

# AppleScript alapok kezdőknek

Második, bővített kiadás.

# Általános információk

Az AppleScript egy Apple-technológia, ami lehetőséget nyújt az alkalmazások közötti kommunikációra. Pl.:

- a Mail programban lévő leveleket tárolhatunk egy adatbázisban;
- utasíthatunk egy képszerkesztő programot, hogy több képnél változtassa meg a felbontását, méretezze át az eredményt és küldje el egy másik számítógépnek, vagy postázza el az interneten stb.

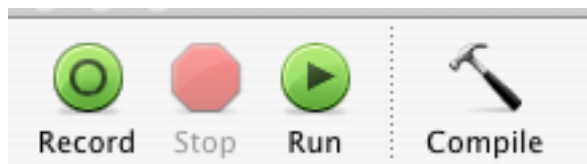
Egy AppleScript szabályszerű utasítások sorozata egy szkript-nyelven amit AppleScript-nek hívnak. Ez a nyelv hasonlít az angol nyelvhez, ezáltal viszonylag könnyű megtanulni, megérteni és használni.

**Az AppleScript-nyelv:** egy az angolra hasonlító nyelv, amivel írásban utasítható a gép és programjai.

**Egy AppleScript:** utasítások sorozata, amely AppleScript-nyelven íródott és része a Mac OS rendszernek, amely végrehajtja az abban leírt utasításokat.

## A Script Editor-ról

A program az **Applications/AppleScript** mappában található. Elindítva egy üres ablak jelenik meg, a fejlécen négy gombbal (gyári beállítás).



- **Record:** Néhány alkalmazás (pl. QuickTime Player) lehetőséget nyújt arra, hogy a benne elvégzett lépéseket ne manuálisan írjuk meg, hanem felvegyük e gomb segítségével, így “írva” meg egy komplexebb szkriptet. Ha szeretnénk megállapítani, hogy egy alkalmazásból felvehetünk-e lépéseket, egyszerűen próbáljuk ki a következő módon:

1. Kattintsunk a **Record** gombbra!
2. Csináljuk meg azokat a lépéseket a programban, amelyeket szeretnénk felvenni.
3. Kattintsunk a **Stop** gombra!

Ha semmi sem jelenik meg az editor ablakában, akkor vagy az elvégzett lépéseket nem tudja rögzíteni, vagy a programból egyáltalán nem vehetők fel ily módon a lépések. Ez az opció — a lehetőségekhez képest — eléggé szegényes, ugyanis gyakran előfordul, hogy olyan lépéseket nem vesz fel, amelyeket egyébként végrehajtana az adott program, ha az utasításokat mi magunk íránk meg.

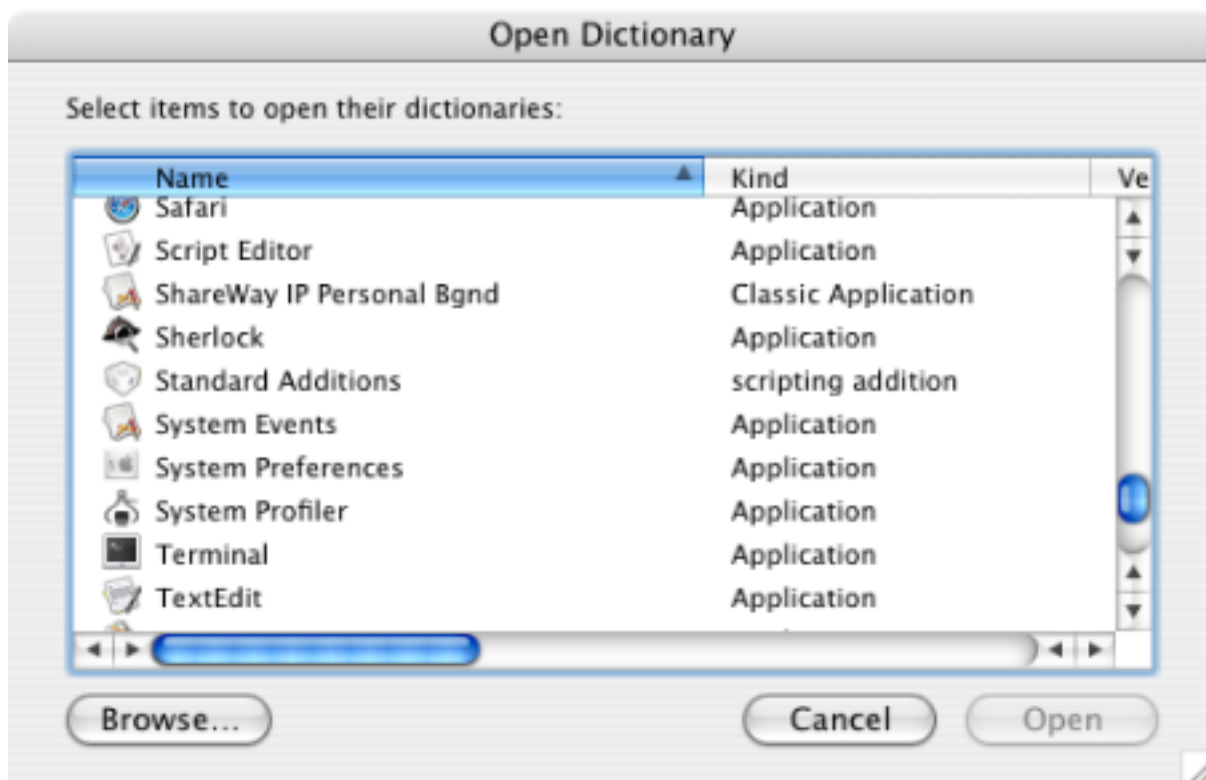
- **Stop:** a felvétel vagy az éppen futó szkript leállítására szolgál.

- **Run:** a megírt szkriptet szintaktikailag leellenőrzi és ha nem talál hibát benne, lefuttatja.
- **Compile:** az általunk írt utasítások helyességét ellenőrizhetjük le. Ez a gomb csak a szintaktikai részt ellenőrzi le. Ha pl. egy szkript utal egy fájlra a gépen, de az nem található, akkor azt csak futás közben fogja jelezni, leállva egy hibüzenettel.

