Tutoriat 4

Tuple

Tuple:

-tip de date iterabil, ce pot stoca tipuri diferite de date.

Ex: lista= (1,'2',[3,'ána'])

-nu pot fi modificate după declarare.

```
1  t=(1,2)
2  t[1]=0
3  print(t)
```

```
Traceback (most recent call last):
  File "<string>", line 3, in <module>
TypeError: 'tuple' object does not support item assignment
```

Metode de declarare:

Exemple:

folosind ():

t1=() (tuple goala)

t2 = (1,2,3,4,5)

folosind constructorul tuple()

tuple(<iterabil>)

t3=tuple("1234") =>t3 = ('1', '2', '3', '4')

t4=tuple([1,2,3]) =>t4=(1,2,3)

folosind *

t5= <tuple> * n => t5 este format prin concatenarea lui <tuple> de n ori , dacă n este negativ sau 0 atunci se va memora []

Ex:

l = (1,2,3) * 3 => l =(1,2,3,1,2,3,1,2,3)

Parcurgerea tuplurilor:

Metoda 1

for <variabila> in <tuple>:

print(<variabila>)

Metoda 2

for <variabila> in range(len(<tuple>)

print(<lista>[<variabila>])

Slicing:

Tip accesare	Rezultat
s[i]	Elementul de pe poziția i în lista
s[i:]	returnează un tuple ce conține elementele incepand cu poziția i în s pana la finalul șirului
s[i:j]	returnează un tuple ce conține elementele incepand cu poziția i în s pana la poziția j-1
s[:]	returnează întreaga secvență
s[:j]	returnează un tuple ce conține elementele incepand cu poziția 0 în s până la poziția j-1
s[::-1]	returnează întreaga secvență în ordine inversa Ex: (1,2,3,4)[::-1]=>[4,3,2,1]

Metodele:

Metoda	Descriere
<tuple>.count(<val>)	Returnează numărul de elemente cu valoarea specificată
len(<tuple>)	Returnează lungimea
<tuple>.index(<val>)	Returnează poziția la care se află valoarea

! Pentru a verifica dacă un tuple ‘t’ conține valoarea ‘val’ putem folosi **in**

Ex:

```
if val in t:
    print(“Exista”)
else:
    print(“Nu exista”)
```

Concatenarea de liste:

Se poate folosi operatorul + , dar și funcțiile append si extend

Ex:

t = (1,2,3,4) + (2,3) => (1,2,3,4,2,3)

Copiere:

t1=(1,2,3)
t2=t1 =>t2=(1,2,3)

Generatori:

Returnează un iterator peste elementele create prin utilizarea comprehensiunii.
(<valoare/variabila> for <variabila> in <iterabil> if <condiție>)
Un obiect generator, spre deosebire de o listă, nu are o lungime.

Pentru a afișa valorile din generator avem 2 variante:

```
for <var> in <generator>:
    print(<var>)

try:
    while <generator>:
        print(next(<generator>)
except:
    pass
```

! dacă am ajuns la finalul generatorului se arunca o excepție pe care o prindem folosind except

Dictionare

Dictionare:

-tip de date iterabil, ce pot stoca tipuri diferite de date sub forma cheie:valoare.

Ex:

```
d={
    1: [1,2,3,4],
    "string": "valoare",
    (1,"string"): 1964
}
```

-cheile pot fi doar valori imutabile și sunt unice. Ex: stringuri, tuple, numere.
-valorile pot avea orice tip de date.
-pot fi modificate după initializare.
-elementele sunt indexate după cheie

Metode de declarare:

Exemple:

folosind {}:
d1={} (dicționarul gol)

```
d={
        1: [1,2,3,4],
        "string": "valoare",
        (1,"string"): 1964
}
```

Accesarea elementelor din dicționar:

<dict>[<key>]

Ex: Pentru dicționarul d declarat mai sus:

d[1]->[1,2,3,4]

! dacă nu exista <key> in dictionar va returna o eroare.

Putem verifica dacă o cheie nu este în dicționar folosind not in:

```
if <key> not  in <dict>:
        print("cheia nu este in dictionar")
```

Parcurgerea unui dicționar:

```
for <key> in <dict>:
        print(<dict>[<key>])
```

Metodele:

Metoda	Descriere
<dict>.clear()	Elimină toate elementele din dicționar
<dict>.copy()	Returnează o copie.
<dict>.keys()	Returnează o listă care conține cheile dicționarului.
<dict>.update({<cheie>:<valoare>})	Actualizează dicționarul cu perechile cheie-valoare specificate
<dict>.pop(<cheie>)	Elimină elementul cu cheia specificată

Set

Set:

-tip de date iterabil, ce pot noate stoca duplicate.

-elementele dintr-un set nu au o ordine definită

Ex:

```
s={1,2,3}
print(s) ->{2,1,3}
print(s) ->{3,2,1}
```

-nu putem schimba elementele după ce setul a fost creat, dar putem elimina și adăuga elemente noi.

Metode de declarare:

Exemple:

folosind {}:

s1={1,2,3}

folosind constructorul tuple()

set(<iterabil>)

s2=set() mulțimea vidă/set gol

s3=(“1234”) =>s3 = {'1', '2', '3', '4'}

Parcurgere:

Metoda 1

```
for <variabila>  in  <set>:
    print(<variabila>)
```

Metodele:

Metoda	Descriere
<set>.add(<item>)	Adaugă un element la set
<set>.clear()	Îndepărtează toate elementele din set
<set>.discard(<item>)	Elimină elementul specificat (dacă <item> nu exista nu se întoarce o eroare)
<set>.pop()	Elimină un element aleator.

<set>.intersection(<set2>)	Returnează o mulțime,care reprezinta intersecția altor două mulțimi
<set>.union(<set2>)	Returnează un set care conține uniunea de mulțimi

Pentru mai multe metode:

-https://www.w3schools.com/python/python_dictionaries_methods.as -dictionare

-https://www.w3schools.com/python/python_sets_methods.asp -seturi