

Wstęp do programowania

INP003203L

rok akademicki 2020/21

semestr zimowy

Laboratorium 12

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 220



Plan prezentacji

- Importowanie modułów
- Wybrane funkcje z modułu `random`
- Wybrane funkcje z modułu `math`
- Tworzenie własnych modułów



Importowanie modułów

- Python jest dostarczany z biblioteką standardową, która zawiera gotowe funkcje
- Przykładowo: `print()`, `input()`, `range()`
- Część funkcji wbudowana jest w interpreter
- Wiele funkcji biblioteki standardowej umieszczono w modułach



Importowanie modułów

- Moduły pomagają w organizacji biblioteki standardowej
- Przykładowo: funkcje matematyczne przechowywane są razem w module, funkcje obsługujące pliki razem w innym module.
- Aby skorzystać w funkcji bibliotecznej zawartej w module, należy zaimportować moduł poleceniem `import`.

```
import nazwa_modułu
```



Importowanie modułów

- W przykładzie wykorzystamy moduł `random`, który zawiera definicje wielu funkcji pozwalających na generowanie liczb losowych (właściwie pseudolosowych).



Importowanie modułów

01_import_random_randint.py

File Edit Format Run Options Window Help

```
# Program pokazuje importowanie modułu (random)
# w celu wywołania funkcji zdefiniowanej w nim (randint).
# Program pokazuje wywołanie funkcji randint().
# Funkcja randint(a, b) zwraca losową liczbę całkowitą
# z przedziału od a do b (włącznie).

# Program symuluje pięć rzutów kostką sześcienną.

import random #import modułu

def main():
    #wykonaj pięć razy
    for i in range(5):
        #rzut kostką (losowa liczba od 1 do 6)
        number = random.randint(1,6)
        #wyświetl wynik
        print(number)

main()
```



Importowanie modułów

01_import_random_randint.py

File Edit Format Run Options Window Help

```
# Program pokazuje importowanie modułu (random)
# w celu wywołania funkcji zdefiniowanej w nim (randint).
# Program pokazuje wywołanie funkcji randint().
# Funkcja randint(a, b) zwraca losową liczbę całkowitą
# z przedziału od a do b (włącznie).
```

```
# Program symuluje pięć rzutów kostką sześcienną.
```

```
import random #import modułu
```

```
def main():
    #wykonaj pięć razy
    for i in range(5):
        #rzut kostką (losowa liczba od 1 do 6)
        number = random.randint(1,6)
        #wyświetl wynik
        print(number)
```

```
main()
```

Wybrane funkcje z modułu `random`

- W module `random` dostępna jest funkcja `randint()`, która zwraca losową liczbę całkowitą z podanego przedziału.

Wybrane funkcje z modulu random

01_import_random_randint.py

File Edit Format Run Options Window Help

```
# Program pokazuje importowanie modulu (random)
# w celu wywołania funkcji zdefiniowanej w nim (randint).
# Program pokazuje wywołanie funkcji randint().
# Funkcja randint(a, b) zwraca losową liczbę całkowitą
# z przedziału od a do b (włącznie).

# Program symuluje pięć rzutów kostką sześcienną.

import random #import modulu

def main():
    #wykonaj pięć razy
    for i in range(5):
        #rzut kostką (losowa liczba od 1 do 6)
        number = random.randint(1,6)
        #wyświetl wynik
        print(number)

main()
```

Wybrane funkcje z modulu random

```
02_randint.py
File Edit Format Run Options Window Help
# Program symuluje serie rzutow moneta.
# Program wykorzystuje funkcje randint()
# z modulu random.

import random #import modulu

HEAD = 1      #stala oznaczajaca awers (orzel)
TAIL = 2      #stala oznaczajaca rewers (reszka)
TOSSES = 10   #stala okreslajaca liczbe rzutow

def main():
    # w kazdym rzucie
    for toss in range(TOSSES):
        # wylosuj liczbe (1 lub 2)
        # 1 oznacza awers (orzel)
        if random.randint(HEAD, TAIL) == HEAD:
            print('orzel')
        # w przeciwnym wypadku to rewers (reszka)
        else:
            print('reszka')

main()
```

Wybrane funkcje z modułu `random`

Inne funkcje dostępne w module `random`:

