Napredno programiranje i programski jezici

11 Python

Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

Složeni objekti, kolekcije/sekvence

- n-torke (tuple)
- liste (list)
- skupovi (set)
- rečnik (dictionary)

```
tacka = (3, 5)
print(tacka)

student = ("Petar", "Petrovic")
print(student)

student = ("Petar", "Petrovic", 123456)
print(student)
```

```
(3, 5)
('Petar', 'Petrovic')
('Petar', 'Petrovic', 123456)
```

n-torka (tuple) - uređeni, nepromenljivi, skup objekata
nepromenljiva sekvenca

```
student = ("Petar", "Petrovic", 123456)

print(student[0])
print(student[0], student[1], student[2])
print(len(student))
```

```
Petar
Petar Petrovic 123456
3
```

```
student = ("Petar", "Petrovic", 123456)

for x in student:
    print(x)



data = (1, 15, -2, 7, 8)

if (15 in data):
    print("tu je 15")
else:
    print("nema 15")
```

```
Petar
Petrovic
123456
tu je 15
```

```
data = (1, 15, -2, 7, 8)
print(min(data), max(data))

results = (min(data), max(data))
print(results)



student = ("Petar", "Petrovic", 123456)
print(min(student), max(student))
```

```
-2 15
(-2, 15)
```

min/max za stringove?

```
student = ("Petar", "Petrovic", 123456)
bri = (42, 2020)
student += bri

print(student)



data = (1, 15, -2, 7, 8)
data += (26,)
#data += 26,

print(data)
```

```
('Petar', 'Petrovic', 123456, 42, 2020)
(1, 15, -2, 7, 8, 26)
```

```
student = ("Petar", "Petrovic", 123456)

bri = (42, 2020)
student += bri

~~student[3] = 45~~
```

```
('Petar', 'Petrovic', 123456, 42, 2020)
```

ne mogu se menjati elementi

```
student = ("Petar", "Petrovic", 123456)
bri = (42, 2020)
student += bri
student[3] = 45


point = (3, 5)
point = (point[0] + 1, point[1])
```

```
('Petar', 'Petrovic', 123456, 42, 2020)
(4, 5)
```

Napravljen je novi objekat, staro ime je sad povezano sa novim objektom

```
x = ()
print(x)

x = (1)
print(x)

x = 1
print(x)

x = (1, )
print(x)

x = (1, 2, 3)
print(x)

x = 1, 2, 3
print(x)

x = 1,
print(x)
```

```
()
1
1
(1,)
(1, 2, 3)
(1, 2, 3)
(1,)
```

probati: type, len.
Kako izgleda ntorka čiji elementi su ntorke?
Kako izgleda tuple čiji je jedini element prazan tuple?

```
t = (3, 5)
print(t)

(x, y) = (3, 5)
print(x, y)
print(x)
print(y)

print((x, y))

(x, y) = t
print(x, y)
```

```
(3, 5)
3 5
3
5

(3, 5)
3 5
```

tuple packing - dodela više vrednosti jednoj n-torci
tuple unpacking - dodela vrednosti n-torke različitim imenima
(umesto pisanja više dodela)

```
t = (3, 5)
print(t)

(x, y) = (3, 5)
print(x, y)

x, y = (3, 5)
print(x, y)

x, y = 3, 5
print(x, y)
```

```
(3, 5)
3 5
3 5
```

swap

```
x = 3
y = 5
print(x, y)

x = y
y = x
print(x, y)
```

```
3 5
5 5
```

```
x = 3
y = 5
print(x, y)

temp = (x, y)
#(x, y) = (temp[1], temp[0])
x, y = (temp[1], temp[0])
print(x, y)
```

```
3 5
5 3
```

```
x = 3
y = 5
print(x, y)

temp = (x, y)
x = temp[1]
y = temp[0]
print(x, y)
```

```
3 5
5 3
```

```
x = 3
y = 5
print(x, y)

# (x, y) = (y, x)
x, y = (y, x)
print(x, y)
```

```
3 5
5 3
```

```
def kolicnik_i_ostatak(x, y):
    kolicnik = x // y
    ostatak = x % y
    return (kolicnik, ostatak)

rez = kolicnik_i_ostatak(5,3)
print(rez)
```

```
(1, 2)
```

Možemo iskoristiti za vraćanje više vrednosti iz funkcije

```python
def kolicnik_i_ostatak(x, y):
    kolicnik = x // y
    ostatak = x % y
    return (kolicnik, ostatak)

q, r = kolicnik_i_ostatak(5, 3)
print(q, r)
```

