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.

19e.py

```
1  adatok = []
2  with open('./nyersanyag/19e_szemelyi_adatok.txt','r',encoding='utf-8') as fajl:
3      for sor in fajl:
4          adatok.append(sor.strip())
5  #print(adatok)
6  diak={}
7  diakok=[]
8  for elem in adatok:
9      diak_adatok = elem.split()
10     diak['vezeteknev'] = diak_adatok[0]
11     diak['keresztnev'] = diak_adatok[1]
12     diak['kor'] = int(diak_adatok[2])
13     diak['szak'] = diak_adatok[3]
14     if diak_adatok[4]=='1':
15         diak['kollegista']=True
16     else:
17         diak['kollegista'] = False
18     diakok.append(diak)
19     diak={}
20 #print(diakok)
21 # a matematika szakos hallagtók neveinek kiíratása
22 print("\n----- Matematika szakos hallagatók listája: -----")
23 for diak in diakok:
24     if diak['szak']== 'matematika':
25         print(diak['vezeteknev'] + ' ' + diak['keresztnev'])
26 # tegyük listába történelem szakos hallgatók adatait
27 print("\n----- Történelem szakos hallagatók adatai: -----")
28 tortenelem_szak=[diak for diak in diakok if diak['szak'] == 'történelem']
29 print(tortenelem_szak)
30 # a diákok átlagéletkora
31 print("\n----- A diákok átlagéletkora: -----")
32 osszeg=0
33 for diak in diakok:
34     osszeg += diak['kor']
35 atlag=osszeg / len(diakok)
36 print("A hallagatók átlagéletkora: %.2f év." % (atlag))
37 # a legidősebb diák nevének és korának kiíratása
38 print("\n----- A legidősebb diák: -----")
39 max_index=0
40 max_kor=diakok[0]['kor']
41 for index, diak in enumerate(diakok):
42     if diak['kor'] > max_kor:
43         max_kor = diak['kor']
44         max_index = index
45 print("A legidősebb hallgató kora %d év" % (max_kor))
46 print("A legidősebb hallgató neve: ",diakok[max_index]['vezeteknev'],diakok[max_index]['keresztnev'])
```

- Ha beolvastuk az adatokat, feltöltöttük a listát a megfelelő típusú adatokkal, akkor végezzünk el pár alapvető feladatot!
- A 21-25. sortól **kiíratjuk a matematika szakos hallgatók neveit**. Megjegyzésbe írjuk le, hogy mit fogunk csinálni a faladatrészben (21. sor), majd a minta szerint egy print() utasítással elválasztjuk a feladatokat egymástól (22. sor). (A többi feladatnál is ezt használjuk.)
- Ha a diakok listában for utasítással végig megyünk (23. sor) és a szakok kulcsnál megegyeznek a ’matematika’ szóval (24. sor), ott írja ki a vezetékneveket és a keresztneveket, szóközzel elválasztva (összefűzve)(25. sor).

- A következő feladatrészben a **történelem szakos hallgatók összes adatát gyűjtjük ki**, és tesszük egy külön listába. (26-29. sor)
- Ezt a műveletet **leképezéssel készítjük** el. (28. sor)

```
28    tortenelem_szak=[diak for diak in diakok if diak['szak'] == 'történelem']
```

- Hozzunk létre egy **új listát „tortenelem_szak" néven**! A lista bejárását írjuk be először (for diak in diakok), majd megadjuk a feltételt, ami alapján szűrjük a diak szótárból a 'szak' kulcsnál megtalálható 'történelem' egyezést (diak['szak'] == 'történekelm'). A műveletnél változatlanul hagyjuk az értéket.
- A feladatrész végén **kiíratjuk** a „történekem_szak" lista tartalmát!
- A következő feladatrészben írassuk ki a **hallgatók átlagéletkorát**! (30-36. sor)
- Ez egy nagyon egyszerű feladat, hiszen már nagyon sokszor számoltunk átlagot. Létrehozunk egy „osszeg" nevű változót, melyben összeadjuk a „diakok" **lista bejárásával** a „diak" szótár 'kor' kulcsához tartozó értékeket. (33-34. sor)
- Majd egy „atlag" nevű változóba beletesszük az összeg és a „diakok" adatit tartalmazó lista hosszának hányadosát. Ahol a hányadost a len() függvénnyel kapunk meg. (35. sor)
- Végül formázottan kiírjuk az eredményt a minta szerint, ahol float adattípust használunk két tizedessel. (36. sor)
- Az utolsó részfeladatban **írassuk ki a legidősebb hallgató nevét és életkorát**! (37-45. sor)
- A részfeladat elején két változót adunk meg, amire szükségünk van. Az egyik a legnagyobb szám indexének eltárolásához kell majd, ezért a neve „max_index" lesz és az elején lenullázzuk. (39. sor)
- A másik változó a legnagyobb kor tárolására fog szolgálni. A kezdő értéknek pedig a „diakok" listánk nulladik elemének 'kor' kulcsához tartozó értéket adjuk meg. (40. sor)
- A következő sorban egy új beépített függvényt fogunk használni. A függvény neve: enumerate(). Az enumerate függvény segítségével számlálót használhatunk, amikor objektumokkal dolgozunk. Tegyük fel, hogy végig akarunk menni minden elemen a listánkban egy for ciklus segítségével. Emellett minden elem indexét is szeretnénk megtudni az iteráció során.
  Ezt megtehetjük például a range használatával. Ekkor le kell írnunk, hogy for x in range, majd használnunk kell a len-t, ami megmutatja a lista hosszát. A kettőspont után írhatunk egy print utasítást, ami az x-et és az x indexét is kinyomtatja. Eddig mindig ezt használtuk.
  Az enumerate-et for ciklussal együtt is lehet használni. Az enumerate függvény egy számlálót ad a listánkhoz, és a kimenet egy tuple lesz indexszámokkal a megfelelő elemek mellett.

```
41    for index, diak in enumerate(diakok):
```

- A következőkben pedig, ahogy végig megyünk, megvizsgáljuk, hogy az aktuálisan tárolt legnagyobb érték (max_kor) nagyobb-e mint a szótár aktuális 'kor' kulcsához tartozó érték. (42. sor)
- Mert ha nagyobb akkor beletesszük az új értéket és ezzel megyünk tovább. (43. sor)
- És ha a vizsgálat két sorral előbb igazzá vált szükség lesz a legnagyobb kor indexére, hogy a nevet is ki tudjuk íratni.
- Végül a minta szerint formázottan kiíratjuk a legidősebb hallgató korát és nevét.

```
Run:  19e
E:\00_MM\12_Python_programozas\programok\venv\Scripts\python.exe E:/00_MM/12_Python_prograi

----- Matematika szakos hallagatók listája: -----
Nagy Béla
Tóth Imre
Fekete Ivett

----- Történelem szakos hallagatók adatai: -----
[{'vezeteknev': 'Kiss', 'keresztnev': 'János', 'kor': 20, 'szak': 'történelem', 'kollegist

----- A diákok átlagéletkora: -----
A hallagatók átlagéletkora: 20.40 év.

----- A legidősebb diák: -----
A legidősebb hallgató kora 23 év
A legidősebb hallgató neve:  Farkas Zsanett
```