UNIVERSITATEA TEHNICĂ A MOLDOVEI

Programare Interactivă

Tema: Colecții de date: Tupluri, Seturi

lect. univ. Victoria ALEXEI
Departament Informatică și Ingineria Sistemelor
Facultatea Calculatoare, Informatică și Microelectronică

UNIVERSITATEA TEHNICĂ A MOLDOVEI

În acest curs vom învăța:

Tuplurile

- Crearea tuplurilor
- Operatori și funcții
- Metode de bază cu tupluri
- Parcurgerea tuplurilor

Seturi / mulțimi

- Crearea seturilor
- Operații pe seturi
- Metode de bază cu seturi
- Parcurgerea seturilor

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Tuplurile

Crearea tuplurilor

Un tuplu este o colecție de date **imutabilă,** ceea ce înseamnă că nu putem schimba, adăuga sau elimina elemente după crearea tuplului.

Tuplurile sunt utilizate pentru a stoca **mai multe elemente într-o singură variabilă separate prin virgulă.** Tuplurile sunt **reprezentate prin paranteze rotunde ()**.

Exemple de tupluri

```
t = (2,3,4)
print(t, type(t))

t1 = 1,2,4
print(t1, type(t1))

t2 = "cifra",
print(t2, type(t2))

t3 = "cifra"
print(t3, type(t3))
```

Rezultate

```
(2, 3, 4) <class 'tuple'>
(1, 2, 4) <class 'tuple'>
```

```
('cifra',) <class 'tuple'>
cifra <class 'str'>
```

NB! Python aplică o virgulă la sfârșitul tuplului cu un sigur element pentru a menționa că este tuplu.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Funcția **tuple() –** crează un tuplu

Exemple

```
t4 = tuple((2,5,6))
print(t4)
t6 = tuple([78,90,67,5])
t7 = tuple(('forma',))
t8 = tuple('forma')
t = tuple()

print(t6,t7,t8, t, sep = '\n')
```

Rezultate

```
(2, 5, 6)
```

```
(78, 90, 67, 5)
('forma',)
('f', 'o', 'r', 'm', 'a')
()
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Funcția **tuple() –** crează un tuplu

Exemplu

```
t5 = tuple(2,5,6)
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Victoria Lucru2021\BazeleProgramariiCalculat
    t5 = tuple(2,5,6)

TypeError: tuple expected at most 1 argument, got 3
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Tupluri imbricate

Un tuplu poate **stoca diferite tipuri de date, este posibilă imbricarea tuplului**

Exemplu

```
t = (5,('date', 'imbricate'), "diferite", True, False, (6,9))
print(t, type(t))
```

Rezultat

```
(5, ('date', 'imbricate'), 'diferite', True, False, (6, 9)) <class 'tuple'>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Tipuri de date mutabile în tupluri.

Liste imbricate în tupluri

Exemplu

```
tt = (5,('date', 'imbricate'), "diferite", [34,89,56,3,7,8], (6,9))
tt[3][-1] = 333
print(tt, type(tt))
```

Rezultat

```
(5, ('date', 'imbricate'), 'diferite', [34, 89, 56, 3, 7, 333], (6, 9)) <class 'tuple'>
```

Exemplu

```
#Modificarea unui element mutabil in tuplu
l=[34,89,56,3,7,8]
tt = (5,('date', 'imbricate'), "diferite", l, (6,9))
tt[3][2] = 555
print(tt)
```

Rezultat

```
(5, ('date', 'imbricate'), 'diferite', [34, 89, 555, 3, 7, 8], (6, 9))
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Packing / Unpacking - Împachetarea și despachetarea

Când creăm un tuplu, îi atribuim valori. Acest proces se numește **packing „împachetare”.**

Exemplu

```
b = ('Ion', 'Postu', 19)
print(b)
```

Rezultat

```
('Ion', 'Postu', 19)
```

Python, ne permite să extragem valorile înapoi în variabile (separăm elementele unui tuplu). Acțiunea se numește **unpacking „despachetare”.**

Exemplu

```
nume, prenume, varsta = b
print(nume, prenume, varsta)
```

Rezultat

```
Ion Postu 19
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Packing / Unpacking - Împachetarea și despachetarea

Exemplu

```
nume, prenume, varsta, elem = b
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Tuple-Set.py", line 37, in <module>
    nume, prenume, varsta, elem = b

ValueError: not enough values to unpack (expected 4, got 3)
```

Exemplu

```
b = ('Ion', 'Postu', 19, 25)
nume, prenume, varsta = b
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Tuple-Set.py", line 42, in <module>
    nume, prenume, varsta = b

