19.) SZÓTÁR ADATTIPUS

A **szótár adattípus** egy a programozásban nagyon gyakran használt típus. Nézzünk egy példát, melyben autók adatait tároljuk (rendszám, márka, típus, gyártási év, megtett kilométer, sérülésmentes-e).

Eddig ezt listában tettük meg, hiszen listában különböző típusú adatokat is tárolhatunk:

auto=['AA-BB-123','Audi','A4','2022','35000','True']

Ha a gyártási évére vagyunk kíváncsiak, akkor az **indexre hivatkozva** tudjuk megadni.

print(auto[3])

Ezzel van egy kis nehézség mert, ha nagyon sok adatot tárolunk, és azzal kell dolgoznunk, bonyolulttá válik a munka. Vagy ha valaki más olvassa a kódunkat, akkor nehezen igazodik ki a sok index között.

Ezért, sokkal **praktikusabb** lenne, ha **a különböző adatokra nem indexekkel hivatkoznánk, hanem valamiféle címkékkel, vagy megnevezésekkel azonosítanánk**. Erre van lehetőségünk a szótár használatánál.

```
auto={
        'rendszam' : 'AA-BB-123',       # a sztring típusokat aposztrófok közé tesszük
        'marka' : 'Audi',
        'tipus' : 'A4',
        'gyart_ev' : 2022,              # a szám típusnál nem kell aposztróf
        'km_ora' : 35000,
        'ser_e' : True,                 # a logikai típusnál sem kell aposztróf
}
```

A szótár nevének megadása után nyitó „kapcsos” zárójelet ({)kezdünk és a minta alapján aposztrófok között elkezdjük megadni a címkék nevét – ezt nevezzük **kulcs**nak(key) -, majd kettőspont után **érték**et (value) adunk. A sorok végén ne felejtsünk el vesszöket rakni! A szótár megadásának végén zárjuk a kapcsos zárójelünket (})

Adatokra hivatkozni úgy tudunk, hogy a címkék nevét adjuk meg:

print(auto['gyart_ev'])

Így egyszerűbb kódot írni, illetve kódot olvasni.

Mikor érdemes a szótár adattípust alkalmazni?

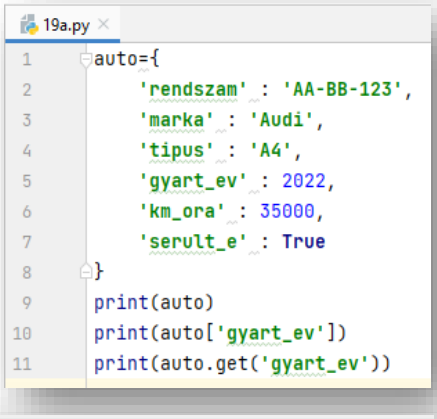
Ha homogén adatokat kell tárolni, mondjuk műszerrel mért adatokat, melyeket különböző időpontokban tettük meg, akkor elég a két dimenziós lista. (Például: hőmérséklet adatok reggel, délben, este)

De ha több, különböző típusú adatokat szeretnénk tárolni, akkor érdemes szótár típust használni.

(19a.py)

Készítsünk programot, melyben az előzőekben látható példát valósítjuk meg.

- Egyszerűen létrehozunk egy „auto” nevű szótár típust a megadott adatokkal. (1-8. sor)
- Aztán kiíratjuk a képernyőre a teljes tartalmat kulcsokkal és értékekkel. (9. sor)
- A minta szerint a gyártási év értékének kiíratására két módszert is megnézünk.



