Python
(#4)

Nemanja Maček

Zanimljivost

- “Prirodno stanište Kraljevskog pitona su tropske šume i suve savane zapadne i centralne Afrike. Obitavaju i na tlu i na drveću. Ove zmije aktivne su u sumrak i tokom noći, dok se danju odmaraju i spavaju.” [1]

Slika preuzeta sa: https://www.xyzreptiles.com/product/baby-super-mystic-ball-python/

Sadržaj

- Rečnici
- Tuple
- Skupovi

Rečnici

Kako se pravi rečnik.

```
>>> # Prazan rečnik
>>> new_dict = dict()
>>> new_dict = {}
>>> print(new_dict)
{}
>>> # Rečnik sa parovima ključ-vrednost
>>> color = {"col1" : "Red", "col2" : "Green", "col3" : "Orange" }
>>> color
{'col2': 'Green', 'col3': 'Orange', 'col1': 'Red'}
```

Rečnici

Čitanje vrednosti na osnovu ključa.

- Vrednosti mogu da se čitaju i upotrebom funkcije get()

```
>>> dict = {1:20.5, 2:3.03, 3:23.22, 4:33.12}
>>> # Čitanje vrednosti na osnovu ključa
>>> dict[1]
20.5
>>> dict[3]
23.22
>>> # Upotrena ugrađene funkcije get()
>>> dict.get(1)
20.5
>>> dict.get(3)
23.22
```

Dodavanje para ključ-vrednost.

- Parovi takođe mogu da se dodaju upotrebom funkcije update()

```
>>> dic = {'pdy1':'DICTIONARY'}
>>> dic['pdy2'] = 'STRING'
>>> print(dic)
{'pdy1': 'DICTIONARY', 'pdy2': 'STRING'}
>>> # Upotrena ugrađene funkcije update()
>>> d = {0:10, 1:20}
>>> d.update({2:30})
>>> print(d)
{0: 10, 1: 20, 2: 30}
```

Rečnici

Rečnici i petlja for.

- Obratite pažnju na (1) uređeni par ključ, vrednost i (2) funkciju items().

```
d = {'Red': 1, 'Green': 2, 'Blue': 3}
for color_key, value in d.items():
     print(color_key, 'corresponds to ', d[color_key])
```

- Šta je izlaz ovog programa?

Rečnici

Rečnici i petlja for.

- Obratite pažnju na (1) uređeni par ključ, vrednost i (2) funkciju items().

```
d = {'Red': 1, 'Green': 2, 'Blue': 3}
for color_key, value in d.items():
     print(color_key, 'corresponds to ', d[color_key])
```

- Šta je izlaz ovog programa?

```
Green corresponds to  2
Red corresponds to  1
Blue corresponds to  3
```

Uklanjanje ključa iz rečnika.

- Koristi se naredba del.

```
myDict = {'a':1,'b':2,'c':3,'d':4}
if 'a' in myDict:
  del myDict['a']
print(myDict)
```

- Šta je izlaz ovog programa?

Rečnici

Uklanjanje ključa iz rečnika.

- Koristi se naredba del.

```
myDict = {'a':1,'b':2,'c':3,'d':4}
if 'a' in myDict:
  del myDict['a']
print(myDict)
```

- Šta je izlaz ovog programa?

```
{'d': 4, 'b': 2, 'c': 3}
```

Sortiranje rečnika po ključu.

- Koristi se ugrađena funkcija sorted().

```
color_dict = {'red':'#FF0000', 'green':'#008000',
              'black':'#000000', 'white':'#FFFFFF'}
for key in sorted(color_dict): print("%s: %s" % (key, color_dict[key]))
```

- Šta je izlaz ovog programa?

Sortiranje rečnika po ključu.

- Koristi se ugrađena funkcija sorted().

```
color_dict = {'red':'#FF0000', 'green':'#008000',
              'black':'#000000', 'white':'#FFFFFF'}
for key in sorted(color_dict): print("%s: %s" % (key, color_dict[key]))
```

- Šta je izlaz ovog programa?

```
black: #000000
green: #008000
red: #FF0000
white: #FFFFFF
```

Pronalaženje najmanje i najveće vrednosti u rečniku.

- Koriste se ugrađene funkcije min() i max().

```
my_dict = {'x':500, 'y':5874, 'z': 560}
key_max = max(my_dict.keys(), key=(lambda k: my_dict[k]))
key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))
print('Maximum Value: ',my_dict[key_max])
print('Minimum Value: ',my_dict[key_min])
```

- Šta je izlaz ovog programa?

