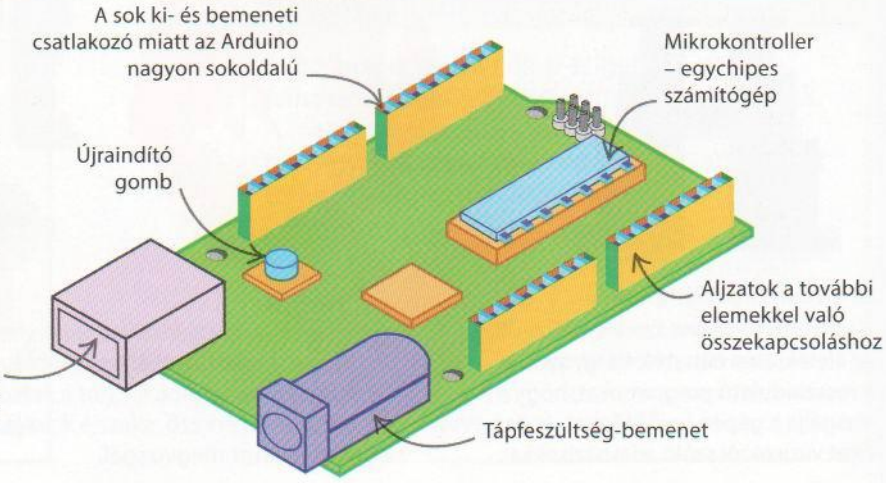
Raspberry Pi

A bankkártya méretű, ám igen hatékony Pi-t a számítógépek működésének oktatására fejlesztettek ki. Hasonló programok futhatnak rajta, mint a hagyományos számítógépeken.



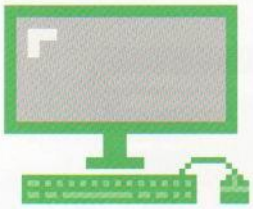
Arduino

Az Arduino olcsóbb, de kisebb képességű is, mint a Pi. Elektronikai eszközök építésére vagy robotok vezérlésére szokták használni.



A miniszámítógépek használata

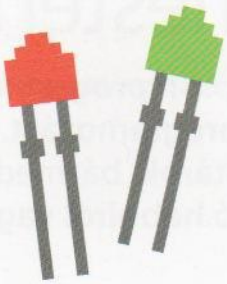
A sokoldalúságuk miatt ezeknek a gépeknek sok felhasználási területük van. Íme néhány ezek közül:



△ **Számítógép**
Csatlakoztass egy monitort, egy billentyűzetet és egy egeret, és máris kész a működő asztali gép!

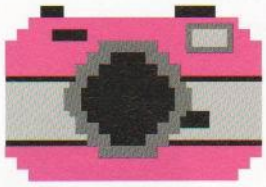
△ **Zenelejátszó**
Csatlakoztass hangszórót, és játssz le zenét a hálózatról!

△ **Mobiltelefon**
Csatlakoztasd a számítógépet az internetre egy mobiltelefon segítségével!



△ **Kütyük**
Csatlakoztass LED-eket vagy egyszerű áramköröket, hogy legyen egy robotod vagy egy kütyüd!

△ **Tévé**
Csatlakoztass tévét, hogy azon mutasd meg a fotóidat és a videóidat!



△ **Kamera**
Egy egyszerű kamera rákötésével állíts össze saját webkamerát!

△ **Pendrive**
Csatlakoztass egy USB-s háttértárat, és oszd meg a fájljaid a hálózaton!

△ **SD-kártya**
Próbálj ki különböző programokat memóriakártyáról!