## A gépen lévő alkalmazások lehetőségei

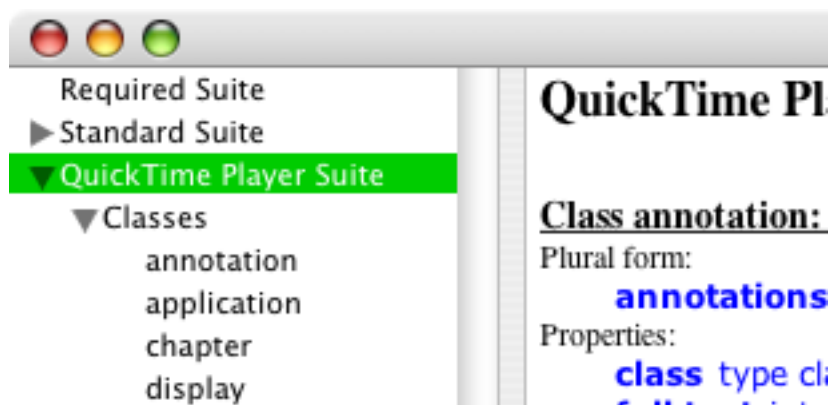
Minden program vezérelhető bizonyos mértékig szkripttel. El lehet indítani, ki lehet léptetni, előre lehet hozni és el lehet bújtatni (hide). Utóbbi két utasítás nem működik csak a háttérben futó alkalmazásoknál. Egy alkalmazás “szkriptelhetőségét” a következő módon állapíthatjuk meg:

A **ScriptEditor**-ban válasszuk ki a **File** menüből az **Open Dictionary...** parancsot! Ekkor egy ablak jelenik meg:



Ha ebben a listában nem találjuk meg a kérdéses programot, kattintsunk a **Browse...** gombra és keressük meg mi magunk!

Az **Open** gobbal megnyitva, hasonló ablakot láthatunk:



Bal oldalon a parancsokat láthatjuk csoportosítva, jobb oldalon pedig a konkrét szintaktikát, példa nélkül.

Ha az **Open Dictionary...** paranccsal nem nyitható egy program (világosszürke marad), akkor az közvetlenül csak az alapparancsokra reagál. Közvetett vezérlésre is van lehetőség, erről bővebben később.

#### **A programokra vonatkozó alapparancsok szintaktikái:**

**tell application "Preview" to activate**

Ez a megoldás két dologra is jó. Ha az alkalmazás nem fut, akkor elindítja és előrehozza, ha pedig már fut, akkor csak előrehozza. A programok nevét mindig úgy kell megadni, ahogy a Finder-ben láthatóak!

Programok kiléptetéséhez ezt a formát használhatjuk:

**tell application "Preview" to quit**

Egy program elbújtatásához (hide) kissé bonyolultabb szkriptet kell írunk.

**tell application "Finder" to set visible of process "Script Editor" to false**

Az **Open Dictionary** fejlécű képen (előző oldal) látható egy **Standard Additions** nevezetű **scripting additions**. Ezek a szkript-bővítmények, amelyek új lehetőségekkel ruházzák fel a szkripteket. Ahhoz, hogy ezeket használni tudjuk, bele kell tennünk őket a **HD/Library/ScriptingAdditions** mappába. Ha nem létezne ilyen mappa, akkor nekünk kell létrehoznunk egyet, ügyelve a pontos névre! Sok — 95 %-ban ingyenes — szkript bővítményt találhatunk a következő oldalon mind Mac OS X-re és Classic-ra:

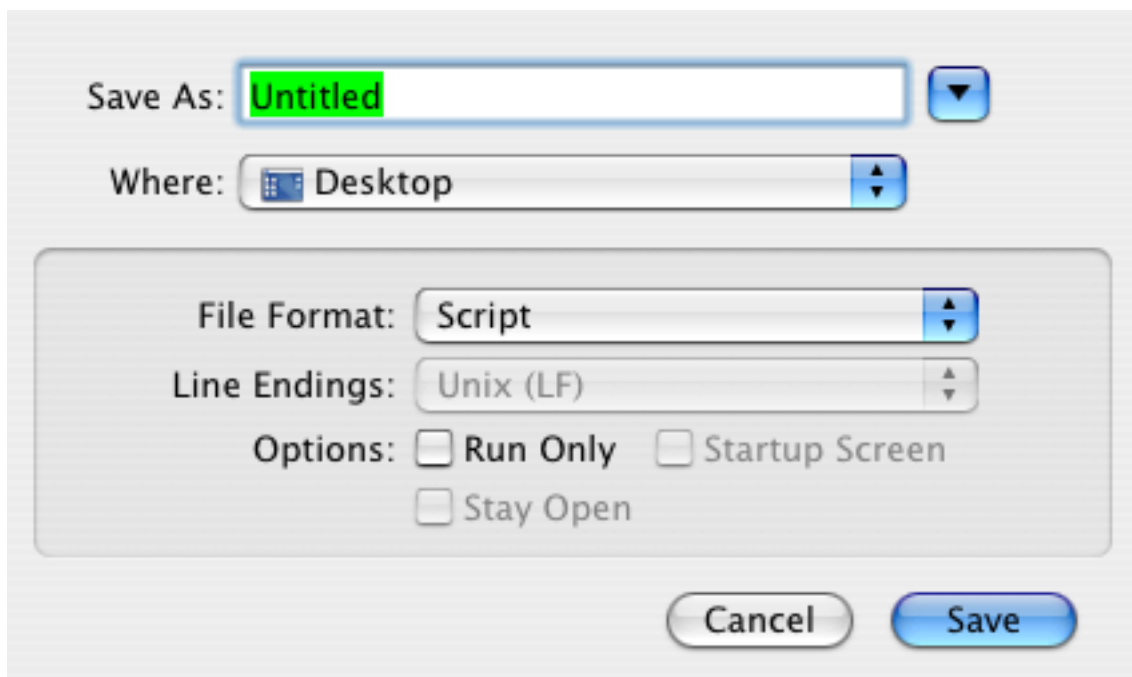
<http://www.osaxen.com>

## A megírt szkript elmentése

Többféleképpen menthetünk el egy szkriptet, de ha az nem volt leellen-ôrizve a gép által (Compile), akkor csak szöveggként.

Ha az ellenôrzés alatt nem volt hiba, akkor elmenthetô, mint egy ellenôrzött szkript (compiled script), vagy mint egy alkalmazás (application).

Válasszuk ki a File menü Save parancsát, adjuk meg a szükséges információkat, majd kattintsunk a Save gombra!



**Figyelem!** Ha a **Run Only** opciót kijelöltük, akkor a szkript, vagy alkalmazás csak futtatható, módosításra nincs lehetőség.

Az ellenôrzött és elmentett szkript ikonja:



Kétszer rákattintva, a **Script Editor** fogja kinyitni, ahol a **Run** gombbal futtathatjuk. Betehetjük a “**Script menu**”-be, és onnan közvetlenül is futtatható. Több programban van

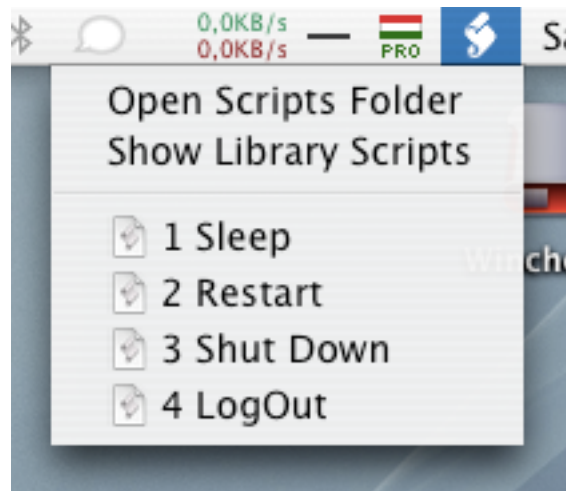
beépített szkript menü a saját szkriptjeink számára, ahol általában ezt a formátumot preferálják.