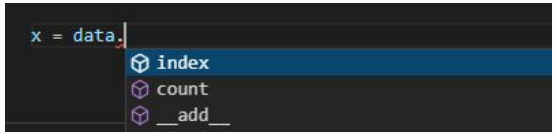
```
(1, 2)
1 2
```

```
data = (1, 15, -2, 7, 8)

x = data[3:5]
print(x)
```

```
(7, 8)
```

```
data = (1, 15, -2, 7, 8, -2)

x = data.count(-2)
print(x)

x = data.index(-2)
print(x)

x = str(data)
print(x)

print(type(data))
print(type(x))
```

```
2
2
(1, 15, -2, 7, 8, -2)
<class 'tuple'>
<class 'str'>
```



metode (index, count)

ugrađene funkcije (str, min, max,...)

```
data = (1, 15, -2, 7, 8, -2)

data = [1, 15, -2, 7, 8, -2]
print(data)
```

```
[1, 15, -2, 7, 8, -2]
```

lista - uređeni, promenljivi, skup objekata
promenljiva sekvenca

```
data = [1, 15, -2, 7, 8, -2]
print(data)

data[2] = 100
print(data)
```

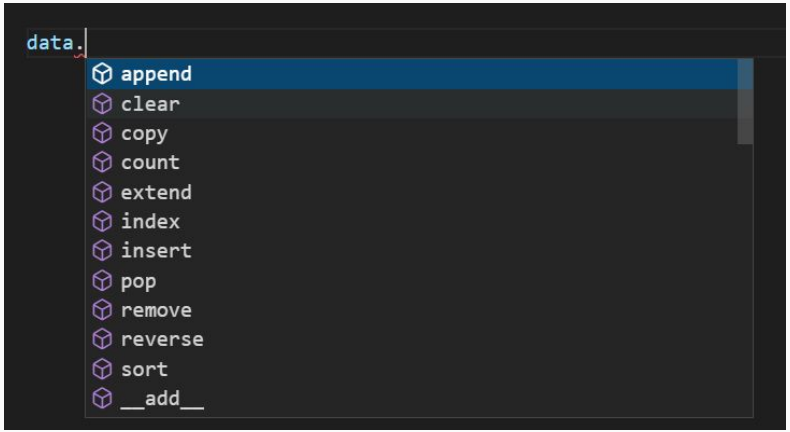
```
[1, 15, -2, 7, 8, -2]
[1, 15, 100, 7, 8, -2]
```

```
data1 = [1, 15, -2, 7, 8, -2]
data2 = data1
data2[3] += 100
print(data1)
print(data2)

data1 = [1, 15, -2, 7, 8, -2]
data3 = data1[:]
data3[3] += 100
print(data1)
print(data3)
```

```
[1, 15, -2, 107, 8, -2]
[1, 15, -2, 107, 8, -2]
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 107, 8, -2]
```

```python
data = [1, 15, -2, 7, 8, -2]
print(data)

data.append(0)
print(data)


print(min(data))
```

```
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2, 0]
-2
```



```
data = [1, 15, -2, 7, 8, -2]
print(data)

data.extend([100, 200])
print(data)

data += [300, 400]
print(data)
```

```
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2, 100, 200]
[1, 15, -2, 7, 8, -2, 100, 200, 300, 400]
```

```
data = [1, 15, -2, 7, 8, -2]
print(data)

s = str(data)
print(s)

s = "Hello world"
s1 = list(s)
print(s1)

s2 = s.split(" ")
print(s2)
```

```
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2]
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r',
'l', 'd']
['Hello', 'world']
```

```
data1 = [1, 15, -2, 7, 8, -2]
print(data1)
data1.sort()
print(data1)


data1 = [1, 15, -2, 7, 8, -2]
print(data1)
data2 = data1.sort()
print(data1)
print(data2)


data1 = [1, 15, -2, 7, 8, -2]
print(data1)
sorted(data1)
print(data1)


data1 = [1, 15, -2, 7, 8, -2]
print(data1)
data2 = sorted(data1)
print(data1)
print(data2)
```

```
[1, 15, -2, 7, 8, -2]
[-2, -2, 1, 7, 8, 15]
[1, 15, -2, 7, 8, -2]
[-2, -2, 1, 7, 8, 15]
None
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2]
[1, 15, -2, 7, 8, -2]
[-2, -2, 1, 7, 8, 15]
```