



Funkcje w języku Python

- [Wprowadzenie](#)
- [Audiobook](#)
- [Przeczytaj](#)
- [Sprawdź się](#)
- [Dla nauczyciela](#)

Bibliografia:

- Dokumentacja funkcji wbudowanych w Python 3 (3.7)
- Dokumentacja funkcji wbudowanych w Python3
- Dokumentacja funkcji wbudowanych w Python 3 (3.6)



Funkcje w języku Python

Źródło: Pixabay, domena publiczna.

W tym e-materiale powtarzamy również wiadomości ze szkoły podstawowej.

Jak w każdym języku programowania, również w Pythonie programista może tworzyć własne funkcje. Są to części programu wielokrotnego użytku. Funkcje pozwalają nadać nazwę blokowi wyrażień, 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.

Więcej informacji o funkcjach znajdziesz w e-materiale [Funkcje](#). Chcesz wiedzieć, jak wygląda omawiane zagadnienie w innych językach programowania? Zapoznaj się z e-materiałami:

- [Funkcje w języku C++](#),
- [Funkcje w języku Java](#).

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.

Audiobook

Problem 1

Specyfikacja problemu:

Napisz program, którego zadaniem będzie policzenie sumy dwóch zmiennych.

Dane:

- a, b - liczby całkowite
- W programie wykorzystaj funkcje.

Lista kroków:

```
1 def suma(a, b):  
2     # tutaj dopisz treść funkcji  
3  
4 # tutaj zdefiniuj zmienne i wywołaj funkcję suma
```

1

Polecenie 1

Porównaj swoje rozwiązanie z filmem.



Wprowadzenie do funkcji

Funkcje w języku Python



Film dostępny pod adresem </preview/resource/R9A41XMFk0zRL>

Źródło: Contentplus.pl Sp. z o.o., licencja: CC BY-SA 3.0.

Film nawiązujący do treści materiału wprowadzenia do funkcji w języku Python.

Polecenie 2

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

Audiobook można wysłuchać pod adresem: <https://zpe.gov.pl/b/P1ew8QMNS>

Wprowadzenie do funkcji w Pythonie

Funkcja, czasami nazywana podprogramem, a rzadziej procedurą, to wydzielona część programu przetwarzająca parametry lub argumenty i ewentualnie zwracająca wartość lub wiele wartości, które następnie mogą być wykorzystane jako argument w innych działaniach lub funkcjach.

Czasami o funkcjach mówi się jak o czarnych skrzynkach. Nie zawsze wiesz, co ona zawiera, ale masz klucz, możesz ją otworzyć i do niej zajrzeć. Możesz zawsze użyć skrzynki do pewnych zadań, które potrafi wykonać dzięki swojej zawartości. Ty wcale nie musisz do końca wiedzieć, jak ona to dokładnie robi. Wystarczy, że doskonale wiesz, co owa skrzynka na końcu dla ciebie zrobi, i jak sprawić, aby to zrobiła. Natomiast szczegóły działania pozostawiasz jej, czyli funkcji – czarnej skrzynce.

W dużych firmach programiści często zajmują się tylko fragmentem całości – przykładowo pojedynczą funkcją. Możesz otrzymać następujące zlecenie: stwórz funkcję, która przyjmuje trzy parametry: pozostałą do spłacenia kwotę kredytu, pozostałe do spłacenia odsetki oraz pozostały czas trwania kredytu. Na podstawie tych danych oblicz i zwróć kwotę następnej raty. Tego typu opis, który mówi, czego potrzebujemy na wejściu i wyjściu do naszej czarnej skrzynki, z pewnością będzie przydatny również w przypadku pracy na innym stanowisku, a nawet gdy zlecamy takie zadanie innym programistom lub zewnętrznej firmie.

Funkcje w programach mają bardzo różne zadania do wykonania, trochę jak w zespole pracowników różne osoby mają różne zadania, jednak wymieniają się pomiędzy sobą informacjami i w ten sposób wykonują swoją pracę. Dzięki współpracy powstają na przykład telewizory czy samochody, piosenki lub teledyski, reklamy czy artykuły prasowe, a także również programy. W przypadku tworzenia tych ostatnich bardzo ważne jest, aby programiści tworzyli funkcje, które mogą współpracować w jak najlepszy sposób.

