Set

2025

1 Set adatszerkezet

A `set` (halmaz) olyan tárolószerkezet, amely nem engedi meg az értékek ismétlődését (egyedi elemek gyűjteménye). Az adatok rendezetlenül kerülnek eltárolásra (nem számít a sorrend), az elemek nem módosíthatók, azonban törölhetők és új elemekkel bővíthető. Két `set` között értelmezhetők a halmazműveletek: unió, metszet, különbség. Duplikátumok kiszűrésére, halmazműveletek és keresések gyors elvégzésére alkalmas.

1.0.1 Set létrehozása

A halmaz elemeit kapcsos `{}` zárójelek között kell felsorolni, vagy használhatjuk a `set()` konstruktort is. Üres halmazt csak a `set()` konstruktorral lehet létrehozni.

[ ]:
```
gyumolcsok = {"alma", "körte", "banán", "alma"}
print(f"Halmaz elemei: {gyumolcsok}")  # rendezetlen, nincs elemismétlés

halmaz = set(("alma", True, 42, 3.14))  # tuple miatt két zárójel kell
halmaz = set(["alma", True, 42, 3.14])  # listából
print(type(halmaz))
```

1.0.2 Elemek bővítése, eltávolítása

Az elemeket nem lehet `[]` operátorral kiválasztani, ezért azok nem is módosíthatók.
Az `add()` függvénnyel hozzá lehet adni új elemet, ha még nem szerepel benne.
Az `update()` függvény segítségével egy másik set vagy gyűjtemény elemei is hozzáadásra kerülnek.
Eltávolítani a `remove()` vagy a `discard()` függvényekkel lehet, a kettő között az a különbség, hogy nem létező elem törlése esetén a `remove()` hibát fog adni.

[ ]:
```
gyumolcsok = {"alma", "körte", "banán", "alma"}
# "citrom" hozzáadása
gyumolcsok.add("citrom")  # elem hozzáadása
# "banán" törlése
gyumolcsok.remove("banán")  # töröl, hibát ad, ha nincs ilyen elem
# "almafa" törlése
gyumolcsok.discard("almafa")  # töröl, nem ad hibát, ha nincs ilyen elem
print(f"Halmaz elemei: {gyumolcsok}")

deli_gyumolcsok = ["narancs", "kivi"]
# deli_gyumolcsok hozzáadása
```

```
gyumolcsok.update(deli_gyumolcsok)  # lista elemeinek hozzáadása
print(f"Halmaz elemei: {gyumolcsok}")
```

Létezik a `pop()` függvény, de ennek hatása kiszámíthatatlan, mert rendezetlen a gyűjtemény és nem lehet tudni melyik elem kerül kidobásra. A véletlenszerűen kivett elemet adja vissza. Ha a halmaz üres, akkor `KeyError` hibát dob.
A `clear()` függvénnyel kiüríthetjük az egész halmazt.
A `del` kulcsszóval az egész változó kerül eltávolításra.

[ ]:
```
gyumolcsok = {"alma", "körte", "banán"}

print(f"Eltávolított elem: {gyumolcsok.pop()}")
print(f"Halmaz megmaradt elemei: {gyumolcsok}")

gyumolcsok.clear()
print(f"Üres halmaz: {gyumolcsok}")

del gyumolcsok
# print(f"Halmaz elemei: {gyumolcsok}") # itt már nem létezik a gyumolcsok␣
 ↪változó, ezért ki sem lehet írni
```

1.0.3 Bejárás

Az elemek bejárására a `for` ciklus alkalmas.

[ ]:
```
gyumolcsok = {"alma", "körte", "banán"}

for gyumolcs in gyumolcsok:
    print(gyumolcs)
```

A ciklushoz az `enumerate()` függvény segítségével egy indexet is hozzárendelhetünk, amelynek kezdő értéke 0-ról indul, de második argumentumként megadható tetszőleges kezdőérték.

[ ]:
```
gyumolcsok = {"alma", "körte", "banán"}

for index, gyumolcs in enumerate(gyumolcsok, 1):
    print(f"{index}. {gyumolcs}")
```

1.0.4 Halmazműveletek

A `set`-ek támogatják az alapvető halmazműveleteket, például uniót, metszetet és különbséget.

Unió: *Az A és B halmazok uniójának nevezzük azt a halmazt, melynek minden eleme benne van valamelyik halmazban. Jelölés: A B* - `union()` függvény: egy `set` és egy vagy több iterálható gyűjtemény között értelmezett - `|` operátor: csak két `set` között értelmezett

[ ]:
```
set_A = {1, 2, 3, 4}
set_B = {3, 4, 5, 6}
```

```
# két halmaz uniója
unio = set_A.union(set_B)
unio = set_A | set_B
print(unio)
```

Metszet: *Az A és B halmazok metszetének nevezzük azt a halmazt, melynek minden eleme benne van mindkét halmazban. Jelölés: A B* - `intersection()` függvény: egy `set` és egy vagy több iterálható gyűjtemény között értelmezett - `&` operátor: csak két `set` között értelmezett

