Python!

Lekcja 5 – funkcje

Plan

- Funkcja – zastosowanie
- Definiowanie funkcji
- return vs yield
- Przypisywanie wartości funkcji do zmiennych
- Parametryzacja funkcji
- Wartości domyślne parametrów
- Parametry nazwane (keyword arguments)

Funkcja – zastosowanie

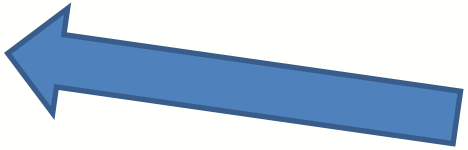
Funkcja (podprogram, procedura) – wydzielona część programu, która wykonuje konkretne zadanie. Użycie funkcji zwiększa przejrzystość kodu i pozwala na jego powtórne wykorzystanie. Ułatwia również późniejszą edycję.

Przy ‘użyciu’ funkcji mówi się o jej *wywołaniu*

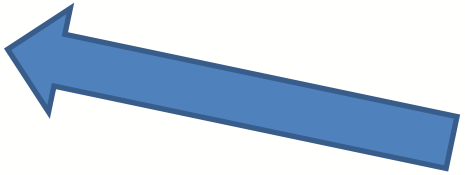
Definiowanie funkcji

```
>>> def mojafunkcja():
...     print("Działa")
...
>>> mojafunkcja()
Działa
>>>
>>> a = mojafunkcja()
Działa
>>> type(a)
<type 'NoneType'>
>>>
```

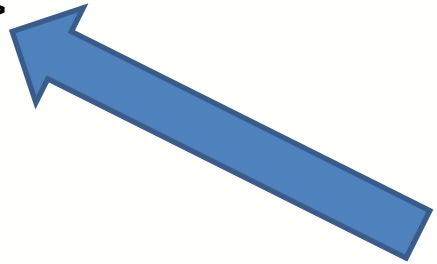
definiowanie funkcji



wywołanie funkcji



przypisanie wyniku funkcji do zmiennej



coś poszło nie tak

return vs yield

return – instrukcja pozawalająca funkcji zwrócić wynik operacji zawartych w funkcji. Mówi się o ‘zwracaniu wartości’. Dana funkcja może zawierać kilka takich instrukcji, jednak powinny być one zawarte w innych blokach kodu. W przypadku braku instrukcji *return* w funkcji, będzie ona zwracała wartość *None*

yeild – instrukcja, która ‘zwykłą’ funkcję przekształca w funkcje generującą, tzw. generator. Generatory upraszczają tworzenie iteratorów – obiektów, po których można iterować (np. poprzez wykorzystanie w pętli for).

return vs yield

```
>>> def funkcjaZreturn():
...     print("Yesh!")
...     return "Zwróciłem Yesh!"
...
>>> funkcjaZreturn()
Yesh!
'Zwr\xa2ci\x88em Yesh!'
>>> a = funkcjaZreturn()
Yesh!
>>> type(a)
<type 'str'>
>>> print(a)
Zwróciłem Yesh!
>>>
```

return vs yield

```
>>> def funkcjaZyield():
...     yield 1
...     yield 2
...
>>> b = funkcjaZyield()
>>> type(b)
<type 'generator'>
>>>
>>> for i in b:
...     print(i)
...
1
2
```

Przypisywanie wartości funkcji do zmiennych

```
>>> def funkcjaZwracaDuzo():
...     return 1,2
...
>>> funkcjaZwracaDuzo()
(1, 2)
>>> a = funkcjaZwracaDuzo()
>>> a
(1, 2)
>>> a,b = funkcjaZwracaDuzo()
>>> a
1
>>> b
2
```

Parametryzacja funkcji

Parametryzacja funkcji pozwala zwiększyć jej uniwersalność. Parametry przekazane do funkcji mogą być dowolnego typu.

Wyróżnia się dwa rodzaje parametrów:

- parametry pozycyjne
- parametry nazwane

Parametry pozycyjne muszą znajdować się przed parametrami nazwanymi

Wartości domyślne parametrów

```
>>> def funkcjaZparametrami(a, b="Domyślna wartość"):
...     print(a)
...     print(b)
...
>>> funkcjaZparametrami(5)
5
Domyślna wartość
>>> funkcjaZparametrami(5, "Nowa wartość")
5
Nowa wartość
>>> funkcjaZparametrami(b=1,a=2)
2
1
>>>
```

Parametry nazwane (keyword arguments)

```
>>> def funkcjaZnazwanymi(**kwargs):
...     for k, v in kwargs.items():
...             print("klucz: {}, wartość:
{}".format(k, v))
...
>>> funkcjaZnazwanymi(a=20, b=30)
klucz: a, wartość: 20
klucz: b, wartość: 30
>>>
```

Parametry nazwane (keyword arguments)

```
>>> def funkcjaZmiksowana(a,
...     b = "Wartość domyślna",
...     **kwargs):
...     print(a)
...     print(b)
...     for k, v in kwargs.items():
...             print("klucz: {}, wartość: {}".format(k, v))
...
>>> funkcjaZmiksowana( 20, c = 30, d = 40)
20
Wartość domyślna
klucz: c, wartość: 30
klucz: d, wartość: 40
>>>
```

Dziękuję za uwagę!