ValueError: too many values to unpack (expected 3)
```

NB!

Numărul de variabile trebuie să se potrivească cu numărul de valori din tuplu

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Crearea tuplurilor

Packing / Unpacking - Împachetarea și despachetarea

Utilizăm **asterisc "*"** pentru a colecta valorile rămase ca listă

Exemplu

```
fructe = ('mere', 'prune', 'capsuna', 'caisa', 'portocala')
verde, violet, rosu, *galben = fructe

print(verde, violet, rosu, galben, sep='\n')
```

Rezultat

```
mere
prune
capsuna
['caisa', 'portocala']
```

Dacă **asteriscul este adăugat la o altă variabilă decât ultima**, Python va atribui valori variabilei cu * până când numărul de valori rămase se potrivește cu numărul de variabile rămase.

Exemplu

```
fructe = ('mere', 'zmeura', 'capsuna', 'caisa', 'mure')
verde, *dulci, negre = fructe
print(verde, dulci, negre, sep='\n')
```

Rezultat

```
mere
['zmeura', 'capsuna', 'caisa']
mure
```

Operatori și funcții

Tuplurile pot **fi comparate**, poziție după poziție.

Două tupluri sunt egale – când au aceeași dimensiune, aceleași valori pe aceeași poziție

Exemplu

```
print((5,7,9) == (5,7,9))
print((5,7,9) == (5,9,7))
print((2,4,0) == (2,4))
```

Rezultat

```
True
False
False
```

Exemplu

```
print((5,7,9) < (5,8,0))
print((0,7) < (0,7,0))
print((0,7) == (0,6,0))
print((5,7,9) > (5,8,9))
```

Rezultat

```
True
True
False
False
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatori și funcții

Apartenența de membru – operatorul **“in”, “not in”**

Exemplu

```
tup = ('cinci', 6,'sapte',8)
print('cinci' in tup)
print(8 in tup)
print('sase' in tup)
```

Rezultat

```
True
True
False
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatori și funcții

Extragerea unui element din tuplu

- Accesul la elementele tuplului se efectuiază similar ca la stringuri și liste.
- Elementele tuplului au o ordine definită ca cele dintr-o listă
- Indecșii tuplurilor se încep de la zero.

'mere'	'prune'	'capsuna'	"caisa"	"portocala"
0	1	2	3	4

Exemplu

```
fruct = ('mere', 'prune', 'capsuna')
print(fruct[1], fruct[0], fruct[-1], sep = '\n')
```

Rezultat

```
prune
mere
capsuna
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatori și funcții

Felierea / Slicing tuplurilor

- Slicing funcționează la fel ca pentru liste.
- La felierea tuplurilor se obține un tuplu nou

'mere'	'prune'	'capsuna'	"caisa"	"portocala"
0	1	2	3	4

L [start : stop : step]

Exemplu

```
fr = ('mere', 'prune', 'capsuna', 'caisa', 'portocala')
print(fr[1:4], fr[0:3], fr[-3:-1], sep = '\n')
print(fr[0:3], fr[::-1], fr[::], sep = '\n')
```

Rezultat

```
('prune', 'capsuna', 'caisa')
('mere', 'prune', 'capsuna')
('capsuna', 'caisa')
```

Rezultat

```
('mere', 'prune', 'capsuna')
('portocala', 'caisa', 'capsuna', 'prune', 'mere')
('mere', 'prune', 'capsuna', 'caisa', 'portocala')
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatori și funcții

Funcția **len() –** lungimea tuplului

Exemplu

```
fr = ('mere', 'prune', 'capsuna', 'caisa', 'portocala')
print(len(fr))
print(len(()))
```

Rezultat

```
5
0
```

Funcția **sorted() –** sortează elementele din tuplu și returnează o listă

Exemplu

```
fr = ('mere', 'prune', 'capsuna', 'caisa', 'portocala')
print(sorted(fr))
print(type(sorted(fr)))
print(tuple(sorted(fr)))
```

Rezultat

```
['caisa', 'capsuna', 'mere', 'portocala', 'prune']
<class 'list'>
('caisa', 'capsuna', 'mere', 'portocala', 'prune')
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operatori și funcții

Funcțiile **min(), max() –** află elementul minim/maxim din tuplu

Exemplu

```
t1 = (5,9,7)
print(min(t1))
print(max(t1))
```

Rezultat

```
5
9
```

Adunarea și multiplicarea tuplurilor

Adunarea cu operatorul ”+” – concatenarea este o combinație de tupluri

Multiplicarea cu operatorul ”*” – crearea unui tuplu nou, în care elementele se repetă de n ori

