

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/350374014>

# Python. Zadania z programowania. Przykładowe imperatywne rozwiązania

Book · March 2021

CITATIONS

0

READS

5,247

1 author:



[Mirosław J. Kubiak](#)

Independent Researcher

25 PUBLICATIONS 2 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Zadania z programowania [View project](#)

\_ Mirośław J. Kubiak \_

# PYTHON

**ZADANIA Z PROGRAMOWANIA**

**Przykładowe  
imperatywne rozwiązania**



Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn  
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/pyzaim>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-7254-2

Copyright © Helion 2021

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

# Spis treści

<b>Od autora</b>	<b>7</b>
<b>Wstęp</b>	<b>11</b>
<b>Rozdział 1. W jaki sposób Python komunikuje się z użytkownikiem</b>	<b>15</b>
Podstawowe operacje wejścia – wyjścia	15
<b>Rozdział 2. Struktury warunkowe i operatory logiczne</b>	<b>25</b>
Struktury warunkowe	25
Konstrukcja if	25
Trójargumentowa instrukcja warunkowa	27
Operatory logiczne	31
Operator trójargumentowy	34
Porównywanie ciągów tekstowych	35
<b>Rozdział 3. Struktury cykliczne, czyli wielokrotne wykonywanie fragmentu kodu</b>	<b>37</b>
Pętle warunkowe i licznikowe	37
<b>Rozdział 4. Funkcje</b>	<b>55</b>
Wprowadzenie do funkcji	55
Definiowanie i wywoływanie funkcji	56
Moduły	60
Rekurencja	68
Funkcje anonimowe — lambda	77
Funkcje — cd.	78
<b>Rozdział 5. Klasy i programowanie zorientowane obiektowo</b>	<b>87</b>
Wprowadzenie	87
Podstawy paradygmatu obiektowego	88

	Obiekty i klasy	88
	Klasa Osoba	99
	Dziedziczenie	101
	Przeciążanie metod	104
<b>Rozdział 6.</b>	<b>Pliki</b>	<b>107</b>
	Wprowadzenie	107
	Typy plików	108
	Serializacja danych — zapis i odczyt danych binarnych	125
<b>Rozdział 7.</b>	<b>Wyjątki</b>	<b>131</b>
	Instrukcja try-except	131
<b>Rozdział 8.</b>	<b>Dekoratory i menedżery kontekstu</b>	<b>137</b>
	Dekoratory	137
	Menedżery kontekstu	141
<b>Rozdział 9.</b>	<b>Iteratory i generatory</b>	<b>143</b>
	Zalety iteratorów i generatorów	143
	Iteratory	145
	Funkcja next()	145
	Iterator range()	149
	Generatory	150
<b>Rozdział 10.</b>	<b>Funkcje wyższego rzędu</b>	<b>157</b>
	<b>Bibliografia</b>	<b>163</b>

## Rozdział 1.

# W jaki sposób Python komunikuje się z użytkownikiem

*W tym rozdziale omawiam sposób, w jaki język Python komunikuje się z użytkownikiem.*

## Podstawowe operacje wejścia – wyjścia

Istnieje kilka sposobów prezentacji wyników programu. Na przykład dane można wydrukować w formie czytelnej dla człowieka lub zapisać do pliku, aby je wykorzystać w przyszłości. W tym rozdziale omówię niektóre podstawowe możliwości języka Python.

Każda aplikacja powinna mieć możliwość komunikowania się z użytkownikiem. Za pomocą prostych przykładów pokażę, w jaki sposób napisany w Pythonie program komunikuje się z użytkownikiem poprzez **standardowe operacje wejścia – wyjścia**.

Podstawowe operacje wejścia – wyjścia w języku Python realizowane są za pomocą dwóch poleceń (funkcji):

- ◆ `print()`, która służy do wypisywania wartości, np. na ekranie komputera;
- ◆ `input()`, która służy do odczytywania wartości podanych przez użytkownika, np. z klawiatury.

Przystępuję teraz do przedstawienia zadań i ich przykładowych rozwiązań w języku Python.

---

**Zadanie**

**1.1** Napisz program, który oblicza pole prostokąta. Wartości boków *a* i *b* są typu `float` i należy je wprowadzić z klawiatury. Wynik działania programu należy wprowadzić na ekran komputera.