Wyobraź sobie program do obsługi dziennika elektronicznego. Nawet nie musisz starać się oszacować ilości funkcji użytych w całym programie – na pewno są ich dziesiątki, jeśli nie setki. Funkcja jest odpowiedzialna za swój mały wycinek pracy, który jest

elementem większej całości. Wyobraź sobie wpisywanie do dziennika ocen uczniów. W programie, aby wpisać ocenę, należy wpisać słowo z klawiatury, na przykład czwórka. Nagle, po kilku latach działania programu, ktoś wpada na genialny pomysł, aby słowa zastąpić cyframi. Samo to nie jest jeszcze problematyczne. Jednak po pewnym czasie okazuje się, że użytkownikom zdarza się wpisać ocenę siedem lub inną, która według skali ocen nie istnieje. Zatem programiści otrzymują zadanie – należy tak zmienić program, aby nie było możliwości wpisania cyfr innych niż od jednego do sześciu.

Aby to zrobić, trzeba sprawdzić w każdym miejscu programu, czy nie ma kawałka kodu odpowiadającego za wpisywanie danych z klawiatury przez użytkownika. Wyobraź sobie, że trzeba przeanalizować na przykład dziesięć tysięcy linii kodu. W tym fragmencie może być ukrytych kilkanaście miejsc, gdzie trzeba zmienić kilkanaście linii. W sumie – powiedzmy – zmiana pięciuset linii kodu wśród dziesięciu tysięcy. To żmudna praca, a do tego obarczona dużym prawdopodobieństwem błędu – w końcu programista może się pomylić i na przykład w szóstym wystąpieniu danego kawałka kodu programu wstawi znak plus zamiast minus lub odwrotnie.

A gdyby wpisywanie danych od użytkownika było funkcją, która jest wywoływana w wielu różnych miejscach, ale jest to TYLKO jedna funkcja? Wówczas wystarczyłoby dbać o JEDNO wystąpienie kodu odpowiedzialnego za pobieranie ocen od użytkownika. A każde zmiany, jakie programiści wprowadzą w JEDNEJ funkcji, od razu są widoczne w wielu miejscach dużego programu.

W ten sposób działają funkcje i taka idea przyświeca od zawsze tworzeniu programów. Duże problemy są dzielone na mniejsze, które to realizowane są poprzez małe funkcje. A wyniki działania wielu małych funkcji składają się na dobrze działający duży program.

Jako ciekawostkę możemy dodać, że w Pythonie – w przeciwieństwie do niektórych języków programowania – funkcja może być argumentem funkcji, może również być wynikiem, czyli wartością zwracaną funkcji lub można ją przypisać do zmiennych.

Źródło: GroMar.eu, licencja: CC BY-SA 3.0.

Polecenie 3

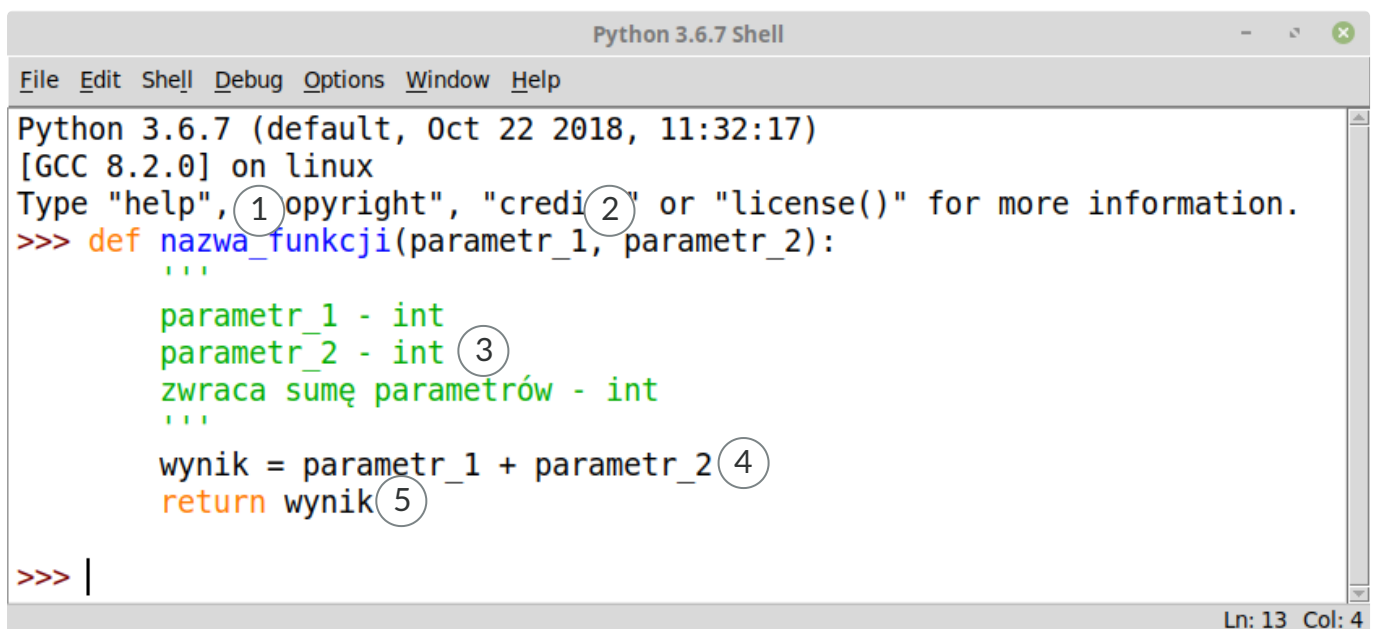
Opisz przykładowy problem, w rozwiązaniu którego możesz użyć funkcji.

