Funkcje

Deklaracja funkcji wykonywana jest słówkiem ‘def’. Funkcje mogą być z argumentami lub bez nich:

PythonFunkcje.py

```
1  #funkcja bez argumentu
2  def wyraz():
3      return "Brawo!!!"
4  print(wyraz())
5
6  #funkcja z argumentem
7  def kwadrat(a):
8      return a**2
9  print(kwadrat(5))
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Sh
Brawo!!!
25
Press any key to continue . . .
```

Nadawanie wartości argumentom w definicji funkcji (wartości domyślne):

PythonFunkcje.py — funkcja1

```
1  #funkcja bez argumentu
2  def wyraz():
3      return "Brawo!!!"
4  print(wyraz())
5
6  #funkcja z argumentem
7  def kwadrat(a):
8      return a**2
9  print(kwadrat(5))
10
11 #funkcja z domyślnymi wartościami argumentów
12 def funkcja1(a,b='O', c='X'):
13     return a,b,c
14 #print(funkcja1()) #funkcja musi mieć minimum 1 argument
15 print(funkcja1('Ala'))
16 print(funkcja1('Ala',' ma '))
17 print(funkcja1('Ala', 'ma ', 'kota'))
```

```
C:\Program Files (x86)\Microsoft V
Brawo!!!
25
('Ala', 'O', 'X')
('Ala', ' ma ', 'X')
('Ala', 'ma ', 'kota')
Press any key to continue
```

Funkcję można wywołać podając konkretny argument:

PythonFunkcje.py — kwadrat

```
1  #funkcja bez argumentu
2  '''def wyraz():...
10 #funkcja z domyślnymi wartościami argumentów
11 def funkcja1(a,b='O', c='X'):
12     return a,b,c
13 #print(funkcja1()) #funkcja musi mieć minimum 1 argument
14 print(funkcja1('Ala'))
15 print(funkcja1('Ala',' ma '))
16 print(funkcja1('Ala', 'ma ', 'kota'))
17 print(funkcja1('Ala', c='kota'))
18 print(funkcja1(c='kota')) #ta linia nie zadziała
```

Funkcja nie wymagająca wywołania printem i nie wymagająca słówka ‘return’:

```
PythonFunkcje.py
1  #funkcja wywoływana bez słowka print
2  def funkcja2():
3      print("funkcja nie wymagaa 'return'")
4
5  funkcja2()
6
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Sh
funkcja nie wymagaa 'return'
Press any key to continue . . .
```

Wpisanie wyniku funkcji do zmiennej w celu użycia jej wyniku w innym miejscu lub wielokrotnie:

```
PythonFunkcje.py
1  #funkcja wywoływana bez słowka print
2  def funkcja3(a,b,c):
3      return a+b+c
4  d = funkcja3(2,3,5)
5  print(d)
6  print(d)
7  print(d)
```

```
C:\Program Files (x86)\Microsoft Visual Studi
10
10
10
Press any key to continue . . .
```

Przekształcenie argumentów funkcji w listę:

```
1  # argumenty funkcji przekształceone w listę
2  def f(*args):
3      print(args)
4  f(8,2,65,'abc')
```

```
C:\Program Files (x86)\Microsoft Visual Studio\
(8, 2, 65, 'abc')
Press any key to continue . . .
```

Stworzenie dokumentacji funkcji i jej wywołanie:

```
fun_doc
1  #dokumnetowanie funkcji do celów pokazu
2  def fun_doc(a):
3      '''To jest opis funkcji - kwadrat a '''
4      return a**2
5  print(fun_doc(2))
6  print(fun_doc.__doc__)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Sha
4
To jest opis funkcji - kwadrat a
Press any key to continue . . .
```

Dokumentacja funkcji wbudowanych:

```
fun_doc
1  #wywołanie dokumentacji dowolnej funkcji
2  print(abs.__doc__)
3  print(ascii.__doc__)
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python37_64\python.exe
Return the absolute value of the argument.
Return an ASCII-only representation of an object.

As repr(), return a string containing a printable representation of an
object, but escape the non-ASCII characters in the string returned by
repr() using \\x, \\u or \\U escapes. This generates a string similar
to that returned by repr() in Python 2.
Press any key to continue . . .
```

Zmienne globalne użyte w funkcjach:

```
#zmienne globalne
i = 2
def fun(x):
    return x**i
z=fun(2)
print(z)
```

```
C:\Program Files (x86)\Microsoft Visual Stu
4
Press any key to continue . . .
```

Przekazanie funkcji do zmiennej o innej nazwie:

```
#przekazanie funkcji
def fun(x):
    return x*x
zm1 = fun
print(zm1(8))
```

```
C:\Program Files (x86)\Microsoft Visual Stu
64
Press any key to continue . . .
```

PRZYKŁAD UŻYCIA FUNKCJI DO STWORZENIA REKURENCJI

PythonFunkcje.py

```
#przekazanie funkcji
x=5
def fun(x):
    return x*x
zm1 = fun
print(zm1(x))
# użycie funkcji przez inną funkcję
def fun2(fun1, x):
    return fun1(x) * x
print(fun2(fun,x))
#użycie funkcji przez samą siebie - rekurencja
def silnia(x):
    if x<=1:
        return 1
    else:
        return x * silnia(x-1)
print(silnia(0))
print(silnia(1))
print(silnia(2))
print(silnia(3))
print(silnia(x))
```

```
C:\Program Files (x86)\Microsoft Visual Stu
25
125
1
1
2
6
120
Press any key to continue . . .
```

Funkcja tworząca zmienną globalną:

```
#zmienna globalna
def myfunc():
    global x
    x = 3
myfunc()
print (x**2)
```

```
C:\P
9
```