VALÓS VILÁG

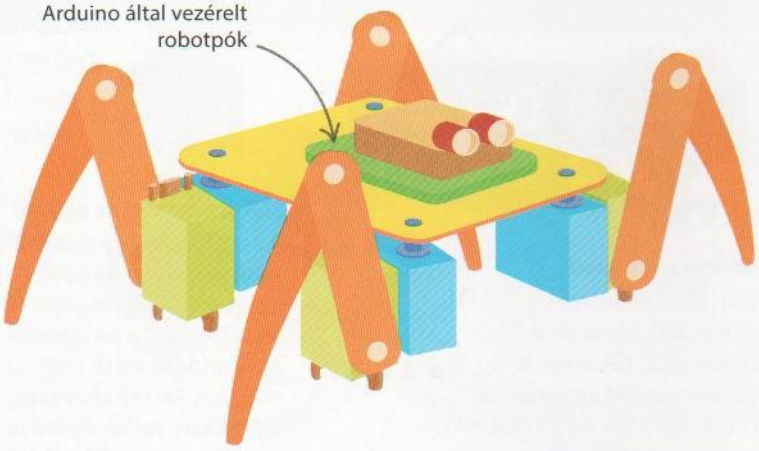
Házi építésű robotok

Alacsony költségük, illetve kis méretük és súlyuk miatt a miniszámítógépeket egyre szélesebb körben használják robotok építéséhez. Például:

Időjárásballonok: a légkör meteorológiai adatait mérik és rögzítik.

Minijárművek: ultrahanggal érzékelik az akadályokat, akár a denevérek.

Robotkarok: nehéz tárgyakat mozgatnak nagy pontossággal.



Programozás mesterfokon

LÁSD MÉG
〈 176–177 Hogyan tovább?
〈 214–215 Miniszámítógépek

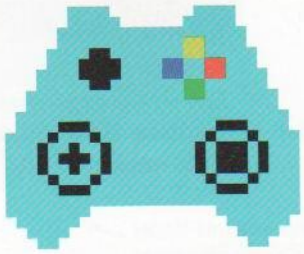
A profi programozóvá válás titka, hogy élvezd a programozást. Ha örömöd leled benne, akkor nincsenek határok, bármeddig el tudsz jutni a fejlődésben, legyen szó hobbiról vagy a leendő szakmádról.

Hogyan lehetsz még jobb programozó?

Akárcsak a síelés, a zenetanulás vagy a tenisz, a programozói tudás is csak gyakorlással fejleszthető. Hosszú évek alatt válhatsz igazi profivá, de ha közben élvezed, amit csinálsz, nem lesz megerőltető. Íme néhány tipp, hogy miként fejlődhet a tudásod.

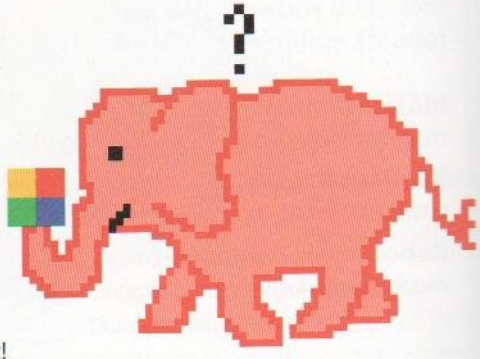
△ **Kódolj sokat!**
Gyakorlat teszi a mestert – ez a programozásra is igaz. Írj minél több programot, és egyre jobb programozó lesz belőled.

◁ **Legyél kíváncsi!**
Olvass weboldalakat és könyveket a programozásról, és próbáld ki mások kódjait. Olyan trükköket leshetsz így el, amelyekre egyedül sokkal lassabban jöttél volna rá.



△ **Vedd át a jó ötleteket!**
Ha látsz egy jó programot, gondold át, te hogyan programoztad volna. Keress jó ötleteket és megoldásokat, majd használd fel őket. A legjobb programozók gyakran egymás ötleteit fejlesztik tovább.

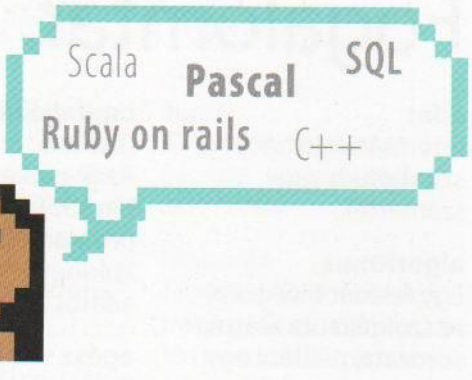
▷ **Tanítsd a barátod!**
Magad is sokat tudsz tanulni abból, ha tanítod a programozást. Amikor el kell magyaráznod, hogyan működik egy kód, akkor derül ki, hogy mindent jól értettél-e meg.