Pronalaženje najmanje i najveće vrednosti u rečniku.

- Koriste se ugrađene funkcije min() i max().

```
my_dict = {'x':500, 'y':5874, 'z': 560}
key_max = max(my_dict.keys(), key=(lambda k: my_dict[k]))
key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))
print('Maximum Value: ',my_dict[key_max])
print('Minimum Value: ',my_dict[key_min])
```

- Šta je izlaz ovog programa?

```
Maximum Value:  5874
Minimum Value:  500
```

Rečnici

Konkatenacija dva rečnika.

- Kreira se prazan rečnik, prođe for petljom kroz listu rečnika i koristi ugrađena funcija update().

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

- Šta je izlaz ovog programa?

Konkatenacija dva rečnika.

- Kreira se prazan rečnik, prođe for petljom kroz listu rečnika i koristi ugrađena funcija update().

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

- Šta je izlaz ovog programa?

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

Kako da proverite da li rečnik sadrži određeni ključ?

- Koristi se ključna reč in.

```
fruits = {}
fruits["apple"] = 1
fruits["mango"] = 2
fruits["banana"] = 4
if "mango" in fruits: print("Has mango")
else: print("No mango")
if "orange" in fruits: print("Has orange")
else: print("No orange")
```

- Šta je izlaz ovog programa?

Kako da proverite da li rečnik sadrži određeni ključ?

- Koristi se ključna reč in.

```
fruits = {}
fruits["apple"] = 1
fruits["mango"] = 2
fruits["banana"] = 4
if "mango" in fruits: print("Has mango")
else: print("No mango")
if "orange" in fruits: print("Has orange")
else: print("No orange")
```

- Šta je izlaz ovog programa?

```
Has mango
No orange
```

Rečnici

Određivanje dužine rečnika.

- Koristi se ugrađena funkcija len().

```
fruits = {"mango": 2, "orange": 6}
print("Length:", len(fruits))
```

- Šta je izlaz ovog programa?

Rečnici

Određivanje dužine rečnika.

- Koristi se ugrađena funkcija len().

```
fruits = {"mango": 2, "orange": 6}
print("Length:", len(fruits))
```

- Šta je izlaz ovog programa?

```
Length: 2
```

Tuple

Kreiranje tuple-a.

```
>>> tuplex = () # prazan tuple
>>> tuplex = ('tuple', False, 3.2, 1) # tuple različitih tipovia podataka
>>> tuplex = 4, 7, 3, 8, 1 # tuple brojevam, notacija bez zagrada
>>> tuplex = 4, # tuple jednog elementa, notacija bez zagrada
>>> tuplex = tuple() # kreiranje praznog tuplea ugrađenom funkcijom tuple()
>>> tuplex = tuple([True, False]) # pretvaranje liste u tuple
>>> print (tuplex)
(True, False)
```

Tuple

Kako se čita element tuple-a?

```
>>> tuplex = ("w", 3, "r", "e", "s", "o", "u", "r", "c", "e")
>>> print(tuplex)
('w', 3, 'r', 'e', 's', 'o', 'u', 'r', 'c', 'e')
>>> # pročitaj element (četvrti) na osnovu indeksa
>>> item = tuplex[3]
>>> print(item)
e
>>> # pročitaj element (četvrti s kraja) na osnovu negativnog indeksa
>>> item1 = tuplex[-4]
>>> print(item1)
u
```

Tuple

Provera da li element postoji u tuple-u?

- Koristi se ključna reč in.

```
>>> tuplex = ("w", 3, "r", "e", "s", "o", "u", "r", "c", "e")
>>> print(tuplex)
('w', 3, 'r', 'e', 's', 'o', 'u', 'r', 'c', 'e')
>>> print("r" in tuplex)
True
>>> print(5 in tuplex)
False
```

Tuple

Pretvaranje liste u tuple.

- Lista se prenosi kao parameter ugrađenoj funkciji tuple().

```
>>> listx = [5, 10, 7, 4, 15, 3]
>>> print(listx)
[5, 10, 7, 4, 15, 3]
>>> tuplex = tuple(listx)
>>> print(tuplex)
(5, 10, 7, 4, 15, 3)
```

Tuple

“Raspakivanje” tuple-a u nekoliko promenljivih.

