Funkcje w języku Python

- Wprowadzenie
- Przeczytaj
- Audiobook
- Sprawdź się
- Dla nauczyciela

Jak w każdym języku programowania, również w Pythonie programista może tworzyć własne funkcje. Są to części programu wielokrotnego użytku. Funkcje pozwalają nadać nazwę blokowi wyrażeń, a następnie uruchamiać ten blok, używając jego nazwy w dowolnym miejscu w programie, potrzebną ilość razy.

Funkcje są kluczowym elementem każdego programu. Pozwalają uporządkować strukturę kodu, co skutkuje znacznym ułatwieniem wprowadzania zmian na szeroką skalę, ułatwia przeprowadzanie testów oraz poprawianie błędów.

> **Twoje cele**
>
> - Zaznajomisz się z pojęciem funkcji jako „czarnej skrzynki” (*black box*).
> - Poznasz sposoby definiowania funkcji w Python 3.
> - Stworzysz podstawową dokumentację (*docstring*) do funkcji w Python 3.

Przeczytaj

Funkcja w programowaniu to czarna skrzynka

Nie zawsze wiesz, co ona zawiera. Masz jednak klucz, możesz ją otworzyć i do niej zajrzeć. Możesz zawsze użyć skrzynki do pewnych zadań, które potrafi wykonać za pomocą tego, co zawiera w środku. Ale nie musisz dokładnie wiedzieć, jak to robi – wystarczy, że wiesz, jaki będzie ostateczny wynik działania skrzynki, i jak sprawić, aby to osiągnąć. Natomiast szczegóły pozostawiasz jej, czyli funkcji – owej czarnej skrzynce.

Jak jest zbudowana funkcja w Python 3?

Funkcja w Python 3 – definicja

```
def nazwa_funkcji(parametr_1, parametr_2):
    '''
    parametr_1 - int
    parametr_2 - int
    zwraca sumę parametrów - int
    '''
    wynik = parametr_1 + parametr_2
    return wynik
```

Funkcję w języku Python definiujesz po to, aby móc używać jej później w wielu miejscach w swoim programie. Zauważ, że w powyższym przykładzie funkcja zawiera dwa parametry: parametr_1 i parametr_2. Następnie w jej wnętrzu, a więc w kodzie, który jest wykonywany, oba te parametry są sumowane, a suma przypisywana jest do zmiennej `wynik`. Na końcu zmienna `wynik` jest zwracana przez słowo kluczowe `return`.

Inne sposoby definiowania parametrów funkcji

W Pythonie możesz przekazywać do funkcji określoną liczbę parametrów, które zdefiniujesz w funkcji. Takie parametry nazywa się **parametrami wymaganymi**.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> def nazwa_funkcji(parametr_1, parametr_2):
    '''
    parametr_1 - int
    parametr_2 - int
    zwraca sumę parametrów - int
    '''
    wynik = parametr_1 + parametr_2
    return wynik

>>> nazwa_funkcji(12, 45)
57
```

Podczas wywoływania funkcji trzeba podać wszystkie parametry, inaczej wykonanie funkcji zakończy się

błędem.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> def nazwa_funkcji(parametr_1, parametr_2):
    '''
    parametr_1 - int
    parametr_2 - int
    zwraca sumę parametrów - int
    '''
    wynik = parametr_1 + parametr_2
    return wynik

>>> nazwa_funkcji(12)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    nazwa_funkcji(12)
TypeError: nazwa_funkcji() missing 1 required positional argument: 'parametr_2'
>>>
```

Parametry mogą mieć wartości domyślne, które zostaną przyjęte, o ile nie podasz takiego parametru jawnie podczas wykonywania funkcji. Oto przykład zdefiniowania takiej wartości domyślnej.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> def nazwa_funkcji(parametr_1, parametr_2 = 10):
    '''
    parametr_1 - int
    parametr_2 - int (wartość domyślna 10)
    zwraca sumę parametrów - int
    '''
    wynik = parametr_1 + parametr_2
    return wynik

