Python bevezető

Mit jelent a programozás?

Számítógépek, telefonok működéséhez programokra van szükségünk. Az eszközök önmagukban csak **HARDVEREK**. A rajtuk futó **SZOFTVEREK** a működtetői. A szoftverek (programok) forráskódját **MAGAS SZINTŰ PROGRAMOZÁSI NYELVEKKEL** készítik el. Vagyis programoznak.

Programozási nyelv főbb összetevői

- **szintaxis**: leírt kód helyességét meghatározó szabályok (*Nyelvtani szabályok kézikönyve*)
- **szemantika**: adott nyelven megírt program jelentését és tartalmi helyességét leíró szabályok

Szintaktikai hibáról akkor beszélünk, amikor a forráskódunk hibás. Ezen hibák pedig megakadályozzák a Pythont, hogy értelmezze (elolvassa) a forráskódot.
Szemantikai hibáról akkor beszélünk, amikor a Python már képes értelmezni a kódot, de nem megfelelően működik. *Ezen hibákról később lesz szó bővebben.*

Fordított és interpretált nyelvek

Fordított nyelvek

Azon nyelvek, melyeknél a **COMPILER** (fordító) egyszer és mindenkorra elvégzi a gépi szintre történő átalakítást. Ilyen nyelv: *Pascal, C#, C*

Előnyei:

- fordított kód gyorsabb végrehajtása, lefutása
- a bináris program másolható más gépre, és ott azonnal futtatható

Interpretált nyelvek

Azon nyelvek, melyeknél az alkalmazás minden egyes futtatásakor újra és újra át kell alakítania a **SZKRIPTET** (interpretált nyelv forráskódját). A forráskódot egy interpreter (parancsértelmező) dolgozza fel. Ilyen nyelv: *JavaScript, Python*

Előnyei:

- gyorsan fejleszthetőek
- más platformokon ugyanaz a forráskód futtatható (*platformfüggetlen*)

Hátrányai:

- lassabb futás (*hardverközeli*)
- nyelv interpretere szükséges

Programozási környezet

Python-programok elkészítéséhez elengedhetetlen a megfelelő programozási környezet kialakítása. Linux és a macOS operációs rendszereken a Python értelmezője alapértelmezetten megtalálható, míg Windows-on ezt külön kell telepíteni.

Az értelmező (interpreter) telepítésével elérhetővé válik a Python *integrált fejlesztői környezete* (röviden: **IDLE** [1]– Integrated Development and Learning Environment). Integrált fejlesztői környezetből (röviden: **IDE** – Integrated Development Environment) több is létezik. Például **VS Code**[2] vagy **PyCharm**. Akár online felület is rendelkezésünkre állhat (pl. a Replit oldala).

Program

A **PROGRAM** *utasítások sorozata*. Az utasítás adatokkal dolgozik.
A legtöbb utasítás **FÜGGVÉNY**, a függvény neve mögé nyitó és csukó zárójelet írunk. A legtöbb függvénynek paramétere(i) van(nak), amit a zárójelen belülre írunk.
A későbbiekben elágazások, ciklusok, függvények létrehozásánál elengedhetetlen a kód megfelelő behúzása. Rossz behúzás szintén hibákhoz vezethet (lásd szemantikai hibák).

Változók, adatok kiírása és bekérése

Változók

Memóriában tárolt hivatkozás egy adatra. A változó névvel, típussal (és értékkel) rendelkezik. A Python **DINAMIKUSAN TÍPUSOS NYELV**, mely azt az előnyt nyújtja számunka, hogy automatikusan típust kap a változó értékadás során. (*A típusról a következő órán bővebben beszélünk majd.*)

1. **Név**
   a. betűket, számokat és aláhúzásokat tartalmazhat
   b. nem kezdődhet számmal
   c. általában kis betűkkel írjuk őket
2. **Típus (érték)**
   a. egész szám (pl.: 4; 1; 5)
   b. törtszám (pl.: 2.5; 3.13)
   c. szöveg (pl.: „Helló, világ!”)
3. **Értékadás**
   a. változó_neve = érték
   b. egy sorban több adatnak **különböző** értékeket is lehet adni

```
pl.: adat1, adat2 = ertek1, ertek2
```

   c. egy sorban több adatnak **ugyanolyan** értékeket is lehet adni

```
pl.: adat1, adat2, adat3 = ertek
```