---

*Przykładowe rozwiązanie — listing 1.1*

---

```
# Zadanie 1.1.
# To jest komentarz.

print("Program oblicza pole prostokąta.")

a = float(input("Podaj bok a = ")) # Czytanie z klawiatury liczby
↳rzeczywistej a.
b = float(input("Podaj bok b = ")) # Czytanie z klawiatury liczby
↳rzeczywistej b.

print() # Wyświetlenie pustej linii.

pole = a * b # Obliczanie pola prostokąta.

print("Dla a =", a, "i b =", b) # Wyświetlenie zmiennych a i b.
print("pole prostokąta =", pole) # Wyświetlenie zmiennej pole.
```

---

Przeanalizuję teraz program linijka po linijce.

Komentarze w języku Python zaczynają się od znaku „#” i kończą z końcem wiersza.

```
# Zadanie 1.1.
# To jest komentarz.
```

Komentarze wieloliniowe można wstawiać do programu w postaci wielowierszowych *stringów* (ograniczonych przez `"""` lub `'''`) bez żadnych działań (np. przypisań); *stringi* te nie są traktowane jako wyrażenia i są pomijane przez interpreter (lub kompilator).

```
'''
To jest wieloliniowy
komentarz.
'''
```

Wywołanie funkcji `input()` i pobranie zmiennej a wpisanej za pomocą klawiatury przedstawia następująca linijka kodu:

```
a = float(input("Podaj bok a = ")) # Czytanie z klawiatury
↳ liczby rzeczywistej a.
```

Argument przekazany funkcji `float()` zostanie przekonwertowany na postać liczby zmiennoprzecinkowej (typ `float`). Zmienna `a` jest typu `float`.

Język Python obsługuje cztery różne typy liczbowe:

- ♦ `int` (liczby całkowite, które mogą być reprezentowane również w systemach ósemkowym i szesnastkowym);
- ♦ `long` (długie liczby całkowite);
- ♦ `float` (liczby rzeczywiste zmiennoprzecinkowe);
- ♦ `complex` (liczby zespolone).

Analogicznie wprowadzamy zmienną `b`. Obie zmienne są typu rzeczywistego. Zmienna `pole` oblicza pole prostokąta według wzoru:

```
pole = a * b # Obliczanie pola prostokąta.
```

Polecenie

```
print() # Wyświetlenie pustej linii.
```

wyświetla na ekranie komputera pustą linię.

Wyświetlenie wartości zmiennych `a` i `b` oraz `pole` na ekranie komputera umożliwiającą następujące linijki kodu:

```
print("Dla a =", a, "i b =", b) # Wyświetlenie zmiennych
↳ a i b.
print("pole prostokąta =", pole) # Wyświetlenie zmiennej
↳ pole.
```

Rezultat działania programu można zobaczyć na rysunku 1.1.

**Rysunek 1.1.** *Efekt działania programu*  
Zadanie 1.1

```
Program oblicza pole prostokąta.
Podaj bok a = 1.02
Podaj bok b = 2

Dla a = 1.02 i b = 2.0
pole prostokąta = 2.04
Press any key to continue . . .
```





Uwaga

Napis `Press any key to continue . . .` (Naciśnij dowolny klawisz, aby kontynuować...) będzie pojawiał się w dalszej części książki **wyłączenie** na ekranie komputera.

Program powyżej można przedstawić trochę inaczej, stosując tzw. sformatowany łańcuch znaków<sup>1</sup> (ang. *Formatted String Literals*) lub f-ciąg. Dzięki niemu kod jest bardziej czytelny niż w poprzednich wersjach Pythona.

```
# Zadanie 1.1a.
```

```
print("Program oblicza pole prostokąta.")
```

```
a = float(input("Podaj bok a = ")) # Czytanie z klawiatury liczby rzeczywistej a.
```

```
b = float(input("Podaj bok b = ")) # Czytanie z klawiatury liczby rzeczywistej b.
```

```
print() # Wyświetlenie pustej linii.
```

```
pole = a * b # Obliczanie pola prostokąta.
```

```
print(f"Dł a = {a} i b = {b}") # Wyświetlenie zmiennych a i b.
```

```
print(f"pole prostokąta = {pole}.") # Wyświetlenie zmiennej pole.
```

Teraz wartości zmiennych `a` i `b` oraz `pole` umożliwiają wyświetlenie na ekranie komputera następującej linii kodu:

```
print(f"Dł a = {a} i b = {b}") # Wyświetlenie zmiennych
```

```
↳ a i b.
```

```
print(f"pole prostokąta = {pole}.") # Wyświetlenie
```

```
↳ zmiennej pole.
```

Rezultat działania programu można zobaczyć na rysunku 1.1a.