Alkalmazás: ugyanúgy viselkedik, mint egy program. Kétszer rákattintva elindul és végrehajtja a feladatokat. Elhelyezhető a “Log-in Items” közé (System Preferences) és reagál az alapparancsokra.

## A ScriptMenü

Ez a szkript menü rendszerszintű, tehát minden programból elérhető mint pl. a bill. kiosztás menüje.



*Én konkrétan csak ezek miatt használom (a számok csak a sorrend miatt vannak előttük), mert így nem kérdez rá, hogy biztosan ezt akarom-e csinálni. :-D*

Természetesen bármilyen ellenőrzött és elmentett szkriptet betehetünk a menübe.

Installálása: Kattintsunk kétszer a **HD/Applications/AppleScript/Install Script Menu** programra, majd a szkripteket tegyük a **~/Library/Scripts** mappába.

# Alapok

Az elkövetkező oldalakon megismerkedünk a szkriptírás alapvető szabályaival. Az AppleScript, mint a Mac OS része, elég kevés lehetőséggel rendelkezik, de ettől függetlenül sok mindenre jó. Például meg tudjuk vele szólaltatni a rendszer figyelmeztető hangját:

```
beep
```

Ez egy nagyon rövid szkript, amit ha végrehajt a gép, megszólal az általunk beállított (System Preferences) figyelmeztető hang.

Több hang egymás utáni lejátszásához ugyanezt a parancsot kell használni, de ki kell egészíteni egy számmal attól függően, hányszor akarjuk megszólaltatni a figyelmeztető hangot:

```
beep 3
```

Ekkor egymás után háromszor fog megszólalni az általunk beállított figyelmeztető hang. A “3” opcionális információ. A **beep** parancs alapértelmezési értéke egy.

Ha szeretnénk megszólaltatni a rendszerbe épített beszédgenerátort, akkor a **say** parancsot kell használnunk:

```
say "This is my one of the first script"
```

Ekkor az általunk beállított (System Preferences) beszédhangon fogja elmondani a gép az idézőjelek közötti szöveget, ami alapbeállításként “Victoria”. Ha más hangon szeretnénk felolvastatni akkor itt is plusz információt kell beírni a szkriptbe:

```
say "This is my one of the first script" using "Zarvox"
```

**Fontos!** Az AppleScript érzékeny a kis és nagybetű használatára az idézőjeleken belül. Pl. “Zarvox” helyett nem használhatjuk a “zarvox” kifelyezést, különben “Parameter error.” hibaüzenettel leáll a szkript futása.

Néhány program népszerűbb, mint mások, de van egy, amit minden Mac- felhasználó használ, ez pedig a Finder. Lehetőséget nyújt arra, hogy fájlokat, mappákat mozgass, másolj, átnevezz stb. Pl. a kuka kiürítésének lehetősége is megadatott szkripttel.

**Figyelem! A következő parancs hatására a Finder azonnal kiüríti a kukát figyelmeztetés nélkül!**

```
tell application "Finder" to empty the trash
```

Ugyanezen parancs másik szintaktikával:

```
tell application "Finder"  
    empty the trash  
end tell
```

A különbség az előzőhöz képest annyi, hogy előbbinél szerepel, a másikonál nem a **to** szócska. Utóbbi viszont akkor hasznos, ha egy alkalmazásnak egymás után több parancsot szeretnénk adni.

```
tell application "Finder"  
    empty the trash  
    open home  
    beep  
end tell
```

Vannak parancsok, amelyeket nemcsak a Finder ért meg, hanem a Mac OS AppleScript része is, ezáltal bizonyos mértékig variálhatóak úgy, hogy a végeredmény ugyanaz.

```
tell application "Finder"  
    empty the trash  
    open home  
end tell  
beep
```

Egy kicsit egyszerűsíthetünk a dolgunkon, ha a szkriptben az **“application”** szó helyett csak az **“app”** szócskát használjuk, a rendszer automatikusan kiegészíti azt.

# Az alapfunkciók

## Vátozók és típusaik

Egy programban szükségünk lehet változókra. Olyan elemekre, amelyekben ideiglenesen tudunk tárolni adatokat, eredményeket, közbenső eredményeket stb. Az AppleScript többféle változót támogat. Alkalmasak mindenféle adat (szám, szöveg, lista stb.) tárolására. Elnevezésükben vannak szabályok: nem lehet olyan nevet adni nekik, ami valamilyen parancsot, utasítást, kódot stb jelent. Ilyen pl. a **“tell”**, **“pi”** vagy a **“to”** és nem kezdőhetnek számmal! Az AppleScript nem veszi figyelembe az idézőjeleken kívüli kis- és nagybetű közötti különbséget. Ha létrehoztunk egy **“alma”** nevű változót, akkor arra **“ALMA”**, **“Alma”** vagy **“aLmA”** stb. névvel hivatkozhatunk, mindegyik ugyanazt a változót fogja jelenteni.

## Számok (integer) és valós számok (real integer)

Egy szám típusú változó, csak számokat tartalmaz, tizedespont nélkül. Az AppleScript tizedes vessző helyett tizedes pontot használ. Egy valós szám már tartalmazhat tizedespontot is. Létrehozásukhoz mindössze értéket kell nekik adnunk:

```
set x to 4553
set y to 345.11
```

Ebben a példában **x** és **y** betűk változók, értékük tetszés szerint változtatható és bármikor hivatkozhatunk rájuk.

Ha már adtunk a változóknak értékeket, tudunk velük számolni:

```
set a to 212
set b to 1234
set a to a * b
set b to 456 - 112
```

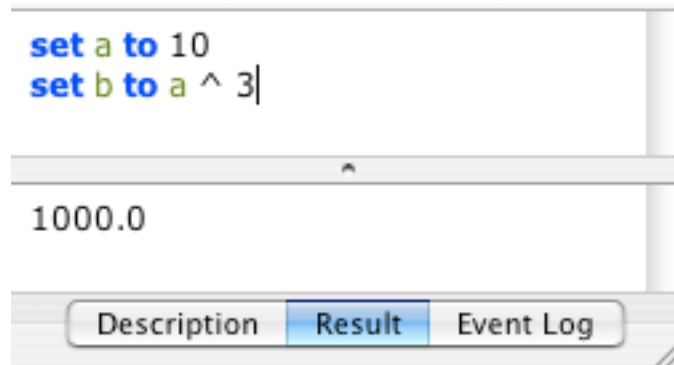
**A** értékét beállítja **a** és **b** szorzatára, **b** értékét pedig 456 mínusz 112-re. Használható jelek a négy alpművelethez: **+ - / \***

Futtassuk le a következőt:

```
set a to 10
set b to a ^ 3
```

Az eredmény tíz a harmadikon, de nem látjuk. Kattinsunk az ablak alján lévő **Result** fülre! Az eredmény ebben a mezőben jelenik meg. Figyelem! Itt mindig csak a szkript futásakor

keletkező legutolsó eredményt tekinthetjük meg amennyiben nem szakadt meg hiba miatt vagy le nem állítottuk mi magunk.