>>> nazwa_funkcji(2, 4)
6
>>> nazwa_funkcji(2)
12
>>>
```

Parametry domyślne możesz wykorzystywać również w inny sposób. Pozwalają one wywoływać funkcję w dosyć czytelny sposób, podając nazwy parametrów przy wywołaniu – oto przykład.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> def nazwa_funkcji(parametr_1 = 20, parametr_2 = 10):
    '''
    parametr_1 - int (wartość domyślna 20)
    parametr_2 - int (wartość domyślna 10)
```

```
    zwraca sumę parametrów - int (domyślnie 30)
    '''
    wynik = parametr_1 + parametr_2
    return wynik

>>> nazwa_funkcji()
30
>>> nazwa_funkcji(parametr_1=2, parametr_2=3)
5
>>> nazwa_funkcji(2, parametr_2=4)
6
>>> nazwa_funkcji(parametr_2=3)
23
>>>
```

W powyższym przykładzie możesz zaobserwować różne sposoby wywoływania funkcji, co w połączeniu z różnymi wartościami domyślnymi parametrów daje różne wyniki.

> **Ciekawostka**
>
> Istnieje wiele wbudowanych funkcji, które pozwolą ci wykonać często powtarzane zadania programistyczne, np. czytanie/zapisywanie plików. To wszystko, aby szybciej pisać bardziej skomplikowane programy.

Aby utworzyć nazwaną funkcję z własnym kodem, zastosuj słowo `def` – skrót od słowa „definiuj”.

Zapamiętaj kilka ciekawych cech funkcji w Python:

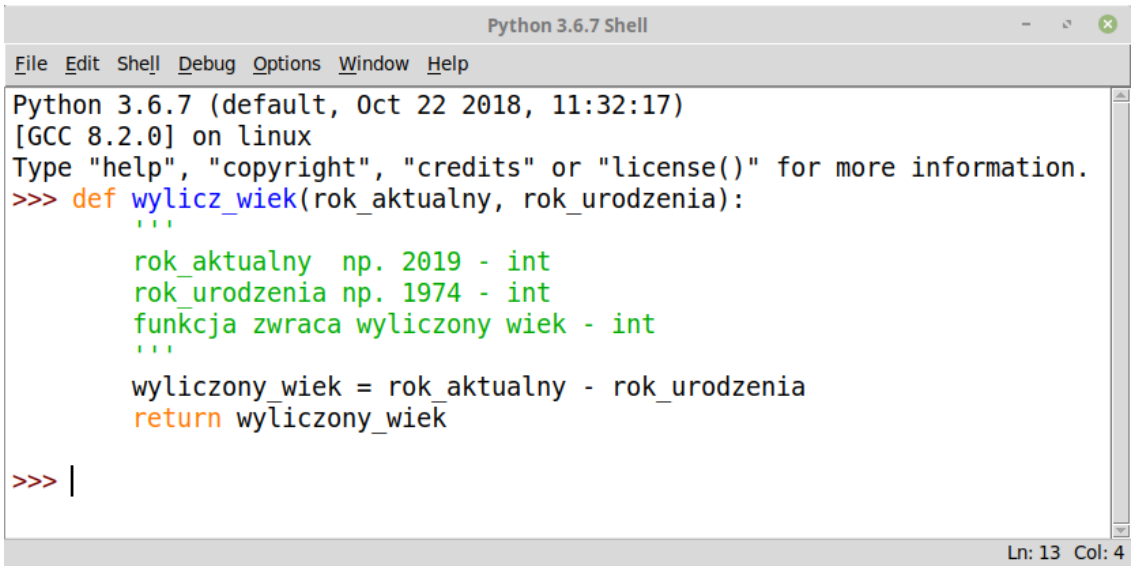
- można ustalać wartości domyślne wybranych argumentów,
- można wywoływać funkcję z użyciem nazw argumentów w dowolnej kolejności,
- jeśli funkcja nie zwraca jawnie żadnej wartości, wówczas zwraca typ `None`.

> **Ciekawostka**
>
> **W Pythonie** – w przeciwieństwie do takich języków jak C++, Java, Pascal – funkcja może być **argumentem jakieś funkcji**, może być **wynikiem (wartością zwracaną) funkcji**, można ją **podstawiać pod zmienne** itp.