▷ **Tornáztasd az agyad!**
Az agyunk is úgy működik, mint az izmok: ha edzésben tartjuk, egyre jobb lesz. Fejleszd a programozói gondolkodást: oldj meg logikai rejtvényeket, fejtörőket, sudokufeladványokat vagy matekpéldákat!

▷ Teszteld a programod!
Vizsgáld meg, hogyan viselkedik, mennyire hibatűrő a kód, amit írtál, eszement bemeneti értékek hatására. Ha kell, módosíts rajta, akár mások programjaiból ellesett trükköket is felhasználva.

◁ Állíts robothadsereget!
A számítógépet sokféle programozható kütyüvel lehet összekötni, az egyszerű LED-lámpáktól a különféle robotokig. A működésükből és az irányításukból rengeteget tanulhatsz.



△ Tanulj újabb programnyelveket!
Légy soknyelvű! Minden újabb programnyelv elsajátításával jobban meg fogod érteni az addig tanult nyelveket is. A legtöbb programnyelvnek van ingyenesen hozzáférhető változata.

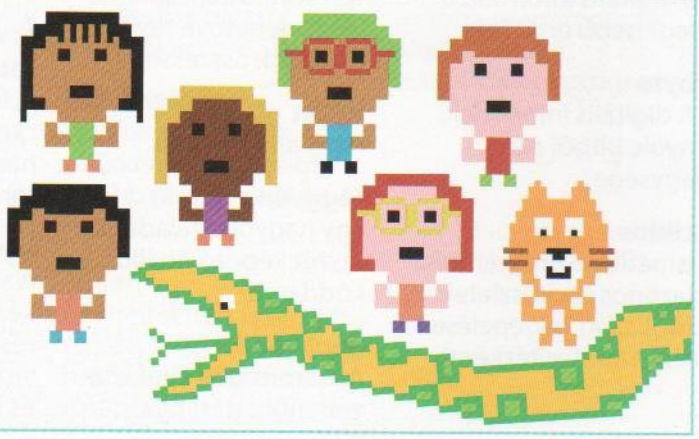
▷ Nézz meg egy gépet belülről!
Szedj darabjaira egy régi számítógépet (persze, csak miután engedélyt kértél rá), hogy megismerd a működését. Ha már tudod, mire valók az egyes komponensek, a legjobb móka saját számítógépet építeni.

▷ Nyerj díjakat!
Ha már elég jártas vagy a kódolásban, miért ne neveznél programozói versenyekre? Az interneten különféle szintű versenyeket találhatsz. A legnehezebbek egyike a Google Code Jam világverseny, de vannak sokkal könnyebbek is.

JEGYEZD MEG!

Élvezd a programozást!

A kódolás annyiban hasonló a rejtvényfejtéshez, hogy komoly kihívásokat rejt, és könnyű benne elakadni. Ez persze idegesítő lehet. Ugyanakkor mindezt feledteti a sikeresen megoldott probléma és a jól működő program feletti öröm. A legjobb módja, hogy a sikerélményed folyamatos legyen, ha mindig a tudásodnak megfelelő feladatokat vállalsz. Ha a feladat túl könnyű, unalmas lesz, ha pedig túl nehéz, hamar elmegy tőle a kedved. Vezessen a kíváncsiságod, hogy felfedezz, kísérletezz, és eltérj a megszokottól. Mindeközben pedig: mulass jól!



Fogalomtár

adat
Információ, pl. szöveg, szimbólum vagy számérték.

algoritmus
Egy feladat elvégzésére szolgáló utasítások sorozata, például egy számítógépes program „gondolatmenete".