## Szövegek (string)

A változók természetesen tartalmazhatnak szöveges adatokat is. A szöveget mindig idézőjelek közé kell tenni! A következő példában **a** értéke egy szóköz, **b** értéke "5" és **c** értéke a Hello szó lesz. **B** értéke ugyan 5, de szöveges változóként, így ezzel matematikai feladatokat nem hajthatunk végre!

```
set a to " "  
set b to "5"  
set c to "Hello"
```

Amennyiben a **Result** mező van nyitva, ott idézőjelek között a **Hello** szó látható.

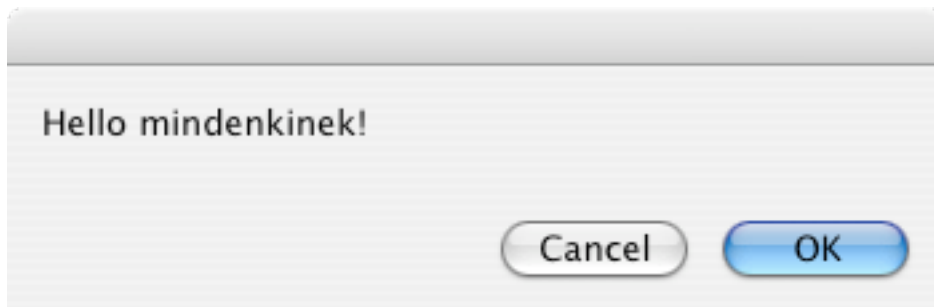
Egy szöveg hosszának megszámlálása:

```
set a to the length of "Próba szöveg"
```

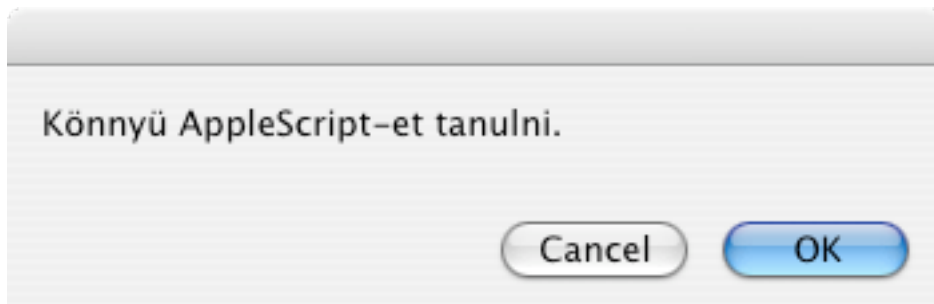
Az eredmény 12 mint szám.

Mivel közbenső adatokra is szükségünk lehet, használhatjuk a "**display dialog**" parancsot, mint egy alternatív kommunikációs eszközt. Ezzel részletesebben a későbbiekben foglalkozunk.

```
display dialog "Hello mindenkinek!"
```



```
set a to "Könnyü"  
set b to "tanulni."  
display dialog a & " AppleScript-et " & b
```



Mivel a szövegek idézőjellel kezdődnek és fejeződnek be, egy apró trükköt kell alkalmazni az idézőjelek megjelenítéséhez:

```
display dialog "Ennyit mondott: \"Szia\""
```

A szöveges és numerikus változók közötti átjárásra is van lehetőség:

```
set a to "15" as number  
set b to 12 as string  
set c to 234  
  
set c to c as string
```

Az eredmények:

**a:** 15 mint szám  
**b:** "12" mint szöveg  
**c:** "234" mint szöveg.

Szövegek "összeadására" van lehetőségünk:

```
set a to "alma" & "körte"  
display dialog a
```

## Boolean

Ez a legegyszerűbb változófajta, mindössze két értéke lehet: igaz (true) vagy hamis (false).

```
set a to true
set b to false
```

## Listák (list)

A listák arra jók, hogy több(féle) adatot tároljunk egy változóban. A lista elemeit kapcsos zárójelbe kell tenni és vesszővel kell őket elválasztani.

```
set a to {213.1, 6, "Hello", 8.5}
display dialog a as text
```

A listában lévő adatok cserélhetőek és lehet rájuk hivatkozni egyenként (**item n**) is:

```
set a to {213.1, 6, "Hello", 8.5}
set item 2 of a to "szöveg"
display dialog a as text
```

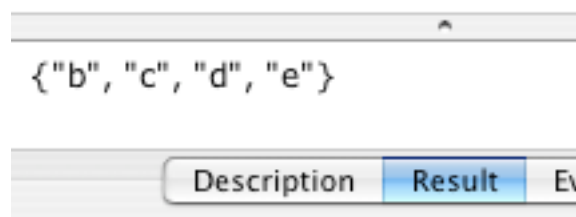
```
set a to {213.1, 6, "Hello", 8.5}
display dialog item 2 of a as text
```

Variációk az “item n”-re:

2nd item  
15th item  
last item  
first item

Egy lista részének kiválasztása:

```
set a to {"a", "b", "c", "d", "e", "f", "g", "h"}
set b to items 2 through 5 of a
```



A lista elemeinek megfordítása:

**set a to reverse of** {3, 2, 1}

Hány elem is található a listában?

**set a to the length of** {"first", "last"}  
**set b to the count of** {"first", "last"}

Mindkettő eredménye 2.

Véletlenszerű kiválasztás:

**set a to some item of** {"hearts", "clubs", "spades", "diamonds"}

Az eredmény egy véletlenszerűen kiválasztott listaelem.

Egy számot vagy egy szöveget is át lehet alakítani egy listaelemmé:

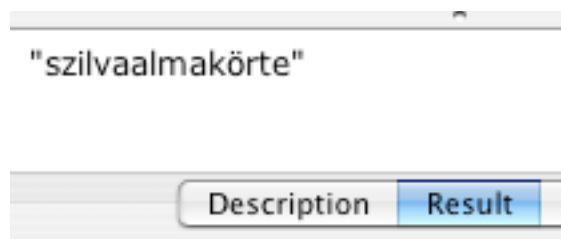
**set a to** "alma"  
**set b to a as list**

Elem hozzáadása egy listához:

**set a to** {"alma", "körte"}  
**set b to** "szilva"  
**set c to a & b**

Futtassuk le a következő szkriptet és figyeljük meg az eredményt:

**set a to** {"alma", "körte"}  
**set b to** "szilva"  
**set c to b & a**



Az eredmény **nem** lista, hanem szöveg azért, mert egy szöveges változóhoz adunk hozzá egy listát, de könnyen megoldható, hogy lista maradjon:

**set a to** {"alma", "körte"}  
**set b to** "szilva"  
**set c to (b as list) & a**

Ebben az esetben az eredmény már lista.