(19b.py)

Ebben a példában hozzunk létre egy „szemely” nevű szótárt, melyben egy főiskolai hallgató adatait tároljuk. (1-7. sor)

- Nézzünk pár hasznos utasítást, majd mindig írassuk ki a képernyőre, hogy lássuk a változást!
- Adjunk hozzá egy újabb mezőt „gyakorlat” néven, melynek logikai típusa legyen!
- Aztán változtassuk meg az egyik tárolt értéket! A hallgató korát írjuk át 19-re!
- Végül töröljünk egy felesleges mezőt a del paranccsal a minta szerint!

```
19b.py
1  szemely = {
2      'veznev' : 'Horváth',
3      'kernev' : 'Zoltán',
4      'kor' : 25,
5      'diakigazolvany': True,
6      'vizsgajegy' : [4,5,3,4,4,5],
7  }
8  print(szemely)
9  szemely['gyakorlat'] = False
10 print(szemely)
11 szemely['kor'] = 19
12 print(szemely)
13 del szemely['diakigazolvany']
14 print(szemely)
```

```
Run: 19b
E:\00_MM\12_Python_programozas\programok\venv\Scripts\python.exe E:/00_MM/12_Python_programozas/programok/19b.py
{'veznev': 'Horváth', 'kernev': 'Zoltán', 'kor': 25, 'diakigazolvany': True, 'vizsgajegy': [4, 5, 3, 4, 4, 5]}
{'veznev': 'Horváth', 'kernev': 'Zoltán', 'kor': 25, 'diakigazolvany': True, 'vizsgajegy': [4, 5, 3, 4, 4, 5], 'gyakorlat': False}
{'veznev': 'Horváth', 'kernev': 'Zoltán', 'kor': 19, 'diakigazolvany': True, 'vizsgajegy': [4, 5, 3, 4, 4, 5], 'gyakorlat': False}
{'veznev': 'Horváth', 'kernev': 'Zoltán', 'kor': 19, 'vizsgajegy': [4, 5, 3, 4, 4, 5], 'gyakorlat': False}
```

(19c.py)

Szótár bejárása (1.):

- Hozzunk létre egy új programot, melybe másoljuk át a „szemely” szótár deklarálását! (1-7. sor)
- Ebben a rövid programban a bejárás egyik módját nézzük meg for ciklussal. (8. sor)
- Tehát kiíratjuk a kulcsot és a hozzá tartozó értéket. A kettő közé beszúrunk egy kettőspontot! (9. sor)

```
19c.py
1  szemely = {
2      'veznev' : 'Horváth',
3      'kernev' : 'Zoltán',
4      'kor' : 25,
5      'diakigazolvany': True,
6      'vizsgajegy' : [4,5,3,4,4,5],
7  }
8  for i in szemely:
9      print(i,":", szemely[i])
```

```
Run: 19c
E:\00_MM\12_Python_programozas\p
veznev : Horváth
kernev : Zoltán
kor : 25
diakigazolvany : True
vizsgajegy : [4, 5, 3, 4, 4, 5]
```

(19d.py)

Szótár bejárása (2.):

A szótár bejárásának másik módja a **values()** használata.
Tehát ha a szótárra közvetlenül meghívjuk ezt a metódust (8. sor), akkor az adatokat egy listában tárolva jeleníti meg.
A 9-10. sorban for ciklussal soronként meghívjuk az értékeket és kiíratjuk a képernyőre.

```
19d.py
1  szemely = {
2      'veznev' : 'Horváth',
3      'kernev' : 'Zoltán',
4      'kor' : 25,
5      'diakigazolvany': True,
6      'vizsgajegy' : [4,5,3,4,4,5],
7  }
8  print(szemely.values())
9  for ertek in szemely.values():
10     print(ertek)
```

```
Run: 19d
E:\00_MM\12_Python_programozas\programok\venv\Scripts\python.exe
dict_values(['Horváth', 'Zoltán', 25, True, [4, 5, 3, 4, 4, 5]])
Horváth
Zoltán
25
True
[4, 5, 3, 4, 4, 5]
```

A legtöbb esetben nem csak egy ember adataival kell dolgoznunk, hanem – maradva az előző példánál – egyszerre több egyetemi hallgató adataival kell feladatokat elvégeznünk. Tehát képzeljük el, hogy egy listában tároljuk a különböző személyek adatait.

hallgatok=[{személy_0},{személy_1},{személy_2},{személy_3},…,{személy_n}]

