

# Wstęp do programowania

INP003203L

rok akademicki 2021/22

semestr zimowy

## Laboratorium 2

Karol Tarnowski

[karol.tarnowski@pwr.edu.pl](mailto:karol.tarnowski@pwr.edu.pl)

L-1 p. 220



# Plan prezentacji

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



# 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



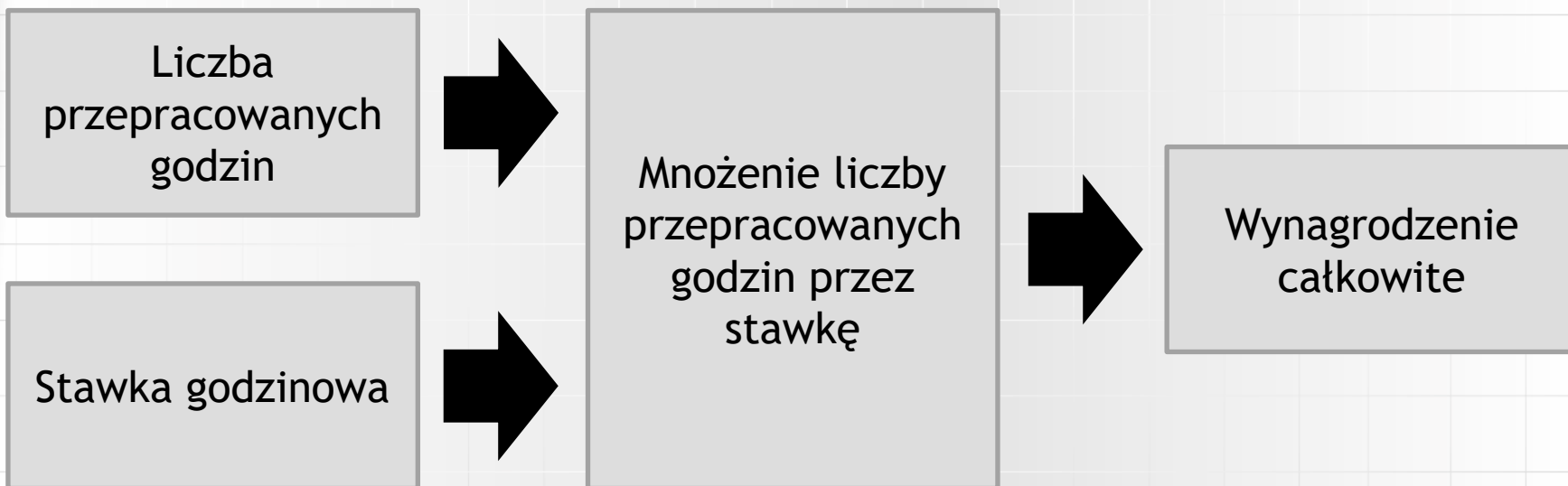
# Projektowanie programu

## Przepływ danych

Dane  
wejściowe

Przetwarzanie

Dane  
wyjściowe





# 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.



# Funkcja print()

## Przykłady

 print01.py

File Edit Format Run Options Window Help

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


 print02.py

File Edit Format Run Options Window Help

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


# Funkcja print()

## Przykłady

 print\_apostrophe.py

File Edit Format Run Options Window Help

```
print("Znak apostrofu ' można umieścić, 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""")
```



# Ciągi tekstowe

- W programach będą się pojawiały dane różnych typów
- Napis `Hello world!` w przykładowym wywołaniu funkcji  
`print('Hello world!')`  
jest **ciągami tekstowym**.
- Gdy ciąg tekstowy pojawia się w kodzie programu nazywamy go **literałem znakowym**.