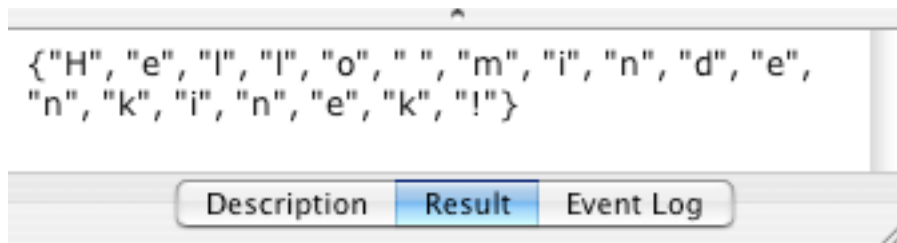
Egy elem hozzáadása a lista végéhez:

```
set a to {1, 2, 3, 4}
set the end of a to 5
get a
```

A harmadik sorban lévő “**get**” parancsra csak az eredmény (result) láthatósága miatt van szükség.

Lehetőségünk van egy szöveg karaktereinek listává alakítására. Ebben az esetben a szöveg minden karakterét — beleértve az írásjeleket és szóközöket is — egyenként teszi listaelemmé:

```
set a to every character of "Hello mindenkinek!"
```



Fordított átalakítás is lehetséges:

```
set a to {"a", "l", "m", "a"}
set a to a as string
```

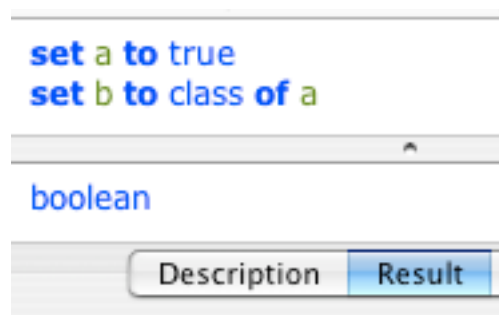
Az eredmény az “alma” szó, mint szöveg.

## Besorolási (class) és napló (log) parancsok

A **class** parancsra akkor lehet szükségünk, ha nem tudjuk eldönteni, egy változóról, hogy milyen típusú.

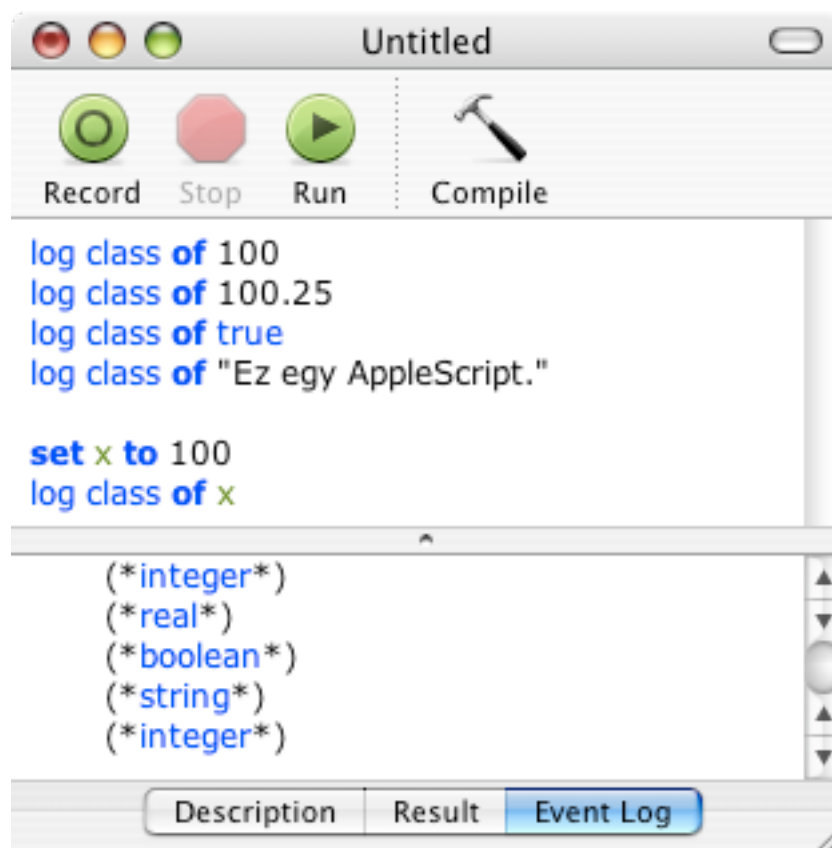
A **log** parancs arra jó, hogy a szkript futása közben keletkező (rész)eredményeket láthatóvá tegyük.

```
set a to true
set b to class of a
```



```
log class of 100
log class of 100.25
log class of true
log class of "Ez egy AppleScript."
```

```
set x to 100
log class of x
```



Az **Event Log** ablakban csak akkor láthatunk eredményeket, ha a **Run** parancs kiadása előtt kinyitjuk azt.

## Megjegyzések

Előfordulhat, hogy szeretnénk megjegyzéseket írni a szkriptbe a magunk vagy mások dolgának megkönnyítése végett. Erre is van lehetőségünk.

Egysoros megjegyzéshez kezdjük a szöveget két kötőjellel, ami ezután következik, azt az AppleScript megjegyzésként kezeli, nem ellenőrzi és nem próbálja meg futtatni:

`beep` -- *Ez a szövegrész a megjegyzés*

Többsoros megjegyzéshez kezdjük a zárójel-csillag karakterkombinációval és zárjuk a csillag-zárójellel.

*(\* Ha ezt a szkriptet lefuttatod, akkor a rendszer figyelmeztető hangja fog megszólalni \*)*

`beep`

## Feltételes lépések

Vannak esetek, amikor egy szkript valamely részét csak akkor kell futtatnunk, ha egy bizonyos feltétel teljesül vagy nem teljesül. Egy feltételes módnak két állapota van: igaz vagy hamis. Feltételes lépéseket az **“if then”** parancskombinációval tudunk készíteni:

```
set a to 25
if a = 25 then
    beep
end if
```

Lefordítva soronként:

**állítsd a értékét 25-re**  
**Ha a egyenlő 25-el, akkor**  
**figyelmeztető hang**  
**feltétel vége**

A figyelmeztető hang megszólal, hiszen a feltétel igaz. Az **“if”** és **“end if”** sorok közötti részt csak akkor hajtja végre, ha a feltétel(ek) igazak.

```
set a to 25
set b to 30
if a = b then
    beep
end if
```

A figyelmeztető hang nem szólal meg, hiszen a feltétel nem teljesül.