- `randrange()`
- `random()`
- `uniform()`

Wybrane funkcje z modulu random

03_random.py

File Edit Format Run Options Window Help

```
# Program pokazuje różne funkcje z modulu random.

import random

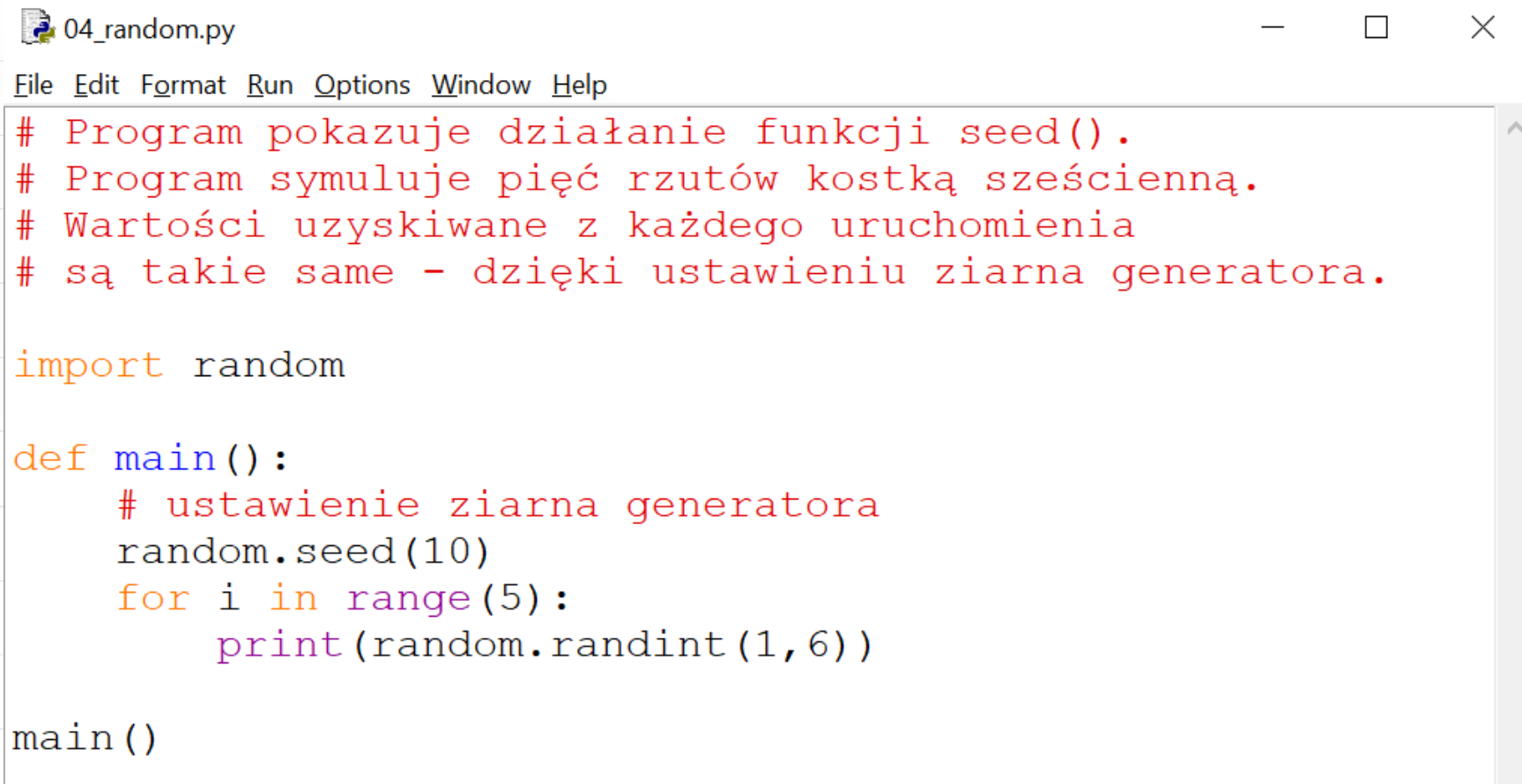
def main():
    #randrange - zwraca losową liczbę z listy
    print(random.randrange(6))          #losowa wartość od 0 do 5
    print(random.randrange(7,10))      #losowa wartość od 7 do 9
    print(random.randrange(0,101,10))  #losowa wartość z listy od 0 co 10 do 100
    #random - zwraca losową liczbę z przedziału [0.0, 1.0)
    print(random.random())
    #uniform - zwraca losową liczbę z przedziału [a, b]
    print(random.uniform(3.0,5.0))     #losowa wartość z przedziału [3, 5]

main()
```

Wybrane funkcje z modułu `random`

- Liczby pseudolosowe generowane są jako wyrazy deterministycznego ciągu bazującego na pewnej informacji początkowej (nazywanej ziarnem lub załączkiem).
- Wartość załączka można zadać funkcją `seed()`.