W dalszej części książki, w ramach ćwiczeń, oba formaty wyprowadzania danych na ekran komputera będę stosował wymiennie.

---

<sup>1</sup> Sformatowany łańcuch znaków, lub krócej: f-łańcuch, jest ciągiem znaków poprzedzonych przedrostkami `f` lub `F`. Ciągi te mogą zawierać pola zastępcze, które są wyrażeniami ograniczonymi nawiasami klamrowymi `{}`. Podczas gdy inne literały ciągów zawsze mają stałą wartość, sformatowane ciągi są naprawdę wyrażeniami obliczanymi w czasie wykonywania. Sformatowany łańcuch znaków stosowany jest od wersji Pythona 3.6.

**Rysunek 1.1a.**

*Efekt działania programu*  
*Zadanie 1.1a*

```
Program oblicza pole prostokąta.  
Podaj bok a = 1.02  
Podaj bok b = 2  
  
Dla a = 1.02 i b = 2.0  
pole prostokąta = 2.04.
```

**Zadanie****1.2**

Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej  $\pi = 3.14\dots$  Należy przyjąć format wyświetlania tej stałej z dokładnością do trzech miejsc po przecinku.

**Wskazówka**

Do programu dołącz moduł `math`<sup>2</sup>.

*Przykładowe rozwiązanie — listing 1.2*

```
# Zadanie 1.2.  
  
import math # Dołączamy moduł math.  
  
print("Program wyświetla stałą pi z zadaną dokładnością.")  
print("pi = %5.3f"% math.pi, ".", sep = "")
```

Linijka kodu w programie:

```
print("pi = %5.3f"% math.pi, ".", sep = "")
```

oznacza, że do wyświetlenia na ekranie liczby  $\pi$  przeznaczono 5 pól, w tym 3 pola na część ułamkową. Natomiast separator `sep = ""` służy do formatowania ciągów znaków wyjściowych<sup>3</sup>.

Moduł `math`, który należy do programu zaimportować poleceniem

```
import math # Dołączamy moduł math.
```

dostarcza nam potrzebną liczbę  $\pi$ .

Rezultat działania programu można zobaczyć na rysunku 1.2.

<sup>2</sup> Moduł `math` znajduje się w bibliotece standardowej Pythona i zawiera szereg funkcji gotowych do użycia w obliczeniach matematycznych.

<sup>3</sup> UWAGA! Zastosowany tutaj zapis `sep = ""` i w rozdziale 2. `end = ""` jest niezgodny z wytycznymi stylu PEP 8 (zob.: <https://www.python.org/dev/peps/pep-0008/#whitespace-in-expressions-and-statements>). Zastosowano go wyłącznie w celu poprawy czytelności listingów programów.

**Rysunek 1.2.** *Efekt działania programu*  
Zadanie 1.2

```
Program wyświetla stałą pi z zadaną dokładnością.  
pi = 3.142.
```

**Zadanie**

**1.3**

Napisz program, który wyświetla na ekranie komputera pierwiastek kwadratowy z wartości predefiniowanej  $\pi = 3.14...$  z dokładnością do czterech miejsc po przecinku.

*Przykładowe rozwiązanie — listing 1.3*

```
# Zadanie 1.3.  
  
import math # Dołączamy moduł math.  
  
print("Program wyświetla pierwiastek kwadratowy z liczby pi")  
print("z dokładnością do czterech miejsc po przecinku.")  
print(f"sqrt(pi) = {math.sqrt(math.pi):.4f}", ".", sep = "")
```

Funkcja `sqrt()` pozwala na obliczenie pierwiastka kwadratowego z dowolnej liczby rzeczywistej. Należy ona do modułu `math`, który trzeba zaimportować do programu poleceniem `import math`.

Rezultat działania programu można zobaczyć na rysunku 1.3.

**Rysunek 1.3.** *Efekt działania programu*  
Zadanie 1.3

```
Program wyświetla pierwiastek kwadratowy z liczby pi  
z dokładnością do czterech miejsc po przecinku.  
sqrt(pi) = 1.7725.
```

Oto przykłady kilku zadań zawierających podstawowe działania arytmetyczne.

**Zadanie**

**1.4**

Napisz program, który oblicza wynik dzielenia całkowitego bez reszty dla dwóch liczb całkowitych  $a = 37$  i  $b = 11$ .