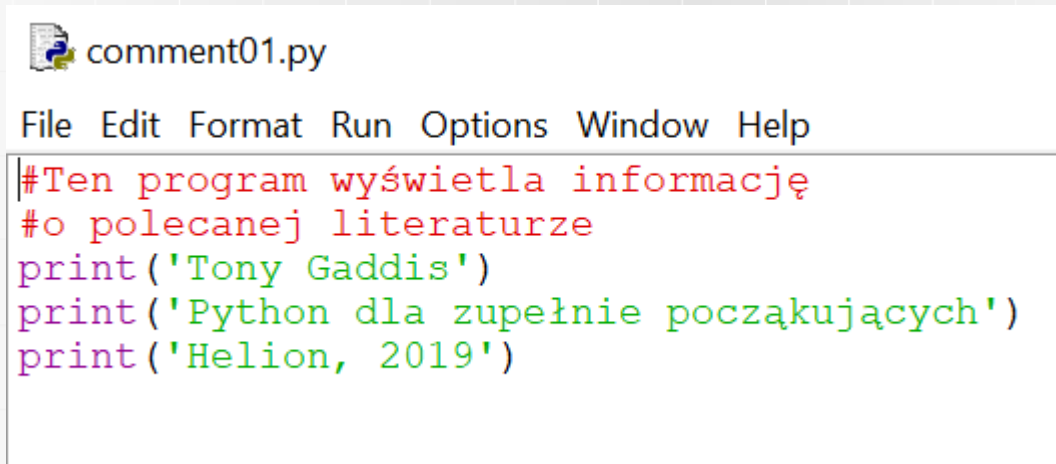
# Komentarze

- 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 począujących')
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
```



# Komentarze

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



# 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**





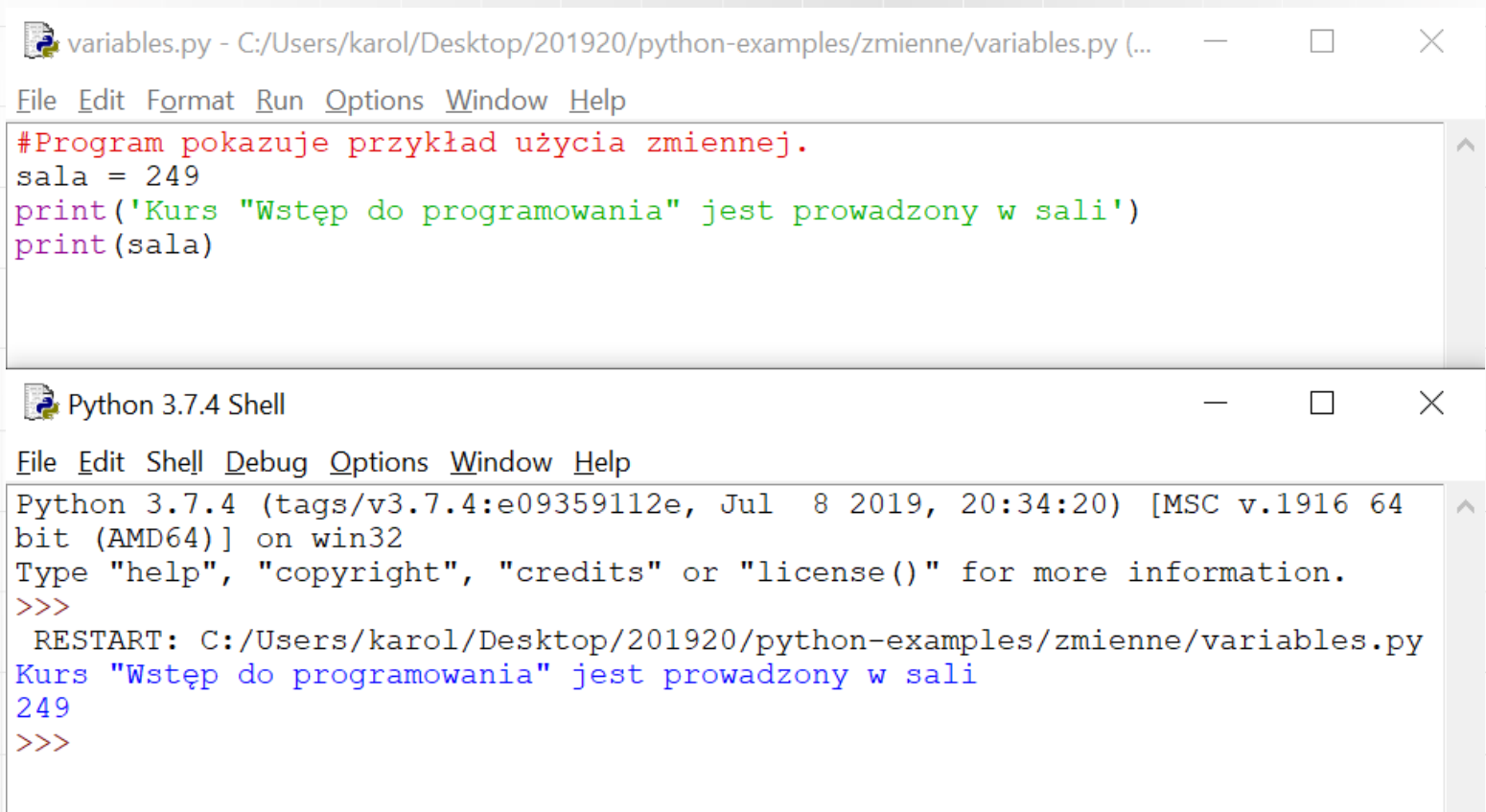


# Przypisanie

- Ogólna postać polecenia przypisania:  
**zmienna = wyrażenie**
- Znak równości jest operatorem przypisania

# Zmienne

- Zmienne można przekazywać do funkcji `print()`



The screenshot displays two windows from a Python IDE. The top window, titled 'variables.py', contains the following Python code:

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


The bottom window, titled 'Python 3.7.4 Shell', shows the execution output:

```
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  
>>>
```



# 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'
```



# 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



# 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)
```

**ocena**



3.5

```
#Ponowne przypisanie zmiennej ocena
```

```
ocena = 5.5
```

```
print('Moja ocena jest wyższa! To:', ocena)
```

**ocena**



3.5



5.5



# 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)
```





# 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**

999

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

**x**

999

Literał znakowy



# 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
>>>
```



# 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'))
```



# 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

- Ogólny schemat definicji funkcji

```
def nazwa_funkcji():
```

```
    polecenie
```

```
    polecenie
```

```
    itd.
```



nagłówek funkcji

blok poleceń

# Funkcje

## Definiowanie i wywoływanie

- Przykładowa definicja funkcji

```
def message():
```

```
    print('Jestem Artur,')
```

```
    print('król Brytyjczyków.')
```



nagłówek funkcji



blok  
poleceń



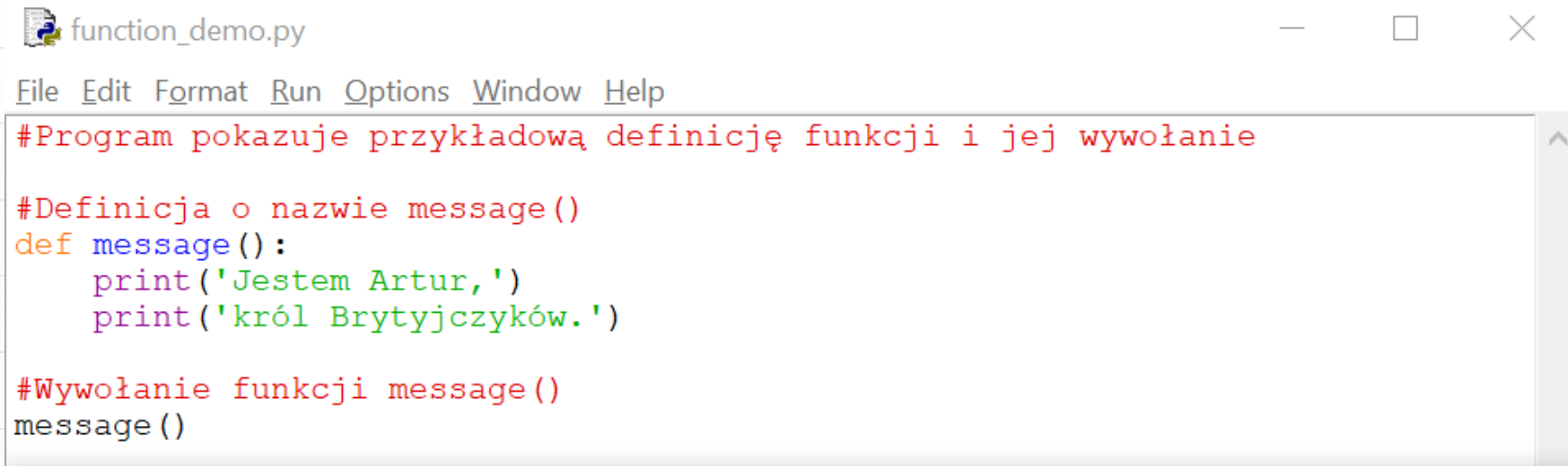
# Funkcje

## Definiowanie i wywoływanie

- Wywołanie funkcji  
`message ()`

# Funkcje

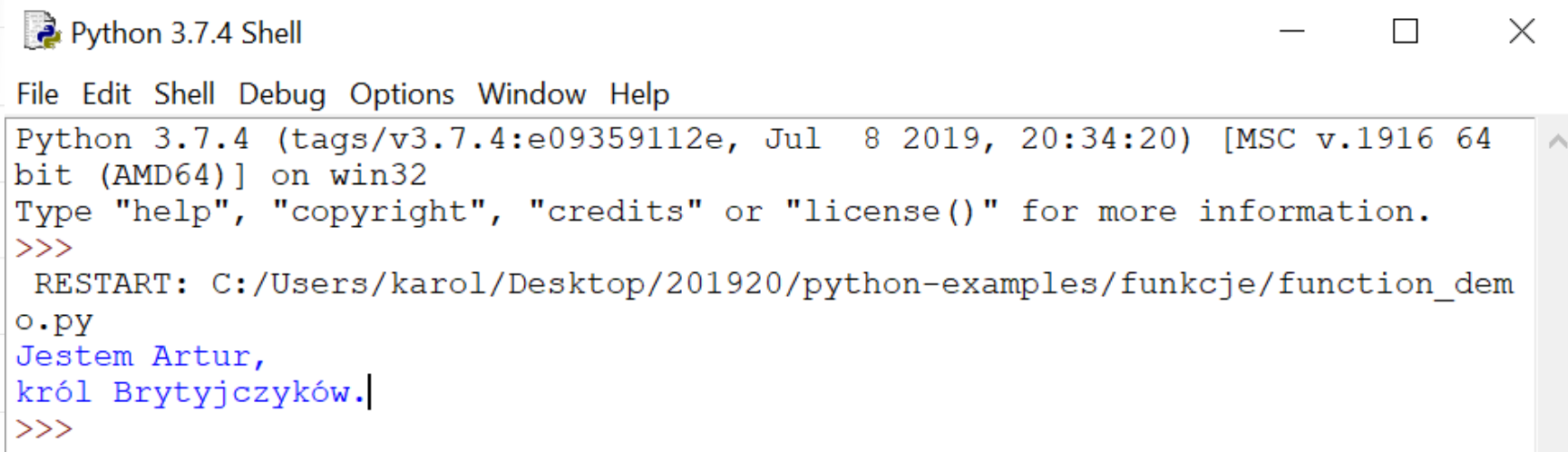
## Definiowanie i wywoływanie



```
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.')

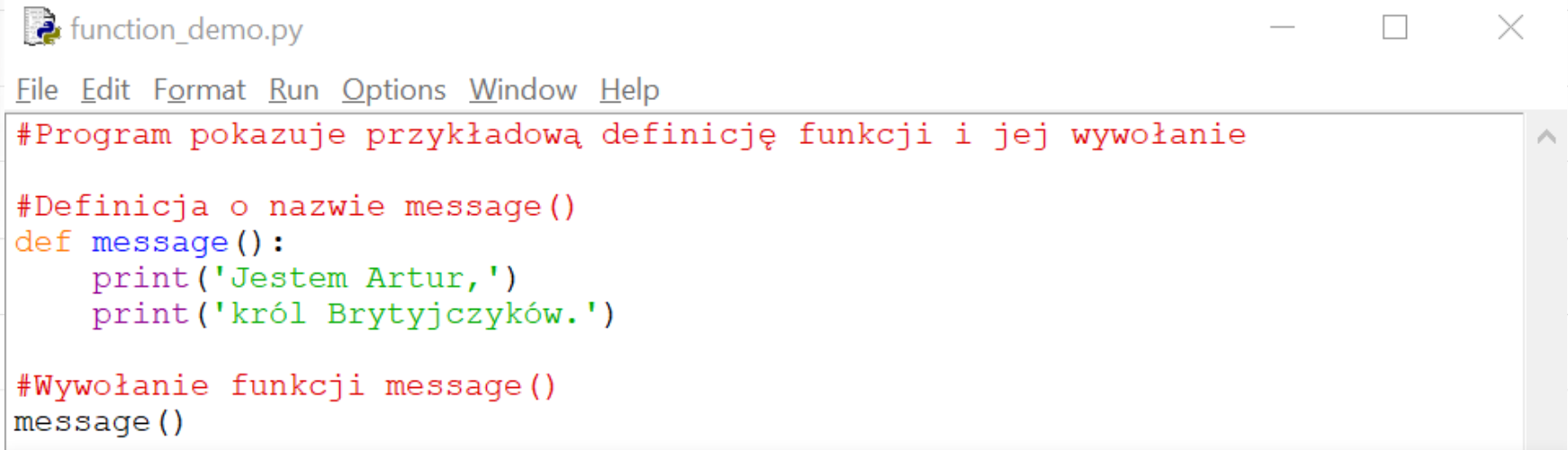
#Wywołanie funkcji głównej
main()
```

```
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()
```

# Funkcje

`main()`

- Tworzenie funkcji głównej nie jest wymogiem Pythona
- Tworzenie funkcji głównej jest konwencją, ale wykorzystywanie konwencji zwiększa czytelność kodu



# Absolutne minimum

- 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