



## 13. hét: adatszerkezetek

 Czirkos Zoltán ·  2023.11.22.  
Listák leképezése és szűrése. A dict és set típusok használata.

Felkészülés a laborra:

- A [listák leképezése és szűrése](#) témakör megértése.
- A [Python beépített tárolóinak](#) ismerete.

### Tartalom

1. Ötödik kis ZH
2. [Listák leképezése és szűrése](#)
3. [Szász Pál verse](#)
4. [Charlie](#)
5. [Zöldséges](#)
6. [DogaDict](#)
7. [DogeDict](#)

### 1. Ötödik kis ZH

A hétfői laborokra csúszott az ötödik kis ZH. Fáj!l használat a témakör.

### 2. Listák leképezése és szűrése

Alakítsd az alábbi programrészeket list comprehension kifejezéssé!

```
l = []
for i in range(0, 10):
    l.append(i * 2)
```

```
l = []
for i in range(100):
    if i % 10 == 3:
        l.append(i)
```

```
l1 = [43, 15, 48, 59, 33, 72, 11, 65, 95, 34]
l2 = []
for i in l1:
    if i % 2 == 1:
        l2.append(i)
```

```
l1 = [43, 15, 48, 59, 33, 72, 11, 65, 95, 34]
l2 = []
for i in l1:
    if i % 2 == 1:
        l2.append(True)
    else:
        l2.append(False)
```

# Vigyázz, ebben átvenés az if.

```
l1 = ["alma", "körte", "barack", "szilva", "ananasz"]
l2 = []
for s in l1:
    if s[0] == 'a':
        l2.append(s)
```

```
l1 = ["alma", "körte", "barack", "szilva", "meggy"]
l2 = []
for s in l1:
    l2.append(len(s))
print(l2)
```

### 3. Szász Pál verse

Alább olvashatod Szász Pál versét. A vers érdekessége, hogy a szavak hossza, azaz a betűik száma kiadja a  $\pi$  tizedesjegyeit. Nem  $\rightarrow$  3, a  $\rightarrow$  1, régi  $\rightarrow$  4, és így tovább.

Nem a régi s durva közelítés,  
Mi szótól szóig így kijön  
betűiket számlálva.  
Ludolph eredménye már,  
ha itt végezzük húsز jegyen.  
De rendre kijő még tíz pontosan,  
azt is bizvást ígérhetem.

A feladatot: megkapod a verset az alább látható formában; a sorokat / jellel elválasztva, ahogy azt irodalom óráról ismerjük. Írd ki a vers alapján a  $\pi$  tizedesjegyeit! Használj lista szűrés és leképezés műveleteket!

vers = "Nem a régi s durva közelítés, / Mi szótól szóig így kijön / betűiket számlálva. /  
Ludolph eredménye már, / ha itt végezzük húsز jegyen. / De rendre kijő még tíz pontosan, /  
azt is bizvást ígérhetem."

A program ezeket a számjegyeket kell majd kiírja:

```
3141592653589793238462643383279
```

### 4. Charlie

Charlie fagyilaltot árul: jelenleg pisztácia, vanília, tutti-frutti, karamell, rumos dió és kávé a választék. A fagyit íz alapján kérhetik a gyerekek, egyszerre csak egy gombócot. A készlet véges, minden gombóc eladásával értelemszerűen eggyel csökken.

Tárold el a rendelkezésre álló mennyiségeket egy szótárban, azaz **dict** tárolóban!

```
fagyik = {
    "pisztácia": 0,
    "vanília": 3,
    "tutti-frutti": 8,
    "karamell": 4,
    "rumos dió": 5,
    "kávé": 9,
}
```

Írj programot, amely a vásárlásokat kezeli! Olvasd be a vásárlásokat (ízeket) üres sorig. Keresd meg az előbb megírt függvénnyel a kapott ízt, és jelezd a vásárlás eredményét: sikeres, vagy kifogyott (volt, de 0-ra csökkent), esetleg nem is volt!

```
vanília
kösz, őcsi!

pisztácia
pisztácia kifogyott!

csokoládé
csokoládé nem is volt!
```

### 5. Zöldséges

Adott egy zöldséges alábbi raktárkészlete és árai:

```
keszlet = {
    "banán": 6,
    "alma": 31,
    "narancs": 32,
    "körte": 15
}

arak = {
    "banán": 100,
    "alma": 80,
    "narancs": 120,
    "körte": 90
}
```

#### Első feladat

A programodban egy vásárlást kell kezelni. A felhasználó megadja, hogy mit szeretne venni – több zöldség vagy gyümölcs nevét is megadhatja, egészen üres sorig tart a bemenet. Ezután írja ki a program, hogy mennyi a vásárlás végösszege! Természetesen csökkenjen is a készlet, vagy ha olyan tételt kért, amiből nincsen, akkor jelezze azt a program.

```
banán
OK
dinnye
Nincs raktáron.
alma
OK

Végösszeg: 180
```

#### Második feladat

Adott ugyanez a két tároló. Mennyit ér a raktárkészlet, azaz mennyi pénzt keresne a zöldséges, ha eladná az összeset?