Tehát ebben a listában ha egy diák adataira akarunk hivatkozni, akkor indexére hivatkozunk.

print(hallgatok[2])

Ha egy adott diák konkrét adatára, akkor az index után megadjuk a kulcsot is, amelyre kíváncsiak vagyunk.

print(hallgatok[2]['kor'])

(19e.py)

A szótár adattípus használatára nézzünk egy komolyabb, bonyolultabb példát!
Készítsünk programot, melyben egy txt fájlból beolvasunk adatokat, amelyeket szótár adattípusba helyezzük el. Aztán feladatokat végzünk a szótárakba helyezett adatokkal.
A példában egyetemi hallgatók adataival fogunk dolgozni.

- Először **hozzunk létre egy üres „adatok" nevű listát**, melyben majd tárolni fogjuk soronként a kiolvasott adatokat. (1. sor)
- Aztán **megnyitjuk a nyersanyag txt –t olvasásra** (2. sor), majd soronként hozzáadjuk az adatok listához (3-4. sor). Az adatok listába az **append() felhasználásával hozzáadjuk a beolvasott sorokat, felesleges soremelés karakterek nélkül (strip()).**
- A program írása közben mindig ellenőrizzük print() utasítással, hogy sikerült-e az előző művelet. Tehát létrejött-e az „adatok" feltöltött lista? Aztán a későbbiekben kettőskereszt (#) elé írásával, kommentbe helyezhetjük.
- A következő sorokban **elkészítjük a szótár típust**. (6-19. sor) Fel **kell építenünk az adatszerkezetet, és fel is kell töltenünk adatokkal.**
- A 6. sorban **létrehozzuk a „diak" nevű üres szótárt**. Ebben tároljuk majd az egyes diákok adatait.
- A 7. sorban pedig **egy „diakok" üres listát hozzuk létre**, melybe majd az összes diák szótár adatait rakjuk bele.
- For ciklussal be kell járnunk az adatok nevű listánkat, hiszen ebbe olvastuk be a nyersanyagot a txt-ből.
- Ha megnézzük a listánk elemeit, azok gyakorlatilag sztringek, szóközökkel elválasztva.

```
['Kiss János 20 történelem 0', 'Nagy Béla 19 matematika 1', 'Horváth Éva 21 biológia 1', 'Kovács
```

- Tehát egy-egy ilyen **sztringet fel kell darabolnunk kisebb részekre, a space-ek mentén**, ahol meg tudjuk adni a típusokat is.
- A feldaraboláshoz **létrehozunk egy listát „diak_adatok" néven (9. sor), melyben a split() metódus segítségével feldaraboljuk**. (Ha nem adunk meg argomentumot a zárójelek között, akkor az alapértelmezett space-ek mentén darabolja fel a sztringünket.)
- print(diak_adatok) paranccsal ideiglenesen tesztelhetjük is, hogy jól dolgoztunk-e. Aztán ezt a sor töröljük.
- Aztán kezdődhet a szótár illetve a lista felöltése. A minta szerint **megadjuk a kulcsokat és a hozzá tartozó „diak_adatok" megfelelő indexét**.
- A „kor" kulcsnál az étréket **int()** paranccsal számmá alakítjuk.
- A kollégiumhoz tartozó 1 illetve 0 számoknál „**if" feltétellel** True vagy False értéket állítunk be.
- A 18. sorban a **diakok.append(diak)** paranccsal becsatoljuk a „diakok" listához az aktuális sort.
- Végül **ki kell ürítenünk a „diak" szótár aktuális adatait** a 19.sorban.
- Aztán nézzük meg, hogy mit tartalmaz a „diakok" nevű lista. (20. sor) A listán belül vannak a szótárak sorban egymás után. És az egy-egy szótárban a gyerek adatait egymás után látjuk.

```
[{'vezeteknev': 'Kiss', 'keresztnev': 'János', 'kor': 20, 'szak': 'történelem', 'kollegista': False}, {'vezet
```

- Tehát sikerült létrehozni az adatszerkezetet és sikerült feltölteni azt.