Politechnika Wrocławska

Wstęp do programowania

INP003203L

rok akademicki 2020/21

semestr zimowy

Laboratorium 3

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Operatory matematyczne
  - kolejność wykonywania działań
  - zmienne w wyrażeniach matematycznych
  - typ danych wyniku

Na podstawie: T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

Plan prezentacji (2)

- Funkcje
  - projektowanie programu używającego funkcji
  - przekazywanie argumentów
  - funkcje zwracające wartość

Na podstawie: T. Gaddis, *Python dla zupełnie początkujących*, Helion, 2019

Politechnika Wrocławska

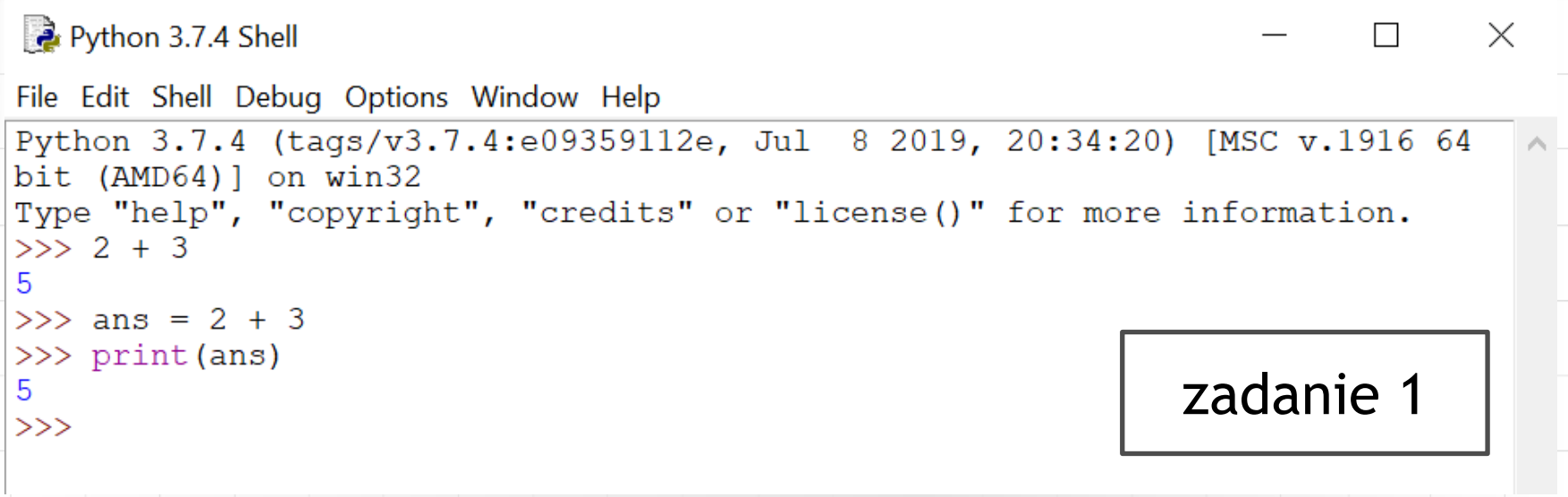
Operatory matematyczne

symbol	działanie
+	dodawanie
-	odejmowanie
*	mnożenie
/	dzielenie
//	dzielenie całkowite
%	reszta z dzielenia
**	potęgowanie

Politechnika Wrocławska

Operatory matematyczne

- Operator matematyczny wykonuje działanie i zwraca wynik
- Wynik można przypisać do zmiennej



Operatory matematyczne

Kolejność wykonywania działań

1. potęgowanie
2. mnożenie, dzielenie, dzielenie całkowite, reszta z dzielenia
3. dodawanie, odejmowanie

Operatory matematyczne

Kolejność wykonywania działań

Po wykonaniu polecenia

```
ans = 5 + 2 * 3
```

wartość zmiennej **`ans`** to 11

Operatory matematyczne

Kolejność wykonywania działań

```
ans = 5 + 2 * 4 / 2 % 3 + 10 - 3

      5 + 8 / 2 % 3 + 10 - 3
        5 + 4 % 3 + 10 - 3
          5 + 1 + 10 - 3
            6 + 10 - 3
              16 - 3
                13
```

