- TUPLURI în PYTHON
- SETURI în PYTHON

CE ESTE UN TUPLU?

Presupunem că dorim să prelucrăm date referitoare la mai multe persoane și să spunem că pentru fiecare cunoaștem *numele*, *prenumele*, *vârsta*, *înălțimea* și locul *nașterii*.

Putem crea 5 liste, una pentru fiecare informație, însă precum ați observat, dacă *alterăm* (adică modificăm cumva) una dintre liste, *indicii nu mai corespund*, iar pozițiile nu mai pot fi accesate corect. În acest caz putem folosi **tuplurile**!

Un **tuplu** este tot un fel de listă care are însă **elementele ordonate** și **nemodificabile**, iar acestea *se definesc între paranteze rotunde*:

EXPLICAȚIE

main.py Run Shell

```
1  student1 = ("Mihai", "Eminescu", 18, 1.82, "Botoșani")
2  student2 = ("Ion", "Creangă", 19, 1.64, "Iași")
3  student3 = ("Grigore", "Vieru", 20, 1.75, "Pererâta")
4  print(student1)
5  print(student2)
6  print(student3)
```

```
('Mihai', 'Eminescu', 18, 1.82, 'Botoșani')
('Ion', 'Creangă', 19, 1.64, 'Iași')
('Grigore', 'Vieru', 20, 1.75, 'Pererâta')
```

Mai sus am definit trei tupluri, în cazul nostru pentru **Mihai**, **Ion** și **Grigore**, ca exemplu. Fiecare element al tuplurilor reține o anumită informație pe care am introdus-o corespunzător, astfel încât să obținem o structură de date coerentă.

Deci tuplurile create rețin *vârsta* la indicele **2**, iar *locul nașterii* la **4**, etc. Astfel, avem mecanismul format pentru oricare alt student și vom crea un tuplu pentru fiecare.

IMPORTANT: *Nu putem adăuga, șterge sau modifica elementele definite într-un tuplu! Orice operație ne oferă o eroare la compilare.*

Cum putem soluționa această problemă?

OPERAȚII CU TUPLURI

Ca și în cazul listelor, putem **accesa elementele** unui tuplu folosind **indici și parantezele pătrate**, cum suntem obișnuiți. În exemplul de mai jos am folosit indicii direct ori am selectat doar o parte dintre ei, apoi indecșii negativi și inversarea tuplului.

Pentru a afla **lungimea** unui tuplu, folosim funcția deja cunoscută len().

main.py Run Shell

```
1  student1 = ("Mihai", "Eminescu", 18, 1.82, "Botoșani")
2  student2 = ("Ion", "Creangă", 19, 1.64, "Iași")
3  student3 = ("Grigore", "Vieru", 20, 1.75, "Pererâta")
4  print(student1[0:2])
5  print(student1[3])
6  print(student1[3:])
7  print(student1[-1], student1[-2])
8  print(student1[::-1])
9  print(len(student1))
10 print("Ion" in student2)
11 print("Grigore" in student3)
12 print("Vadim" not in student1)
```

```
('Mihai', 'Eminescu')
1.82
(1.82, 'Botoșani')
Botoșani 1.82
('Botoșani', 1.82, 18, 'Eminescu', 'Mihai')
5
True
True
True
```

OPERAȚII CU TUPLURI

Operatorul in și grupul not in

Deja cunoscuți de la liste, acești operatori testează dacă o valoare se regăsește sau nu în colecția de date respectivă. Rezultatul este unul boolean, deci adevărat (**True**) sau fals (**False**).

În exemplul anterior am verificat dacă anumite nume (șiruri de caractere) există sau nu în tuplul **student**.