Wybrane funkcje z modulu random



```
04_random.py
File Edit Format Run Options Window Help
# Program pokazuje dzialanie funkcji seed().
# Program symuluje piec rzutow kostka szezcienna.
# Wartości uzyskiwane z kazdego uruchomienia
# są takie same - dzięki ustawieniu ziarna generatora.

import random

def main():
    # ustawienie ziarna generatora
    random.seed(10)
    for i in range(5):
        print(random.randint(1, 6))

main()
```

Wybrane funkcje z modułu `math`

- Moduł `math` zawiera funkcje matematyczne
- Przykładowo, w module `math` dostępna jest funkcja `sqrt()`, obliczająca pierwiatek kwadratowy.
- Przykładowo, w module `math` dostępna jest zmienna `pi`, przechowująca przybliżenie liczby π .

Wybrane funkcje z modulu math

05_import_math.py

File Edit Format Run Options Window Help

```
# Program pokazuje dzialanie funkcji sqrt()
# z modulu math.
```

```
#importowanie modulu math
import math
```

```
def main():
    #wczytanie liczby rzeczywistej
    number = float(input('Podaj liczbe: '))
    #obliczenie pierwiastka
    #funkcja sqrt() z biblioteki math
    square_root = math.sqrt(number)
    #wyswietlenie wynikow
    print('Pierwiastek kwadratowy liczby',
          number, 'wynosi', square_root)
```

```
main()
```




Wybrane funkcje z modułu math

```
06_math_pi.py
File Edit Format Run Options Window Help
# Program oblicza pole koła o podanym promieniu.
# Program wykorzystuje wielkość pi z modułu math.

import math

def main():
    #wczytanie liczby rzeczywistej
    radius = float(input('Podaj promień koła: '))
    area = math.pi * radius ** 2
    #wyświetlenie wyników
    print('Pole koła o promieniu',
          radius, 'wynosi', area)

main()
```

Wybrane funkcje z modulu `math`

Funkcja	Opis
<code>acos(x)</code>	arcus cosinus x (w radianach)
<code>asin(x)</code>	arcus sinus x (w radianach)
<code>atan(x)</code>	arcus tan x (w radianach)
<code>ceil(x)</code>	zaokrąglenie w górę (sufit)
<code>cos(x)</code>	cosinus dla x (wyrażonego w radianach)
<code>exp(x)</code>	e^x
<code>floor(x)</code>	zaokrąglenie w dół (podłoga)
<code>log(x)</code>	logarytm naturalny x
<code>log10(x)</code>	logarytm dziesiętny x
<code>sin(x)</code>	sinus dla x (wyrażonego w radianach)
<code>sqrt(x)</code>	pierwiastek kwadratowy z x
<code>tan(x)</code>	tangens dla x (wyrażonego w radianach)



Tworzenie własnych modułów

- Złożone programy powinny być podzielone na funkcje
- Kod programu zawierającego wiele funkcji można podzielić na moduły
- Przykładowy program oblicza pola i obwody kół i prostokątów



Tworzenie własnych modułów

- Funkcje obsługujące koło zebrano w module (plik `circle.py`).
- Funkcje obsługujące prostokąt zebrano w module (plik `rectangle.py`)
- Program importuje te moduły



Tworzenie własnych modułów

```
circle.py
File Edit Format Run Options Window Help
# Moduł circle.py
# Zawiera definicje funkcji służących do obliczania:
# pola i obwodu koła oraz do wczytywania promienia.

# import modułu math - wykorzystywana wartość math.pi
import math

# Funkcja oblicza pole koła.
# I: promień
# P: oblicza pole z zależności: pole = pi*r^2
# O: obliczona wartość pola
def area(radius):
    return math.pi * radius ** 2

# Funkcja oblicza obwód koła.
# I: promień
# P: oblicza obwód z zależności: obwód = 2*pi*r
# O: obliczona wartość obwodu
def circumference(radius):
    return 2 * math.pi * radius

# Funkcja pobiera promień koła od użytkownika.
# I: brak
# P: wczytanie liczby rzeczywistej podanej przez użytka
# O: promień
def get_radius():
    return float(input('Podaj promień: '))
```



