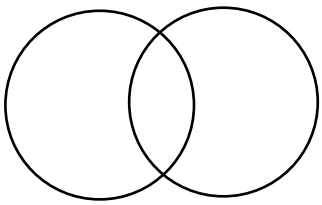
20.) HALMAZ TÍPUS (SET)

Az eddig megtanult, alkalmazott adattípusok között voltak az egyszerű típusok:

- int (egész számok)
- float (tizedes törtek)
- str (sztringek)
- bool (logikai típus: True / False)

Aztán tanultuk az összetett adattípusokat, melyekben nem csak egy értéket, hanem egymással logikailag összetartozó adatokat tudunk tárolni:



- lista (list)
- szótár (dict)

Ebben a leckében pedig egy új adattípusról lesz szó:

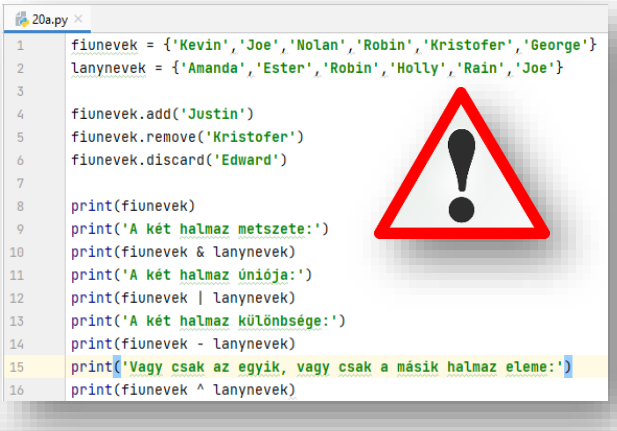
- **halmaz (set)**

Nézzük meg a halamazok jellemzőit:

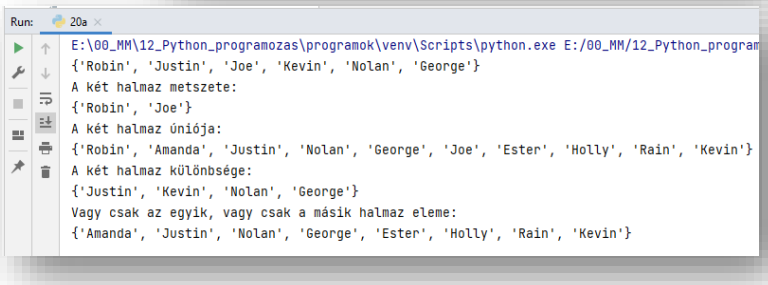
- a halmazokban **egy elem csak egyszer fordulhat elő** (a listákban megengedett volt az ismétlődés, itt nem)
- a **listák rendezetlen tárolók** (gondoljunk egy zsákra, belepakolunk dolgokat, majd tetszőleges sorrendben kivesszük belőle)
- **többféle adattípust tárolhatunk** a halmazokban (ebben hasonlít a listákra)
- az **elemeket nem lehet megváltoztatni** (a listákban például megduplázhattuk a tagok értékét, itt a halmazokban ezt nem lehet megtenni)
- a halmazokkal viszont a matematika órán megtanult műveletek elvégezhetőek (**metszet, unió, különbség**)
- sok hétköznapi probléma megoldásnál használhatjuk ezt a típust

Nézzünk példát a halmaz típus használatára:

(20a.py)



```
20a.py
1  fiunevek = {'Kevin','Joe','Nolan','Robin','Kristofer','George'}
2  lanynevek = {'Amanda','Ester','Robin','Holly','Rain','Joe'}
3
4  fiunevek.add('Justin')
5  fiunevek.remove('Kristofer')
6  fiunevek.discard('Edward')
7
8  print(fiunevek)
9  print('A két halmaz metszete:')
10 print(fiunevek & lanynevek)
11 print('A két halmaz uniója:')
12 print(fiunevek | lanynevek)
13 print('A két halmaz különbsége:')
14 print(fiunevek - lanynevek)
15 print('Vagy csak az egyik, vagy csak a másik halmaz eleme:')
16 print(fiunevek ^ lanynevek)
```



```
Run: 20a
E:\00_MM\12_Python_programozas\programok\venv\Scripts\python.exe E:/00_MM/12_Python_progra
{'Robin', 'Justin', 'Joe', 'Kevin', 'Nolan', 'George'}
A két halmaz metszete:
{'Robin', 'Joe'}
A két halmaz uniója:
{'Robin', 'Amanda', 'Justin', 'Nolan', 'George', 'Joe', 'Ester', 'Holly', 'Rain', 'Kevin'}
A két halmaz különbsége:
{'Justin', 'Kevin', 'Nolan', 'George'}
Vagy csak az egyik, vagy csak a másik halmaz eleme:
{'Amanda', 'Justin', 'Nolan', 'George', 'Ester', 'Holly', 'Rain', 'Kevin'}
```