### 6. DogaDict

A [zheredmeny.txt](#) egy képzeletbeli évfolyam NEPTUN kódjait és dolgozatok eredményeit tárolja. Minden sor egy dolgozat adatait tárolja: NEPTUN kód és pontszám, kettősponttal elválasztva.

```
DHSF7F:24
ZPD9PY:17
LGNUK6:3
XMO05D:12
...
```

Tudjuk, hogy a NEPTUN kódok egyediek, ezért egy szótárban (**dict** típusban) kulcsnak használhatóak.

- Írj függvényt, amelyik beolvassa egy fájlt, és létrehoz egy szótárt. A szótár kulcsai a NEPTUN kódok, értékei pedig a dolgozat pontszámok lesznek.
- Hányan írták meg a dolgozatot? Írd ki a programban a **dict** ismeretében!
- Készíts statisztikát, hány pontos dolgozatból hány darab született! Vedd észre, hogy ehhez nem kell a ismerni a következő adatokat! Pl. `kiknek[7]` azt fogja tárolni, hogy kik azok (mi a NEPTUN kódjuk azoknak), akik 7 pontos dolgozatot írtak. Ellenőrzésképp néhány adat:

```
0: 1
7: 3
25: 2
```

- Készíts egy másik szótárt is, amely szintén pontszámmal indexelhető, de most nem létszámot, hanem listát tartalmaz! Pl. `kiknek[7]` azt fogja tárolni, hogy kik azok (mi a NEPTUN kódjuk azoknak), akik 7 pontos dolgozatot írtak. Ellenőrzésképp pár adat:

```
0: ['E3YX24']
7: ['OI0IOI', 'UPAXFK', 'NDAI3P']
25: ['IP1WBY', 'K8TO00']
```

Írj függvényt, amelyik pontszámok szerint növekvő sorban listáz egy ilyen szótárt! Mivel a szótár elemel össze-vissza tárolódnak, ezért rendezni kell őket. Kulcsok szerint iterálással, `sorted(d.keys())`, vagy `sorted(d.items())`.

### 7. DogeDict

Egy kutyatenyésztő számára kell programot írnod, amely a kutyákat tartja nyilván. Egy kutyáról a következő adatokat kell megjegyezni: 1) ID, vagyis azonosító, egész szám, 2) név, 3) papa és 4) mama, a kutya szüleinek azonosítói, vagy -1 értékek, ha ismeretlenek.

Mivel egy kutyát mindig ID alapján kell megtalálni, ezért tárolhatod őket egy **dict**-ben. Pl. az 5-ös azonosítójú kutya a `doges[5]` lesz:

```
doges = {}
doges[5] = Doge(5, "Woof")
doges[7] = Doge(7, "Floof")
```

Így bár úgy viselkedik, mint egy lista, de az indexek akármilyen egész számok lehetnek.

Az alábbi részfeladatok megvalósítása után mindig teszteld a megírt programrészeket!

- Defináld a **Doge** osztályt!
- Írj függvényt, amelyik kiírja az összes tárolt kutya nevét és azonosítóját!
- Írj függvényt, amely egy a paraméterként kapott tárolóban megkeresi a szintén paraméterként kapott azonosítójú kutyát! A visszatérési érték a megtalált elem, vagy **None**. Teszteld ezt olyan módon, hogy a felhasználótól kérsz azonosítókat!
- Írj függvényt, amely egy megadott azonosítójú kutyát töröl a tárolóból! Figyelj arra, hogy ilyenkor át kell vizsgálni az adatbázist: ha a törölt listaelemet valamelyik másik elem szülőként (papa, mama) hivatkozik, akkor azoknál -1-et kell beírni.
- Írj függvényt, amely a paraméterként kapott nevű szöveges fájlba kiírja a kutyák adatait, soronként íd, név, papa id, mama id formában, szőközökkel elválasztva. Ha ismeretlenek a szülők, az azonosítók helyére a fájlban is -1 kell kerüljön.
- Írj függvényt, amely visszaolvas egy ilyen fájlt, betölti onnan a kutyák adatait!