Wskazówka

Zastosuj operator `//` (**operator dzielenia bez reszty**), który wykonuje całkowitą operację dzielenia dwóch liczb całkowitych. Na przykład w języku F# w przypadku zastosowania operatora dzielenia `/` dla liczb całkowitych reszta wyniku jest pomijana (tak samo jest w niektórych językach imperatywnych: C/C++ i Java).

---

*Przykładowe rozwiązanie — listing 1.4*

---

```
# Zadanie 1.4.

a = 37
b = 11

print("Program wyświetla wynik dzielenia całkowitego")
print("bez reszty dwóch liczb całkowitych a i b.")

print("Dla liczb: a = %2i i b = %2i" % (a, b))
print("%2i // %2i = " % (a, b), a // b, ".", sep = "")
```

---

gdzie zapis `a // b` oznacza operację całkowitego dzielenia bez reszty.

Rezultat działania programu można zobaczyć na rysunku 1.4.

**Rysunek 1.4.** Efekt  
działania programu  
Zadanie 1.4

```
Program wyświetla wynik dzielenia całkowitego
bez reszty dwóch liczb całkowitych a i b.
Dla liczb: a = 37 i b = 11
37 // 11 = 3.
```

---

**Zadanie****1.5**

Napisz program, który oblicza resztę z dzielenia całkowitego dwóch liczb całkowitych `a = 37` i `b = 11`.



Wskazówka

Należy zastosować operator reszty z dzielenia całkowitego modulo, który oznaczamy w języku Python jako `%`. Podobnie jak w językach imperatywnych C/C++ i Java, operator ten umożliwia uzyskanie tylko reszty z dzielenia, natomiast wartość całkowita jest odrzucana.

---

*Przykładowe rozwiązanie — listing 1.5*

---

```
# Zadanie 1.5.

a = 37
b = 11

print("Program oblicza resztę z dzielenia całkowitego")
print("dwóch liczb całkowitych a i b.")
print(f"Dla liczb: a = {a} i b = {b}")
print(f"{a} % {b} = {a % b}", ".", sep = "")
```

---

Rezultat działania programu można zobaczyć na rysunku 1.5.

**Rysunek 1.5.** *Efekt działania programu*  
*Zadanie 1.5*

```
Program oblicza resztę z dzielenia całkowitego
dwóch liczb całkowitych a i b.
Dla liczb: a = 37 i b = 11
37 % 11 = 4.
```

**Zadanie**

**1.6** Napisz program, który wczytuje imię, nazwisko, wiek oraz cenę chleba, a następnie te cztery zmienne drukuje na ekranie komputera.

*Przykładowe rozwiązanie — listing 1.6*

```
# Zadanie 1.6.

print("Podaj swoje imię.")
imię = input()

print("Podaj swoje nazwisko.")
nazwisko = input()

wiek = int(input("Ile masz lat? "))

cena = float(input("Ile płaciłeś za chleb? "))

print()

print("Oto wprowadzone przez Ciebie dane:")
print("Imię: ", imię, ".", sep = "")
print("Nazwisko: ", nazwisko, ".", sep="")
print("Wiek:", wiek, "lata.")
print("Chleb kosztuje:", cena, "zł.")
```

Wywołanie funkcji `input()` i pobranie wartości tekstowej wpisanej za pomocą klawiatury przedstawia następująca linijka kodu:

```
imię = input()
```

Wywołanie funkcji `input()` i pobranie całkowitej wartości wpisanej za pomocą klawiatury przedstawia następująca linijka kodu:

```
wiek = int(input("Ile masz lat? "))4
```

---

<sup>4</sup> Wartość zwrótna funkcji `input()`, czyli ciąg tekstowy, zostaje przekazana jako argument funkcji `int`.

Zmienna jest typu `int`. Wywołanie funkcji `input()` i pobranie wartości zmiennoprzecinkowej wpisanej za pomocą klawiatury przedstawia następująca linijka kodu:

```
cena = float(input("Ile płać się za chleb? "))
```

Zmienna jest typu `float`.

W programie zastosowano mechanizm separatora `sep = ""`, który kasuje niepotrzebne spacje pomiędzy elementami funkcji `print()`.

Ilustruje to następująca linijka kodu:

```
print("Imię: ", imię, ".", sep = "")
```

Rezultat działania programu można zobaczyć na rysunku 1.6.