Több feltételt is tehetünk egy sorba:

```
set a to 25
set b to 30
set c to 35

if a = b and c = 35 then
    beep
end if
```

A feltételes módot lefordítva, a szkript jelentése a következő:

**Ha a egyenlő b-vel ÉS c=35-el akkor**  
**beep**  
**feltétel vége**

Próbáljuk ki úgy, hogy az **“and”** szót kicseréljük **“or”**-ra! Lefuttatva hallani fogjuk az ismerős hangot.

**and:** jelentése **“és”**. Az eredmény akkor igaz, ha mindkét feltétel igaz.

**or:** jelentése **“vagy”**. Az eredmény akkor igaz (a köznapi szóhasználatától eltérően), ha vagy az egyik, vagy a másik, vagy pedig mindkettő feltétel igaz.

Egy szimpla **“if then”** feltételes módban két állapot lehetséges: igaz vagy hamis. Ha alternatív választási lehetőségre van szükségünk használhatjuk az **“if else then”** vagy az **“if else if then”** parancskombinációkat.

```
set a to 25
set b to 30
```

```
if a = b then
    beep
else
    display dialog "A két szám nem egyezik."
end if
```

Ebben az esetben nem szólal meg a figyelmeztető hang, hiszen **a** nem egyenlő **b**-vel. Az **“else”** és **“end if”** közötti rész mindenféleképpen végrehajtódik, amennyiben az első feltétel nem teljesül.

```
set a to 25
set b to 30
```

```
if a = b then
    beep
else if a > b then
    display dialog "a értéke nagyobb mint b értéke"
end if
```

A szkriptet lefuttatva gyakorlatilag semmi sem történik, hiszen sem az első, sem a második feltétel nem teljesül. Ha az első teljesült volna, megszólal a figyelmeztető hang, ha a második, akkor az **“a értéke nagyobb, mint b értéke”** dialógust kaptuk volna.

Számok összehasonlításához az alábbi jeleket használhatjuk:

**=** egyenlő  
**<** kisebb  
**>** nagyobb  
**<=** kisebb vagy egyenlő  
**>=** nagyobb vagy egyenlő  
**not** nem egyenlő

Szövegek összehasonlításához is használhatjuk a feltételes módokat:

```
set a to "Az AppleScript jó dolog."
```

```
if a begins with "az app" then  
    beep  
end if
```

A hang megszólal, ugyanis ilyen esetekben az AppleScript nem tesz különbséget kis- és nagybetű között.

Szövegek összehasonlítására az alábbi kódokat használhatjuk:

```
begins with  
ends with  
is equal to  
comes before  
comes after  
is in  
contains  
does not begins with  
does not ends with  
stb...
```

Szóközök nélkül is össze lehet hasonlítani szövegeket:

```
set string1 to "Stev e Jobs"  
set string2 to "Steve Jobs"  
ignoring white space  
    if string1 = string2 then beep  
end ignoring
```

Listaelemek összehasonlításához:

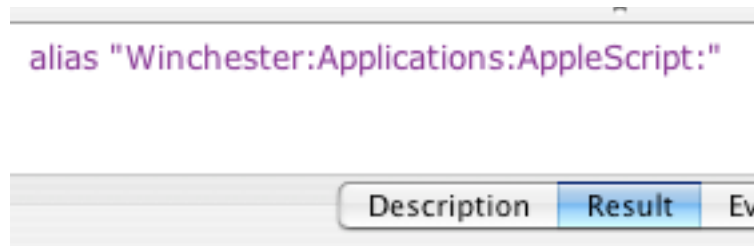
```
begins with  
ends with  
contains  
is equal to  
is in
```

## Fájlok, mappák és programok elérése

Futtassuk le az alábbi egysoros szkriptet és válasszunk ki egy mappát!

```
choose folder
```

Eredményként megkapjuk a mappa elérési útvonalát (alias):



*Fenti képen az idézőjelen belüli szöveg a "Winchester" szóval kezdődik. Ennek mindössze az az oka, hogy az én gépem merevlemezének ez a neve.*

Az elérési útvonal lényege:

**a winchester neve:mappa neve:almappa neve:... ...:a választott mappa neve:**

Egy mappa elérési útvonala, mindig kettősponttal végződik! Ha megvan a mappa, akkor ki is tudjuk nyitni.

**Figyelem!** Az elkövetkezendő példák csak abban az esetben működnek ha a szkriptben kicseréled a "**Winchester**" szót a saját géped winchesterének **pontos** nevével.

```
tell application "Finder"  
    activate  
    open folder "Winchester:Applications:AppleScript:"  
end tell
```

Egy másik lehetséges megoldás:

```
tell application "Finder"  
    open folder "AppleScript" of folder "Applications" of startup disk  
end tell
```

Ez a megoldás azért jó, mert ilyenkor mindegy, hogy pontosan mi a neve a winchesternek, a mappát mindenféleképpen kinyitja a Finder, amennyiben megtalálható.

Természetesen fájlok eléréséhez is használható az AppleScript, de ilyenkor nem kell a végére kettőspontot tennünk. Fontos viszont, hogy a fájl nevét mindig a **teljes** neve alapján kell megadni kiterjesztéssel! A Mac OS X-ben ezek többnyire el vannak rejtve. A pontos

névről tájékoztatást kaphatsz, ha egyszer rákattintasz a fájlra, majd a **Get Info...** (CMD+I) ablakban megkeresed a “**Name & Extension:**” részt.



A fenti képen szereplő PDF kép kinyitása:

```
tell application "Finder"  
    open file "Picture 1.pdf" of folder "Desktop" of home  
end tell
```

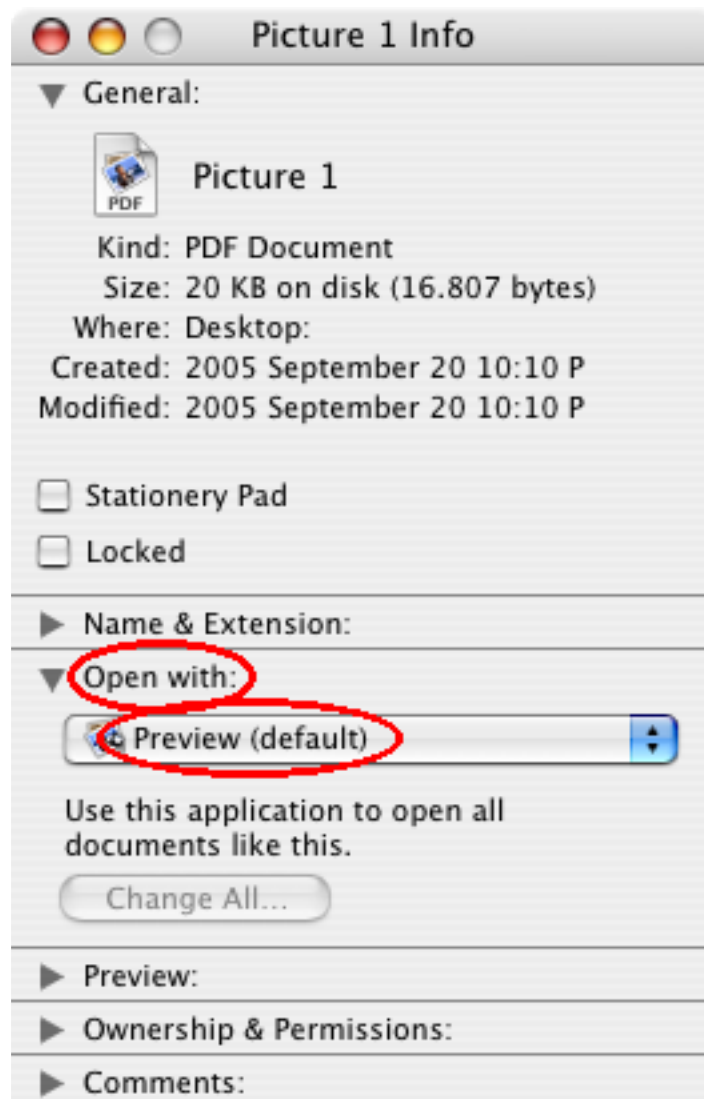
vagy:

```
tell application "Preview"  
    open "Winchester:Users:te:Desktop:Picture 1.pdf"  
end tell
```

