Számok és karakterláncok a programunkban

Eddig még csak karakterláncot (szöveget) tároltunk a változóinkban. Ebben a leckében változtatunk ezen, és számokat is használunk.

Hány éves a felhasználó?

Olyan programot fogunk írni, amely választ ad a fenti kérdésre. Az első részfeladat egy olyan program megírása, amely megkérdi, hogy mikor születtünk, és ezt ki is írja nekünk. Ez még nem tartalmaz új tudáselemet, úgyhogy készítsük el egyedül, majd olvassunk tovább!

A következő részfeladat a felhasználó korának meghatározása. Ehhez a programunknak tudnia kell, hogy melyik évben futtatják. A jelenlegi év megkérdezhető az operációs rendszertől, de egyelőre megelégszünk azzal, hogy változóként felvesszük a programunkba.

Azokat az értékeket, amelyek később nem változnak, konstansnak nevezzük, és vannak olyan programozási nyelvek, amelyeknél külön jelöljük. A Pythonban úgy *szokás*, hogy az ilyen értéket tároló változóknak csupa nagybetűs nevet adunk. A konstansnak tekintett változókat szokás a program elején megadni, azaz a programunk mostanra nagyjából így néz ki:

```
1.  IDEI_ÉV = 2021
2.  felhasználó_kora = input('Hány éves vagy?')
3.  print('Te most', felhasználó_kora, 'éves vagy.')
```

A negyedik sorunk alighanem egy kivonás lesz, például

```
születési_év = IDEI_ÉV - felhasználó_kora
```

vagy hasonló. Ha ebben az állapotban lefuttatjuk a programunkat, a kérdésre még megvárja a választ, de utána hibaüzenettel leáll.

Hányadik sorra vonatkozik a hibaüzenet?

A hibaüzenet az újonnan beírt sorban van, és átböngészve találunk benne olyat is, hogy `int` és `str`, előttük meg egy kivonásjel. Ez a három dolog a lényeg.

A Python azt igyekszik elmagyarázni, hogy nem tud egy egész számból (angolul: integer, röviden `int`) kivonni egy karakterláncot, más szóval szöveget (angolul: string, `str`).

Ha úgy gondoljuk, hogy itt valami tévedés lesz, mi rendes felhasználó módjára a programban feltett kérdésre számmal válaszoltunk, akkor igazunk van, de el kell fogadnunk, hogy a Python meg óvatos. Nem tudhatja, hogy amit válaszoltunk a kérdésére, azt biztosan számnak, tízes számrendszerbeli számnak gondoltuk. Ezért minden, amit az `input` a programnak átad, szöveg, azaz `str` marad. Ha 15-öt válaszoltunk az előző kérdésre, a program lát egy 1-es és egy 5-ös karaktert, de csak mint két egymás utáni karaktert, nem pedig egy számot.

Mik is azok a műveleti jelek?

Még iskoláskorunk legelején megtanultuk, hogy mik azok, de ha meg kell fogalmazni, esetleg elbizonytalanodunk. Végül talán arra gondolunk, hogy a műveleti jel olyan jel, ami a mellette álló adattal, adatokkal egy műveletet végez.
Futtassuk az alábbi egyszerű programot, és gondolkodjunk el a kimenetén!

```
1. print(10 + 3)
2. print('10' + '3')
3. print('Ej' + 'nye!')
4. print(10 * 3)
5. print(10 * '3')
6. print(10 * 'Abc')
```

Az aposztrófok közé írt szám NEM szám!

Fogalmazzuk meg a tanulságokat:

Az összeadásjel:

- két számot összead,
- két karakterláncot egymás mellé ír.

A szorzásjel:

- két számot összeszoroz,
- számot észlelve a karakterláncot egymás mellett a számnak megfelelően megismétli.

Az, hogy a műveleti jel pontosan milyen műveletet végez, *attól is függ, hogy az adat milyen* ***típusú****.* Ezért nem találgat a Python, hanem azt várja, hogy pontosan adjuk meg az adat típusát.

Típusátalakítás

Térjünk vissza a felhasználó születési évét kiíró programunkhoz. Azt már tudjuk, hogy az `input` utasítás karakterláncot ad vissza, és azt is, hogy nekünk számra van szükségünk. A típusátalakítást, típuskonverziót a Pythonban a céladattípus nevével megegyező utasításokkal végezzük el: az `int('2021')` utasítás eredménye 2021, számként. Ennek figyelembevételével programunk így alakul:

```
1. IDEI_ÉV = 2021
2. felhasználó_kora = input('Hány éves vagy? ')
3. print('Te most', felhasználó_kora, 'éves vagy.')
4. felhasználó_kora = int(felhasználó_kora)
5. születési_év = IDEI_ÉV - felhasználó_kora
6. print('Ekkor születtél: ', születési_év, '.', sep='')
```

„Kozmetikai” szóköz, hogy szebb legyen a kimenet.

A **típuskonverzió** a negyedik sorban van. Próbáljuk ki a kész programot!
Szeretnénk *pontosan* érteni, hogy mit csinál a típuskonverziót végző utasítássor, úgyhogy most mondjuk ki fennhangon, hogy mit csinálnak az alábbiak!

