Szám és karakterlánc adattípusok

A python **DINAMIKUSAN TÍPUSOS NYELV**, azaz a változókhoz rendelt adatok automatikusan kapnak típust értékadáskor. Egy változó típusát a **type()** függvény segítségével kérdezhetjük le.

```
pelda = 123
print(type(pelda))
```

*Például a fenti kód hatására a konzolon megjelenik a **<class 'int'>** szöveg, mely jelzi, hogy a **pelda** nevezetű változónk **int** típusú, azaz egész számot tartalmaz.*

A változók típusának ismerése elengedhetetlen, hiszen különböző típusú adatokkal más és más műveletek végezhetőek el.

Szám (műveletek, konverzió)

Ide tartoznak az **egész** (*python: int*) és **valós** számok (*python: float*).
Egész számok például: -2; -1; 0, 1; 2
Valós számok például: -2.1; 0; 1.2

Műveletek: (Több művelet esetén az ismert matematikai precedencia szabály érvényesül.)

- *hatványozás* (**)
- *szorzás* (*)
- *osztás* (/) – egész számok osztása esetén is MINDIG valós számot ad eredményül.
- *összeadás* (+)
- *kivonás* (-)
- *maradékos osztás* eredménye (div vagy //)
- *maradék* meghatározása (mod vagy %)

Konverzió

Egész és valós számok közötti típusváltást értjük a konverzió alatt jelen esetben. (Általánosságban: *Konverzió során egy változó típusát megváltoztatjuk, változásra kényszerítjük egy másik típusra.*)

int() függvény segítségével int-é alakíthatjuk az adott változót
float() függvény segítségével pedig float-á

Leggyakrabban ezeket szám-adatok bekérésénél használjuk. Így a bekért adattal – melyet változóba mentettünk – elvégezhetjük a fent említett műveleteket.

```
pelda_szam = int(input("Adj meg egy számot! "))
```

Ha ezt nem tennénk, szövegként kezelné a Python az adatot. Lássuk erre milyen szabályok vonatkoznak.

Karakterlánc (hibaüzenetek, összefűzés, szeletelés)

Ide tartoznak a **szöveges** (*python: str – string*) típusú adatok.
Például: „Helló, világ!”; ’Python’

Műveletek:

- *összeadás* – összefűzi a szövegeket/szavakat/betűket
- *szorzás* – szorzónak megfelelően megismétli az adott szöveget/szót/betűt

Ezen műveletek a számoknál is definiált, viszont más végeredményt adnak a változó típusától függően.

```
szo1 = "alma"
szo2 = "fa"
szo3 = szo1 + szo2
```

*Például a fenti kódban a **szo3** eredménye két **str** típusú változó összefűzése. Ha kiíratnánk a szo3 tartalmát a képernyőre a következőt kapnánk: „almafa”*

```
szo1 = "alma"
print(szo1 * 3)
```

*Ezen kód során a képernyőre pedig háromszor kiíródna a **szo1** tartalma, vagyis: „almaalmaalma”*

- **len()** – meghatározza hány karakterből áll a szöveg/szó

```
szo1 = "alma"
print(len(szo1))
```

*A képernyőn megjelenik, hogy a **szo1** változó hossza: 4*

- *szoveg[index]* – visszaadja az *index*-edik helyen álló karaktert a *szoveg* változóban
  - legelső index értéke MINDIG: **0**
  - legutolsó index értéke MINDIG: **-1** vagy **len()-1**

```
szo1 = "alma"
betu = szo1[1]
print(betu)
```

0: ’a’ | 1: ’l’ | 2: ’m’ | 3: ’a’
*A képernyőn megjelenik a **szo1** változó 1. helyen szereplő karaktere, vagyis az ’l’.*

- *szeletelés kettősponttal: szoveg[n:m]* – visszaadja az n-edik és m-edik indexértékek közé eső karaktereket. FONTOS!!! Az m-edik karaktert már nem adja vissza.

```
szo1 = "alma"
print(szo1[2:4])
```

*A képernyőn megjelenik a **szo1** változó 2. hely és 4. hely közötti karakterek, beleértve a 2. helyen szereplőt is: ma*

Konverzió

str() függvény segítségével string-é alakíthatunk változókat

```
szam1 = 123
szoveg1 = str(szam1)

szam2 = 456
szoveg2 = str(szam2)

szoveg3 = szoveg1 + szoveg2
print(szoveg3)
```

Például a fenti kód során a szam1 és szam2 változókat sztringé alakítjuk és eltároljuk őket a szoveg1, illetve szoveg2 változókban. Mikor a szoveg3 változó eredményeként összefűzzük az előző str típusú változóinkat, és ezt kiírjuk a képernyőn a következő jelenik meg: „123456”