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 języku Python 3?



```
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", (1)copyright, "credits" or "license()" for more information.
>>> def nazwa_funkcji(parametr_1, parametr_2):
    ...
    parametr_1 - int
    parametr_2 - int (3)
    zwraca sumę parametrów - int
    ...
    wynik = parametr_1 + parametr_2 (4)
    return wynik (5)

>>> |
```

Ln: 13 Col: 4

1

Nazwa funkcji

Funkcję warto nazywać tak, aby później odzwierciedlała swoją funkcjonalność.

2

Parametry

Funkcja może mieć różne parametry. Na początek skup się na tym, aby tworzyć proste funkcje z maksymalnie kilkoma parametrami.

Docstring

Krótki opis funkcji widoczny dla innych programistów.

Ciało funkcji

Wszystkie operacje, które są wykonywane wewnątrz funkcji.

Return

Wartość zwracana przez funkcję (funkcja może zwracać więcej wartości).

Funkcja w Python 3 – definicja

```
1 def nazwa_funkcji(parametr_1, parametr_2):
2     '''
3     parametr_1 - int
4     parametr_2 - int
5     zwraca sumę parametrów - int
6     '''
7     wynik = parametr_1 + parametr_2
8     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**.

```

1 Python 3.6.7 (default, Oct 22 2018, 11:32:17)
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> def nazwa_funkcji(parametr_1, parametr_2):
5     '''
6     parametr_1 - int
7     parametr_2 - int
8     zwraca sumę parametrów - int
9     '''
10    wynik = parametr_1 + parametr_2
11    return wynik
12
13 >>> nazwa_funkcji(12, 45)
14 57

```

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

```

1 Python 3.6.7 (default, Oct 22 2018, 11:32:17)
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> def nazwa_funkcji(parametr_1, parametr_2):
5     '''
6     parametr_1 - int
7     parametr_2 - int
8     zwraca sumę parametrów - int
9     '''
10    wynik = parametr_1 + parametr_2
11    return wynik
12
13 >>> nazwa_funkcji(12)
14 Traceback (most recent call last):
15   File "<pyshell#3>", line 1, in <module>
16     nazwa_funkcji(12)
17 TypeError: nazwa_funkcji() missing 1 required positional argument
18 >>>

```

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

```

1 Python 3.6.7 (default, Oct 22 2018, 11:32:17)
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> def nazwa_funkcji(parametr_1, parametr_2 = 10):
5     '''
6     parametr_1 - int
7     parametr_2 - int (wartość domyślna 10)
8     zwraca sumę parametrów - int
9     '''
10    wynik = parametr_1 + parametr_2
11    return wynik
12
13 >>> nazwa_funkcji(2, 4)
14 6
15 >>> nazwa_funkcji(2)
16 12
17 >>>

```

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

```

1 Python 3.6.7 (default, Oct 22 2018, 11:32:17)
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> def nazwa_funkcji(parametr_1 = 20, parametr_2 = 10):
5     '''
6     parametr_1 - int (wartość domyślna 20)
7     parametr_2 - int (wartość domyślna 10)
8     zwraca sumę parametrów - int (domyślnie 30)
9     '''
10    wynik = parametr_1 + parametr_2
11    return wynik
12
13 >>> nazwa_funkcji()
14 30
15 >>> nazwa_funkcji(parametr_1=2, parametr_2=3)
16 5
17 >>> nazwa_funkcji(2, parametr_2=4)
18 6

```

```
19 >>> nazwa_funkcji(parametr_2=3)
20 23
21 >>>
```

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 jakiejś 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`).

