Politechnika Wrocławska

Wstęp do programowania

INP001126W

rok akademicki 2021/22

semestr zimowy

Wykład 2

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220

Politechnika Wrocławska

Plan prezentacji (1)

- Projektowanie programu
- Funkcja `print()`
- Ciągi znakowe
- Komentarze
- Zmienne
- Typy danych
- Funkcja `input()`
- Funkcje

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

Politechnika Wrocławska

Plan prezentacji (2)

- Operatory matematyczne
  - kolejność wykonywania działań
  - zmienne w wyrażeniach matematycznych
  - typ danych wyniku
- Funkcje
  - projektowanie programu używającego funkcji
  - przekazywanie argumentów
  - funkcje zwracające wartość

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

Politechnika Wrocławska

Projektowanie programu

- Projektowanie programu można sprowadzić do dwóch kroków:
  1. Określenie, jakie zadania ma wykonywać program
  2. Określenie kroków, za pomocą których program wykona to zadanie

Projektowanie programu

Przykład

- Funkcjonalność: Program ma obliczać i wyświetlać wynagrodzenie pracownika
- Lista kroków:
  1. Pobranie liczby przepracowanych godzin
  2. Pobranie stawki godzinowej pracownika
  3. Pomnożenie liczby przepracowanych godzin przez stawkę godzinową
  4. Wyświetlenie wyniku działania z punktu 3.

Projektowanie programu

Pseudokod

```
Podaj liczbę przepracowanych godzin
Podaj stawkę godzinową
Oblicz wynagrodzenie, mnożąc liczbę
przepracowanych godzin przez stawkę
godzinową
Wyświetl obliczone wynagrodzenie
```

Projektowanie programu

Schemat blokowy

Projektowanie programu

Przepływ danych

Politechnika Wrocławska

Funkcja print()

- **Funkcja** to przygotowany fragment kodu przeprowadzający pewną operację
- Uruchomienie funkcji nazywa się **wywołaniem**
- W nawiasach umieszcza się **argument**

- Przykładowo wywołanie funkcji:

```
print('Hello world!')
```

spowoduje wypisanie napisu na ekran.

Politechnika Wrocławska

Funkcja print()

Przykłady

print01.py

File Edit Format Run Options Window Help

```
print('Tony Gaddis')
print('Python dla zupełnie początkujących')
print('Helion, 2019')
```

print02.py

File Edit Format Run Options Window Help

```
print("Tony Gaddis")
print("Python dla zupełnie początkujących")
print("Helion, 2019")
```

Funkcja print()

Przykłady

print_apostrophe.py

File Edit Format Run Options Window Help

```
print("Znak apostrofu ' można umieści, jeśli literał znakowy jest ujęty w cudzysłów.")
print('Tak wygląda znak cudzysłowu: " .')
print("""W tym łańcuch znakowym mamy ' i " """)
print('''oraz w tym: ' i " ''')
```

print_multiline.py

File Edit Format Run Options Window Help

```
print("""Jeden
Dwa
Trzy""")
```

Politechnika Wrocławska

Ciągi tekstowe

- Napis **`Hello world!`** w przykładowym wywołaniu funkcji

  ```
  print('Hello world!')
  ```

  jest **ciągiem tekstowym**.
- Gdy ciąg tekstowy pojawia się w kodzie programu nazywamy go **literałem znakowym**.

Politechnika Wrocławska

Ciągi tekstowe

- Ciąg tekstowy może być zamknięty w pojedynczy lub w podwójny cudzysłów

```
'Hello world!'
"Hello world!"
```

- Ciąg tekstowy obejmujący wiele linii zamyka się trzy razy powtórzonym znakiem cudzysłowu

```
"""Hello
world!"""
```

Politechnika Wrocławska

Komentarze

- Komentarz – informacja umieszczona w kodzie źródłowym programu, która objaśnia jego działanie
- Komentarze są ignorowane przez interpreter
- Komentarze ułatwiają zrozumienie kodu

Komentarze

Przykład 1

- Komentarz wyjaśnia przeznaczenie programu

comment01.py

File Edit Format Run Options Window Help

```
#Ten program wyświetla informację
#o polecanej literaturze
print('Tony Gaddis')
print('Python dla zupełnie początkujących')
print('Helion, 2019')
```

Komentarze

Przykład 2

- Komentarz na końcu wiersza objaśnia jego działanie

comment02.py

File Edit Format Run Options Window Help

```
print('Tony Gaddis')                          #Wyświetlenie informacji o autorze
print('Python dla zupełnie początkujących')   #Wyświetlenie tytułu
print('Helion, 2019')                         #Wyświetlenie wydawcy i roku wydania
```

Politechnika Wrocławska

Komentarze

- W trakcie pracy nad programem można wykorzystywać komentarze do wyłączania pewnych fragmentów kodu

Politechnika Wrocławska

Notki dokumentacyjne

- Literał znakowy, który pojawia się jako pierwszy element klasy, funkcji lub modułu, stanowi notkę dokumentacyjną – najczęściej taką notkę stanowi wielowierszowy literał znakowy

Politechnika Wrocławska

Zmienne

- **Zmienna** to miejsce w pamięci komputera reprezentowane przez określoną nazwę
- **Przypisanie** jest używane do utworzenia zmiennej i określenia jej jako odwołania do pewnego fragmentu danych

```
wiek = 19
```

Politechnika Wrocławska

Przypisanie

- Ogólna postać polecenia przypisania:

  `zmienna = wyrażenie`

- Znak równości jest **operatorem przypisania**

Politechnika Wrocławska

Zmienne

- Zmienne można przekazywać do funkcji **print()**

variables.py - C:/Users/karol/Desktop/201920/python-examples/zmienne/variables.py (...

