

Funkcje

Definicja i wywołanie funkcji,
przekazywanie argumentów, zwracanie wartości

Funkcje w matematyce

Funkcja (łac. functio, -onis „odbywanie, wykonywanie, czynność”) – dla danych dwóch zbiorów X i Y przyporządkowanie każdemu elementowi zbioru X dokładnie jednego elementu zbioru Y .

Przykład:

$$f(x) = x + 1$$



$$y = f(4)$$

$$y = 4 + 1$$

$$y = 5$$

Zobacz też: <https://pl.wikipedia.org/wiki/Funkcja>



Funkcje

Definicja i wywołanie funkcji

Funkcja to wydzielona część programu wykonująca pewne operacje. Nazwa funkcji ma istotne znaczenie, określa ona czynność, którą dana funkcja wykonuje. Poprzez użycie odpowiedniej nazwy można wywołać konkretną funkcję, czyli wykonać kod w niej zapisany.

Dobrze napisana funkcja wykonuje tylko jedną czynność.

Zobacz też: <https://pl.wikipedia.org/wiki/Podprogram>



Funkcje

Definicja i wywołanie funkcji

Tworzenie funkcji w języku Python rozpoczynamy od słowa kluczowego `def`. Zaraz po nim następuje nazwa funkcji, nawiasy okrągłe oraz dwukropek.

Przypomnijmy: dwukropek na końcu linii oznacza, że w kolejnej linii następuje odpowiednio wcięty blok kodu złożony z co najmniej jednej linii.

```
def czesc():  
    print("Czesc!")
```

```
czesc()
```



Definicja i wywołanie funkcji

Każdy blok kodu powinien zawierać dokładnie cztery spacje. Jest to wymaganie określone w dokumencie **PEP 8**, do którego zaleceń stosują się programiści Pythona.

```
def czesc():  
    print("Czesc!")  
  
czesc()
```



Funkcje

Przekazywanie argumentów

Parametry funkcji umieszczane są w nawiasach okrągłych umieszczonych za nazwą funkcji.

Argumenty przekazujemy w nawiasach okrągłych podczas wywoływania funkcji.

```
def czesc(imie):  
    print("Czesc " + imie + "!")  
  
czesc("Janusz")
```

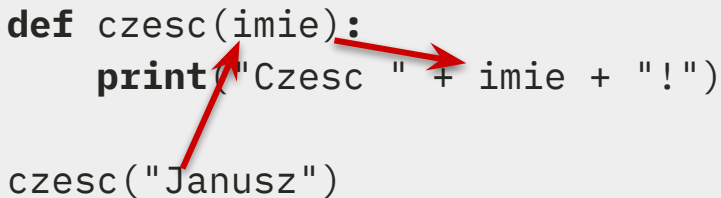


Przekazywanie argumentów

Parametry funkcji
umieszczane są w nawiasach
okrągłych umieszczonych
za nazwą funkcji.

Argumenty przekazujemy
w nawiasach okrągłych
podczas wywoływania
funkcji.

```
def czesc(imie):  
    print("Czesc " + imie + "!")  
  
czesc("Janusz")
```



Przekazywanie argumentów

Podczas przekazywania argumentu możemy również podać nazwę parametru.

```
def czesc(imie):  
    print("Czesc " + imie + "!")  
  
czesc(imie="Janusz")
```



Przekazywanie argumentów

Pytanie:

Założmy, że mamy funkcję przyjmującą dwa parametry: imie oraz miasto.

Czy korzystając z nazw parametrów możemy zmienić kolejność ich występowania?

```
def czesc(imie, miasto):  
    print("Czesc " + imie + "!")  
    print("Widze, ze jestes z miasta", miasto)  
  
czesc(miasto="Wroclaw", imie="Janusz")
```



Przekazywanie argumentów

Pytanie:

Założmy, że mamy funkcję przyjmującą dwa parametry imie oraz miasto.

Czy korzystając z nazw parametrów możemy zmienić kolejność ich występowania?

TAK!

```
def czesc(imie, miasto):  
    print("Czesc " + imie + "!")  
    print("Widze, ze jesteś z miasta", miasto)  
  
czesc(miasto="Wroclaw", imie="Janusz")
```



Funkcje

Przekazywanie argumentów

Pytanie 2:

Założmy, że mamy funkcję przyjmującą dwa parametry imie oraz miasto.

Czy program zadziała dobrze, gdy prześlemy argumenty ze zmienioną kolejnością nie podając nazw parametrów?

```
def czesc(imie, miasto):  
    print("Czesc " + imie + "!")  
    print("Widze, ze jesteś z miasta", miasto)  
  
czesc("Wroclaw", "Janusz")
```



Funkcje

Przekazywanie argumentów

Pytanie 2:

Założmy, że mamy funkcję przyjmującą dwa parametry imie oraz miasto.

Czy program zadziała dobrze, gdy prześlemy argumenty ze zmienioną kolejnością nie podając nazw parametrów?

NIE!

```
def czesc(imie, miasto):  
    print("Czesc " + imie + "!")  
    print("Widze, ze jesteś z miasta", miasto)  
  
czesc("Wroclaw", "Janusz")
```



Funkcje

Przekazywanie argumentów - Argumenty domyślne

Dla parametry funkcji można ustanowić pewne konkretne wartości, nazywamy je **argumentami domyślnymi**.

Argumenty domyślne pozwalają wywołać funkcję bez podawania jednego lub większej liczby argumentów.

```
def czesc(imie, miasto, komunikat="Czesc"):  
    print(komunikat, imie + "!")  
    print("Widze, ze jestes z miasta", miasto)
```

```
czesc("Janusz", "Wroclaw")  
czesc("Alicja", "Wroclaw", "Milego dnia")
```



Funkcje

Przekazywanie argumentów

Pytanie 3:

Czy argument domyślny
możemy przypisać
do parametru miasto?