[1] A(z) „IDLE” szóra kattintva el lehet navigálni a ***python.org*** oldalra, ahol letölthető ezen szoftver.
[2] Segítség a python telepítéséhez VS Code-ban: https://youtu.be/fAW8aMCobAA

Alapvető műveletek változókkal

Zárójelezés **()**; szorzás ***;** osztás **/**; összeadás **+**; kivonás **–**

Speciális karakterek

- **\n** → soremelés (**n**ew line)
- **\t** → **t**abulátor
- **\”** → idézőjelek megjelenítése
- **\’** → aposztrófok megjelenítése

Adatok kiírása (Output)

A **print()** függvény segítségével írunk a képernyőre (terminálba).

- több adat kiírása vesszővel elválasztva történik

```
pl.: print(adat1, adat2, adat3)
```

- matematikai műveletek is tehetők bele

```
pl.: print(4 + 2)
```

Két fő paraméterrel rendelkezik:

1. **sep** paraméter
   a. *elválasztó karakter megadására szolgál*
   b. **alapértelmezett**: *space*
   c. utolsó karakterre nem tesz elválasztót! ☹/😊

```
print(1, 2, 3, sep='; ')
```

Például a fenti kód eredménye a képernyőn: 1; 2; 3

2. **end** paraméter
   a. *adatok kiírása után milyen karakter jelenjen meg*
   b. **alapértelmezett**: *\n*

```
print("alma", "körte", sep=', ' end='és ')
print("cseresznye")
```

Például a fenti kód eredménye a képernyőn: alma, körte és cseresznye

Adatok bekérése (Input)

A(z) **input()** függvény segítségével kérünk be adatot a felhasználótól.

- ENTER leütésének hatására olvassa be a terminálba beírt adatokat
- legtöbb esetben valamilyen változót rendelünk hozzá

```
adat = input("Add meg az adatot! ")
```

- paraméterben lehet tájékoztatni a felhasználót, hogy milyen adatot várunk tőle
- a bekért adat MINDIG szöveg típusú!

Szintaktikai és szemantikai hibák

Szintaktikai hibák – SyntaxError: invalid syntax

- nincs megfelelő behúzás
- lemaradt zárójelek, vesszők
- függvények rossz használata

```
print(hello world)
```

Például az alábbi kód hatására a program szintaktikai hibát kapna, hiszen a print csak változót vagy szöveget tud terminálra írni. A hello world nem lehet változó, hiszen kellene akkor a szóköz helyére egy elválasztó karakter, és nem lehet szöveg, hiszen hiányoznak róla aposztrófok.

```
print("Szia"
```

A fenti kód szintén szintaktikai hibába fut, mivel lemaradt a függvény (print) záró zárójele.

Szemantikai hibák

- algoritmus rossz implementálása
- a program nem úgy működik, ahogy szeretnénk

```
jo_adat = input("Add meg a neved! ")
rossz_adat = 123

print("A felhasználó neve: ", rossz_adat)
```

Például a fenti kód sikeresen ír ki a terminálra, azonban nem várt eredményt. Ez azért van, mert szemantikai hibát vétettünk. A rossz_adat helyett, a jo_adat változót kellett volna használnunk.

Kommentek írása

A programozás során kommenteket szoktunk hagyni a kódba, ezzel

- a bonyolult kódokat elmagyarázhatjuk a forráskódban (de ezt röviden)
- jelzést tehetünk magunknak, hogy ez a rész még kódolásra vár

```
# Ez egy egysoros komment.
print("alma") # Így is írhatunk kommentet.

'''
Többsoros kommentet írhatunk
3 aposztróf vagy 3 idézőjel közé írva.
'''
```

Kikommentelés során az adott utasítás elé kettőskeresztet teszünk, így az nem hajtódik végre.

```
# print("alma")
```