Liste. Tupluri. Seturi. Dictionare.

LISTE

- Mutabile

```
l = [1,2,3,4]
print (f'l={l}, id={id(l)}')
l.append(5)
print (f'l={l}, id={id(l)}') # 1, 2, 3, 4, 5
```

- Ordonate

```
l = [1,2,3,4]
print(l) # 1, 2, 3, 4
```

Accesare

- **Indexare**

(pozitiva/negativa)

```
l = [1, 2, 3, 4, 5]
#    0  1  2  3  4
#   -5 -4 -3 -2 -1
print(l[2], l[-3]) # 3 3
```

- **Slicing**

(L[i : j : pas], i = inclusiv, j este exclusiv)

```
l = [1, 2, 3, 4, 5]
#    0  1  2  3  4
#   -5 -4 -3 -2 -1
print(l[1:3]) # [2, 3]
print(l[1:-3]) # [2]
print(l[-1:-4]) # []
print(l[-1:-4:-1]) # [5, 4, 3]
l[1:3] = []
print(l) # [1, 4, 5]
```

Creare liste

- lista constanta
- list comprehension

(secvente de initializare)

```
# secvență de inițializare
L = [x + 1 for x in range(10)]
print(L) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# secvență de inițializare cu placeholders (_)
L = [_ + 1 for _ in range(10)]
print(L) # [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# secvențe de inițializare condiționale
L = [x**2 for x in range(10) if x % 2 == 0]
print(L) # [0, 4, 16, 36, 64]
L = [x**2 if x % 2 == 0 else -x**2 for x in range(10)]
print(L) # [0, -1, 4, -9, 16, -25, 36, -49, 64, -81]
L1 = [1, 3, 5, 6, 8, 3, 13, 21]
L2 = [18, 3, 7, 5, 16]
L3 = [x for x in L1 if x in L2]
print(L3) # [3, 5, 3]
```

Metode de baza

- Append
- Remove
- Index
- Pop
- Insert
- Length

```
l = [1, 2, 3, 4, 5]
#    0  1  2  3  4
#   -5 -4 -3 -2 -1
l.append(6)
print(l) # [1, 2, 3, 4, 5, 6]
l.remove(2)
print(l) # [1, 3, 4, 5, 6]
print(l.index(3)) # 1
l.pop()
print(l) # [1, 3, 4, 5]
l.insert(2, 10)
print(l) # [1, 3, 10, 4, 5]
print(len(l)) # 5
```

Complexitate operatii

Operation	Average case	Worst case
Append	O(n)	O(n)
Pop last	O(1)	O(1)
Pop	O(n)	O(n)
Insert	O(n)	O(n)
Get/Set item	O(1)	O(1)
Get length	O(1)	O(1)
x in l	O(n)	O(n)
min(x), max(x)	O(n)	O(n)

Situatii utile

- Vrei sa faci un to do list simplu si tii task-urile intr-o lista
- Ai o baza de date cu studenti si notele lor la diferite materii, cand iei notele din baza de date, le tii intr-o lista

```
tasks = ["Cumpărături", "Exerciții", "Citit"]
tasks.append("Proiect Python")   # Adaugă un task nou
tasks.remove("Citit")            # Șterge un task


grades = [9.5, 8.0, 7.5, 10]
average_grade = sum(grades) / len(grades)   # Calculează media
```

TUPLURI

Destul de similare listelor, diferenta principala fiind imutabilitatea lor

- Imutabile

```
l = (1,2,3)
print(l) # (1,2,3)
l.append(5) # EROARE, nu poti schimba
```

- Ordonate

```
l = (1,2,3,4)
print(l) # 1, 2, 3, 4
```

Situatii utile

Returnarea mai multor variabile din functii

```
def get_student_info():
    name = "Alex"
    grade = 9.5
    return (name, grade)

student_name, student_grade = get_student_info()
```

Stocarea de date care nu se pot schimba

```
point = (10, 20)
x, y = point
```

SETURI

- Mutabile

```
s = {1,2,3,4}
print (s) # 1, 2, 3, 4
s.add(5)
print(s) # 1, 2, 3, 4, 5
```