Ștergerea unui tuplu

Ștergerea unui element din cadrul unui tuplu este interzisă. Totuși, putem *șterge complet tuplul* prin folosirea cuvântului cheie **del**. *Exemplu*: **del student2**. După această comandă, bineînțeles că în program nu mai puteți accesa variabila **student2**.

main.py

Run

```
1  student1 = ("Mihai", "Eminescu", 18, 1.82, "Botoșani")
2  student2 = ("Ion", "Creangă", 19, 1.64, "Iași")
3  student3 = ("Grigore", "Vieru", 20, 1.75, "Pererâta")
4  print(student1)
5  del student2
6  print(student3)
```

Shell

```
('Mihai', 'Eminescu', 18, 1.82, 'Botoșani')
('Grigore', 'Vieru', 20, 1.75, 'Pererâta')
>
```

OPERAȚII CU TUPLURI

Operatorii + și *

Similar aplicării pentru liste, operatorul "**+**" alătură două sau mai multe tupluri, iar operatorul "*****" are rolul de multiplicare a elementelor.

main.py

Run

Shell

```
1  t1 = (1,2,3)
2  t2 = (4,5)
3  print(t1+t2)
4  print((t1+t2)*2)
```

```
(1, 2, 3, 4, 5)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
>
```

Evident că multiplicarea trebuie făcută cu *un număr natural*, altfel vom obține eroare de interpretare.

METODE și FUNCȚII utile pentru TUPLURI

Există două metode care pot fi folosite pentru tipul de date **tuple** și anume **count()** și **index()**.

Metoda **count(dată)** întoarce numărul de apariții ale obiectului trimis ca parametru în cadrul tuplului.

Metoda **index(obiect)** întoarce prima poziție (indexul) a obiectului în cadrul tuplului.

Observație: Dacă valoarea nu este găsită, metodele întorc o eroare de interpretare și nu o anumită valoare! Așadar, atenție la crearea programelor deoarece se sistează executarea acestora la apariția unei astfel de erori.

main.py

Run

Shell

```
1  tuplu = (1,2,3,4,5,2,3)
2  print("3 apare de", tuplu.count(3), "ori.")
3  print("Indexul lui 5 este", tuplu.index(5))
4  print("Valoarea minimă este", min(tuplu))
5  print("Valoarea maximă este", max(tuplu))
6  print("Valorile sortate ca listă:", sorted(tuplu))
7  print("Suma elementelor este:", sum(tuplu))
```

```
3 apare de 2 ori.
Indexul lui 5 este 4
Valoarea minimă este 1
Valoarea maximă este 5
Valorile sortate ca listă: [1, 2, 2, 3, 3, 4, 5]
Suma elementelor este: 20
```

MODIFICAREA UNUI TUPLU

Să presupunem că am introdus greșit valorile elementelor unui tuplu ori, în cadrul programului, pur și simplu *avem nevoie de modificări*. Obiectul este implicit *nemodificabil*, asadar vom folosi – **conversia explicită**.

main.py

Run

Shell

```
1  scriitor = ("Grigore", "Vieru", 70, 1.75, "Pererâta")
2  #am greșit vârsta, trebuie reținut 25
3  #P1. Creăm o listă ce conține elementele tuplului
4  temp = list(scriitor)
5  #P2. Listele pot fi modificate, deci...
6  temp[2] = 25
7  #P3. Rețin în scriitor elementele listei ca tuplu
8  scriitor = tuple(temp)
9  print(scriitor)
```

```
('Grigore', 'Vieru', 25, 1.75, 'Pererâta')
>
```

Pasul 1. Listele sunt obiecte modificabile, deci creăm una numită **temp** care să conțină toate elementele tuplului **scriitor**.

Pasul 2. Astfel, putem să rescriem elementul de pe poziția a doua din lista **temp** corespunzător.

Pasul 3. Acum că este totul corect și cum ne-am dorit, reținem în variabila **sriitor** datele, convertind explicit lista **temp** spre tipul **tuple** cu ajutorul constructorului său.

CE ESTE UN SET?

Putem privi **seturile** (tipul **set**) precum niște *mulțimi* de la matematică. Elementele unui astfel de tip de date sunt **neordonate** (nu putem folosi indici), iar fiecare dintre ele este **unic** (se regăsește o singură dată). Totuși, este permis să adăugăm ori să ștergem elemente.