ASCII
Szabványos amerikai információcsere-kód; karakterek bináris tárolására használt kódolás.

bemenet
A számítógépbe bevitt adatok összessége, pl. mikrofonnal, billentyűzettel vagy egérrel.

bináris kód
Számok és egyéb adatok leírása 0 és 1 számjegyekkel.

bit
Kettes számrendszerbeli számjegy: 0 vagy 1, a digitális információ legkisebb egysége.

byte
A digitális információ nyolc bitből álló egysége.

ciklus
Ismétlődő programrész (azonos kódrészletek többszöri begépelése alóli mentesítésként).

csatlakozópont (socket)
Az IP-cím és a port kombinációja, amely által a programok közvetlenül küldhetnek egymásnak adatot az interneten.

egész szám (integer)
Tizedesvessző vagy törtvonal nélkül leírható szám.

elágazás
A számítógépes program olyan pontja, ahol kétféleképpen folytatódhat a végrehajtás.

esemény
A számítógép által érzékelhető történés, pl. billentyű lenyomása vagy egérkattintás.

fájl
Tárolt adatok névvel ellátott halmaza.

felület
A felhasználó számára a szoftverrel vagy a hardverrel való kapcsolatba lépést lehetővé tevő eszközök összessége.

futtat
Programot elindít.

függvény
Egy nagyobb feladat részét képező önálló kóddarab.

gépi kód
A számítógép által közvetlenül értett nyelv – a programnyelveket gépi kódra kell fordítani, hogy a processzor el tudja olvasni.

grafika
A képernyőn megjelenő vizuális elemek a szövegeken kívül, pl. képek, ikonok és szimbólumok.

grafikai processzor (GPU)
Képeknek a számítógép képernyőjén való megjelenítését lehetővé tevő egység.

grafikus felhasználó felület (GUI)
A programnak azt a részét felépítő gombok és ablakok stb. összessége, amelyet látunk, és amelyen keresztül kapcsolatba lépünk vele.

hacker
Számítógépes rendszert feltörő személy – a „fehér kalapos" hackerek a számítógép-biztonsági cégeknek dolgoznak, és azért keresik a hibákat, hogy kijavíthassák; a „fekete kalapos" hackerek károkozás vagy haszonszerzés céljából törnek fel rendszereket.

hardver
A számítógép látható, kézzel fogható fizikai részeinek összessége, pl. vezetékek, billentyűzet és képernyő.

hexadecimális rendszer
Tizenhatos alapú számrendszer, amelyben a 10 és 15 közötti értékeket betűk jelölik A-tól F-ig.

hiba (bug)
A program elvárt működését gátló téves rész a kódban.

hibakeresés (debugging)
A program hibáinak felderítése és kijavítása.

hibakereső (debugger)
Más programok hibáinak felderítésére szolgáló számítógépes program.

hív
Függvényt használ egy programban.

index
Egy lista vagy karakterlánc elemét azonosító szám – Pythonban az első elem indexe 0, a másodiké 1, stb.

IP-cím
Az internetre csatlakozó eszközök egyedi címéül szolgáló számsorozat.

kapu
A számítógép által döntésekre használt áramkör, egy vagy több bemenetből szabály alapján kimeneti jelet állít elő, pl. az ÉS kapu-

nál csak akkor lesz a kimenő jel bekapcsolva, ha mindkét bemenő jel is be van kapcsolva; létezik még többek közt VAGY és NEM kapu is.

karakterlánc (sztring)
Karakterek sorozata, amely tartalmazhat számjegyeket, betűket, szimbólumokat, pl. kettőspontot.

kimenet
Számítógépes program által létrehozott, a felhasználó számára megjelenített adatok összessége.

könyvtár (függvénykönyvtár)
Más projektek által újrahasznosítható függvények gyűjteménye.

lebegőpontos szám (float)
Tizedesvesszőt (tizedespontot) tartalmazó szám – tizedes tört.

logikai kifejezés
Kizárólag igaz vagy hamis értéket felvenni képes kifejezés.

mappa
Fájlok rendszerezett tárolásának helye.

memória
Adatok tárolására szolgáló lapka a számítógépben.

modul
Egy nagyobb program önálló részfeladatát ellátó kódrészlet.