*Fenti példában a “te” szót ki kell cserélned a “home” mappád pontos nevére ahhoz, hogy működjön a szkript.*

Mi a különbség? Az első példában a Finder-t utasítjuk arra, hogy nyissa ki a dokumentumot. Ez gyakorlatilag ugyanazt jelenti, mintha kétszer rákattintanánk a fájlra, tehát az a program

nyitja ki, ami alapbeállításként szerepel. Egy mp3 fájrt az iTunes nyit ki alapbeállítás esetén, de a QuickTime Player is le tudja játszani, ha azt utasítjuk úgy, mint a második példában.



## Ismétlések

Ha egy parancsot szeretnénk többször végrehajtani, két lehetőségünk van. Vagy többször leírjuk, vagy ismétlést használunk:

```
repeat 3 times
  say "Hello!"
end repeat
```

Ekkor háromszor mondja el a gép a “Hello” szót.

A számot helyettesíthetjük változóval is:

```
set a to 3

repeat a times
  say "Hello!"
end repeat
```

Feltételes módú ismétléseket is létrehozhatunk:

```
set a to 0

repeat until a = 3
  say "Hello"
  set a to a + 1
end repeat
```

Ekkor az ismétlés addig folytatódik, amíg a feltétel logikai értéke igaz nem lesz.

Lehetőség van az ismétlés közbeni kilépésre is:

```
set a to 0

repeat until a = 10
  say "Hello"
  display dialog "Írj be egy számot!" default answer ""
  set b to the text returned of the result as integer
  if b = 5 then exit repeat
  set a to a + 1
end repeat
```

Itt két lehetőség van. Vagy tízszer ismétli el a “Hello” szót a gép, vagy beírunk ötöt, és akkor kilép az ismétlésből.

Ismétlések közben nem tudjuk megállapítani, éppen melyik periódusnál tart a program, hacsak nem teszünk bele egy külön számlálót. Ez megnehezíti a szkriptírást és feleslegesen bonyolítja. Dolgunk egyszerűsítéséhez használhatjuk a “**repeat with**”utasítást:

```
repeat with a from 10 to 20  
    display dialog a as text  
end repeat
```

Előnye ennek a megoldásnak, hogy lépésegységet is megadhatunk, ami alapállapotban egy.

```
repeat with a from 10 to 20 by 2  
    display dialog a as text  
end repeat
```

## Hibakezelés

Szkriptírás közben megeshet, hogy hibát ejtünk valamilyen szinten. Ha szintaktikai, akkor el sem tudjuk indítani a programot, ha egyéb típusú, akkor menet közben fog leállni a futása. Utóbbit ki lehet kerülni a “**try, end try**” (próbáld) paranccsal. Működésének megértéséhez futtassuk le az alábbi szkriptet:

```
tell application "System Events"  
  get every process  
end tell
```

Mielőtt a “Run” parancsot kiadnánk, nyissuk ki az “Event Log” ablakot! Itt láthatjuk majd az összes, jelenleg futó alkalmazást, amit a **System Event**-en keresztül elérünk. Tegyük fel, hogy tíz futó programot “találtunk”, és ezeket szeretnénk egy listába tenni. Elvileg nem sok olyan számítógép-használó van, aki egyszerre több mint 50 programot futtat a gépén, de biztos, ami biztos alapon tegyük egy listába az első ötvenet az alábbi módon:

```
set a to {} -- egy üres, "lista" változó létrehozása  
repeat with x from 1 to 50  
  tell application "System Events"  
    set the end of a to name of process x  
  end tell  
end repeat  
get a
```

A szkript futása (hibaüzenettel) leáll, mivel nem talál 50 futó alkalmazást. Olyan esetekben (mint például ez is), amikor bizonytalan az eredmény, vagy számítunk bizonyos hibá(k)ra, alkalmazhatjuk a “**try, end try**” hibakezelési kombinációt. Kiegészítve az előző szkript:

```
set a to {} -- egy üres, "lista" változó létrehozása  
repeat with x from 1 to 50  
  tell application "System Events"  
    try  
      set the end of a to name of process x  
    end try  
  end tell  
end repeat  
get a
```

Így lefut a szkript hibajelzés nélkül, **a**-ban pedig megtaláljuk az első 50 futó alkalmazást ha van ennyi.

Hibakezelés közben szükségünk lehet visszajelzésre is. Ehhez használjuk az “**on error**” parancsot, ami hasonlóképpen működik, mint egy feltételes lépés.

```
set a to {} -- egy üres, "lista" változó létrehozása
```

```
repeat with x from 1 to 50
  tell application "System Events"
    try
      set the end of a to name of process x
    on error
      display dialog "Valami nem stimmel!"
    end try
  end tell
end repeat

get a
```

Az “on error” és “end try” közötti rész, csak hiba esetén fog lefutni.

## Display dialog

Ez a parancs szövegek megjelenítésére szolgál. Bizonyos mértékig testreszabható, alapvető szintaktikája pedig a következő:

**display dialog** plain text - a megjelenítésre szánt szöveg  
[**default answer** plain text] - az alapértelmezett, szerkeszthető szöveg  
[**buttons** a list of plain text] - lista egy, kettő vagy három gomb nevével  
[**default button** number or string] - az alapértelmezett gomb neve vagy száma  
[**with icon** stop/note/caution] - a megjelenítendő ikon  
[**giving up after** integer] - idő másodpercben, letelte után bezáródik a dialógus ablak.

Természetesen ezek opcionális paraméterek, annyit használunk amennyit akarunk.

```
display dialog "Ez" & return & return & "egy példa." buttons {"Rendben"} ↵  
default button 1 with icon note giving up after 5
```

A túl hosszú sorok elkerüléséhez használhatjuk az option+return bill. kombinációt (↵ jel jelenti a sortörést).