[ ]:
```
set_A = {1, 2, 3, 4}
set_B = {3, 4, 5, 6}

# két halmaz metszete
metszet = set_A.intersection(set_B)
metszet = set_A & set_B
print(metszet)
```

Különbség: *Az A és B halmazok különbségének nevezzük azt a halmazt, melynek minden eleme benne van A-ban, de nincs benne B-ben. Jelölés: A B* - `difference()` függvény: egy `set` és egy vagy több iterálható gyűjtemény között értelmezett - `-` operátor: csak két `set` között értelmezett

[ ]:
```
set_A = {1, 2, 3, 4}
set_B = {3, 4, 5, 6}

# két halmaz különbsége
kulonbseg = set_A.difference(set_B)
kulonbseg = set_A - set_B
print(kulonbseg)
```

Szimmetrikus különbség: *Az A és B halmazok szimmetrikus differenciájának nevezzük azt a halmazt, melynek minden eleme az A és a B halmazok közül pontosan az egyikben van benne. Jelölés: A B.* - `symmetric_difference()` függvény: egy `set` és egy vagy több iterálható gyűjtemény között értelmezett - `^` operátor: csak két `set` között értelmezett

[ ]:
```
set_A = {1, 2, 3, 4}
set_B = {3, 4, 5, 6}

# két halmaz szimmetrikus különbsége
szimmetrikus_kulonbseg = set_A.symmetric_difference(set_B)
szimmetrikus_kulonbseg = set_A ^ set_B
print(szimmetrikus_kulonbseg)
```

1.0.5 Tagság ellenőrzése

Az `in` operátorral tudjuk megvizsgálni, hogy egy elem része-e a halmaznak vagy sem.

[ ]:
```
set_A = {1, 2, 3, 4}
print(2 in set_A)  # True
print(5 not in set_A)  # True
```

1.0.6 Tartalmazás ellenőrzése

Részhalmaz ellenőrzése: *Az A halmaz a B halmaznak részhalmaza, ha minden A-beli elem egyben B-nek is eleme. Jelölés: A B.* - `issubset()` függvény ellenőrzi, hogy részhalmaza-e a megadott halmaznak - `<=` operátor ellenőrzi, hogy részhalmaza-e a megadott halmaznak

Tartalmazás ellenőrzése: - `issuperset()` függvény ellenőrzi, hogy tartalmazza-e a megadott halmaz összes értékét - `>=` operátor ellenőrzi, hogy tartalmazza-e a megadott halmaz összes értékét

Különálló halmazok-e: *Az A halmaz és a B halmaz különálló (független) halmazok, ha nincs egyetlen közös elemük sem.* - `isdisjoint()` függvény ellenőrzi, hogy különálló halmazok-e

```
[ ]: set_A = {1, 2, 3, 4}
set_B = {1, 2, 3, 4, 5, 6}
set_C = {5, 6}

# A részhalmaza B-nek?
print(set_A.issubset(set_B))
print(set_A <= set_B)

# B tartalmazza-e A-t?
print(set_B.issuperset(set_A))
print(set_B >= set_A)

# B különálló halmaz-e A-tól?
print(set_B.isdisjoint(set_A))
# C különálló halmaz-e A-tól?
print(set_C.isdisjoint(set_A))
```

1.1 Felhasználása (duplikátumok kiszűrése)

Olyan gyűjteményeknél, ahol a sorrend nem számít, de mindenből csak egyszeres előfordulás a megengedett. A `set` ideális arra, hogy egy listából vagy más gyűjteményből eltávolítsuk az ismétlődő elemeket. Csak átalakítjuk a listát egy `set`-té, majd vissza egy listává.

```
[ ]: ismetlodo_lista = [5, 1, 2, 3, 1, 2, 4, 5]
# duplikátumok kiszűrése
egyedi_elemek = list(set(ismetlodo_lista))
print(egyedi_elemek)
```

2 Gyakorló feladatok

1. Feladat

   A budapesti metrók megállóinak neve szerepel egy-egy set-ben.
   Határozza meg, hogy melyik Metró-nak van a legtöbb megállója. Határozza meg, hogy az M2-ről hol lehet átszállni az M3-ra.