művelet
Speciális függvényt végrehajtó szimbólum, pl. „+" (összeadás) vagy „–" kivonás.

operációs rendszer
Egyéb programok futásához megfelelő környezetet és a hardverhez történő kapcsolódást biztosító rendszer.

port
A számítógép által egy program címeként használt számsorozat.

processzor
A programok futtatására szolgáló speciális elektronikus lapka a számítógépben.

program
A számítógép által egy feladat elvégzése érdekében végrehajtott utasítások sorozata.

programnyelv
Számítógép irányítására használt nyelv.

rosszindulatú program (malware)
Számítógép megrongálására vagy tönkretételére megalkotott program.

számítógép-hálózat
Két vagy több számítógép összekapcsolásának módja.

szereplő
Mozgatható objektum.

szerver
Hálózaton elérhető fájlokat tároló számítógép.

szintaxis
Egy programnak a megfelelő működéshez szükséges felépítését meghatározó szabályok összessége.

szoftver
A számítógépen futó és annak működését meghatározó programok összessége.

tároló (konténer)
További adatok halmazának tárolására használt programrész.

titkosítás
Adatok kódolása, hogy csak a megfelelő személyek tudják olvasni vagy elérni őket.

tömörítés
Adatok méretének csökkentése úgy, hogy kisebb tárhelyen férjenek el.

trójai
Önmagát más programnak álcázó, ezáltal a felhasználót megtévesztő rosszindulatú program.

tuple
Elemek vesszővel elválasztott, zárójelek közé tett listája.

Unicode
Szimbólumok és betűk, írásjegyek ezreit ábrázoló univerzális számítógépes kód.

utasítás
Egy programnyelv legkisebb, felbonthatatlan, teljes értékű eleme.

változó
Változtatható adatok tárolására szolgáló, névvel ellátott rekesz.

végrehajt
Lásd futtat.

véletlen szám
Megjósolhatatlan kimenetű függvény által adott érték, játékok készítésekor hasznos.

vírus
Önmagának lemásoltatásával fertőző, számítógépek között terjedő rosszindulatú program.

Név- és tárgymutató

A **félkövérrel** szedett oldalszámok a legfontosabb előfordulást jelölik.

Köszönetnyilvánítás

A DORLING KINDERSLEY köszönettel tartozik Vicky Short, Mandy Earey, Sandra Perry és Tannishtha Chakraborty tervező asszisztenseknek, Olivia Stanford szerkesztő asszisztensnek, Caroline Hunt olvasószerkesztőnek, Helen Petersnek a Név- és tárgymutató összeállításáért és Adam Brackenbury technikai segítségéért.

DORLING KINDERSLEY INDIA köszönettel tartozik Kanika Mittalnnek, Pawan Kumarnak és Saloni Singh-nek.

A Scratch fejlesztése: A Lifelong Kindergarten csoport az MIT kereteiben; http://sratch.mit.edu

= x 0  { 0 1 0) : 1 0 = *

(y = 0 * > 0 1 : 1 @ [/ 

Érdekelnek a számítógépes játékok és animációk? Szeretnél megtanulni programozni?

Saját játékokat készíteni élvezetes, hasznos és meglepően egyszerű kreatív tevékenység. A programozás tudománya egyesíti a művészetet, a történetmesélést és fejleszti a logikus gondolkodást.

Ismerd meg a kezdő programozók leghasznosabb programnyelveit! A könyvben található színes grafikák, egyszerű magyarázatok és érdekes projektek segítségével könnyedén megtanulhatod a Scratch és a Python alapjait.

„Hiszem, hogy a programozás a 21. század írástudása. Aki érti a gépek nyelvét, tud algoritmikusan gondolkozni, az hatékonyabb, boldogabb és egészségesebb lesz. Ráadásul a programozás szórakoztató, izgalmas és kreatív alkotás."

Halácsy Péter,
a Prezi társalapítója

 } 0 * 1 \ =  > : 1 0 1)

10+ = 0 * 1 @

hvg könyvek junior

3900 Ft
ISBN 978-963-304-320-2
9 789633 043202

0 ^ 1  # @ 1