Tworzenie własnych modułów

```
rectangle.py
File Edit Format Run Options Window Help
# Moduł rectangle.py
# Zawiera definicje funkcji służących do obliczania:
# pola i obwodu prostokąta oraz
# do wczytywania jego wymiarów.

# Funkcja oblicza pole prostokąta.
# I: długość i szerokość
# P: oblicza pole z zależności: pole = długość*szerokość
# O: obliczona wartość pola
def area(length, width):
    return length * width

# Funkcja oblicza obwód prostokąta.
# I: długość i szerokość
# P: oblicza obwód z zależności: obwód = 2*(długość+szerokość)
# O: obliczona wartość obwodu
def perimeter(length, width):
    return 2 * (length + width)

# Funkcja pobiera wymiary prostokąta od użytkownika.
# I: brak
# P: wczytanie liczby rzeczywistej podanej przez użytkownika
# O: (długość, szerokość)
def get_dimensions():
    length = float(input('Podaj długość prostokąta: '))
    width = float(input('Podaj szerokość prostokąta: '))
    return (length, width)
```



Tworzenie własnych modułów

07_geometry.py

File Edit Format Run Options Window Help

```
# Program pokazuje importowanie własnych modułów.  
# Program wykorzystuje pliki circle.py oraz rectangle.py.  
#  
# Plik circle.py zawiera definicje funkcji:  
# - area(radius),  
# - circumference(radius),  
# - get_radius().  
#  
# Plik rectangle.py zawiera definicje funkcji:  
# - area(length, width),  
# - perimeter(length, width),  
# - get_dimensions().  
  
#importowanie własnych modułów  
import circle  
import rectangle
```



Tworzenie własnych modułów

07_geometry.py

File Edit Format Run Options Window Help

```
#definicje stałych do obsługi menu
AREA_CIRCLE_CHOICE      = 1
CIRCUMFERENCE_CHOICE   = 2
AREA_RECTANGLE_CHOICE   = 3
PERIMETER_RECTANGLE_CHOICE = 4
QUIT_CHOICE             = 5
```

```
def main():
```

```
    print('Program geometria pozwala obliczyć pole i obwód koła i prostokąta.')
```

```
    # W pętli while pobierana jest od użytkownika liczba całkowita
    # odpowiadająca wybranej czynności (od 1 do 5), a następnie
    # w bloku if-elif-else odczytywane są dane i wykonywane obliczenia.
    choice = 0 #inicjalizacja zmiennej choice
```

```
    while choice != QUIT_CHOICE:
```

```
        display_menu() #wyświetlenie menu
```

```
        #pobranie wartości od użytkownika
```

```
        choice = int(input('Wybierz opcję: '))
```

```
    #wybór czynności
```

```
    if choice == AREA_CIRCLE_CHOICE:                #pole koła
```

```
        radius = circle.get_radius()
```

```
        print('Pole koła wynosi', circle.area(radius))
```

```
    elif choice == CIRCUMFERENCE_CHOICE:            #obwód koła
```

```
        radius = circle.get_radius()
```

```
        print('Obwód koła wynosi', circle.circumference(radius))
```

```
    elif choice == AREA_RECTANGLE_CHOICE:           #pole prostokąta
```

```
        (length, width) = rectangle.get_dimensions()
```

```
        print('Pole prostokąta wynosi', rectangle.area(length, width))
```

```
    elif choice == PERIMETER_RECTANGLE_CHOICE:      #obwód prostokąta
```

```
        (length, width) = rectangle.get_dimensions()
```

```
        print('Obwód prostokąta wynosi', rectangle.perimeter(length, width))
```

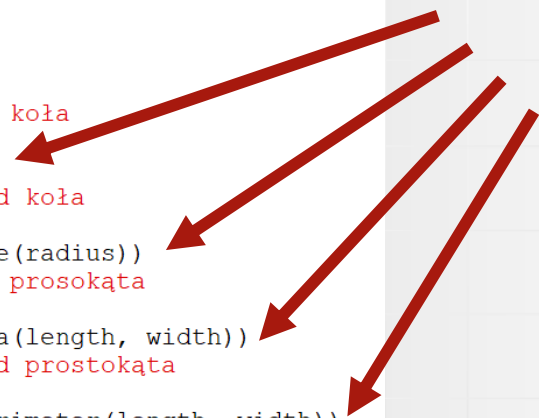
```
    elif choice == QUIT_CHOICE:                      #koniec programu
```

```
        print('Zakończenie działania programu.')
```

```
    else:                                             #wartość spoza listy
```

```
        print('Błąd: nieprawidłowa opcja.')
```

Wykorzystanie funkcji
z modułów **circle**
oraz **rectangle**





Tworzenie własnych modułów

- Przykładowy program, wykorzystuje pętlę **while** oraz instrukcję **if-elif-else** do obsługi menu



Absolutne minimum

- Importowanie modułów biblioteki standardowej
- Znajomość wybranych funkcji z modułów **random** oraz **math**
- Organizowanie kodu własnych programów z wykorzystaniem modułów