File Edit Format Run Options Window Help

```
#Program pokazuje przykład użycia zmiennej.
sala = 249
print('Kurs "Wstęp do programowania" jest prowadzony w sali')
print(sala)
```

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/zmienne/variables.py
Kurs "Wstęp do programowania" jest prowadzony w sali
249
>>>
```

Politechnika Wrocławska

Zmienne

- **Zmiennej nie można używać przed przypisaniem jej wartości**

variables_error01.py

File Edit Format Run Options Window Help

```
#Program pokazuje ZŁY przykład użycia zmiennej.
print('Kurs "Wstęp do programowania" jest prowadzony w sali')
print(sala) #uzycie zmiennej przed przypisaniem jej wartości
sala = 249
```

- Należy uważać na literówki

variables_error02.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykład BŁĘDU przy użyciu zmiennej.
sala = 249
print('Kurs "Wstęp do programowania" jest prowadzony w sali')
print(Sala) #błędna nazwa zmiennej - jest: 'Sala' - powinno być: 'sala'
```

Politechnika Wrocławska

Zmienne

- Nazwą zmiennej nie mogą być słowa kluczowe Pythona
- Nazwa zmiennej nie może zawierać spacji
- Pierwszym znakiem musi być litera lub podkreślenie (_)
- Każdym kolejnym znakiem może być litera, cyfra lub podkreślenie
- Rozróżniana jest wielkość liter

Politechnika Wrocławska

Zmienne

- Nazwa zmiennej powinny wskazywać do czego służy dana zmienna
- Jeśli nazwa zmiennej ma się składać z kilku wyrazów to wygodnie jest je oddzielić znakiem podkreślenia

  `liczba_produktow` zamiast **`liczbaproduktow`**
- Inna możliwa konwencja, to rozpoczynanie kolejnych wyrazów wielką literą

  `liczbaProduktow`

Funkcja print()

Dwa argumenty wywołania

variables01.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykład wywołania funkcji print z dwoma argumentami.
sala = 249
print('Kurs "Wstęp do programowania" jest prowadzony w sali', sala)
```

Zmienne

Ponowne przypisanie

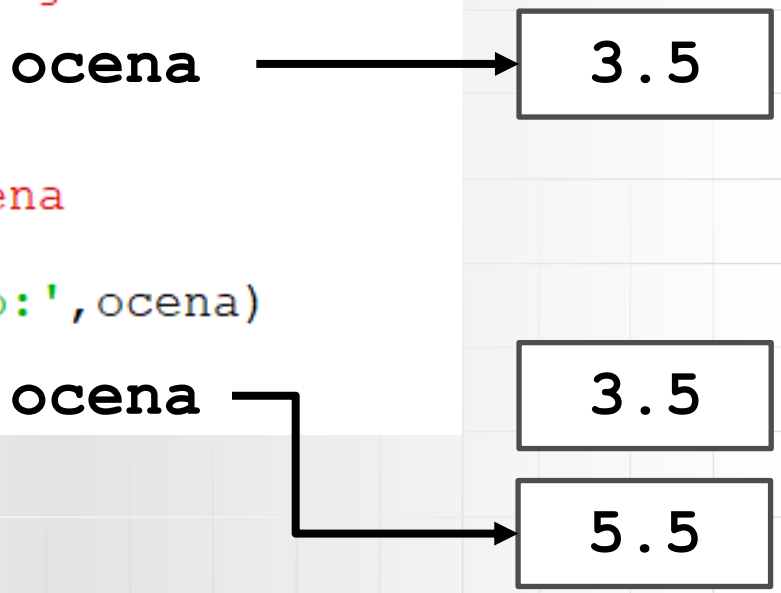
variables02.py

File Edit Format Run Options Window Help

```
#Program pokazuje ponowne przypisanie zmiennej.