Operatory matematyczne

Zmienne w wyrażeniach

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 23
>>> b = 5
>>> a + b
28
>>> a - b
18
>>> a * b
115
>>> a / b
4.6
>>> a // b
4
>>> a % b
3
```

Operatory matematyczne

Nawiasy

- niewłaściwie obliczona średnia

```
srednia = i + j + k + l / 4
```

- dobre wyrażenie na średnią

```
srednia = (i + j + k + l) / 4
```

Operatory matematyczne

Różne typy danych

- Typ danych wyniku zależy od typów danych argumentów

zadanie 2

Politechnika Wrocławska

Funkcje

Projektowanie programu

two_functions.py

File Edit Format Run Options Window Help

```
#Ten program zawiera dwie funkcje.

#Definicja funkcji głównej - main()
def main():
    print('Mam dla Ciebie wiadomość.')
    message()
    print('Żegnaj')

#Definicja funkcji message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji głównej
main()
```

Funkcje

Projektowanie programu

two_functions.py

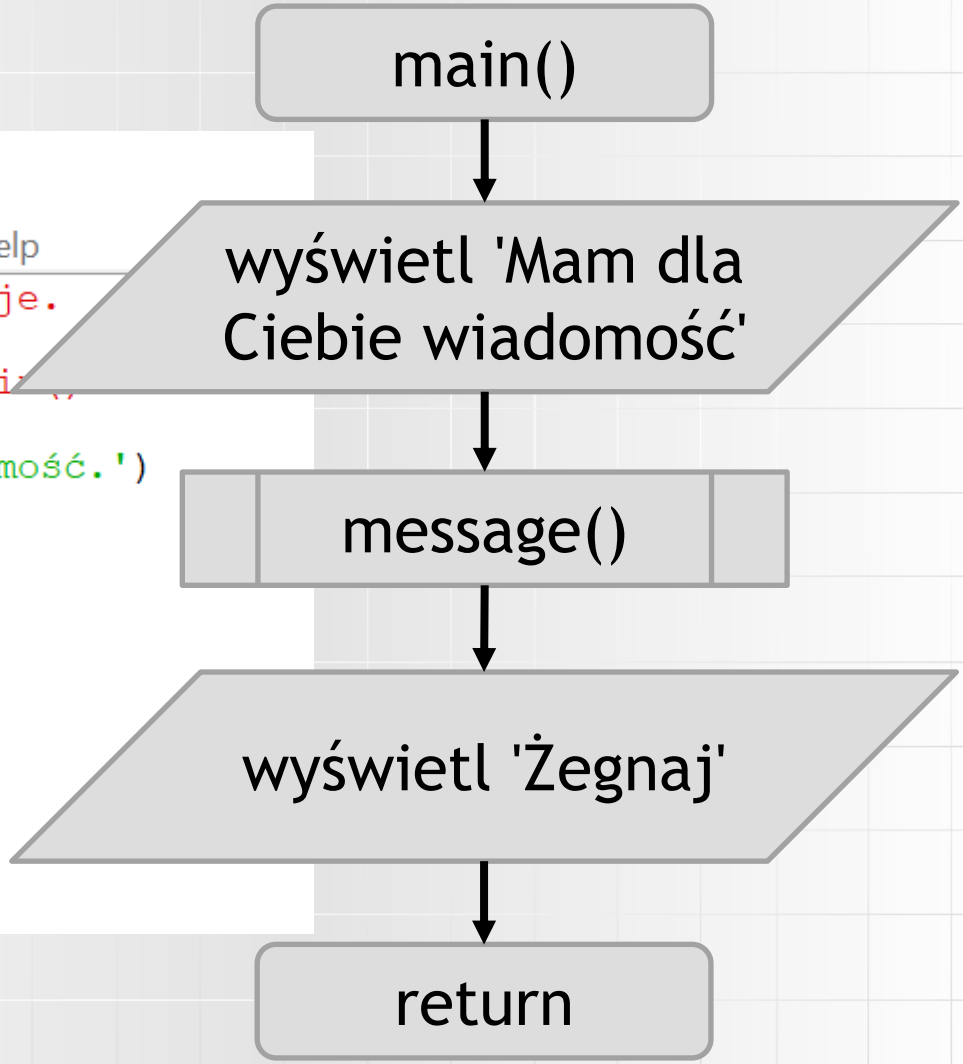
File Edit Format Run Options Window Help

```
#Ten program zawiera dwie funkcje.