- Broj promenljivih mora biti jednak broju elemenata tuple-a.

```
>>> tuplex = 4, 8, 3
>>> print(tuplex)
(4, 8, 3)
>>> n1, n2, n3 = tuplex
>>> print(n1 + n2 + n3)
15
>>> n1, n2, n3, n4 = tuplex
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: need more than 3 values to unpack
>>>
```

Tuple

Dodavanje elemenata.

- Elemente tuple-a ne možete menjati, ali možete da dodate nove.

```
>>> tuplex = (4, 6, 2, 8, 3, 1)
>>> tuplex = tuplex + (9,)
>>> print(tuplex)
(4, 6, 2, 8, 3, 1, 9)
>>> tuplex = tuplex[:5] + (15, 20, 25) + tuplex[:5]
>>> print(tuplex)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3)
>>> listx = list(tuplex)
>>> listx.append(30)
>>> tuplex = tuple(listx)
>>> print(tuplex)
(4, 6, 2, 8, 3, 15, 20, 25, 4, 6, 2, 8, 3, 30)
```

Tuple

Kloniranje.

- Koristi se funkcija deepcopy(). Obratite pažnju da dodavanje elementa klonu utiče na original.

```
>>> from copy import deepcopy
>>> tuplex = ("HELLO", 5, [], True)
>>> print(tuplex)
('HELLO', 5, [], True)
>>> tuplex_clone = deepcopy(tuplex)
>>> tuplex_clone[2].append(50)
>>> print(tuplex_clone)
('HELLO', 5, [50], True)
>>> print(tuplex)
('HELLO', 5, [], True)
>>>
```

Tuple

Kako možete da odredite koliko se puta element ponovio?

- Koristi se metoda count().

```
>>> tuplex = 2, 4, 5, 6, 2, 3, 4, 4, 7
>>> print(tuplex)
(2, 4, 5, 6, 2, 3, 4, 4, 7)
>>> count = tuplex.count(4)
>>> print(count)
3
>>> count = tuplex.count(7)
>>> print(count)
1
>>> count = tuplex.count(5)
>>> print (count)
1
```

Seckanje tuple-a.

```
>>> tuplex = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
>>> _slice = tuplex[3:5]
>>> print(_slice)
(5, 4)
>>> _slice = tuplex[:6]
>>> print(_slice)
(2, 4, 3, 5, 4, 6)
>>> _slice = tuplex[5:]
>>> print(_slice)
(6, 7, 8, 6, 1)
>>> _slice = tuplex[-8:-4]
>>> print(_slice)
(3, 5, 4, 6)
```

Seckanje tuple-a i upotreba step parametra.

```
>>> tuplex = tuple("HELLO WORLD")
>>> print(tuplex)
('H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D')
>>> _slice = tuplex[2:9:2] # tuple[start:stop:step]
>>> print(_slice)
('L', 'O', 'W', 'R')
>>> _slice = tuplex[::4] >>> # vraća svaki četvrti element.
>>> print(_slice)
('H', 'O', 'R')
>>> _slice = tuplex[9:2:-3] # ako je step negativan, ide se unazad
>>> print(_slice)
('L', 'W', 'L')
```

Tuple

Uklanjanje elementa.

- Ne možete direktno da uklonite element!

```
>>> tuplex = "w", 3, "d", "r", "e", "s", "l"
# Način 1 – konkatenacija iseckanih tuplea.
# Hoću da uklonim treći element.
>>> tuplex = tuplex[:2] + tuplex[3:]
>>> print(tuplex)
('w', 3, 'r', 'e', 's', 'l')
# Način 2 – pretvaranje u listu, uklanjanje elementa, pretvaranje u tuple.
# Hoću da uklonim element “l”.
>>> listx = list(tuplex)
>>> listx.remove("l")
>>> tuplex = tuple(listx)
>>> print(tuplex)
('w', 3, 'r', 'e', 's')
```

Pronalaženje indeksa elementa.

```
>>> tuplex = tuple("index tuple")
>>> print(tuplex)
('i', 'n', 'd', 'e', 'x', ' ', 't', 'u', 'p', 'l', 'e')
>>> index = tuplex.index("p")
>>> print(index)
8
>>> index = tuplex.index("p", 5) # počinje pretragu od šestog elementa
>>> print(index)
8
>>> index = tuplex.index("e", 3, 6) # definisanje opsega pretrage
>>> print(index)
3
```

Tuple

Određivanje veličine tuple-a.

