



Sprawozdanie z laboratorium

Języki skryptowe

Py3 104: Funkcje, moduły i zakresy nazw

02.04.2025

Prowadzący: dr Beata Zieba

Grupa: 24-INF-SP/A

Hubert Jarosz

Oliwer Pawelski

1 Wstęp

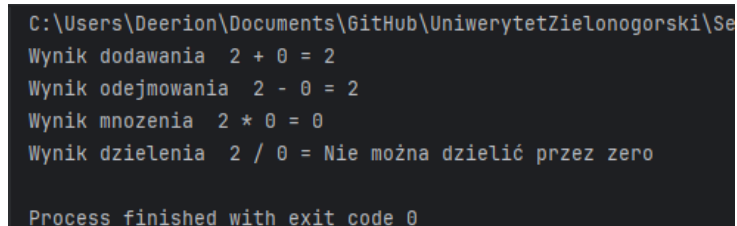
W ramach ćwiczeń zrealizowano zestaw zadań dotyczących operacji arytmetycznych oraz pracy z modułami i pakietami w języku Python. Utworzono funkcje realizujące podstawowe działania matematyczne, które następnie zostały rozszerzone o możliwość pracy z wektorami i macierzami. Kolejnym krokiem było uporządkowanie kodu poprzez przeniesienie funkcji do modułów, utworzenie pakietów oraz ich odpowiednia strukturyzacja. W zadaniach wykorzystano również mechanizmy Pythona, takie jak importowanie modułów, przechowywanie metadanych w plikach `init .py`, a także przeładowywanie kodu w czasie działania programu. Dodatkowo zaprezentowano zastosowanie funkcji lambda jako sposobu tworzenia prostych funkcji anonimowych.

2 Zadania

Zadanie 1 Utworzyć funkcje dodaj, odejmij, pomnoz które będą wykonywane zadane operacje arytmetyczne, odpowiednio `+`, `-`, `*`.

```
1 def dodaj(a, b):
2     return a + b
3
4 def odejmij(a, b):
5     return a - b
6
7 def pomnoz(a, b):
8     return a * b
9
10 def podziel(a,b):
11     if b == 0:
12         return("Nie mozna dzielic przez zero")
13     return a / b
14
15 a = 2
16 b = 2
17 print("Wynik dodawania ", a , "+", b, "=",dodaj(a, b))
18 print("Wynik odejmowania ", a , "-", b, "=",odejmij(a, b))
19 print("Wynik mnozenia ", a , "*", b, "=",pomnoz(a, b))
20 print("Wynik dzielenia ", a , "/", b, "=",podziel(a, b))
```

Listing 1: Kod do zadania 1



```
C:\Users\Deerion\Documents\GitHub\UniwersytetZielonogorski\Se
Wynik dodawania  2 + 0 = 2
Wynik odejmowania  2 - 0 = 2
Wynik mnozenia  2 * 0 = 0
Wynik dzielenia  2 / 0 = Nie można dzielić przez zero
Process finished with exit code 0
```

Zadanie 2 Zmodyfikować funkcje z zad. 8.1. tak aby wspierały wykonywanie operacji arytmetycznych na wektorach (w postaci list).

```
1 def dodaj(a, b):
2     if (len(a)==len(b)):
3         return [x + y for x, y in zip(a, b)]
4     else:
5         return("Nie mozna dodac wektorow o roznych dlugosciach")
6
7 def odejmij(a, b):
8     if (len(a)==len(b)):
9         return [x - y for x, y in zip(a, b)]
10    else:
11        return("Nie mozna odjac wektorow o roznych dlugosciach")
12
13 def pomnoz(a, b):
14     if (len(a)==len(b)):
15         return [x * y for x, y in zip(a, b)]
16     else:
17         return("Nie mozna pomnozyc wektorow o roznych dlugosciach")
18
19
20
21 c = [1, 2, 3,]
22 d = [4, 5, 6]
23 print("Wynik dodawania wektorow ", c , "+", d, "=", dodaj([1, 2, 3,], [4, 5, 6]))
24 print("Wynik odejmowania wektorow ", c , "-", d, "=", odejmij([1, 2, 3,], [4, 5,
25     6]))
26 print("Wynik mnozenia wektorow ", c , "*", d, "=", pomnoz([1, 2, 3,], [4, 5, 6]))
```

