Praktikum iz programiranja 3

2023/24

Python
Torke (Tuples)

- Torka je nepromenljiva lista – ne može da se menja po formiranju

```
L = [1, 2, 3]
T = (1, 2, 3)
```

- Torke se navode unutar malih zagrada ili bez njih.
- Indeksiranje i segmentacija su isti kao i kod liste.
- Kao i kod liste funkcijom `len` možete dobiti dužinu torke.
- Torke imaju kao i liste `count` i `index` metode.
- Međutim, pošto su torke nepromenljive, torke nemaju neke metode koje imaju liste, kao što su `sort` ili `reverse`.

Python
Torke (Tuples)

```
          nole = ('Novak', 'Đoković', 1987, 'Srbija’, 21)
>>> nole[4]
12
>>> nole[4]=22     #greška
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nole[4]=22
TypeError: 'tuple' object does not support item assignment
>>> nov_nole = nole[0:3] + (22,)
>>> nov_nole
('Novak', 'Đoković', 1987, 22)
>>>
```

Python
Torke (Tuples)

- Jedan od razloga zašto postoje i liste i torke je što su nam u nekim situacijama potrebne nepromenljive liste. Na primer:
  - lista ne može da bude ključ u rečniku (lista se može menjati, problem je kako pratiti promene ključa)
  - torke mogu da igraju ulogu ključa u rečnicima.

```
ocene = {('Petar', 'Ana'): 95, ('Milovan', 'Igor'): 87}
```

Python
Torke (Tuples)

- Brže od listi
- Za konverziju nekog objekta u torku koristimo funkciju `tuple`.

```
t1 = tuple([1,2,3])     (1, 2, 3)
t2 = tuple('abcde’)     ('a', 'b', 'c', 'd', 'e')
```

- Prazna torka je `()`.
- Torka sa jednim elementom `(1,)`.
- `(1)` matematički izraz koji ima vrednost 1.

- Torke se koriste za vraćanje više vrednosti iz funkcije
- Torke mogu biti elementi liste

Python
Torke (Tuples)

- Torke mogu biti elementi liste

```
meseci = [("јануар", 31), ("фебруар", 28), ("март", 31), \
          ("април", 30), ("мај", 31), ("јун", 30), \
          ("јул", 31), ("август", 31), ("септембар", 30), \
          ("октобар", 31), ("новембар", 30), ("децембар", 31)]
broj = int(input("Унеси редни број месеца:"))
mesec = meseci[broj - 1]
print("Назив:", mesec[0], "Број дана:", mesec[1])
```

Primer

- Napravi program koji će simulirati skupljanje bodova na kvizu. Svaki igrač će imati određeni broj poena nakon svakog pitanja. Igrači će biti predstavljeni torkama oblika (ime, poeni). Program treba da omogući dodavanje novih igrača, ažuriranje poena postojećih igrača, kao i ispis trenutnog stanja sa sortiranjem igrača po broju poena od najvećeg ka najmanjem.

- Dodatno, program treba da podrži dodatnu funkcionalnost u kojoj se ispituje da li je neki igrač prešao određeni prag poena (npr. 100 poena) i ispisuje se poruka ako jeste.

Primer

```
def dodaj_igraca(igraci, ime, poeni=0):
    igraci.append((ime, poeni))

def azuriraj_poene(igraci, ime, dodatni_poeni):
    for i, (ime_i, poeni_i) in enumerate(igraci):
        if ime_i == ime:
            igraci[i] = (ime, poeni_i + dodatni_poeni)
            break

def sortiraj_igrace(igraci):
    return sorted(igraci, key=lambda x: x[1], reverse=True)

def ispisi_igrace(igraci):
    for ime, poeni in igraci:
        print(f"{ime}: {poeni}")
```

Primer

```
def proveri_prag(igraci, prag):
    for ime, poeni in igraci:
        if poeni >= prag:
            print(f"{ime} je prešao prag od {prag} poena!")

igraci = []
dodaj_igraca(igraci, "Ivan")
dodaj_igraca(igraci, "Ana")
dodaj_igraca(igraci, "Marko")

azuriraj_poene(igraci, "Ivan", 10)
azuriraj_poene(igraci, "Ana", 15)
azuriraj_poene(igraci, "Marko", 5)

igraci = sortiraj_igrace(igraci)
ispisi_igrace(igraci)

proveri_prag(igraci, 20)
```

Python
Skupovi (Sets)

- Python ima i strukturu koja se zove `set (skup)`.
- Set struktura radi slično matematičkim skupovima.
- Skupovi su poput listi u kojima nije dozvoljeno ponavljanje elemenata.
- Skupovi se označavaju vitičastim zagradama, `{}`

```
S = {1,2,3,4,5}
```

- Prazan skup dobijamo pomoću funkcije `set()`

```
S = set()
```

Python
Skupovi (Sets)