A “**default answer**” kiegészítéssel szöveget vihetünk be a programba futás közben:

```
repeat  
  display dialog "Hány éves vagy?" default answer ""  
  try  
    set a to the text returned of the result as integer  
    exit repeat  
  on error  
    display dialog "Csak számot!"  
  end try  
end repeat  
display dialog "Te " & a * 12 & " hónapos vagy."
```

Egyéb típusú adatok megjelenítéséhez (pl. szám) vagy átalakítjuk az adatot szöveggé, vagy szöveggként jelenítjük meg:

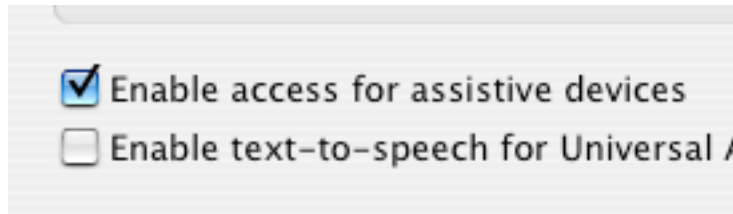
```
set a to 25  
set a to a as text  
display dialog a
```

```
display dialog 43 as text
```

## Közvetett szkriptelés

Ha egy alkalmazás közvetlenül nem szkriptelhető, lehetőségünk van vezérelni annak menüelemeit, gombjait. Ennek előfeltétele, hogy ezeket a vezérlési módokat engedélyezzük:

**System Preferences/Universal Access/Enable access for assistive devices**



A Script Editor egyik menüpontjának aktiválása:

```
tell application "System Events"  
  tell application process "Script Editor"  
    click menu item "Minimize" of menu "Window" of menu bar 1  
  end tell  
end tell
```

Közvetett vezérlésre csak a System Events képes.

Egy program menüelemeinek feltérképezése:

```
tell application "System Events"  
  tell process "Finder"  
    tell menu bar 1  
      get every menu  
      get every menu item of every menu  
      get every menu of every menu item of every menu  
      get every menu item of every menu of every menu item of  
        every menu  
    end tell  
  end tell  
end tell
```

Az eredményt az **Event Log** ablakban láthatjuk. Elsőre kissé bonyolultnak tűnik, de egyszerűen kitalálható, hogy milyen szintaktikát kell alkalmazni. Az egyes elemek vesszőkkel vannak elválasztva. Fentről lefelé keressük meg azt a részt, amelyikben először szerepel a kérdéses menüpont! Pl. keressük meg az **“Arrange”** menüpont **“by Name”** almenüjét! Eredményként ezt kapjuk:

```
menu item "by Name" of menu "Arrange" of menu item "Arrange" of menu  
"View" of menu bar item "View" of menu bar 1 of application process "Finder"
```

```
tell application "Finder" to activate
tell application "System Events"
    tell application process "Finder"
        click menu item "by Name" of menu "Arrange" of menu item →
            "Arrange" of menu "View" of menu bar item "View" of menu bar 1
    end tell
end tell
```

Egy példa arra, hogy hogyan léphetünk be a “beszeljukmac” szobába az iChat-el:

```
tell application "iChat" to activate
tell application "System Events"
    tell application process "iChat"
        keystroke "g" using command down
        keystroke "a" using command down
        keystroke "beszeljukmac"
        tell application "iChat" to activate
        delay 1
        keystroke return
    end tell
end tell
```

Fenti példában megoldást látunk arra, hogyan szimulálhatjuk a billentyűzet leütéseket a **keystroke** utasítással. Az idézőjeleken belüli **y** és **z** betű felcserélődik magyar billentyűzetkiosztás használata esetén.

Több módosító bill. használata is megengedett:

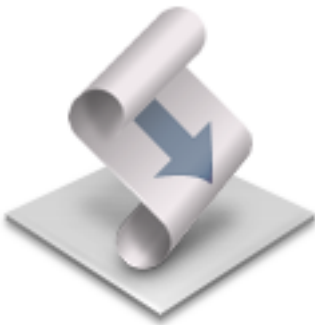
```
keystroke a using {command down, shift down, control down}
```

## Dropletok

A megírt szkripteket elmenthetjük többféleképpen, de a dropletokról még nem esett szó. Ezek olyan, alkalmazásként elmentett AppleScript-ek, amelyekre rádobhatunk akármit így aktivizálva a benne lévő kódot. Ilyen dropletet egyszerűbb készíteni, mint leírni. Mindössze két plusz sort kell a programhoz adni: az **“on open”** és **“end open”** kódokat.

```
on open
    beep
    display dialog "Rámdobtál valamit."
end open
```

Ezt a rövidke szkriptet elmentve alkalmazásként egy dropletet kapunk, amelyre bármit rádobva megszólal a rendszer figyelmeztető hangja. Az így elmentett szkript-alkalmazás ikonja kissé eltér a megszokottól, jelezve, hogy dropletről van szó:



Természetesen a programra dobott elemekre tudunk hivatkozni, de egy kissé módosítani kell a szkriptet:

```
on open a
    set a to a as alias
    tell application "Finder"
        display dialog name of a as text
    end tell
end open
```

Ebben a példában **“a”** az a változó, ami tartalmazza annak a fájlnek az elérési útvonalát, amit rádobtunk. A változó típusa **mindig szöveg**. Ha szeretnénk valamit csinálni a rádobott elemmel, ami lehet akár egy mappa is, akkor a változót át kell alakítani!

Kombinált program (droplet és alkalmazás) létrehozására is van lehetőség, de ebben az esetben két részre kell osztani a kódot!

Az **“on open”** és **“end open”** közötti rész akkor fut le, ha valamit rádobunk, az **“on run”** és **“end run”** közötti rész pedig akkor, ha elindítjuk a programot.

```
on open a  
    display dialog a as string  
end open
```

```
on run  
    display dialog "Most elindítottál."  
end run
```

# Források, megjegyzések

Ezzel a leírással (tulajdonképpen fordítás) igyekeztem bemutatni az AppleScript alapfunkcióit természetesen a teljesség igénye nélkül, mindazonáltal remélem, hogy hasznát tudod venni amennyiben szeretnél megismerkedni az AppleScript-el!

Miként érdemes tanulni? Legegyszerűbb példákon keresztül, amelyek találhatóak a gépeden is:

**HD/Library/Scripts/**

Ezeket a szkripteket a ScriptEditor programból is elérheted, csak egy jobb-egér (CTRL +klick) kattintásra van szükséged egy nyitott Script Editor ablakon belül.

**<http://macscriter.net>**

A legnagyobb angol nyelvű AppleScript fórum, rengeteg működő szkripttel. Itt érdemes keresgélni!.

Köszönet Dujunak korrektori munkájáért!

web: bakman.hu  
mail: bachman1@gmail.com  
iChat: bachman1@mac.com (AIM)  
MSN: bachman111@hotmail.com



Bachman