- A halmazok létrehozása, a név megadása után egyenlőség jel, majd kapcsoszárójelek között felsoroljuk a halmaz tagjait!
- A programban fiúneveket és lányneveket adunk meg.
- A halmazok bővítésére az „add()” metódust használjuk. egyszerűen a halmaz neve után ponttal beírjuk a parancsot, majd sima zárójelek között megadjuk azt a nevet, amivel bővíteni szeretnénk.
- Az eltávolításhoz használjuk a „remove()” metódust.
- Ezt csak akkor alkalmazhatjuk, ha biztos hogy a név szerepel a halmazban. Ha nem létezik, akkor hibát jelez.
- Van egy olyan metódus, amellyel ez a hibaüzenet elkerülhető, ha nem tagja a halmaznak a megadott érték. Ez pedig a „discard()” metódus.
- Aztán nézzük meg a minta szerint a két halmaz metszetét (&), unióját (|), különbségét (-), és azokat a taggokat, amelyek vagy csak az egyik, vagy csak a másik halmazban szerepelnek (^)!

(20b.py)

- Az új programban hozzunk létre egy „gyumolcs” nevű halmazt, melyben soroljunk fel gyümölcs neveket!
- Majd akarjunk létrehozni egy „zoldsegek” nevű üres halamazt! Itt egy kis problémába ütközünk. Ha az egyszerű logika szerint haladunk akkor két kapcsos zárójelet írunk egymás után. Írassuk ki „type” függvénnyel, hogy milyen típust hoz létre a python! Látjuk, hogy szótár (’dict’) lesz. Tehát ez így nem jó.
- A megoldás az, hogy meghívjuk az úgynevezett konstruktorát „set()

```
20b.py
gyumolcs = {'alma','banán','citrom','körte','szilva''dinnye'}
zoldseg = {}

print(type(zoldseg))
```

```
Run: 20b
E:\00_MM\12_Python_pr
<class 'dict'>
```

```
20b.py
gyumolcs = {'alma','banán','citrom','körte','szilva''dinnye'}
zoldseg = set()

print(type(zoldseg))
```

```
Run: 20b
E:\00_MM\12_Python_prog
<class 'set'>
```

- De adhatunk meg a jobb oldalon látható módon halmazt, és elemeit!
- Egyébként látszik a kiíratásnál, a sorrend az nem ugyanaz, mint ahogyan én megadtam eredetileg.

```
20b.py
gyumolcs = {'alma','banán','citrom','körte','szilva''dinnye'}
zoldseg = set(['paradicsom','paprika','zoldborsó','zöldbab','patiszon'])

print(type(zoldseg))
print(zoldseg)
```

```
Run: 20b
E:\00_MM\12_Python_programozas\programok\venv\Scripts\python.ex
<class 'set'>
{'zoldborsó', 'paradicsom', 'zöldbab', 'paprika', 'patiszon'}
```

- Ha létrehozunk egy olyan halmazt, melyben különböző típusú elemek vannak, azzal nincsen gond.
- Viszont, ha elemismétlés van, azt csak egyszer veszi figyelembe! Látjuk a kiíratásnál.

```
20b.py
gyumolcs = {'alma','banán','citrom','körte','szilva''dinnye'}
zoldseg = set(['paradicsom','paprika','zoldborsó','zöldbab','patiszon'])
egyeb = {'beton','tolgyfa',True,True}

print(type(egyeb))
print(egyeb)
```

```
Run: 20b
E:\00_MM\12_Python_programozas\programok\ve
<class 'set'>
{True, 'tolgyfa', 'beton'}
```

(20c.py)

Nézzünk egy olyan példát, ahol bútorok fajtáját soroljuk fel és ebben a halmazban vannak ismétlődések.
Sokszor fordul elő, hogy ezeket az ismétlődéseket kell megszüntetni ténylegesen.

```
20c.py
butorok = {'asztal','szék','polc','szekrény','fotel','asztal'}
fajta = set()
for butor in butorok:
    fajta.add(butor)
print(fajta)
```

```
Run: 20c
E:\00_MM\12_Python_programozas\programok\venv\Script
{'szék', 'szekrény', 'asztal', 'polc', 'fotel'}
```

- Tehát a program elején felsoroljuk egy „butorok” halmazban az elemeket.
- Majd létrehozunk egy üres „fajta” nevű halmazt, konstuktor segítségével.
- Aztán for ciklussal végig megyünk az elemeken (bejárjuk), és az add() metódussal beletesszük az üres halmazba az elemeket.
- Végül kiíratjuk az ismétlődésmentes halmazunkat.