Pentru a crea **un set de date**, se folosesc *acoladele.*

main.py — Run — Shell

```
1  set1 = {1,2,3,4,5,6,7,8,9}
2  #afișăm setul creat
3  print(set1)
4  print('numărul de element din set=',len(set1))
5  #folosim unii operatori
6  print(1 in set1)
7  print(5 not in set1)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
numărul de element din set= 9
True
False
```

Observație importantă. După formarea unui set, Python nu respectă ordinea introducerii elementelor.

ADĂUGAREA ȘI STERGEREA ELEMENTELOR

Pentru a **adăuga** elemente noi în cadrul unui set de date, putem folosi metodele **add(element)** sau **update(element1, element2, ...)**.

Pentru a **șterge** un element putem folosi metodele **discard()** sau **remove()**, diferența dintre ele fiind faptul că cea din urmă ridică o excepție (eroare) și programul se oprește din interpretare dacă elementul nu este găsit.

main.py Run Shell

```
1  set1 = {'A','B','C'}
2  print('Setul inițial:',set1)
3  set1.add('D') #doar un element
4  set1.update('E','F','G') #mai multe deodată
5  print('După adăugări:',set1)
6  #ștergem elementul 'E'
7  set1.discard('E')
8  #ștergem elementul 'B'
9  set1.remove('B')
10 #afișăm lista modificată
11 print('După ștergeri:',set1)
```

```
Setul inițial: {'A', 'B', 'C'}
După adăugări: {'A', 'E', 'D', 'B', 'G', 'F', 'C'}
După ștergeri: {'A', 'D', 'G', 'F', 'C'}
>
```

OPERAȚII CU MULȚIMI

"|" (reuniunea),
"& " (intersecția),
"-" (diferența)
"^" (diferența simetrică)

main.py

Run

Shell

```
A = {1,2,3,4,5,6,7}
B = {5,6,7,8,9,10}
#reuniunea
print(A|B)
#intersecția
print(A&B)
#diferența dintre A și B
print(A-B)
#diferența dintre B și A
print(B-A)
#diferența simetrică
print(A^B)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{5, 6, 7}
{1, 2, 3, 4}
{8, 9, 10}
{1, 2, 3, 4, 8, 9, 10}
```

OPERAȚII CU MULȚIMI

Putem folosi și metodele clasei **set**, precum:
union(), **intersection()** și **difference()**,

main.py

Run

Shell

```
1  A = {1,2,3,4,5,6,7}
2  B = {5,6,7,8,9,10}
3  #reuniunea
4  print(A.union(B))
5  #intersecția
6  print(A.intersection(B))
7  #diferența dintre A și B
8  print(A.difference(B))
9  #diferența dintre B și A
10 print(B.difference(A))
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{5, 6, 7}
{1, 2, 3, 4}
{8, 9, 10}
```

OPERAȚII CU MULȚIMI

Pentru a actualiza un set în urma efectuării operației, există metodele **update()**, **intersection_update()** sau **difference_update()**:

main.py

Run

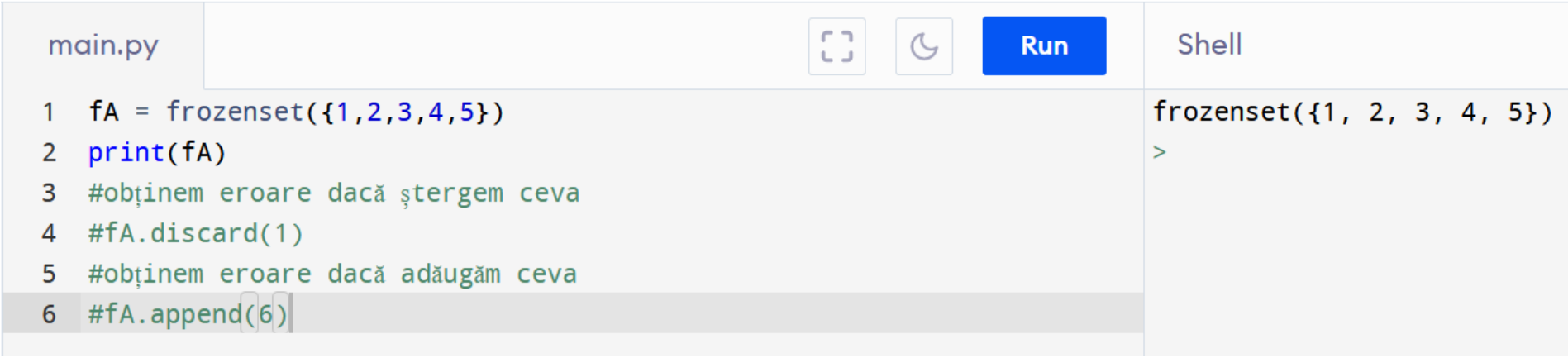
Shell

```
1  A = {1,2,3,4,5,6,7}
2  B = {5,6,7,8,9,10}
3  A.update(B) # A va reține A reunit cu B
4  print(A)
5  A.intersection_update(B) # A va reține A intersectat cu B
6  print(A)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
{5, 6, 7, 8, 9, 10}
```

FROZENSET

În limbajul Python există și tipul de date **frozenset**, care permite crearea unui **set nemodificabil**. Funcția (constructorul) **frozenset()** primește ca argument o colecție de date pe care o transformă spre un set "**înghețat**", deci cu *elemente unice, nemodificabile și neordonate.*



Puteți considera valorile conținute ca niște **date constante** ori **cuvinte cheie**, care uneori sunt utile în cadrul programelor.

LECȚII PRACTICE *03/04.11.2025

MULȚIMI (SET-uri)

1. Fie mulţimile A={1,3,a,4,c,d,5,8,2} și B={2,a,c,8,4,9,e,3}. Să se calculeze mulțimea C=(A∪B)\(A∩B).

2. Se dau mulțimile X și Y de numere naturale mai mici decât 200. Să se determine mulțimile:
   a) X∪Y b) X∩Y c)X\Y d)(X\Y)∪(Y\X) e)(X\Y)∩(Y\X)

3. Considerând mulțimile X,Y,Z – mulțimi de numere naturale date mai mici decât 200. Să se verifice legile lui Morgan:

   a) $\overline{X \cup Y \cup Z} = \overline{X} \cap \overline{Y} \cap \overline{Z}$; b) $\overline{X \cap Y \cap Z} = \overline{X} \cup \overline{Y} \cup \overline{Z}$.

4. Să se afișeze la ecran toate submulțimile mulțimii A={1,2,3,A,B,C}

LECȚII PRACTICE *10/11.11.2025

TUPLURI

1. Pentru un grup din 5 elevi se cunoaște: numele, prenumele, sexul(genul), 3 note de la tezele semestriale. Să se afișeze:
   - nota medie obținută la sesiune de fiecare elev
   - numele, prenumele elevilor restanțieri
   - numele, prenumele elevilor eminenți (nota medie mai mare sau egală cu 9.00 și nici o notă mai mică decât 9 obținută la teză)

2. Este declarat un tuplu de tip dată calendaristică curentă (ziua, luna, denumirea zilei săptămânii). Să se calculeze câte zile cu această denumire au fost de la începutul anului curent până în ziua curentă.

3. Pentru un eșantion social de persoane se cunoaște:
   - numărul de persoane (cuprins între 2...10 persoane)
   - vârsta, înălțimea, masa (greutatea), sexul(genul), starea civilă(căsătorit sau nu)

Să se determine:
   - procentul persoanelor sub 20 de ani
   - procentul persoanelor cu înălțimea mai mare de 170 cm
   - masa medie a unei persoane de peste 18 ani
   - ce procent din numărul persoanelor de sex feminin au peste 20 de ani și nu sunt căsătorite
   - ce procent din numărul persoanelor între 20 și 50 ani au greutatea mai mare decât greutatea medie.