- szám = 2 + 4517
- majmok = orangutánok + cerkófok

Remélhetőleg nagyjából ezeket mondtuk:

- Adjuk össze a két számot, és az eredményt tegyük a „szám” változóba!
- Olvassuk ki az orangutánok és a cerkófok változó tartalmát, és az eredményt tegyük a „majmok” változóba!

Ha megfigyeljük a mondatainkat, látjuk, hogy előbb foglalkozunk a fenti sorok egyenlőségjeltől jobbra álló oldalával, és csak utána a bal oldallal.

Eddig három műveleti jelünk (operátorunk) volt, a „+”, a „–” és a „*” jel, de mostanra el kell fogadnunk, hogy programozáskor az egyenlőségjel is műveleti jel. Alapvetően mást jelent ilyenkor az egyenlőségjel, mint matematikaórán. Ott állításokat, kijelentéseket fogalmaztunk meg vele (Kettő egyenlő: háromból egy. Hatszor hat egyenlő harminchattal.). Programozáskor az egyenlőségjel egy művelet elvégzésére való *felszólítás*, nézzük csak meg az előbbi számos és majmos mondatot!

Programíráskor az egyenlőségjel az **értékadás** műveletének műveleti jele, olvashatjuk „legyen egyenlő” formában. Értékadáskor mindig az egyenlőségjel jobb oldalát értékeli ki a program elsőként, majd a kapott eredményt értékül adja az egyenlőségjel bal oldalán lévő változónak.

A `felhasználó_kora = int(felhasználó_kora)` sor tehát a következőt jelenti:

1. Olvassuk ki a `felhasználó_kora` változó tartalmát (ez az egyenlőségjel jobb oldalán lévő `felhasználó_kora`)!
2. A kapott értéket adjuk át az `int` utasításnak (ami majd egész számmá alakítja)!
3. Az `int` utasítás eredményét adjuk értékül a `felhasználó_kora` változónak (felülírva ezzel a régi értéket)!

Szemléletes, ha úgy képzeljük el, hogy a dobozból kivesszük a benne lévő szöveget, átgyúrjuk számmá, aztán visszatesszük ugyanabba a dobozba.

És amit nem lehet átalakítani számmá?

Abból bizony hibaüzenet lesz.

Nézzük a programunk alábbi két futtatását!

```
C:\Users\raerek\programjaim>hanyeves.py
Hány éves vagy?
Te most  éves vagy.
Traceback (most recent call last):
  File "C:\Users\raerek\programjaim\hanyeves.py", line 4, in <module>
    felhasználó_kora = int(felhasználó_kora)
ValueError: invalid literal for int() with base 10: ''

C:\Users\raerek\programjaim>hanyeves.py
Hány éves vagy? csillijómillijó
Te most csillijómillijó éves vagy.
Traceback (most recent call last):
  File "C:\Users\raerek\programjaim\hanyeves.py", line 4, in <module>
    felhasználó_kora = int(felhasználó_kora)
ValueError: invalid literal for int() with base 10: 'csillijómillijó'
```

Nem adunk meg értéket, csak Entert nyomunk.

A semmit a Python nem tudja számmá alakítani.

A szöveget sem.

Természetesen kezelhetők az ilyen jellegű hibák, csak mi még nem tanultuk meg a módját. Még sokáig abból indulunk ki programkészítéskor, hogy a felhasználótól érkező bemenetet nem kell ellenőriznünk, validálnunk. Egy „igazi” alkalmazás esetében a hibákra való felkészülés (felhasználótól kapott rossz bemenet, elfogyó háttértár, megszakadó hálózati kapcsolat stb.) a programnak igen jelentős része.

Karakterlánccá alakítás

Láttuk már, hogy két karakterlánc összeadható, és azt is láttuk, hogy a `print` utasításnak több kiírnivaló is átadható, és ezeket vesszővel választjuk el. De lehet olyat írni, hogy `print('alma'+ ' ' + 'körte' + ' ' + 'dió')`? Hogyne! Ilyenkor előbb „összeadódnak” a karakterláncok, és ezt követően egyetlen karakterláncot kap meg a `print`. Persze itt éppen megvagyunk e módszer nélkül, mert a vesszővel való felsorolás remekül működik (lásd a programunk 6. sorát), de van, ahol ez probléma.

Egészítsük ki a programunkat: kérdezze meg, hogy „És milyen N évesnek lenni?”, ahol N természetesen a felhasználó korábbi válaszában szereplő szám!

Logikusnak tűnik egy `ilyen = input('És milyen', felhasználó_kora, 'évesnek lenni?')` megoldás, de sajnos az `input` nem ismeri a `print` vesszős összefűzési módszerét. Ha a „+” operátort használjuk, akkor egy másféle hibába csöppenünk, de azért próbáljuk csak ki, könnyű lesz korrigálni!

A hibaüzenet ismét `int`-ről és `str`-ről beszél, és ekkor már gyanítjuk, hogy az lehet a baj, hogy a `felhasználó_kora` a 4. sor óta egész szám, amit nem tud a Python „összeadni” egy karakterlánccal. Amikor egész számmá akartunk alakítani, az `int` utasítást használtuk. Amikor karakterlánccá alakítunk, az `str` utasításra van szükség. Programunk utolsó két sora ezt a formát ölti:

```
7. felhasználó_kora = str(felhasználó_kora)
8. ilyen = input('És milyen ' + felhasználó_kora + ' évesnek lenni? ')
```