[ ]:
```
M1 = {
    "Vörösmarty tér",
    "Deák Ferenc tér",
    "Bajcsy-Zsilinszky út",
    "Opera",
    "Oktogon",
    "Vörösmarty utca",
    "Kodály körönd",
    "Bajza utca",
    "Hősök tere",
    "Széchenyi fürdő",
}

M2 = {
    "Örs vezér tere",
    "Pillangó utca",
    "Puskás Ferenc Stadion",
    "Keleti pályaudvar",
    "Blaha Lujza tér",
    "Astoria",
    "Deák Ferenc tér",
    "Kossuth tér",
    "Batthyány tér",
    "Széll Kálmán tér",
    "Déli pályaudvar",
}

M3 = {
    "Újpest-Központ",
    "Újpest-Városkapu",
    "Gyöngyösi utca",
    "Forgách utca",
    "Árpád híd",
    "Dózsa György út",
    "Lehel tér",
    "Nyugati pályaudvar",
    "Arany János utca",
    "Deák Ferenc tér",
    "Ferenciek tere",
    "Kálvin tér",
    "Ferenc körút",
    "Klinikák",
    "Nagyvárad tér",
    "Népliget",
    "Népliget",
    "Ecseri út",
    "Pöttyös utca",
```

```
    "Határ út",
    "Kőbánya-Kispest",
}

M4 = {
    "Kelenföld vasútállomás",
    "Bikás park",
    "Újbuda-központ",
    "Móricz Zsigmond körtér",
    "Szent Gellért tér",
    "Fővám tér",
    "Kálvin tér",
    "Rákóczi tér",
    "II. János Pál pápa tér",
    "Keleti pályaudvar",
}

megallok = [len(M1), len(M2), len(M3), len(M4)]
hely = megallok.index(max(megallok))
print(
    f"A legtöbb megállóval rendelkező metró az M{hely+1}, {megallok[hely]} db␣
 ↪megállóval"
)
print(f"Az M2-ről az M3-ra a", *(M2 & M3), "megállóban lehet átszállni.")
```

2. Feladat

   A metrómegállókat mentsük ki egy metrok.csv fájlba. A fájlt olvassuk vissza és határozzuk meg, hogy az M2-es és M4-es metrókon hol lehet átszállni.

[ ]:
```
M1 = [
    "Vörösmarty tér",
    "Deák Ferenc tér",
    "Bajcsy-Zsilinszky út",
    "Opera",
    "Oktogon",
    "Vörösmarty utca",
    "Kodály körönd",
    "Bajza utca",
    "Hősök tere",
    "Széchenyi fürdő",
]

M2 = [
    "Örs vezér tere",
    "Pillangó utca",
    "Puskás Ferenc Stadion",
    "Keleti pályaudvar",
```

```
    "Blaha Lujza tér",
    "Astoria",
    "Deák Ferenc tér",
    "Kossuth tér",
    "Batthyány tér",
    "Széll Kálmán tér",
    "Déli pályaudvar",
]

M3 = [
    "Újpest-Központ",
    "Újpest-Városkapu",
    "Gyöngyösi utca",
    "Forgách utca",
    "Árpád híd",
    "Dózsa György út",
    "Lehel tér",
    "Nyugati pályaudvar",
    "Arany János utca",
    "Deák Ferenc tér",
    "Ferenciek tere",
    "Kálvin tér",
    "Ferenc körút",
    "Klinikák",
    "Nagyvárad tér",
    "Népliget",
    "Népliget",
    "Ecseri út",
    "Pöttyös utca",
    "Határ út",
    "Kőbánya-Kispest",
]

M4 = [
    "Kelenföld vasútállomás",
    "Bikás park",
    "Újbuda-központ",
    "Móricz Zsigmond körtér",
    "Szent Gellért tér",
    "Fővám tér",
    "Kálvin tér",
    "Rákóczi tér",
    "II. János Pál pápa tér",
    "Keleti pályaudvar",
]
# mentés metrok.cs fájlba
with open("./file/metrok.csv", "w", encoding="utf-8") as f:
```

```
    for elem in M1:
        f.write(f"M1;{elem}\n")
    for elem in M2:
        f.write(f"M2;{elem}\n")
    for elem in M3:
        f.write(f"M3;{elem}\n")
    for elem in M4:
        f.write(f"M4;{elem}\n")
```

[ ]:
```
# fájl visszaolvasása soronként
M2 = set()
M4 = set()

with open("./file/metrok.csv","r",encoding="utf-8") as f:
    for sor in f:
        sor  = sor.strip().split(';')
        if sor[0] == "M2":
            M2.add(sor[1])
        if sor[0] == "M4":
            M4.add(sor[1])

print(M2&M4)
```

[ ]:
```
# fájl visszaolvasása csv reader-rel
import csv
M2 = set()
M4 = set()

with open("./file/metrok.csv","r",encoding="utf-8") as f:
    csvf = csv.reader(f, delimiter=";")

    for sor in csvf:
        if sor[0] == "M2":
            M2.add(sor[1])
        if sor[0] == "M4":
            M4.add(sor[1])

print(M2&M4)
```

3. Feladat

   Az automatak.csv fájlban 2 üdítő és 2 csokoládé automata termékei találhatók meg. Összesen hányféle termék közül tudunk választani ismétlődésmentesen.

[ ]:
```
# fájl beolvasása soronként
termekek = set()

with open("./file/automatak.csv", encoding="utf-8") as f:
```

```
    f.readline()
    # fejléc eldobása
    for sor in f:
        termek = sor.split(";")[1].strip()
        termekek.add(termek)

print(f"{len(termekek)} db termékből lehet választani")
print(termekek)
```

[ ]:
```
# fájl beolvasása csv reader-rel
import csv

termekek = set()

with open("./file/automatak.csv") as f:
    csvf = csv.reader(f, delimiter=";")

    next(csvf)

    for sor in csvf:
        termekek.add(sor[1])

print(f"{len(termekek)} db termékből lehet választani")
print(termekek)
```