#Definicja funkcji głównej - mai
def main():
    print('Mam dla Ciebie wiadomość.')
    message()
    print('Żegnaj')

#Definicja funkcji message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji głównej
main()
```



Funkcje

Projektowanie programu

two_functions.py

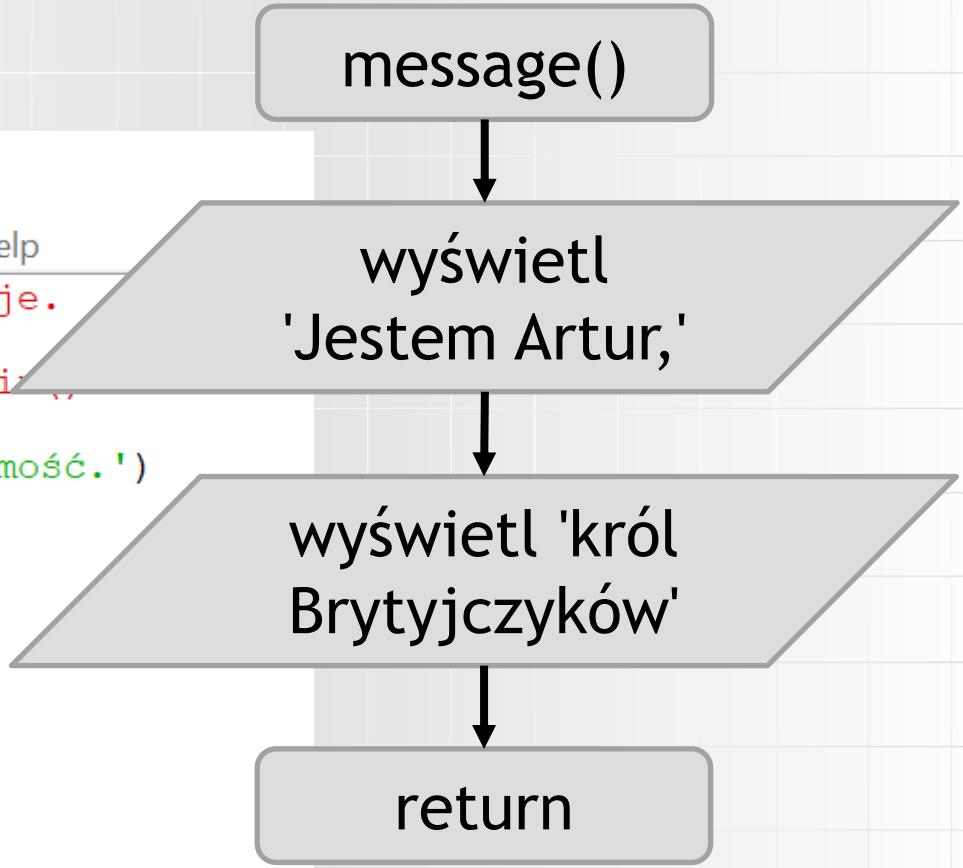
File Edit Format Run Options Window Help

```
#Ten program zawiera dwie funkcje.

#Definicja funkcji głównej - mai
def main():
    print('Mam dla Ciebie wiadomość.')
    message()
    print('Żegnaj')

#Definicja funkcji message()
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')

#Wywołanie funkcji głównej
main()
```



Politechnika Wrocławska

Funkcje

Projektowanie programu

Projektowanie od ogółu do szczegółu

- Zadanie główne jakie ma wywołać program, dzielimy na mniejsze podzadania
- Każde podzadanie analizujemy i sprawdzamy, czy nie da się podzielić na mniejsze. Kontynuujemy dzielenie na podzadania, tak długo jak możemy
- Tworzymy kod dla każdego podzadania

Funkcje

Przykład

elephant.py

File Edit Format Run Options Window Help

```
#Ten program wyświetla instrukcję krok po kroku
#jak schować słonia do lodówki.