**Rysunek 1.6.** *Efekt działania programu*  
Zadanie 1.6

```
Podaj swoje imię.  
Janusz  
Podaj swoje nazwisko.  
Nowak  
Ile masz lat? 24  
Ile płać się za chleb? 6  
  
Oto wprowadzone przez Ciebie dane:  
Imię: Janusz.  
Nazwisko: Nowak.  
Wiek: 24 lata.  
Chleb kosztuje: 6.0 zł.
```

#### Zadanie

##### 1.7

Napisz program, w który generuje 5 liczb pseudolosowych z przedziału od 1 do 100.

*Przykładowe rozwiązanie — listing 1.7*

```
# Zadanie 1.7.
```

```
import random # Importujemy do programu moduł random.
```

```
print("Liczby pseudolosowe: ")
```

```
print()
```

```
for i in range(5):
```

```
    liczba = random.randint(1, 100) # Generowanie liczby pseudolosowej.  
    print(liczba)
```

Rezultat działania programu można zobaczyć na rysunku 1.7.

**Rysunek 1.7.** *Efekt działania programu Zadanie 1.7*

Liczby pseudolosowe:

36  
90  
84  
4  
87

Moduł `random` znajduje się w bibliotece standardowej i zawiera wiele funkcji przeznaczonych do pracy z liczbami pseudolosowymi.

Do wyświetlenia liczb pseudolosowych w programie skorzystano z pętli `for` (zob. rozdział 3.) oraz z funkcji `range()` (zob. rozdział 9.):

```
for i in range(5):  
    liczba = random.randint(1, 100) # Generowanie liczby  
    ↪ pseudolosowej.  
    print(liczba)
```

Więcej o funkcji `range()` w rozdziale 3. w **PI** i w **PF**.

# PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
  2. PREZENTUJ KSIĄŻKI
  3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA  
**Helion** 



# PROGRAMUJ IMPERATYWNIE W PYTHONIE!

- Poznaj język Python od strony praktycznej
- Naucz się czytać i analizować kod Pythona
- Rozwiązuj problemy programistyczne

Python to obecnie jeden z najpopularniejszych języków programowania, a jego znajomość zapewnia zatrudnienie w największych firmach i przy najciekawszych projektach w branży informatycznej. Szerokie możliwości, duża elastyczność i wszechstronność, przejrzystość i zwięzłość składni, czytelność i klarowność kodu, rozbudowany pakiet bibliotek standardowych, niemal nieograniczone zastosowanie w różnych dziedzinach nauki i biznesu — wszystko to sprawia, że język ten z pewnością utrzyma swoją pozycję, a programujące w nim osoby jeszcze długo będą należały do najbardziej pożądaných specjalistów na rynku IT.

Jedną z niewątpliwych zalet Pythona jest to, że wspiera różne paradygmaty programowania, w tym programowanie imperatywne. Jeśli chcesz poznać język od podstaw i dowiedzieć się, jak wykorzystać jego możliwości w podejściu imperatywnym i obiektowym, sięgnij po książkę *Python. Zadania z programowania. Przykładowe imperatywne rozwiązania*. Dzięki zamieszczonym w niej zadaniom o różnym poziomie trudności oraz ich rozwiązaniom szybko i gruntownie nauczysz się programować w Pythonie i czytać kod napisany w tym języku, a praktyczne wskazówki pomogą Ci zrozumieć bardziej zawile zagadnienia.

- Podstawowe operacje wejścia-wyjścia w języku Python
- Konstrukcje warunkowe, operatory logiczne i pętle
- Definiowanie i wywoływanie funkcji oraz rekurencja
- Programowanie zorientowane obiektowo w Pythonie
- Odczytywanie i zapisywanie plików tekstowych i binarnych
- Obsługa wyjątków, dekoratory i menedżery kontekstu
- Iteratory, generatory, funkcje wyższego rzędu

## Przekonaj się, jak prosty może być język Python!

Jeśli chcesz poszerzyć swoją wiedzę i interesuje Cię programowanie funkcyjne w języku Python, sięgnij również po inną książkę tego autora:

**Python. Zadania z programowania. Przykładowe funkcyjne rozwiązania**

**Helion**



helion.pl



**HELION SA**  
ul. Kościuszki 1c  
44-100 Gliwice  
tel.: 32 230 98 63  
helion@helion.pl

Sprawdź nasze szkolenia!



AKADEMIA IT & BUSINESS

HELIONSZKOLENIA.PL

**KOD KORZYŚCI**  
Sięgnij po więcej! ▶



ISBN 978-83-283-7254-2



9 788328 372542

INFORMATYKA W NAJLEPSZYM WYDANIU

Cena: 39,90 zł