#Wartość jest przypisywana zmiennej ocena
ocena = 3.5
print('Moja ocena to:', ocena)

#Ponowne przypisanie zmiennej ocena
ocena = 5.5
print('Moja ocena jest wyższa! To:',ocena)
```



Politechnika Wrocławska

Literały i liczbowe typy danych

- Poszczególne typy liczb przechowywane są w odmienny sposób
- Typy danych służą do kategoryzowania wartości w pamięci
- Wartości z przykładów:

```
sala = 249   # liczba całkowita
             # typ int   (integer)
ocena = 3.5  # liczba rzeczywista
             # typ float (floating-point)
```

Politechnika Wrocławska

Literały i liczbowe typy danych

- Liczba zapisana w kodzie programu nazywa się **literałem liczbowym**
- Literał liczbowy zapisany w postaci liczby całkowitej jest uznawany za typ **`int`**

```
249, -9, 7
```

- Literał liczbowy zapisany w postaci liczby z częścią ułamkową jest uznawany za typ **`float`**

```
1.5, 3.14, 5.0
```

Przypisanie zmiennej wartości innego typu

- Zmienna w Pythonie może odwoływać się do wartości dowolnego typu

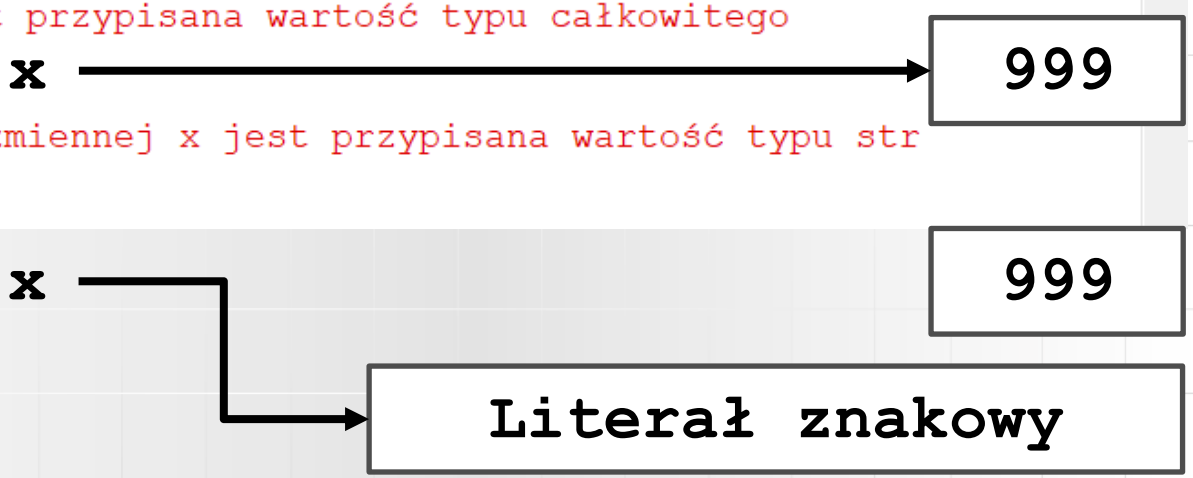
variables03.py

File Edit Format Run Options Window Help

```
#Program pokazuje przypisanie zmiennej wartości innego typu.

x = 999 # zmiennej x jest przypisana wartość typu całkowitego
print(x)

x = 'Literał znakowy' # zmiennej x jest przypisana wartość typu str
print(x)
```



Politechnika Wrocławska

Odczyt danych wejściowych

- Do pobierania danych wejściowych dostarczanych za pomocą klawiatury służy funkcja **`input()`**
- Przed pobraniem danych należy poinformować użytkownika, jakie dane są potrzebne
- Ogólna postać wywołania

`zmienna` **`= input(`*`ciąg tekstowy`*`)`**

- Przykładowo

```
name = input('Jak masz na imię?')
```

Odczyt danych wejściowych

input_string.py

File Edit Format Run Options Window Help

```
#Program pokazuje pobieranie łańcuchów znakowych z klawiatury

#Pobranie pseudonimu
nickname = input('Podaj pseudonim ')

#Wyświetlenie wiadomości powitalnej
print('Witaj,', nickname)
```

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/input_string.py =
==
Podaj pseudonim Karol
Witaj, Karol
>>>
```

Politechnika Wrocławska

Odczyt danych wejściowych

- Funkcja **`input()`** zawsze zwraca wprowadzone dane wejściowe jako ciąg tekstowy
- Aby zmienić typ danych (skonwertować dane) można wykorzystać funkcje **`int()`** oraz **`float()`**
- Przykładowe wykorzystanie funkcji **`int()`**

```
x = int(input('Podaj liczbę całkowitą'))
```

Politechnika Wrocławska

Funkcje

Korzyści z dzielenia programu na funkcje:

- Czytelniejszy kod
- Wielokrotne wykorzystanie kodu
- Lepsze testowanie (łatwiej testować podzadanie umieszczone w osobnej funkcji)
- Szybsze tworzenie oprogramowania
- Łatwiejsza praca w zespołach

Funkcje

Definiowanie i wywoływanie

- Prosty schemat definicji funkcji

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

nagłówek funkcji

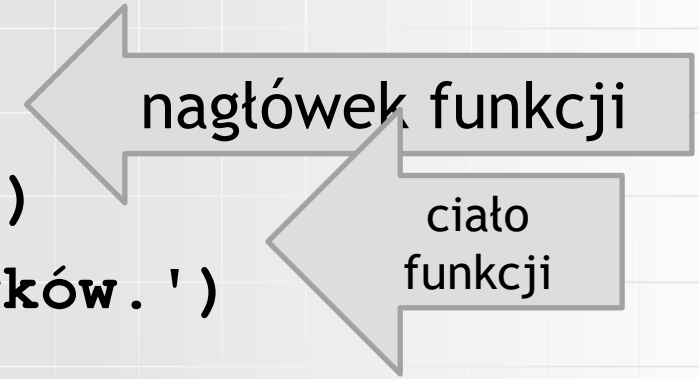
ciało funkcji

Funkcje

Definiowanie i wywoływanie

- Przykładowa definicja funkcji

```
def message():
    print('Jestem Artur,')
    print('król Brytyjczyków.')
```