#Funkcja main() zawiera logikę główną programu
def main():
    #Wyświetlenie komunikatu początkowego
    startup_message()
    input('Naciśnij Enter, aby przejść do kroku 1.')
    #Wyświetlenie instrukcji dla kroku 1.
    step1()
    input('Naciśnij Enter, aby przejść do kroku 2.')
    #Wyświetlenie instrukcji dla kroku 2.
    step2()
    input('Naciśnij Enter, aby przejść do kroku 3.')
    #Wyświetlenie instrukcji dla kroku 3.
    step3()
    input('Naciśnij Enter, aby zakończyć.')
    final_message()
```

Politechnika Wrocławska

Funkcje

Przykład

elephant.py

File Edit Format Run Options Window Help

```
#Funkcja startup_message() wyświetla na ekranie
#komunikat początkowy.
def startup_message():
    print('Ten program przeprowadzi Cię,')
    print('przez proces chowania słonia do lodówki.')
    print('Proces składa się z trzech prostych kroków.')
    print()

#Funkcja step1() wyświetla instrukcję dla kroku 1.
def step1():
    print('Krok 1.')
    print('Otwórz lodówkę.')
    print()

#Funkcja step2() wyświetla instrukcję dla kroku 2.
def step2():
    print('Krok 2.')
    print('Włóż słonia do lodówki.')
    print()
```

Funkcje

Przykład

elephant.py

File Edit Format Run Options Window Help

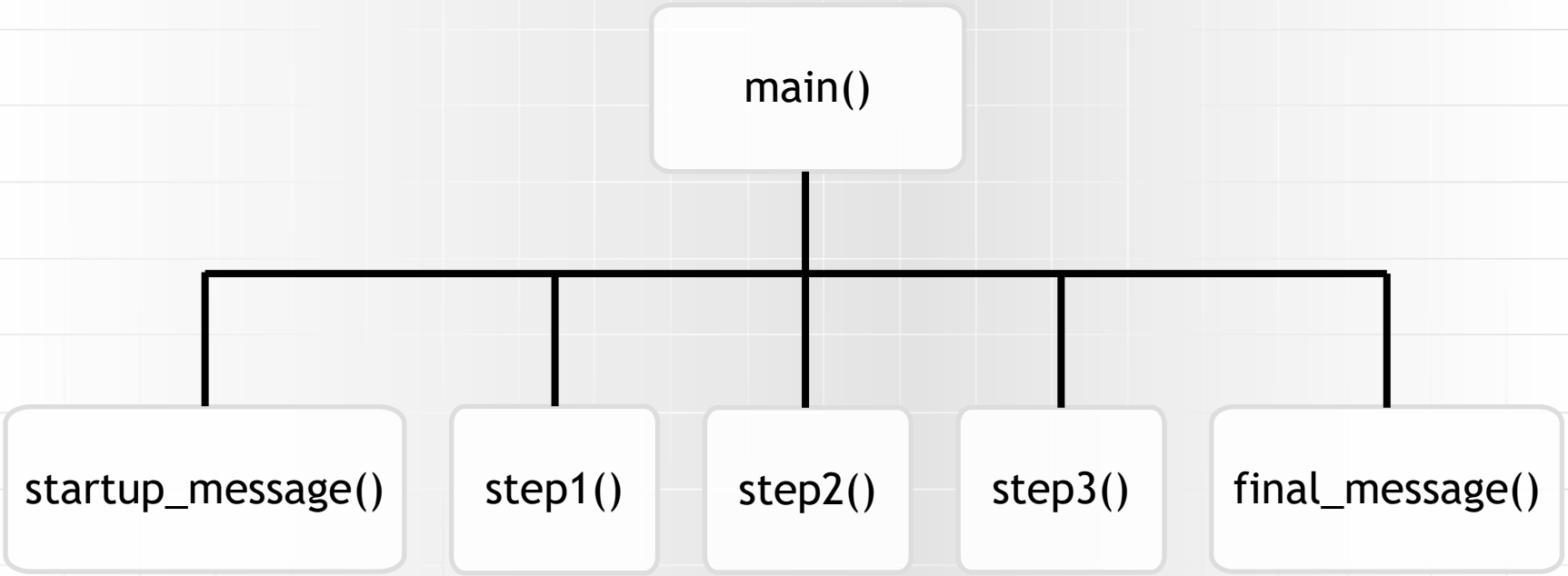
```
#Funkcja step3() wyświetla instrukcję dla kroku 3.
def step3():
    print('Krok 3.')
    print('Zamknij lodówkę.')
    print()