```
1 def wylicz_wiek(rok_aktualny, rok_urodzenia):
2     '''
3     rok_aktualny np. 2019 - int
4     rok_urodzenia np. 1974 - int
5     funkcja zwraca wyliczony wiek - int
6     '''
7     wyliczony_wiek = rok_aktualny - rok_urodzenia
8     return wyliczony_wiek
```

```
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
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 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

>>> |
```

Ln: 13 Col: 4

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 IDE stwórz definicję funkcji, która 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.

```
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> print('To jest podstawowa funkcja - wypisuje na ekran')
5 To jest podstawowa funkcja - wypisuje na ekran
6 >>>
```

Ćwiczenie 2

Połącz w pary odpowiednie treści.

Funkcja print w Python 3 a w Python 2

różni się – w Python 2 jest poleceniem, a nie funkcją.

Funkcja print w języku Python 3

działa tak samo w Windows i w Linuksie.

Funkcja w Pythonie

może przyjąć kilka argumentów/parametrów.

Ważne!

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

```
1 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)`

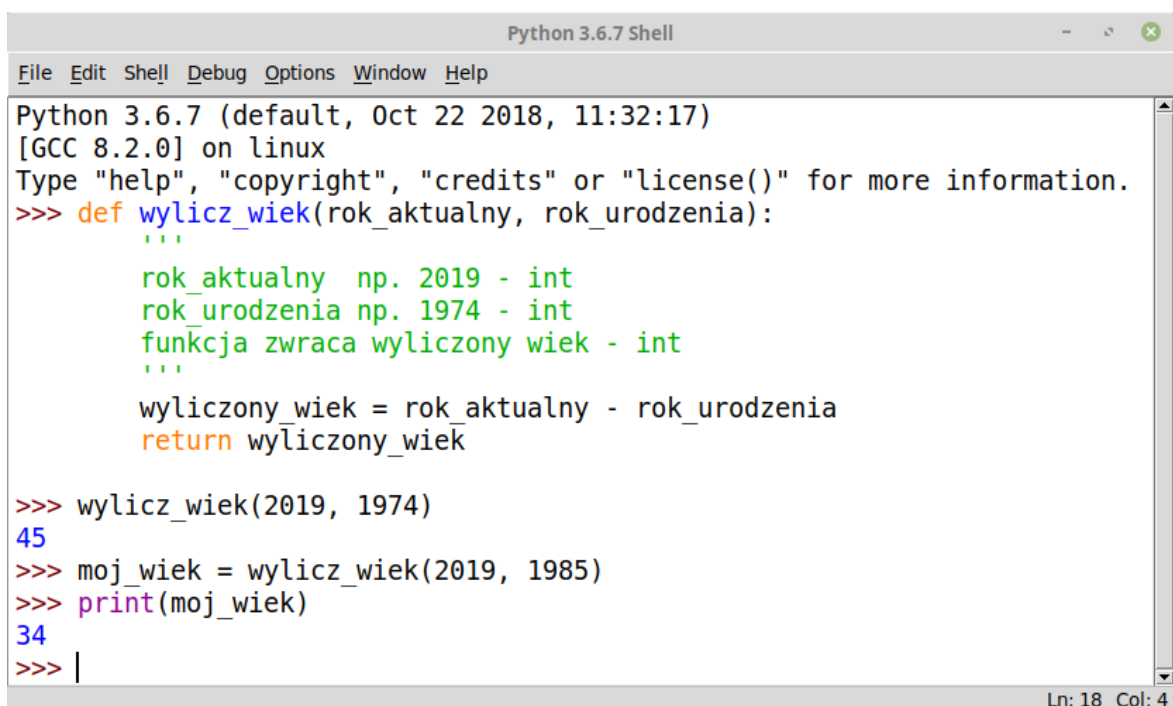
Print objects to the text stream file, separated by sep and followed by end. sep, end, file and flush, if present, must be given as keyword arguments. All non-keyword arguments are converted to strings like `str()` does and written to the stream, separated by sep and followed by end. Both sep and end 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`.



```
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
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 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

>>> wylicz_wiek(2019, 1974)
45
>>> moj_wiek = wylicz_wiek(2019, 1985)
>>> print(moj_wiek)
34
>>> |
```

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:

```
1 return wartosc_1, wartosc_2
```

Przeanalizuj taki kod funkcji:

```

1 def jaki_wiek(rok_urodzenia, aktualny_rok=2019):
2     '''
3     funkcja zwraca wiek w latach - int
4     oraz informację czy osoba jest pełnoletnia - bool
5     '''
6
7     wiek = aktualny_rok - rok_urodzenia
8
9     if wiek > 17:
10         dorosla = True
11     else:
12         dorosla = False
13
14     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.