Politechnika Wrocławska

Funkcje

Definiowanie i wywoływanie

- Wywołanie funkcji – zawsze z nawiasami okrągłymi

```
message()
```

Funkcje

Definiowanie i wywoływanie

function_demo.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową definicję funkcji i jej wywołanie

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

#Wywołanie funkcji message()
message()
```

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/function_dem
o.py
Jestem Artur,
król Brytyjczyków.
>>>
```

Funkcje

Definiowanie i wywoływanie

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()
```

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/two_function
s.py
Mam dla Ciebie wiadomość.
Jestem Artur,
król Brytyjczyków.
Żegnaj
>>>
```

Funkcje

Wcięcia

- Każdy wiersz bloku musi być wcięty
- Ostatni wcięty wiersz jest ostatnim blokiem kodu

function_demo.py

File Edit Format Run Options Window Help

```
#Program pokazuje przykładową definicję funkcji i jej wywołanie

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

#Wywołanie funkcji message()
message()
```

Politechnika Wrocławska

Funkcje

`main()`

- Python nie wyróżnia funkcji głównej
- Część programistów przyjmuje konwencję, w której tworzy się funkcję **`main()`**, ale nie jest to wymogiem języka

Politechnika Wrocławska

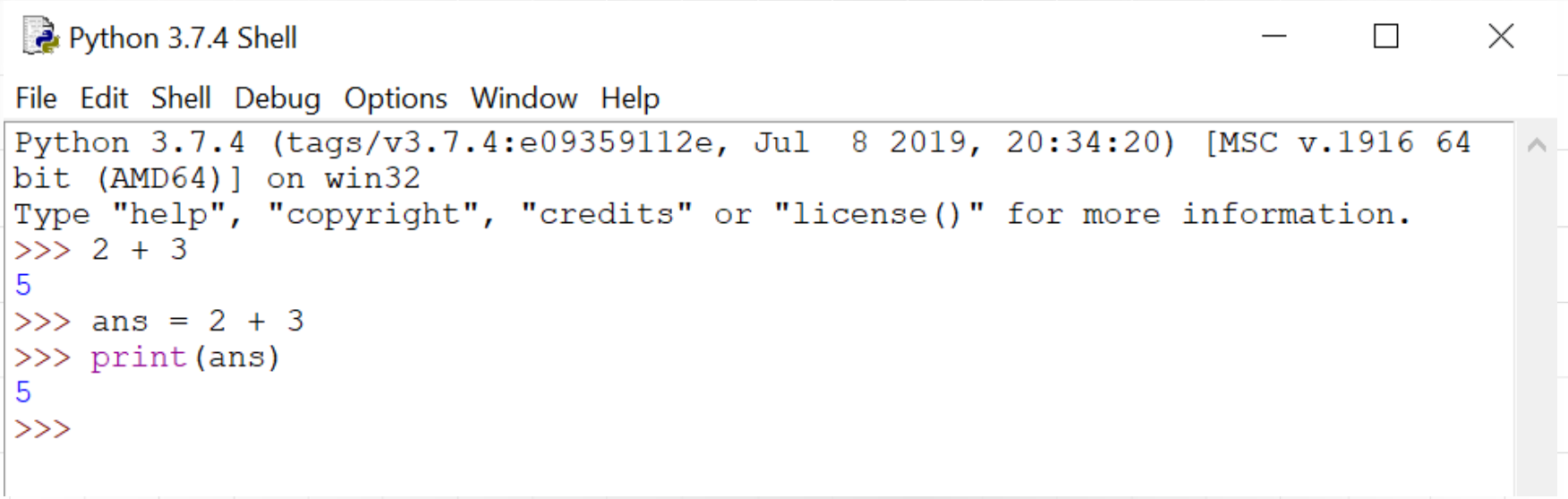
Operatory matematyczne

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