#Funkcja final_message() wyświetla na ekranie
#komunikat końcowy.
def final_message():
    print('\nJeśli postępowałeś zgodnie z instrukcjami,')
    print('to słoń jest w lodówce.\n')

#Wywołanie funkcji main() i rozpoczęcie działania programu
main()
```

Funkcje

Schemat hierarchiczny - przykład



Funkcje

Zmienne lokalne

bad_local.py

File Edit Format Run Options Window Help

```
#Program pokazuje BŁĘDNE użycie zmiennej lokalnej

#Defincja funkcji main()
def main():
    get_name()
    print('Witaj,', name) #Ten wiersz powoduje błąd.

#Funkcja get_name() pyta użytkownika o imię
def get_name():
    name = input('Jak masz na imię? ')

#Wywołanie funkcji głównej
main()
```

- Zmienna **`name`** ma przypisaną wartość w funkcji **`get_name()`**
- Zmienna **`name`** jest niedostępna w funkcji **`main()`**

Funkcje

Zmienne lokalne

population.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje, użycie dwóch zmiennych lokalnych
2  #o tych samych nazwach, w różnych funkcjach
3
4  def main():
5      dolnoslaskie()
6      mazowieckie()
7
8  #Funkcja dolnoslaskie() tworzy lokalną zmienną population.
9  def dolnoslaskie():
10     population = 2902365
11     print('Liczba mieszkańców województwa dolnośląskiego to ',\
12           population,'.')
13
14 #Funkcja mazowieckie tworzy lokalną zmienną population.
15 def mazowieckie():
16     population = 5391813
17     print('Liczba mieszkańców województwa mazowieckiego to ',\
18           population,'.')
19
20 main()
21
```

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.

def main():
    value = 13 #przypisanie zmiennej value wartości 13
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
    show_double(value)

#Funkcja show_double() pobiera argument
#i wyświetla jego podwojoną wartość
def show_double(number):
    result = number * 2
    print(result)

main()
```

Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

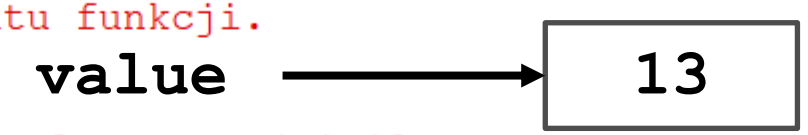
File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.

def main():
    value = 13 #przypisanie zmiennej value wartości 13
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
    show_double(value)

#Funkcja show_double() pobiera argument
#i wyświetla jego podwojoną wartość
def show_double(number):
    result = number * 2
    print(result)

main()
```



Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

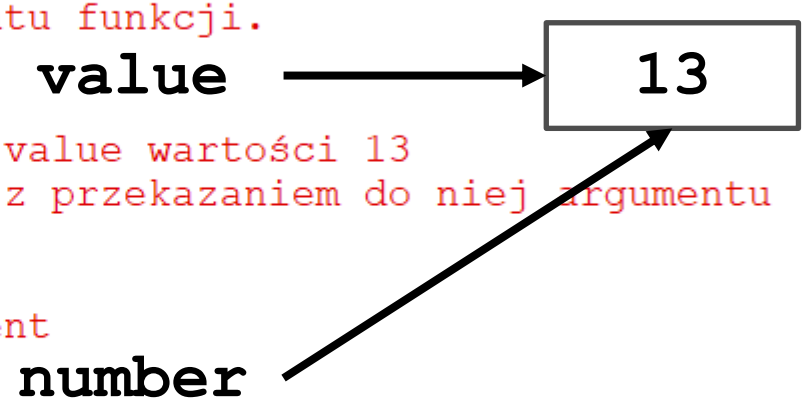
File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.

def main():
    value = 13 #przypisanie zmiennej value wartości 13
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
    show_double(value)