Definiujemy własną funkcję

W tym ćwiczeniu zdefiniujesz własną funkcję, która na podstawie dwóch wartości (bieżącego roku i roku urodzenia) będzie obliczała wiek osoby. Aby ta funkcja była dostępna dla każdego, opiszesz jej sposób wykonywania (`docstring`).

```
def wylicz_wiek(rok_aktualny, rok_urodzenia):
 '''
 rok_aktualny  np. 2019 - int
 rok_urodzenia np. 1974 - int
 funkcja zwraca wyliczony wiek - int
 '''
 wyliczony_wiek = rok_aktualny - rok_urodzenia
 return wyliczony_wiek
```



Zwróć uwagę na kolejne elementy, które musisz zastosować:

- słowo kluczowe `def`,
- nazwę funkcji,
- parametry funkcji,
- opis funkcji – `docstring`,
- obliczenie wieku przez zastosowanie zmiennej wewnątrz funkcji i operatora odejmowania,
- słowo kluczowe `return`.

> **Ważne!**
>
> Jeżeli przy słowie kluczowym `return` nie stoi żadne wyrażenie inne niż `None` lub funkcja nie zawiera słowa kluczowego `return`, to w języku Python funkcja zawsze zwraca wartość `None`.

> **Ćwiczenie 1**
> W środowisku IDLE stwórz definicję funkcji, ktora będzie mnożyć dwa parametry, a następnie spróbuj ją wywołać dla różnych danych wejściowych, aby uzyskać różne wyniki.

Funkcja `print`

> **Polecenie 1**
> W środowisku Python 3 użyj funkcji `print` jak we fragmencie kodu poniżej. Zauważ, że system operacyjny nie ma tu większego znaczenia.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('To jest podstawowa funkcja - wypisuje na ekran')
To jest podstawowa funkcja - wypisuje na ekran
>>>
```

Ćwiczenie 2

Połącz w pary odpowiednie treści.

różni się – w Python 2 jest poleceniem, a nie funkcją., działa tak samo w Windows i w Linuksie., może przyjąć kilka argumentów/parametrów.

Funkcja `print` w Python 3 a w Python 2	
Funkcja `print` w języku Python 3	
Funkcja w Pythonie	

Ważne!

Przyjrzyj się dokładnie utworzonej funkcji `print` i jej parametrowi.

```
print('To jest podstawowa funkcja - wypisuje na ekran')
```

`print` – nazwa funkcji; w języku angielskim oznacza „drukuj", więc drukuje/wypisuje informacje tekstowe na ekranie, ale nie tylko na ekranie;

`'To jest podstawowa funkcja - wypisuje na ekran'` – argument, w postaci ciągu znaków, przekazywany do funkcji. Podprogramy mogą przyjmować różne parametry. Do funkcji print również można przekazać ich więcej. Aby dowiedzieć się więcej na temat możliwych operacji, warto zajrzeć do dokumentacji.

Ciekawostka

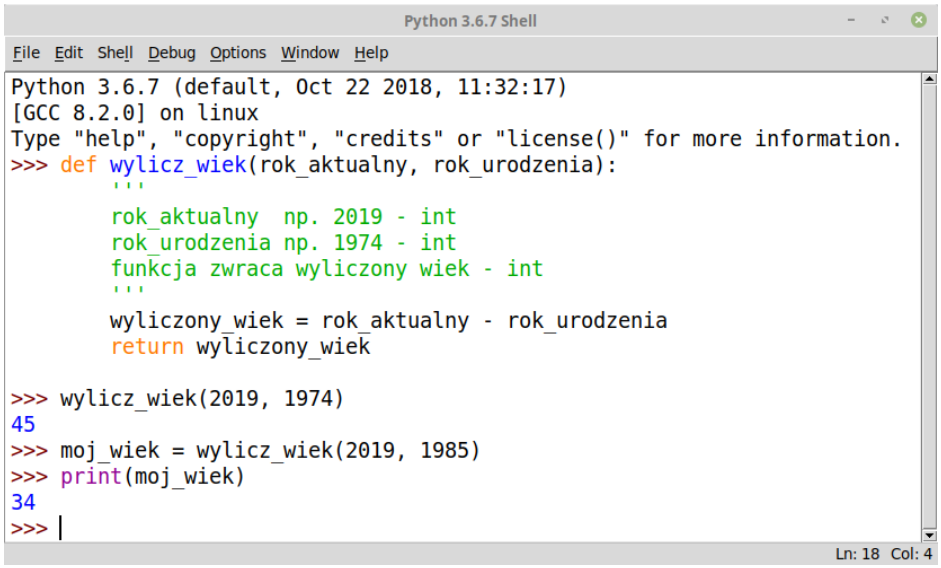
Oto przykład oryginalnej dokumentacji do funkcji `print`:

print(*objects, sep=' ', end='\\n', file=sys.stdout, flush=False)

Print objects to the text stream file, separated by sep and followed by end. sep, end, file and flush, if present, must be given as keyword arguments. All non-keyword arguments are converted to strings like str() does and written to the stream, separated by sep and followed by end. Both sep and end must be strings; they can also be None, which means to use the default values. If no objects are given, print() will just write end. The file argument must be an object with a write(string) method; if it is not present or None, sys.stdout will be used. Since printed arguments are converted to text strings, print() cannot be used with binary mode file objects. For these, use file.write(...) instead. Whether output is buffered is usually determined by file, but if the flush keyword argument is true, the stream is forcibly flushed.

Changed in version 3.3: Added the flush keyword argument.

Ćwiczenie 3
Napisz funkcję z przypisaniem wyniku do zmiennej, a następnie wydrukuj ją na ekranie, korzystając z funkcji `print`.



Funkcja zwracająca więcej niż jeden parametr

W języku Python funkcja może zwrócić więcej niż jedną wartość. Zastanów się, jak zapisać funkcję, która na podstawie roku urodzenia oblicza wiek osoby i od razu sprawdza, czy to osoba pełnoletnia. Funkcja ta ma zwrócić obie te informacje. Zapisujesz to w następujący sposób:

```
return wartosc_1, wartosc_2
```

Przeanalizuj taki kod funkcji:

```
def jaki_wiek(rok_urodzenia, aktualny_rok=2019):
    '''
    funkcja zwraca wiek w latach - int
    oraz informację czy osoba jest pełnoletnia - bool
    '''

    wiek = aktualny_rok - rok_urodzenia

    if wiek > 17:
        dorosla = True
    else:
        dorosla = False

    return wiek, dorosla
```

W ten sposób możesz sprawić, że funkcja będzie bardziej przydatna, gdy zrobi kilka rzeczy, a ty tylko poznasz wyniki. To naprawdę bardzo ciekawy sposób programowania. Musisz tylko pamiętać, aby w wywołaniu takiej funkcji odpowiednio odebrać te dane.

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
```

```
>>> def jaki_wiek(rok_urodzenia, aktualny_rok=2019):
    '''
    funkcja zwraca wiek w latach - int
    oraz informację czy osoba jest pełnoletnia - bool
    '''
    wiek = aktualny_rok - rok_urodzenia
    if wiek > 17:
        dorosla = True
    else:
        dorosla = False
    return wiek, dorosla

>>> jaki_wiek(1974)
(45, True)
>>> jaki_wiek(1974, 2019)
(45, True)
>>> a, b = jaki_wiek(1974, 2019)
>>> a
45
>>> b
True
>>>
```

> **Ważne!**
>
> Zwróć uwagę, że w linii, w której przypisujesz wynik funkcji do zmiennych a i b, używasz przecinka między tymi zmiennymi. Tyle, ile wartości zwraca funkcja, tyle musisz zarezerwować zmiennych odbierających.

Słownik

> **funkcja**
>
> wydzielona część kodu, nazywana także podprogramem, wykonująca określone operacje, możliwa do wywołania wiele razy podczas działania programu

> **parametr**
>
> umożliwia przekazanie argumentów, czyli danych wejściowych, do funkcji. W większości języków programowania wartości parametrów definiuje się podczas procesu deklaracji podprogramu

Audiobook

Polecenie 1
Wysłuchaj audiobooka i zastanów się, jakie są główne zalety funkcji.

Polecenie 2
Opisz przykładowy problem, w rozwiązaniu którego możesz użyć funkcji.

Sprawdź się

Ćwiczenie 1

Funkcja `print` w Python...

- jest funkcją w Python 3, a poleceniem w Python 2.
- jest funkcją w Python 2, a poleceniem w Python 3.
- jest funkcją w Python 2 i w Python 3.
- jest poleceniem w Python 2 i w Python 3.

Ćwiczenie 2

Uporządkuj w kolejności części funkcji.

- opis – `docstring`
- słowo kluczowe `def`
- nazwa funkcji
- słowo kluczowe `return` z wartością zwracaną
- ciało funkcji
- parametry funkcji

Ćwiczenie 3

Oznacz kluczowe słowa języka Python.

słowo kluczowe

def nazwa_funkcji(parametr):

'''

docstring

'''

return parametr

Ćwiczenie 4

Ćwiczenie 5

Ćwiczenie 6

Ćwiczenie 7

Praca domowa

Poszukaj w internecie różnych dokumentacji do funkcji `print`. Poszukaj również różnych przykładowych programów, które zawierają tę funkcję i przygotuj notatkę na podstawie zebranych informacji.