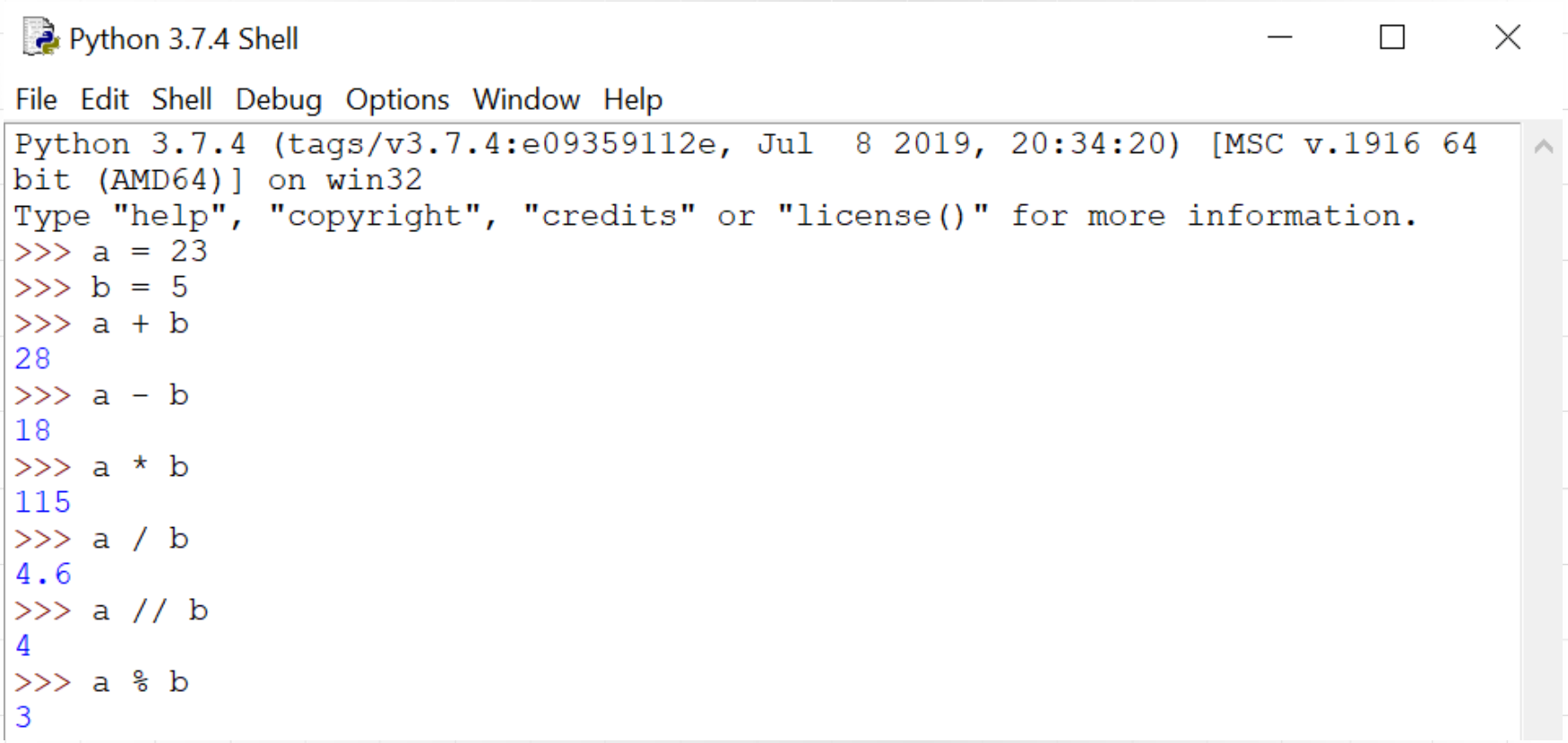
Kolejność wykonywania działań

```
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



Operatory matematyczne

Nawiasy

- niewłaściwie obliczona średnia

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

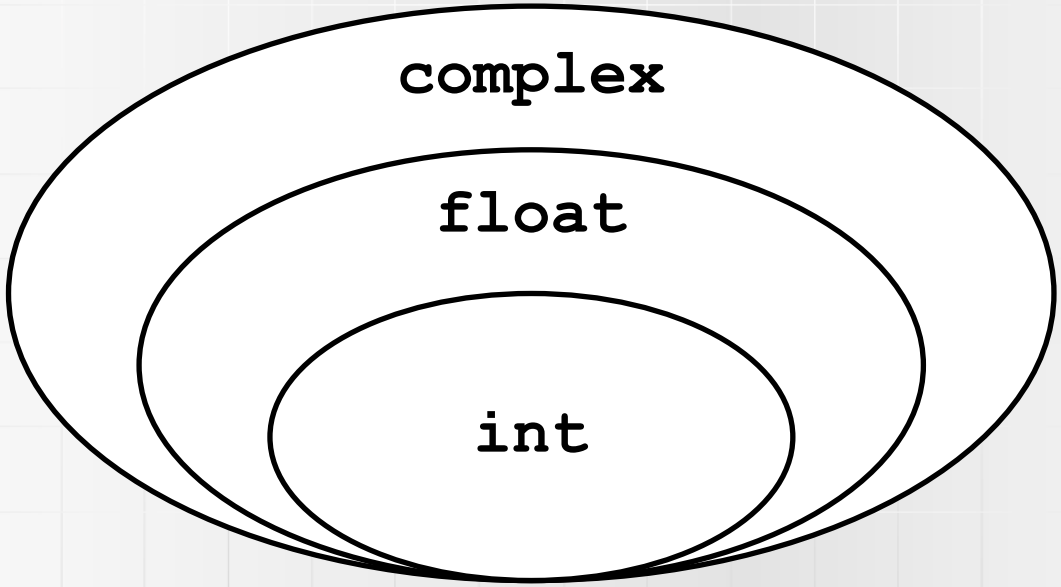
- dobre wyrażenie na średnią

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

Operatory matematyczne

Różne typy danych

- Typ danych wyniku zależy od typów danych argumentów
- Konwersja następuje do „szerszego” typu danych



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

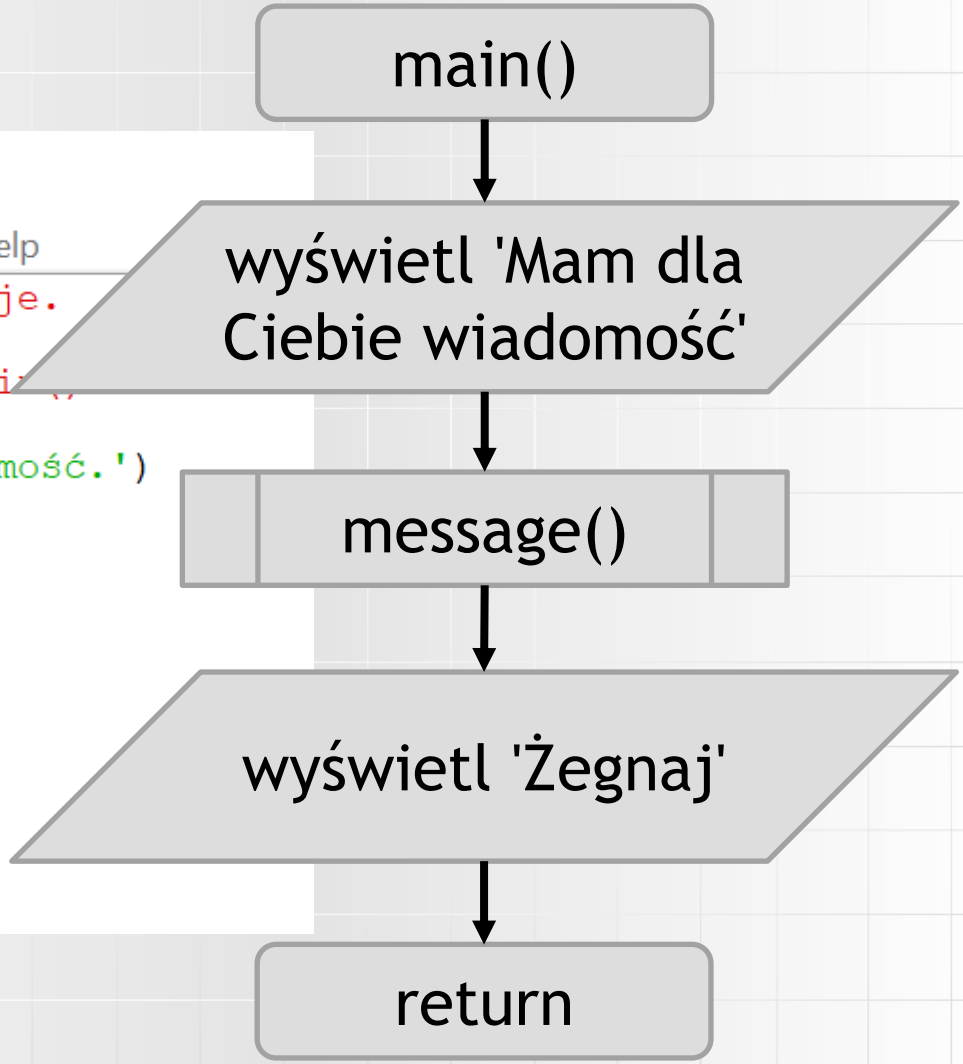
File Edit Format Run Options Window Help