#Funkcja show_double() pobiera argument
#i wyświetla jego podwojoną wartość
def show_double(number):
    result = number * 2
    print(result)

main()
```



Funkcje

Przekazywanie argumentów do funkcji

pass_argument.py

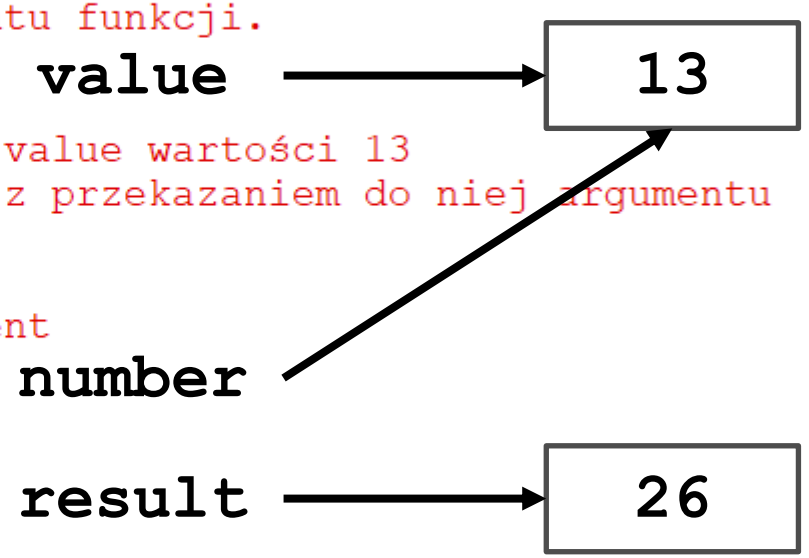
File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie argumentu funkcji.

def main():
    value = 13 #przypisanie zmiennej value wartości 13
    #wywołanie funkcji show_double() z przekazaniem do niej argumentu
    show_double(value)

#Funkcja show_double() pobiera argument
#i wyświetla jego podwojoną wartość
def show_double(number):
    result = number * 2
    print(result)

main()
```



Funkcje

Przekazywanie argumentów do funkcji

multiple_arguments.py

File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie dwóch argumentów do funkcji.

def main():
    print('Suma liczb 12 i 45 wynosi')
    #Wartości wyrażeń użytych w miejscu wywołania
    #zostaną nadane zmiennym w funkcji show_sum().
    show_sum(12,45)

#Funkcja show_sum() pobiera dwa argumenty
#i wyświetla ich sumę.
#W nawiasie umieszczono listę argumentów.
#Nazwy argumentów są rozdzielone przecinkiem.
def show_sum(num1, num2):
    result = num1 + num2
    print(result)

main()
```

Funkcje

Przekazywanie argumentów do funkcji

multiple_arguments.py

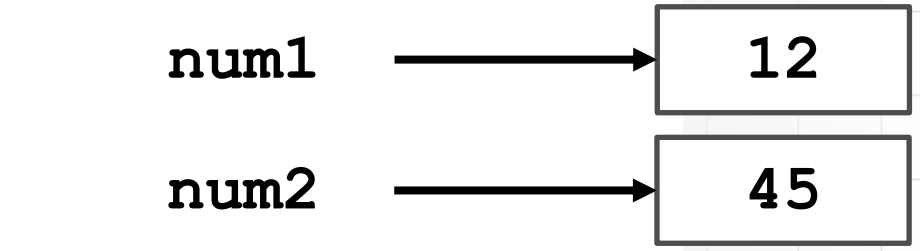
File Edit Format Run Options Window Help

```
#Program pokazuje przekazanie dwóch argumentów do funkcji.

def main():
    print('Suma liczb 12 i 45 wynosi')
    #Wartości wyrażeń użytych w miejscu wywołania
    #zostaną nadane zmiennym w funkcji show_sum().
    show_sum(12,45)

#Funkcja show_sum() pobiera dwa argumenty
#i wyświetla ich sumę.
#W nawiasie umieszczono listę argumentów.
#Nazwy argumentów są rozdzielone przecinkiem.
def show_sum(num1, num2):
    result = num1 + num2
    print(result)

main()
```



Funkcje

Zmiana argumentu funkcji

change_me.py

File Edit Format Run Options Window Help