- Neordonate

```
s = {1,2,3,4}
print(s) # ordinea nu e neaparat 1, 2, 3, 4
```

Accesare

- De obicei, set-ul se foloseste pentru a elimina duplicatele
- Acesta nu poate fi accesat precum o lista (l[2]), ci trebuie iterat cu un for

```
s = {1,2,3,4}
for x in s:
    print(x)
```

Metode de baza

- Add
- Remove
- Update
- Pop - random item
- Union
- Difference
- Intersection

```
s = {1,2,3}
s.add(4) # 1, 2, 3, 4
s.remove(2) # 1, 3, 4
l = [101, 102]
s.update(l)
s.pop() # nu putem stii exact ce elemente au
          ramas
s1 = {1, 3, 53, 54, 55}
s2 = s1.union(s) # 1, 3, 4, 53, 54, 55, 101, 102
                   mai putin elementul eliminat
                    la pop
s3 = s1.difference(s)
s4 = s1.intersection(s)
print(f's = {s}, s1={s1}, s2={s2}, s3={s3}, s4={s4}')
```

Complexitate operatii

Operation	Average case	Worst case
Add	O(1)	O(n)
x in l	O(1)	O(n)
Pop	O(1)	O(n)
Update	O(n)	O(n)
Union s|t	O(len(s) + len(t))	
Intersection s & t	O(min(len(s), len(t)))	O(len(s) * len(t))
Difference	O(len(s))	

Situatii utile

Seturile sunt foarte utile in multe situatii, spre exemplu:

- Eliminarea duplicatelor dintr-o listă
- Operații de intersecție și diferență între grupuri
- Verificarea eficienta daca un element este continut

```
l = [1,2,3,4,2,3,2,4]
s = set(l) # 1, 2, 3, 4

s1 = {1, 2, 3}
s2 = {3, 4, 5}
sIntersectie = s2.intersection(s1)
sDiferenta = s2.difference(s1)
```

```
#Se dau multe numere si apoi se
# da un numar sa fie verificat
# daca a fost dat deja
s = {...multe numere}
#le punem intr-un set
x = int(input()) # primim un numar
if x in s:
    print(“DA”)
```

DICTIONARE

- Cateodata avem nevoie de o mapare intre un cuvant cheie si un tip de date sau o colectie de date (e.g. student->notele sale) => dictionarele sunt modul perfect de a stoca datele

si acesta mapeaza

- Exemple de utilizare:

  Persoana -> ce masini detine

  Student -> Materii -> Note (acesta este un scenariu in care folosim 2 dictionare, unul pentru a tine studentul, mapand catre alt dictionar unde este tinuta materia si notele la materia respectiva)

Creare dictionar

- Prin valori constante

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
 "colors": ["red", "white", "blue"]
}
```

- dict comprehension (secvente de initializare)

```
keys = ['a','b','c','d','e']
values = [1,2,3,4,5]
myDict = { k:v for (k,v) in zip(keys, values)}
```

Accesare

- Accesare dupa cheie

```
d = {
 "brand": "Ford",
 "electric": False,
 "year": 1964,
}
d[”brand”] = “Audi”
print(d[”brand”])
```

- Get

```
d = {
 "brand": "Ford",
 "electric": False,
 "year": 1964,
}
print(d.get(“brand”))
```

Metode de baza

- Add
- Pop
- Get

```
d = {
 "brand": "Ford",
 "electric": False,
 "year": 1964,
}
masina = d[”brand”] # operatia de get
d.pop(“electric”) # am eliminat electric din d
print(d)
d[”color”] = “red” # am adaugat campul color
print(d)
```

Complexitate operatii

Operation	Average case	Worst case
Add	O(1)	O(n)
Pop	O(1)	O(n)
Get	O(1)	O(n)
Set	O(1)	O(n)
k in d	O(1)	O(n)

Situatii utile

- Stocarea datelor unui student
- Maparea prețurilor la produse într-un magazin

```
student = { "nume": "Alex", "varsta": 21, "universitate": “Unibuc” }
print(student["nume"])


prices = { "lapte": 5.5, "oua": 3.2, "paine": 2.0 }
print(prices["lapte"])
```