```
#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

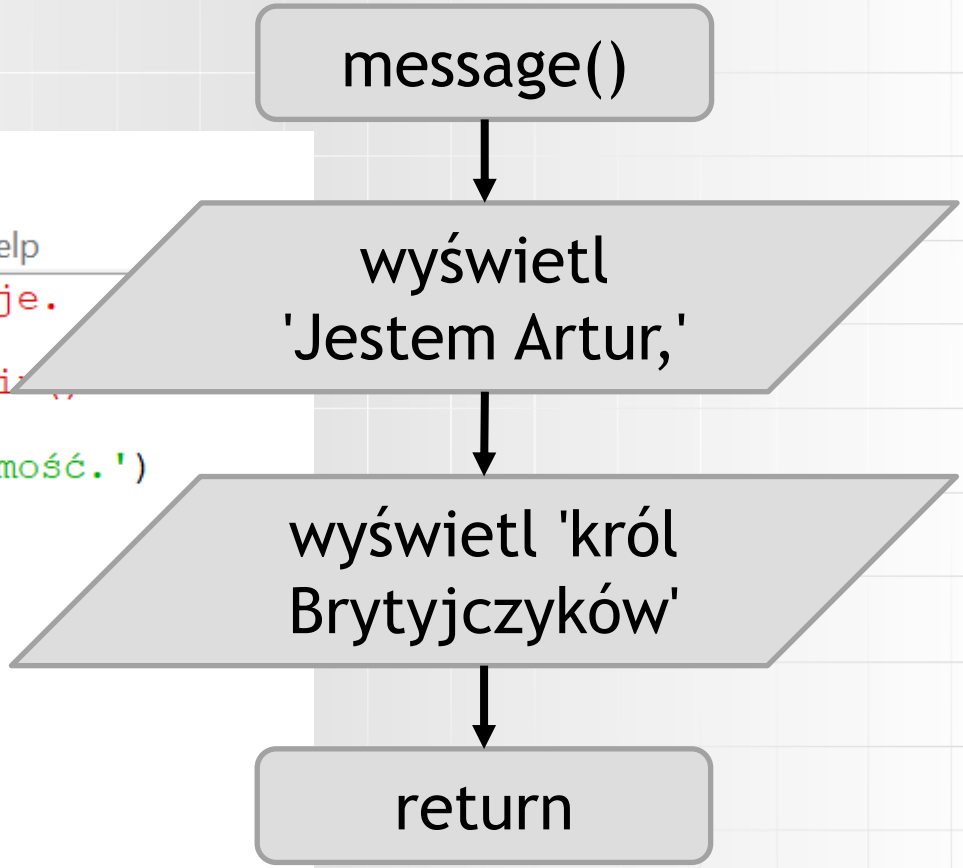
File Edit Format Run Options Window Help

```
#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

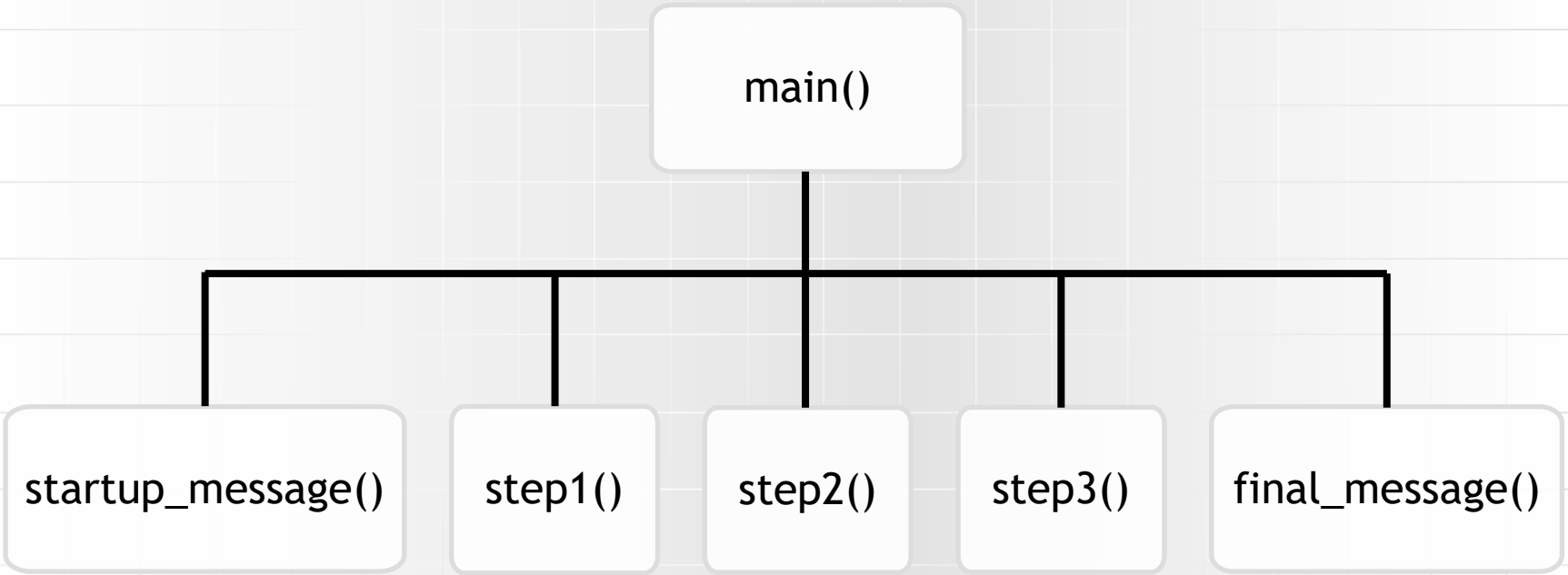
```
#Funkcja step3() wyświetla instrukcję dla kroku 3.
def step3():
    print('Krok 3.')
    print('Zamknij lodówkę.')
    print()