```
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje

Zmiana argumentu funkcji

change_me.py

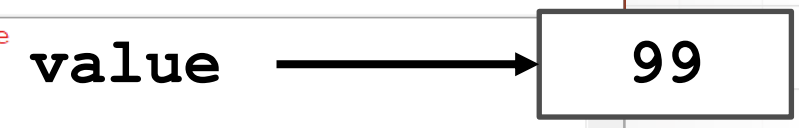
File Edit Format Run Options Window Help

```
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```



Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje

Zmiana argumentu funkcji

change_me.py

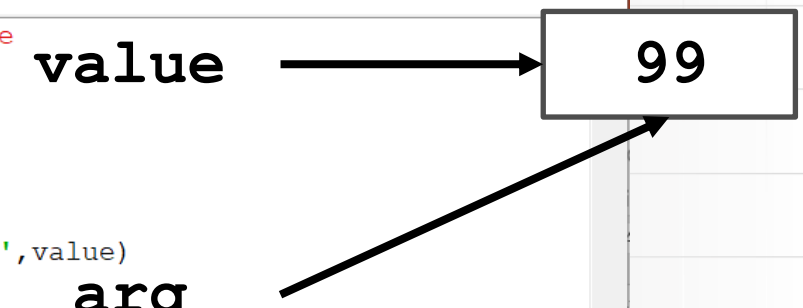
File Edit Format Run Options Window Help

```
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```



Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Funkcje

Zmiana argumentu funkcji

change_me.py

File Edit Format Run Options Window Help

```
#Program pokazuje, co się stanie po zmianie
#wartości argumentu w funkcji.

def main():
    value = 99
    print('Wartość wynosi',value)
    change_me(value)
    print('Wartość w funkcji main() wynosi',value)

def change_me(arg):
    print('Do funkcji change_me() przekazano wartość',arg)
    arg = 0
    print('Przypisano nową wartość do zmiennej arg')
    print('Teraz wartość wynosi',arg)

main()
```

value → 99

arg → 0

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/karol/Desktop/201920/python-examples/funkcje/change_me.py
Wartość wynosi 99
Do funkcji change_me() przekazano wartość 99
Przypisano nową wartość do zmiennej arg
Teraz wartość wynosi 0
Wartość w funkcji main() wynosi 99
>>>
```

Politechnika Wrocławska

Funkcje

Funkcje zwracające wartość

- Ogólna postać funkcji zwracającej wartość

```
def nazwa_funkcji():
    polecenie
    polecenie
    polecenie
    itd.
    return wyrażenie
```

Funkcje

Funkcje zwracające wartość

return_result_01.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje definicję i wywołanie funkcji zwracajacej wartość.
2
3  #Program pyta użytkownika o liczbę jabłek i pomarańczy.
4  #Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców
5  #i wyświetla stosowny komunikat.
6  def main():
7      apples  = int(input('Ile masz jabłek? '))
8      oranges = int(input('Ile masz pomarańczy? '))
9      print('Liczba wszystkich owoców',sum(apples,oranges))
10
11 #Funkcja sum oblicza sumę dwóch liczb
12 def sum(num1, num2):
13     result = num1 + num2 #zapisanie wyniku w zmiennej pomocniczej
14     return result        #zwrócenie wyniku
15
16 main()
17
```

Funkcje

Funkcje zwracające wartość

return_result_02.py

File Edit Format Run Options Window Help

```
1  #Program pokazuje definicję i wywołanie funkcji zwracajacej wartość.
2
3  #Program pyta użytkownika o liczbę jabłek i pomarańczy.
4  #Następnie wywołuje funkcję sum() do obliczenia łącznej liczby owoców
5  #i wyświetla stosowny komunikat.
6  def main():
7      apples  = int(input('Ile masz jabłek? '))
8      oranges = int(input('Ile masz pomarańczy? '))
9      print('Liczba wszystkich owoców',sum(apples,oranges))
10
11 #Funkcja sum oblicza sumę dwóch liczb
12 def sum(num1, num2):
13     return num1 + num2 #polecenie return może zwracać wartość wyrażenia
14
15 main()
16
```

Politechnika Wrocławska

Absolutne minimum

- Operatory matematyczne
  - pamiętaj o kolejności działań
  - zwróć uwagę na typ wyniku, dla operandów różnych typów
- Funkcje
  - projektuj programy przed pisaniem kodu
  - przekazywanie argumentów do funkcji
  - funkcje zwracające wartość