```

1 Python 3.6.7 (default, Oct 22 2018, 11:32:17)
2 [GCC 8.2.0] on linux
3 Type "help", "copyright", "credits" or "license()" for more infor
4 >>> def jaki_wiek(rok_urodzenia, aktualny_rok=2019):
5     '''
6     funkcja zwraca wiek w latach - int
7     oraz informację czy osoba jest pełnoletnia - bool
8     '''
9     wiek = aktualny_rok - rok_urodzenia
10    if wiek > 17:
11        dorosla = True
12    else:
13        dorosla = False
14    return wiek, dorosla
15
16 >>> jaki_wiek(1974)
17 (45, True)
18 >>> jaki_wiek(1974, 2019)
19 (45, True)
20 >>> a, b = jaki_wiek(1974, 2019)
21 >>> a
22 45

```

```
23 >>> b
24 True
25 >>>
```

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

Sprawdź się

Pokaż ćwiczenia:   

Ćwiczenie 1



Uzupełnij poniższe definicje funkcji i sprawdź, czy działają poprawnie. Dawne statki żaglowe pływały ze średnią prędkością 15 węzłów (jednostka prędkości używana w marynarce).

Zdefiniuj funkcję `predk_km(wezel)`, która będzie zamieniać prędkość podaną w węzłach na prędkość w km/h i zwracać taką prędkość.

Twoje zadania

1. Utwórz funkcję `predk_km(wezel)`
2. Zwróć prędkość w km/h

```
1 def predk_km(wezel):  
2  
3     return 1  
4  
5 ### drukowanie testowego wyniku:  
6 print(predk_km(15))
```

```
1
```



Ćwiczenie 2



Zdefiniuj funkcję, która zwróci wartość `True`, jeśli podana liczba jest parzysta, lub `False`, jeśli liczba jest nieparzysta.

Twoje zadania

1. Utwórz funkcję `parzysta(liczba)`
2. Zwróć `True` gdy argument funkcji jest liczbą parzystą
3. Zwróć `False` gdy argument funkcji jest liczbą nieparzystą

```
1 def parzysta(liczba):  
2     ### tutaj zmodyfikuj kod  
3     return True  
4  
5 ### drukowanie testowego wyniku:  
6 print(str(parzysta(15)))
```

1



Ćwiczenie 3



Samoloty mają najczęściej dwie klasy pasażerskie (biznes i ekonomiczną). W klasie biznes są trzy miejsca w rzędzie, a w klasie ekonomicznej sześć. Zdefiniuj funkcję `miejsca_w_samolot(liczba1, liczba2)`, która w wyniku poda liczbę miejsc w samolocie. Parametr `liczba1` oznacza liczbę rzędów foteli w klasie biznes, a parametr `liczba2` oznacza liczbę rzędów foteli w klasie ekonomicznej. Parametr `liczba1` może przyjmować wartości od 3 do 7, a parametr `liczba2` od 15 do 40, w zależności od wersji. Jeśli którykolwiek z parametrów będzie poza dopuszczalnym zakresem, funkcja powinna zwrócić wartość -1.

Twoje zadania

1. Utwórz funkcję `miejsca_w_samolot(liczba1, liczba2)`
2. Zwróć -1 jeśli argumenty funkcji są poza zakresami podanymi w poleceniu
3. Zwróć liczbę miejsc w samolocie

```
1 def miejsca_w_samolot(liczba1, liczba2):
2
3     return True
4
5 ### drukowanie testowego wyniku:
6 print(miejsca_w_samolot(4, 26))
```

1



Ćwiczenie 4



Zdefiniuj funkcję `maksimum_suma_iloczyn(a1, a2)`, która zwróci większą z podanych liczb, ich sumę oraz ich iloczyn. Wszystkie trzy wartości powinny być zwrócone poprzez jedno słowo kluczowe `return`.

Twoje zadania

1. Utwórz funkcję `maksimum_suma_iloczyn`
2. Zwróć obliczone wartości za pomocą instrukcji `return a, b, c`

```
1 def maksimum_suma_iloczyn(a1, a2):  
2  
3     return 1, 2, 3  
4  
5 ### drukowanie testowego wyniku:  
6 print(maksimum_suma_iloczyn(11, 15))
```

```
1
```



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 przygotowują wyjaśnienie problemu. Potem łączą się w czwórki, porównują swoje propozycje, weryfikują je i tworzą wspólne rozwiązanie. Następnie 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 nawzajem ze znajomości poruszanych w audiobooku zagadnień.