#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

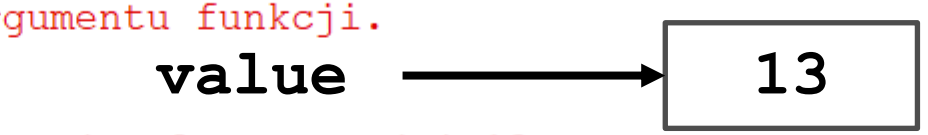
File Edit Format Run Options Window Help

```
#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

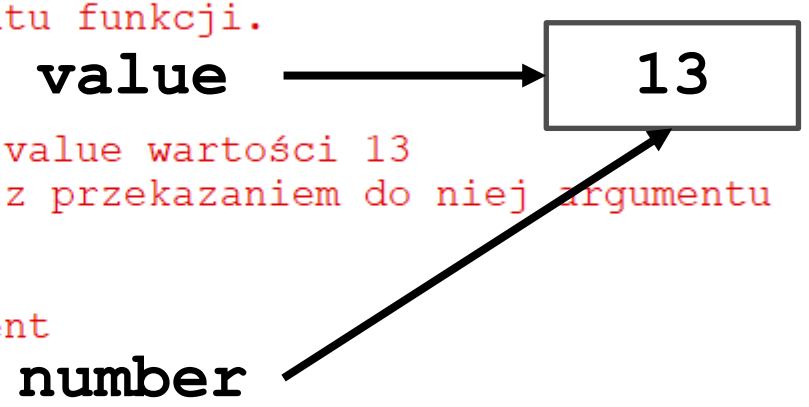
File Edit Format Run Options Window Help

```
#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

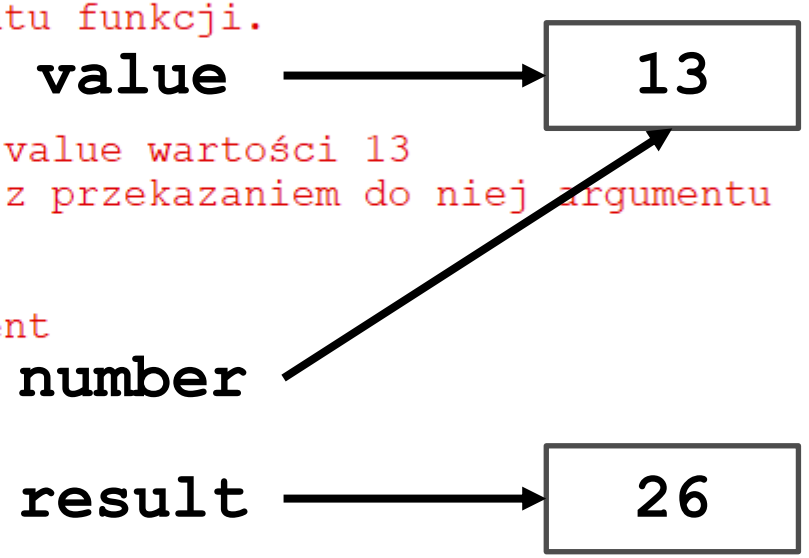
File Edit Format Run Options Window Help

```
#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

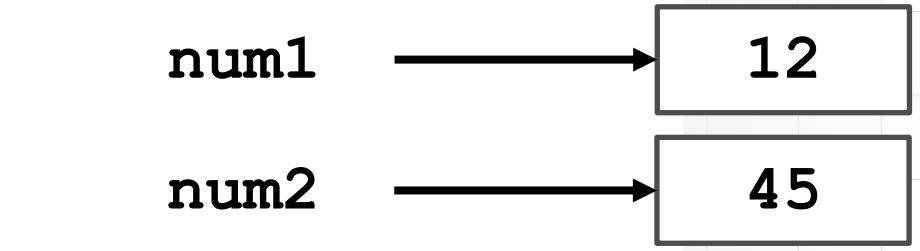
File Edit Format Run Options Window Help

```
#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

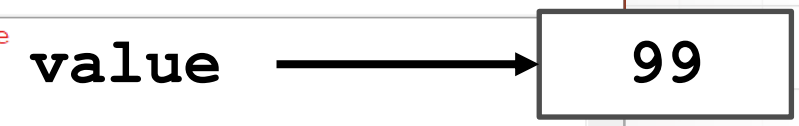
File Edit Format Run Options Window Help

```
#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

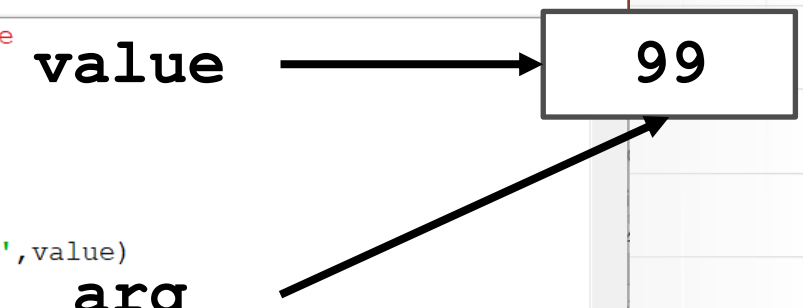
File Edit Format Run Options Window Help

```
#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

Przekazywanie argumentów do funkcji

- Przedstawiony sposób przekazywania argumentów do funkcji jest jednym z kilku możliwych (przez pozycje, przez nazwę, argumenty domyślne).

Politechnika Wrocławska

Funkcje

Funkcje zwracające wartość

- Prosta 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 (1)

- Przed przystąpieniem do pisania programu zaprojektuj jego działanie
- Używaj komentarzy
- Stosowanie funkcji podnosi czytelność kodu i ułatwia pisanie
- Stosowanie znanych konwencji również podnosi czytelność kodu

Politechnika Wrocławska

Absolutne minimum (2)

- 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