Napredno programiranje i programski jezici

11 Python

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

```
def foo(a) :
    a += 5

x = 3
foo(x)
print(x)

a = 2
foo(a)
print(a)
```

```
3
2
```

U nastavku - prosleđivanje parametara, detaljnije

```
n = 1

def foo() :
    n = 5

foo()
print(n)
```

```
1
```

```
n = 1

def foo() :
    global n
    n = 5

foo()
print(n)
```

```
5
```

```
def foo(a):
    if (type(a) == str):
        print(a.upper())
    elif (type(a) == int):
        print(a + 5)

foo('dobar dan')
foo(3)
```

```
DOBAR DAN
8
```

```
def foo(a):
    if (type(a) == str):
        print(a.upper())
    elif (type(a) == int):
        print(a + 5)

foo('dobar dan')
foo(3)
```

```
DOBAR DAN
8
```

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

print(foo('dobar dan'))
print(foo(3))
```

Možemo vratiti objekat bilo kog tipa

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

xf = foo
```

sve je objekat

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

xf = foo
print(xf)


print(foo)
```

```
<function foo at 0x7f2ebd664dc0>
```

Sve je objekat

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

xf = foo
print(xf)

xf_call = foo(5)
print(xf_call)
```

```
<function foo at 0x7f2ebd664dc0>
10
```

Funkcija vs poziv funkcije

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

def just_print(arg) :
    print(arg)

just_print(foo)
```

```
<function foo at 0x7f2ebd664dc0>
```

Prosleđujemo funkciju kao argument

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

def just_print(arg) :
    print(arg)

just_print(foo)
just_print(foo(3))
```

```
<function foo at 0x7f2ebd664dc0>
8
```

Prosleđujemo funkciju kao argument
(arg je objekat: funkcija foo)

Prosleđujemo povratnu vrednost funkcije
(arg je objekat: int - povratna vr funkcije foo)

```
def foo(a):
    if (type(a) == str):
        return a.upper()
    elif (type(a) == int):
        return a + 5

def just_print(arg) :
    print(arg)

def just_run(arg) :
    return arg(5)

def run_and_print(arg) :
   temp = arg(5)
   print(temp)
   return temp

just_print(foo)
print(just_run(foo))
run_and_print(foo)
print(run_and_print(foo) + 1)
```

```
<function foo at 0x7f2ebd664dc0>
10
10
10
11
```

```
def zbir(a, b):
    return a + b

def razlika(a, b):
    return a - b

def izaberi_operaciju():
    op = input("Unesite + za sabiranje ili - za oduzimanje")
    if (op == '+'):
        return zbir
    else:
        return razlika

fun = izaberi_operaciju()
print(fun(3, 5))
```

```
Unesite + za sabiranje ili - za
oduzimanje
+
8
```

```
Unesite + za sabiranje ili - za
oduzimanje
-
-2
```

Funkcija kao povratna vrednost druge funkcije

```
def fun1(n):
    return n

def fun2(n):
    return n * n

def fun3(n):
    return (n + 1)**3

def sum(n, fun):
    sum = 0
    i = 0
    while (i <= n):
        sum += fun(i)
        i += 1
    return sum

print(sum(4, fun1))
print(sum(4, fun2))
print(sum(4, fun3))
```

```
10
20
225
```

```
def sum(n):
    sum = 0
    i = 0
    while (i <= n):
        sum += i
        i += 1
    return sum

def sum(n, fun):
    sum = 0
    i = 0
    while (i <= n):
        sum += fun(i)
        i += 1
    return sum
```

Nema preklapanja imena funkcija

Poslednja defincija je pridružena imenu

Funkcije višeg reda / Higher order functions

- funkcije su objekti
- ravnopravne su sa ostalim objektima
- mogu se prosleđivati kao argumenti
- mogu biti povratna vrednost
- mogu biti deo složenijih objekata