- Koristi se funkcija len().

```
>>> tuplex = tuple("w3resource")
>>> print(tuplex)
('w', '3', 'r', 'e', 's', 'o', 'u', 'r', 'c', 'e')
>>> print(len(tuplex))
10
```

Tuple

Kako se ponašaju operatori + i *?

```
>>> tuplex = 5,
>>> print(tuplex * 6)
(5, 5, 5, 5, 5, 5)
>>> tuplex = (5, 10, 15) * 4
>>> print(tuplex)
(5, 10, 15, 5, 10, 15, 5, 10, 15, 5, 10, 15)
>>> tuplex1 = (3, 6, 9, 12, 15)
>>> tuplex2 = ("w", 3, "r", "s", "o", "u", "r", "c", "e")
>>> tuplex3 = (True, False)
>>> tuplex = tuplex1 + tuplex2 + tuplex3
>>> print(tuplex)
(3, 6, 9, 12, 15, 'w', 3, 'r', 's', 'o', 'u', 'r', 'c', 'e', True, False)
```

Skupovi

Kreiranje skupa.

```
>>> >>> setx = set() # prazan skup
>>> print(setx)
set()
>>> >>> n = set([0, 1, 2, 3, 4, 5]) # neprazan skup
>>> print(n)
{0, 1, 2, 3, 4, 5}
```

Prolaz kroz skup petljom for.

- Koristi se ključna reč in.

```
>>> num_set = set([0, 1, 2, 3, 4, 5])
>>> for n in num_set: print(n)
0
1
2
3
4
5
```

Dodavanje elementa skupu.

- Koristi se metode add() i update().

```
>>> color_set = set() # kreiranje praznog skupa
>>> color_set.add("Red") # dodavanje jednog elementa
>>> print(color_set)
{'Red'}
>>> color_set.update(["Blue", "Green"]) # dodavanje više elemenata
>>> print(color_set)
{'Red', 'Blue', 'Green'}
```

Uklanjanje elemenata skupa.

- Funkcija pop().

```
>>> num_set = set([0, 1, 2, 3, 4, 5])
>>> num_set.pop()
0
>>> print(num_set)
{1, 2, 3, 4, 5}
>>> num_set.pop()
1
>>> print(num_set)
{2, 3, 4, 5}
>>>
```

Uklanjanje elemenata skupa.

- Funkcija remove().

```
>>> num_set = set([0, 1, 2, 3, 4, 5])
>>> num_set.remove(0)
>>> print(num_set)
{1, 2, 3, 4, 5}
>>>
```

Uklanjanje elemenata skupa.

- Funkcija discard().

```
>>> num_set = set([0, 1, 2, 3, 4, 5])
>>> num_set.discard(3)
>>> print(num_set)
{0, 1, 2, 4, 5}
>>>
```

Presek skupova.

- Koristi se operator &

```
>>> setx = set(["green", "blue"])
>>> sety = set(["blue", "yellow"])
>>> setz = setx & sety
>>> print(setz)
{'blue'}
>>>
```

Unija skupova.

- Koristi se operator |

```
>>> setx = set(["green", "blue"])
>>> sety = set(["blue", "yellow"])
>>> seta = setx | sety
>>> print (seta)
{'yellow', 'blue', 'green'}
>>>
```

Razlika skupova.

- Koristi se operator -

```
>>> setx = set(["green", "blue"])
>>> sety = set(["blue", "yellow"])
>>> setz = setx & sety
>>> print(setz)
{'blue'}
>>> >>> setb = setx - setz
>>> print(setb)
{'green'}
>>>
```

Hvala na pažnji

Pitanja su dobrodošla.

Literatura

1. http://moj.pet-centar.rs/Teraristika/5404-Kraljevski-piton.html
2. Milan Bjelica (2016): Programski jezik Python. Dostupno u sekciji “download” na sledećem linku: http://www.etf.bg.ac.rs/etf_files/udzbenici/python.pdf
3. Charles R. Severance (2016): Python for Everybody – Exploring Data Using Python 3. Dostupno u sekciji “download” na stranici predmata i na sledećem linku: http://do1.dr-chuck.com/pythonlearn/EN_us/pythonlearn.pdf
4. A Bite of Python. Dostupno u sekciji “download” na sledećem linku (na engleskom jeziku): https://python.swaroopch.com/.
5. https://www.w3resource.com/python/python-tutorial.php