- Funkcija set se može koristiti i za konverziju objekata u skupove.

```
set([1,4,4,4,5,1,2,1,3])
set('this is a test')
{1, 2, 3, 4, 5}
{'a', ' ', 'e', 'i', 'h', 's', 't’}
```

- Skupove možemo sastavljati na različite načine:

```
s = {i**2 for i in range(12)}
{0, 1, 4, 100, 81, 64, 9, 16, 49, 121, 25, 36}
```

Python
Skupovi (Sets)

- Nad skupovima je moguće primeniti relacijske operatore:
  - == (jednakost)
  - <= (podskup)
  - < (pravi podskup)
  - >= (nadskup)
  - > (pravi nadskup)

Python
Skupovi (Sets)

- Python smešta podatke u skup u proizvoljnom redosledu, i ne uvek u redosledu koji ste vi postavili.
- Kod skupova nije važan redosled, već samo podaci koji ga čine. To znači da indeksiranje kod skupova nema smisla. Zato se ne može , na primer, tražiti s[0], za set s.
- Operacije za rad sa skupovima

Operator	Opis	Primer
|	Unija	{1,2,3} | {3,4}→{1,2,3,4}
&	Presek	{1,2,3}&{3,4}→{3}
-	Razlika	{1,2,3}-{3,4}→{1,2}
^	Simetrična razlika	{1,2,3}^{3,4}→{1,2,4}
in	Jeste u skupu	3 in {1,2,3}→True

Python
Skupovi (Sets)

- Metode koje se primenjuju na skupove

Operator	Opis
S.add(x)	Dodaje x u skup
S.remove(x)	Sklanja x iz skupa
S.issubset(A)	Vraća True ako je $S \subset A$, a False ako nije
S.issuperset(A)	Vraća True ako je $A \subset S$, a False ako nije.

Python
Skupovi (Sets)

```
>>> {'crvena', 'plava'} == {'plava', 'crvena'}
True
>>> a, b, c = {1, 2, 3, 4, 5}, {3, 4, 5}, {1, 2}
>>> c.add(3) # metod za ubacivanje elementa u skup
>>> c
{1, 2, 3}
>>> c.discard(3) # uklanja element ako postoji
>>> c
{1, 2}
>>> a & b # presek skupova
{3, 4, 5}
>>> b & c # presek je prazan
set()
```

Python
Skupovi (Sets)

```
>>> b | c # unija skupova
{1, 2, 3, 4, 5}
>>> a - b # razlika skupova
{1, 2}
>>> a >= b # a nadskup b
True
>>> b <= a # b podskup a
True
>>> a.clear() # uklanja sve elemente iz skupa
>>> a
set()
```

Python
Skupovi (Sets)

- Brisanje duplih elemenata iz liste
- Možemo iskoristiti činjenicu da skupovi ne mogu imati ponovljene elemente.

```
L = [1,4,4,4,5,1,2,1,3]    # lista sa ponavljanjem elemenata
L = list(set(L))           # konvertujemo listu u skup, pa nazad u listu
```

```
[1,2,3,4,5]
```

Python
Skupovi (Sets)

- Napisati program u kojem se unosi broj n a zatim n celih brojeva. Za sve cifre koje su se javile u zapisu unetih brojeva odrediti koliko puta su se ukupno javile.

```
n = int(input())
brojevi = []
for i in range(n):
    brojevi = brojevi + [int(input())]
print(brojevi)

lista_cifara = []
for x in brojevi:
    while x > 0:
        lista_cifara = lista_cifara + [x % 10]
        x = x // 10
print(lista_cifara)

skup_cifara = set(lista_cifara)
for x in skup_cifara:
    print(x," : ", lista_cifara.count(x))
```

```
3
121
23568
5
[121, 23568, 5]
[1, 2, 1, 8, 6, 5, 3, 2,
5]
1  :  2
2  :  2
3  :  1
5  :  2
6  :  1
8  :  1
>>>
```

Primer

Na programskom jeziku Python sastaviti funkciju koja proverava da li je zadata rečenica

- heterogram (nijedno slovo se ne pojavljuje više od jednom),
- pangram (sva slova se pojavljuju bar jednom).

Primer

```
def isHeterogram(sentence):
    letters = [char for char in sentence if char.isalpha()]
    return len(set(letters)) == len(letters)

def isPangram(sentence):
    alphabet = {chr(i) for i in range(ord("a"), ord("z")+1)}
    letters = {l.lower() for l in sentence if l.isalpha()}
    return len(letters) == len(alphabet)

sentence = input("Unesi rečenicu: ")
if isHeterogram(sentence):
    print("\"{}\"JESTE heterogram".format(sentence))
else:
    print("\"{}\"NIJE heterogram".format(sentence))

if isPangram(sentence):
    print("\"{}\"JESTE pangram".format(sentence))
else:
    print("\"{}\"NIJE pangram".format(sentence))
```