Exemplu

```
t1 = (5,9,7)
t2 = ('n',)

t1 = t1 + t2
print(t1)
```

Exemplu

```
t2= t2*5
print(t2)
```

Rezultat

```
(5, 9, 7, 'n')
('n', 'n', 'n', 'n', 'n')
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu tupluri

count() – calculează numărul de apariții ale unui element al tuplului
index() - Caută în tuplu o valoare specificată și returnează prima poziție unde a fost găsit

Exemplu

```
t = ('s', 'a', 'r', 'a', 'd','e')
print(t.count('a'))
print(t.index('r'))
```

Rezultat

```
2
2
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Tuplurile sunt mai eficiente

- Deoarece tuplurile sunt structuri ce nu se modifică, ele sunt mai simple și mai eficiente la utilizarea memoriei și performanței decît listele, sunt un fel de constante
- Deci în programe cînd se crează ”variabile temporare” noi vom prefera tuplurile în loc de liste

Ocupă mai puțin spațiu de memorie și pot fi manipulate mai rapid de către interpretator

Exemplu

```
a=[1,2,3,4,5,6,7,8,9]
b=(1,2,3,4,5,6,7,8,9)
print(a.__sizeof__())
print(b.__sizeof__())
```

Rezultat

```
112
96
```

Pot fi utilizate dept chei în dicționare

Exemplu

```
d = {(1,1,1) : 1}
print(d)
```

Rezultat

```
{(1, 1, 1): 1}
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Tuplurile sunt mai eficiente

Tuplurile ca valori returnabile

Funcțiile întotdeauna returnează doar o singură valoare, dar făcând din această valoare un tuplu, putem grupa efectiv câte valori dorim și le putem returna.

De exemplu, am putea scrie o funcție care returnează atât aria, cât și circumferința unui cerc cu raza r:

Exemplu

```
import math
def f(r):
    c = 2 * math.pi * r
    a = math.pi * r * r
    return (c, a)

print(f(5))
```

Rezultat

```
(31.41592653589793, 78.53981633974483)
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Parcurgerea tuplurilor

Ca și în cazul listelor, putem parcurge iterativ un tuplu

Bucla for

Exemplu

```
tp = (87,56,23,"s",11,"ac")
for i in tp:
    print(i)
```

Rezultat

```
87
56
23
s
11
ac
```

Bucla for cu range()

Exemplu

```
tp = (87,56,23,"s",11,"ac")
for i in range(0,len(tp)):
    print(i, ' ',tp[i])
```

Rezultat

```
0   87
1   56
2   23
3   s
4   11
5   ac
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Parcurgerea tuplurilor

Ca și în cazul listelor, putem parcurge iterativ un tuplu

Bucla while

Exemplu

```
tp = (87,56,23,"s",11,"ac")
i=0
lungimea = len(tp)
while i < lungimea:
    print(tp[i])
    i = i+1
```

Rezultat

```
87
56
23
s
11
ac
```

Funcția enumerate()

Exemplu

```
t = ('s', 'a', 'r', 'a', 'd','e')
for elem in enumerate(t):
    print(elem)
```

Rezultat

```
(0, 's')
(1, 'a')
(2, 'r')
(3, 'a')
(4, 'd')
(5, 'e')
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Seturi / mulțimi

Crearea seturilor

Un set (o mulțime) este un obiect care stochează o colecție de date. Un set are câteva caracteristici:

- Toate elementele setului sunt unice, adică două elemente nu pot avea aceeași valoare
- Seturile sunt structuri neordonate, ceea ce înseamnă că elementele lui pot sta în orice ordine
- Elementele setului pot fi de diferite tipuri și nu sunt păstrate într-o anumită ordine.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Seturi / mulțimi

Crearea seturilor

Funcția **set()**

Exemplu

```
setul=set('sir ordonat')
print(setul)
```

Rezultat

```
{'d', 'a', 't', 'i', 'r', ' ', 'o', 'n', 's'}
```

Exemplu

```
setul_meu= set([23,56,7,34,45])
print(setul_meu)
```

Rezultat

```
{34, 7, 45, 23, 56}
```

Exemplu

```
setul = {}
print(type(setul))

print(set())
```

Rezultat

```
<class 'dict'>

set()
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Seturi / mulțimi

Crearea seturilor

Exemplu

```
cuvinte = ['salut', 'tata', 'mama', 'salut', 'mama']
print(set(cuvinte))
```

Rezultat

```
{'mama', 'salut', 'tata'}
```

Inițializarea unui set

Exemplu

```
s1= {'oras','sat','localitate','comuna', 'municipiu', 'sat'}
print(s1, type(s1))
```

Rezultat

```
{'localitate', 'municipiu', 'comuna', 'sat', 'oras'} <class 'set'>
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Seturi / mulțimi

Operații pe seturi

Funcția **len() –** numărul de elemente din set

Exemplu

```
s1= {'oras','sat','localitate','comuna', 'municipiu', 'sat'}

print(len(s1))
```

