Python trükkök, rövidítések

Az emelt szintű érettségi programozási feladatinak megoldási idejét és a programkód hosszát a Pythonban jelentősen lerövidítheti egy-két olyan fogás, amely a programnyelv sajátja. Mivel a Python egy jól átgondolt, magas szintű programozási nyelv, jó pár olyan szerkezet megtalálható benne, ami az algoritmusok kódolásának hatékonyságát megnöveli. Ezekből szedtem csokorba néhányat.

Listák, halmazok kezelése

A lista és a halmaz pár alapvető dologban különbözik:

- A listában egy elem többször is előfordulhat, a halmazban csak egyszer. Tehát ilyen van: lista első eleme: `lista[0]`, utolsó eleme: `lista[-1]`, de halmazra nincs.
- A lista rendezhető, a halmaz nem (nem lehet névsor vagy nagyság szerint sorba állítani az elemeit, viszont lehet benne maximumot vagy minimumot számolni).

Viszont vannak egyezések:

- Mindkettőnek meghatározható az elemszáma a len függvénnyel, pl. `len(nevek)`
- Mindkettő esetén vizsgálható az `in` operátorral, hogy benne van-e a listában vagy halmazban, pl. `if név in nevek: ...`
- Mindkettő elemein végig lehet lépkedni a `for` utasítással.

Rövidítések, trükkök:

- **lista és halmaz is létrehozható egy sorban** a `for` utasítással, akár `if`-fel kombinálva, pl. páros számok 0-tól 100-ig

```
páros_számok = [i for i in range(101) if i%2==0]
```

- vagy a köv. listából kigyűjtjük az embereket halmazba, hogy mindegyik név csak egyszer szerepeljen, meg a 30 alattiakat is, hogy lásd az `if`-fet:

```
lista = [['Jakab',31],['Petra',29],['Józsi',40]]

nevek = {adat[0] for adat in lista}

harminc_alatt = {adat[0] for adat in lista if adat[1]<30}
```

- ha már itt tartunk, **ki is lehet íratni a neveket vesszővel elválasztva, ha a `join()` függvényt használjuk** (az utolsó után nem ír vesszőt!!!):

```
print(', '.join({adat[0] for adat in lista}))
```

- ha az kell, hogy **hányféle** név van, használjuk a **halmaz méretét adó** `len` függvényt:

```
print('emberek száma:',len({adat[0] for adat in lista}))
```

- ha az kell, hogy Petra **hányszor fordul elő** a nevek között, megcsináljuk a nevek listáját és **megszámoljuk** benne Petra előfordulásait: `lista.count(elem)`

```
print('Petra ennyiszer fordul elő:',[adat[0] for adat in lista].count('Petra'))
```

- **átlag**életkor számításához az életkorok összegének és darabszámának hányadosát kell venni:

```
atlag = sum([adat[1] for adat in lista])/len([adat[1] for adat in lista])
```

- **maximum**, **minimum** számolás:

```
max_kor = max({adat[1] for adat in lista})

min_kor = min({adat[1] for adat in lista})
```

- **Lista rendezése:**
  - **szimpla elemeket** tartalmazó lista esetén, pl.: `lista = [5, 6, 2, 3]` után a `lista.sort()` eredménye: `[2, 3, 5, 6]`
  - listákat tartalmazó (vagyis **több dimenziós**) lista esetén a `lambda` kulcsszót használjuk, pl. ha a köv. listát életkor szerint akarjuk rendezni, akkor így is csinálhatjuk:

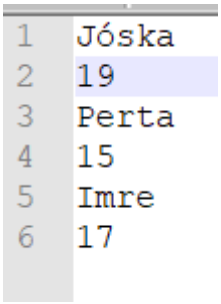
    ```
    lista = [['Jakab',31],['Petra',29],['Józsi',40]]

    lista = lista.sort(key = lambda e:e[1])
    ```

    vagy kor szerint csökkenőbe:
    ```
    lista.sort(key = lambda e:e[1], reverse=True)
    ```

- **Törlés a listából:**
  - egy adott elem törlése: `lista.remove(elem)`
  - egy adott indexű elem törlése: `del lista[index]`
  - listaelemek törlése egy adott indextől másik indexig: `del lista[index1:index2]`

Adatok beolvasása fájlból:




- ```
  with open('lista.txt') as forrasfajl:
      sorok = forrasfajl.read().splitlines()
  ```

- Ha a fájlban egymás alatti sorokban vannak az összetartozó adatok, pl. úgy néz ki a fájl, mint a képen:

  ```
  adatok = [sorok[i:i+2] for i in range(0,len(sorok),2)]
  ```

  (Nem tévedés, kettőt fog beolvasni, mert sorok[i:i+2] esetén az i max. értéke i+1 lesz!)

- Vagy ha tabulátorral elválasztott adatok vannak a sorokban:

  ```
  adatok = [sorok[i].split('\t') for i in range(len(sorok))]
  ```

  vagy

  ```
  adatok = [sor.split('\t') for sor in sorok]
  ```
  ```

Output, input

- Az `input()` függvény szöveget ad vissza, amit kapásból egy listába darabolhatunk:

  ```
  datum = input('Adjon meg egy dátumot (éééé.hh.nn)! Dátum= ').split('.')
  ```

- De lehet egyből egész számmá is átalakítani a bekérést:

  ```
  darab = int(input('Darabszám: '))
  ```

- Az értéktő függő kiíratás rövidíthető, ha **a print függvénybe if-fet** írunk:

  ```
  print('fiatalok' if atlag_életkor<45 else 'nem annyira fiatalok')
  ```

  És ez kombinálható mindenféle számítással is.

- Ha fájlba írunk, akkor a fájl megnyitása után a sorokat a '\n'.join() függvénnyel is kiírathatjuk. Itt egy példa, ami össze-vissza kombinál mindent, de működik:

  ```
  fajl = open('eredmény.txt','w')
  print('\n'.join([adat[1]+'. állomás: '+adat[2]+':'+adat[3] for adat in
  adatok if adat[0]==vonat and adat[4]=='E']),file=fajl)
  fajl.close()
  ```

  A `close()` mindenképpen kell, különben a fájl nyitott marad, megsérülhet, és nem is történik meg a fájlba írás.

Számítások:

- **Százalék** kiírása 12,25% formátumban (tizedes vesszővel, nem tizedes ponttal, ahogy alapból a Python kiírná):

  ```
  latta = 1225.36789 # vagyis a százalékérték 100-szorosa
  print('Látta: {},{}%.'.format(latta//100,latta%100))
  ```

- **Idő számolása**: ha össze kell hasonlítani időpontokat, mindent átváltunk másodpercbe. Ekkor

  ```
  idő = nap*24*60+óra*60+perc
  ```

  óra=idő//3600, perc=idő%3600//60 és másodperc=idő%60

- **Ha dátumokat kell összehasonlítani**, és nem számít a pontosság, csak hogy melyik volt előbb vagy később. átváltjuk őket napba (1 év: 365 nap és 1 hónap: 31 nap). Pl.

  ```
  datum = '2017.10.18'.split('.') # eredménye: ['2017','10','18']
  datum_napban = int(datum[0])*365+int(datum[1])*31+int(datum[2])
  ```