Innymi słowy, czy dowolny
parametr bez podanego
argumentu domyślnego może
znajdować się za
parametrami z argumentami
domyślnymi?

```
def czesc(imie, miasto="Wroclaw", komunikat):  
    print(komunikat, imie + "!")  
    print("Widze, ze jesteś z miasta", miasto)  
  
czesc("Alicja", "Milego dnia")
```



Przekazywanie argumentów

Pytanie:

Czy argument domyślny możemy przypisać do parametru miasto?

Innymi słowy, czy dowolny parametr bez podanego argumentu domyślnego może znajdować się za parametrami z argumentami domyślnymi?

NIE!

```
def czesc(imie, miasto="Wroclaw", komunikat):  
    print(komunikat, imie + "!")  
    print("Widze, ze jesteś z miasta", miasto)  
  
czesc("Alicja", "Miłego dnia")
```



Funkcje

Zwracanie wartości

Z czasem stopień skomplikowania naszych funkcji rośnie, ich zadaniem będzie wykonanie pewnych obliczeń i zwrócenie wyniku.

Wartości z funkcji zwracane są przy pomocy słowa kluczowego `return`.

```
def dodawanie(a, b):  
    z = a + b  
    return z
```

```
wynik = dodawanie(2, 3)  
print("Wynik:", wynik)
```



Zwracanie wartości

Pytanie:

Co to znaczy zwrócić wartość z funkcji?

```
def dodawanie(a, b):  
    z = a + b  
    return z
```

```
wynik = dodawanie(2, 3)  
print("Wynik:", wynik)
```



Zwracanie wartości

Pytanie:

Co to znaczy zwrócić wartość z funkcji?

Bardzo ogólnie możemy powiedzieć, że zwrócenie wartości oznacza podstawienie obliczonego wyniku (wartości wskazanej słowem `return`) w miejsce wywołania funkcji.

```
def dodawanie(a, b):  
    z = a + b  
    return z
```

```
wynik = dodawanie(2, 3)  
print("Wynik:", wynik)
```



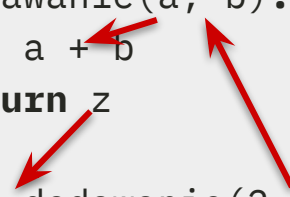
Zwracanie wartości

Pytanie:

Co to znaczy zwrócić wartość z funkcji?

Bardzo ogólnie możemy powiedzieć, że zwrócenie wartości oznacza podstawienie obliczonego wyniku (wartości wskazanej słowem `return`) w miejsce wywołania funkcji.

```
def dodawanie(a, b):  
    z = a + b  
    return z  
  
wynik = dodawanie(2, 3)  
print("Wynik:", wynik)
```

The diagram consists of three red arrows. One arrow points from the `return z` line in the function definition to the `dodawanie(2, 3)` call in the code below. A second arrow points from the `dodawanie(2, 3)` call to the `wynik` variable in the `print` statement. A third arrow points from the `z = a + b` line in the function definition to the `z` variable in the `return` statement.

Zwracanie wartości

Wartości zwracane z funkcji nie muszą być tylko liczbami. Funkcje mogą zwracać dowolne obiekty (np. listy, krotki, słowniki).

Pytanie:

Co robi kod po prawej stronie?

```
def produce_numbers(n):  
    i = 0  
    numbers = []  
    while i < n:  
        numbers.append(i)  
        i = i + 1  
    return numbers  
  
for i in produce_numbers(5):  
    print("Element:", i)
```



Przekazywanie argumentów - ciąg dalszy

Funkcje w języku Python mogą przyjmować dowolnie wiele argumentów. W tym celu został opracowany specjalny parametr `*args` przechowujący dodatkowe nienazwane argumenty przekazane do funkcji.

Nazwa `args` jest umowna.

```
def czesc(imie, *args):  
    print("Czesc " + imie + "!")  
    for s in args:  
        print("Czesc " + s + "! (args)")  
  
czesc("Janusz", "Maciej", "Mateusz")
```



Funkcje

Przekazywanie argumentów - ciąg dalszy

Dostępny jest także parametr `**kwargs` przechowujący dodatkowe **nazwane** argumenty przekazane do funkcji.

`kwargs` jest słownikiem, gdzie kluczem jest nazwa parametru, a wartością przekazany argument.

Nazwa `kwargs` jest umowna.

```
def czesc(imie, **kwargs):  
    nazwa = imie  
    if "nazwisko" in kwargs:  
        nazwa = nazwa + " " + kwargs["nazwisko"]  
    print("Czesc " + nazwa + "!")
```

```
czesc("Janusz")  
czesc("Anna", nazwisko='Nowak')
```



Pytania

1. Czym jest funkcja w Pythonie?
2. Ile spacji powinno zawierać wcięcie bloku kodu? Skąd pochodzi ta zasada?
3. Czy korzystając z nazw parametrów możemy zmienić kolejność ich występowania?
4. Czy dowolny parametr bez podanego argumentu domyślnego może znajdować się za parametrami z argumentami domyślnymi?
5. Czy istnieją techniczne przeszkody, by użyć własnych nazw zamiast nazw `args` i `kwargs`?



Podstawowe elementy języka

Literatura

1. Funkcje, <https://docs.python.org/3/tutorial/controlflow.html#defining-functions>
2. Więcej o funkcjach,
<https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions>