Rezultat

```
5
```

Python păstrează doar elementele distincte

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații pe seturi

Testul de apartenență – operatorul **“in” “not in”**

Nu putem accesa elementele unei mulțimi prin indecși, putem răsfoi elementele cu bucla **for** sau să testăm dacă **este o valoare specificată în set**

Exemplu

```
s1= {'oras','localitate','comuna', 'municipiu', 'sat'}
print('comuna'in s1)
print('tara' in s1)
print('cahul' not in s1)
```

Rezultat

```
True
False
True
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații pe seturi

Egalitatea mulțimilor

Operațiile de comparație a mulțimilor testează dacă două mulțimi sunt egale sau disjuncte
Verifică dacă o mulțime se conține în cealaltă

Exemplu

```
s2 = {'trei', 'doi',6}
s3 = {'trei', 'doi',6}
print(s2==s3)
print(s2!=s3)
```

Rezultat

```
True
False
```

NB! Operatorii de comparare a mulțimilor clasici nu au sens, deoarece elementele unei mulțimi nu sunt ordonate.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu seturi

Adăugarea și ștergerea 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.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu seturi

Adăugarea și ștergerea elementelor

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

Exemple

```
s3 = {'trei', 'doi',6,4}
print('s3:',s3)
s3.add('masa')
s3.add(56)
print('s3 modificat add:',s3)
```

Rezultat

```
s3: {'doi', 'trei', 4, 6}
s3 modificat add: {'doi', 4, 6, 'masa', 'trei', 56}
```

```
s6 = {'trei', 'doi',6,4}
s6.update('eseu','foc','G',[5,7,9])
print('s6 modificat update:',s6)
```

Rezultat

```
{'doi', 4, 5, 6, 7, 'G', 9, 'c', 'trei', 'u', 'o', 'f', 's', 'e'}
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu seturi

Adăugarea și stergerea elementelor

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

Exemple

```
s3 = {'trei', 'doi',6,4}
s3.discard('6')
s3.discard('trei')
s3.remove('B')
```

Exemple

```
print(s3.pop())
```

Rezultate

```
doi
```

```
{4, 6}
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Tuple-Set.py", line 204, in <module>
    s3.remove('B')

KeyError: 'B'
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu seturi

Operații cu mulțimi

Limbajul Python, prin seturile de date, ne oferă acces facil la unele dintre **operațiile uzuale cu mulțimi**.
Operațiile cu mulțimi sunt ***reuniune***, ***intersecție***, ***diferență*** **și** ***diferență simetrică***

Operatorii folosiți au fost: "**|**" (reuniunea), "**&** " (intersecția), "**-**" (diferența) și "**^**" (diferența simetrică).

Metodele clasei **set**, sunt **union()**, **intersection()** și **difference()**
Toate cele trei metode întorc un nou set care conține rezultatul.

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

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Metode de bază cu seturi

Operații cu mulțimi

Exemple

```
A = {1,2,3,4}
B = {2,3,4,5,6,7,8}

print(A|B)
print(A.union(B))

print(A&B)
print(A.intersection(B))

print(A-B)
print(A.difference(B))

print(B-A)
print(B.difference(A))

#diferența simetrică
print(A^B)
```

Rezultate

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

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Parcurgerea seturilor

Exemplu

```
s1= {'oras','localitate','comuna', 'municipiu', 'sat'}
for i in s1:
    print(i)
```

Rezultat

```
localitate
municipiu
comuna
sat
oras
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Frozenset

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

Exemplu

```
sf = frozenset({1,2,3,4,5})
print(sf)
```

Rezultat

```
frozenset({1, 2, 3, 4, 5})
```

```
sf.add(3)
sf.add('cinci')
```

Rezultat

```
Traceback (most recent call last):

  File "C:\Users\Admin\Desktop\Tuple-Set.py", line 260, in <module>
    sf.add(3)

AttributeError: 'frozenset' object has no attribute 'add'
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Exemple de probleme cu tupluri

```
def conversie_data(luna, zi, an):
    """ Avand data scrisa sub forma (6, 17, 2021)
    sa se scrie sub forma Iunie 17, 2021 """

    tup=("Ianuarie", "Februarie", "Martie", "Aprilie", "Mai", "Iunie", "Julie", \
    "August", "Septembrie", "Octombrie", "Noiembrie", "Decembrie")

    print(str(zi),str(tup[luna-1]) +",",an)
```

```
In [75]: conversie_data(9,1,2021)
1 Septembrie, 2021

In [76]: conversie_data(8,27,2021)
27 August, 2021
```

UNIVERSITATEA TEHNICĂ
A MOLDOVEI

Mulțumesc de atenție