Python – lekce 9

Podprogramy (Funkce)

Jazyk Python v podstatě žádné podprogramy nezná. Vše jsou funkce, všechny funkce vracejí hodnotu (i když někdy je to None).
Výhody funkcí:

1) Přehlednost
2) Opakované použití - ušetříme psaní

Deklarace funkce začíná klíčovým slovem **def**. Následuje jméno funkce a v závorce pak argumenty. Více argumentů se odděluje čárkami.

Syntaxe:

```
def Název (parametr1,parametr2,...):
    blok příkazů
    return výsledný výraz
```

Pozn. funkce nedefinuje typ návratové hodnoty. Funkce v jazyce Python neurčují datový typ návratové hodnoty. Neurčují dokonce ani to, jestli vracejí hodnotu nebo ne. (Ve skutečnosti každá pythonovská funkce vrací hodnotu. Pokud funkce provede příkaz return, vrátí v něm uvedenou hodnotu. V ostatních případech vrací None, což je pythonovský ekvivalent hodnoty null, nil, nic, žádná hodnota.)

Počet parametrů:

a) žádné	def Fce()
b) konkrétní počet	def Fce(x,y,z)
c) proměnlivý počet	def Fce(x,*param)

```
def Pozdrav():
    print("Dobré ráno!")
Pozdrav()       #volání funkce

def Pozdravy(pocet):
    for i in range(pocet):
        print("Buenos dias!")
Pozdravy(3)     #volání funkce

def SuperPozdrav(pocet,text):
    for i in range(pocet):
        print (text)

SuperPozdrav(5,'Guten Tag!')            #volání funkce
SuperPozdrav(4," Good morning")
```

Implicitní hodnoty parametrů

V definici funkce může být do parametru předána implicitní hodnota, která se použije v případě, že tento parametr nebude dosazen.
Vpravo od implicitního parametru mohou být již jen implicitní parametry. Při volání funkce se objekty dosazují nejdříve do parametrů bez implicitních hodnot a poté do parametrů s implicitními hodnotami (postupuje se zleva doprava). Z toho je zřejmé, že chcete-li dosadit za implicitní parametr, musíte dosadit za všechny parametry vlevo od něj, nebo použít použít jména parametrů.

```
def mocnina(x,y=2):
    return x**y

print (mocnina(2))
print (mocnina(2,3))
```

Jména parametrů

Nechcete-li přemýšlet, v jakém pořadí jsou zapsány parametry v definici funkce, můžete je volat jejich jmény. Tato možnost se také hodí pro implicitní parametry, chcete-li přiřadit hodnotu implicitnímu parametru a přitom nepřiřazovat hodnoty všem implicitním parametrům vlevo od něj.

```
def funkce(a,b,c='c',d='d',e=None,f=''):
    print (a,b,c,d,e)

funkce(b='b',a='A',e='E')
```

Proměnlivý počet parametrů

Pokud nevíte dopředu, kolik předáte funkci argumentů, můžete použít parametr s hvězdičkou. Ten bude obsahovat všechny argumenty, které bude funkce obsahovat "navíc" oproti obyčejným a pojmenovaným argumentům. Musí být uveden až za všemi obyčejnými a pojmenovanými parametry.

```
def moje_fce(a,b,c=5,*ostatni):
    print(a,b,c)
    print(ostatní)

moje_fce(3,4)
 moje_fce(3,4,5)
 moje_fce(3,4,5,6,7)
```

Globální a lokální proměnné

- proměnné definované uvnitř funkcí, jsou tzv. lokální = po skončení funkce přestanou existovat
- global x = definice globální proměnné x (existuje po celou dobu trvání programu)
- pokud chceme uvnitř funkce zapisovat do globální proměnné, musíme před ni napsat také global

```
def funkce1():
    x=10                    #vznik lokální proměnné
    print( x)               #tisk lok. proměnné
funkce1()
#print(x)                   #pokus o tisk lok. proměnné, která neexistuje


z=1                         #vznik globální proměnné z
def funkce2():
    z=10                    #vznik lokální proměnné z!!!!!!
    print(z)                #tisk lok. proměnné
funkce2()
#print(z)                   #tisk globální proměnné


z=1                         #vznik globální proměnné z
def funkce3():
    global z
    print( z)
funkce3()
print (z)
```