Dla nauczyciela

Autor: Adam Jurkiewicz

Przedmiot: informatyka

Temat: Wprowadzenie do funkcji

Grupa docelowa: III etap edukacyjny, liceum, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

2) stosuje zasady programowania strukturalnego i obiektowego w rozwiązywaniu problemów.

Kształtowane kompetencje kluczowe:

- kompetencje w zakresie rozumienia i tworzenia informacji,
- kompetencje w zakresie wielojęzyczności,
- kompetencje cyfrowe,
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się.

Cele operacyjne:

Uczeń:

- zna podstawowe metody tworzenia funkcji w Pythonie;
- używa różnych sposobów zapisu parametrów funkcji;
- rozumie koncepcję programistyczną funkcji jako „czarnej skrzynki”
- wskazuje problemy, które mogą być zaprezentowane za pomocą funkcji (podzielone na mniejsze).

Strategie nauczania:

- bezpośrednia strategia poznawcza;
- strukturalna.

Metody i techniki nauczania:

- pogadanka;
- kula śniegowa;
- mapa skojarzeń;
- dyskusja;
- metoda przypadków.

Formy pracy:

- praca indywidualna;
- praca w grupach.

Środki dydaktyczne:

- komputery ze środowiskiem Python 3 / IDLE;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg zajęć:

Faza wstępna

1. Krótka pogadanka o podziale większych problemów na mniejsze.
2. Nauczyciel przedstawia temat i cele lekcji.
3. Kula śniegowa – Jak jest zbudowana funkcja w Python 3? Uczniowie w parach przygotowyją wyjaśnienie problemu. Potem łaczą się w czwórki, porównują swoje propozycje, weryfikują je i tworzą wspólne rozwiązanie. astępnie łączą się w liczniejsze grupy, aż do stworzenia ogólnoklasowego rozwiązania.
4. Weryfikacja utworzonej definicji z wykorzystaniem ilustracji interaktywnej w treści lekcji.

Faza realizacyjna

1. Omówienie innych sposobów definiowania parametrów funkcji – praca w parach. Uczniowie zapoznają się z fragmentem treści lekcji, następnie wybrane pary przedstawiają zebrane informacje na forum klasy.
2. Praca z multimedium bazowym – audiobook. Po odsłuchaniu materiału uczniowie zastanawiają się, jakie są główne zalety funkcji – tworzą mapę skojarzeń.
3. Podział klasy na cztery grupy. Zadaniem każdej z nich jest przygotowanie problemu, w rozwiązaniu którego można użyć funkcji. Prezentacja na forum klasy: pozostali uczniowie weryfikują pomysły kolegów, uzupełniają je, dyskutują.
4. Praca w parach. Uczniowie tworzą i testują własne funkcje na podstawie przykładów z e-materiału, m.in. funkcja `print`, funkcja zwracająca więcej niż jeden parametr. Prezentacja wyników na forum klasy.
5. Praca w grupach. Uczniowie wymyślają problem, który można podzielić na mniejsze elementy/etapy i zakodować jako funkcje. Grupy tworzą rozwiązanie problemów i prezentują je pozostałym uczniom.
6. Wspólna dyskusja o zrealizowanych funkcjach.
7. Uczniowie uruchamiają program składający się z wytworzonych przez nich funkcji.

Faza podsumowująca

1. Wybrany uczeń podsumowuje lekcję, wskazując na umiejętności, które nabył podczas zajęć.

Praca domowa:

Napisz program, który będzie symulował X rzutów monetą. Zdefiniuj funkcję `rzut_monety(ilosc)`, której wynikiem będzie liczba wyrzuconych orłów. Niech parametr **ilosc** ma wartość domyślną 26.

Materiały pomocnicze:

Dokumentacja dla Python 3.

Wskazówki metodyczne opisujące różne zastosowania multimedium:

Nauczyciel może wykorzystać audiobook w nauczaniu wyprzedzającym. Uczniowie słuchają jego treści przed lekcją, a podczas zajęć w parach przygotowują pytania do tekstu i przepytują się nazwajem ze znajomości poruszanych w audiobooku zagadnień.