Listing 2: Kod do zadania 2

```
C:\Users\Deerion\Documents\GitHub\UniwerytetZielonogorski\Semestr_4
Wynik dodawania wektorów [1, 2, 3] + [4, 5, 6] = [5, 7, 9]
Wynik odejmowania wektorów [1, 2, 3] - [4, 5, 6] = [-3, -3, -3]
Wynik mnozenia wektorów [1, 2, 3] * [4, 5, 6] = [4, 10, 18]

Process finished with exit code 0
```

Zadanie 3 Zmodyfikować funkcje z zad. 8.2. tak aby wspierały operacje na macierzach.

```
1 def dodaj(a, b):
2     if len(a) != len(b) or len(a[0]) != len(b[0]):
3         return "Nie mozna dodac macierzy o roznych wymiarach"
4
5     return [[x + y for x, y in zip(row_a, row_b)] for row_a, row_b in zip(a, b)]
6
7
8 def odejmij(a, b):
9     if len(a) != len(b) or len(a[0]) != len(b[0]):
10        return "Nie mozna odjac macierzy o roznych wymiarach"
11
12    return [[x - y for x, y in zip(row_a, row_b)] for row_a, row_b in zip(a, b)]
13
14
15 def pomnoz(a, b):
16     if len(a[0]) != len(b):
17         return "Nie mozna pomnozyc macierzy, poniewaz liczba kolumn w pierwszej
18         macierzy rozni sie od liczby wierszy w drugiej"
19
20     result = []
21     for i in range(len(a)):
22         row = []
23         for j in range(len(b[0])):
24             row.append(sum(a[i][k] * b[k][j] for k in range(len(b))))
25         result.append(row)
26     return result
27
28 # Test z macierzami
29 e = [[1, 2, 3], [4, 5, 6]]
30 f = [[7, 8, 9], [10, 11, 12]]
31 print("Wynik dodawania macierzy ", e, "+", f, "=", dodaj(e, f))
32 print("Wynik odejmowania macierzy ", e, "-", f, "=", odejmij(e, f))
33
34
35 # Test mnozenia macierzy
36 g = [[1, 2], [3, 4], [5, 6]]
37 h = [[7, 8, 9], [10, 11, 12]]
38 print("Wynik mnozenia macierzy ", g, "*", h, "=", pomnoz(g, h))
```

Listing 3: Kod do zadania 3

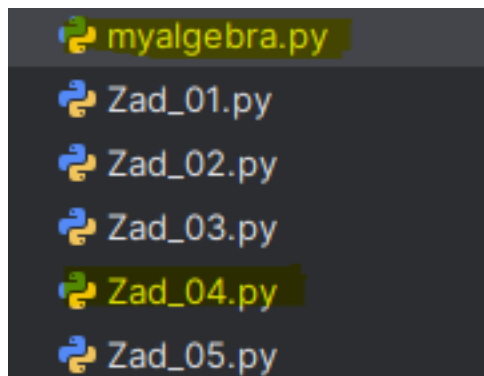
```
Wynik dodawania macierzy [[1, 2, 3], [4, 5, 6]] + [[7, 8, 9], [10, 11, 12]] = [[8, 10, 12], [14, 16, 18]]
Wynik odejmowania macierzy [[1, 2, 3], [4, 5, 6]] - [[7, 8, 9], [10, 11, 12]] = [[-6, -6, -6], [-6, -6, -6]]
Wynik mnozenia macierzy [[1, 2], [3, 4], [5, 6]] * [[7, 8, 9], [10, 11, 12]] = [[27, 30, 33], [61, 68, 75], [95, 106, 117]]
Process finished with exit code 0
```

Zadanie 4 Przenieść funkcje z zadania 8.3. do modułu myalgebra

```
1 #from myalgebra import dodaj, odejmij, pomnoz
2 from myalgebra import *
3 # Test z macierzami
4 e = [[1, 2, 3, 4], [4, 5, 6]]
5 f = [[7, 8, 9], [10, 11, 12]]
6
7 print("Wynik dodawania macierzy ", e, "+", f, "=", dodaj(e, f))
8 print("Wynik odejmowania macierzy ", e, "-", f, "=", odejmij(e, f))
9
10 # Test mnożenia macierzy
11 g = [[1, 2], [3, 4], [5, 6]]
12 h = [[7, 8, 9], [10, 11, 12]]
13 print("Wynik mnożenia macierzy ", g, "*", h, "=", pomnoz(g, h))
```

Listing 4: Kod do zadania 4

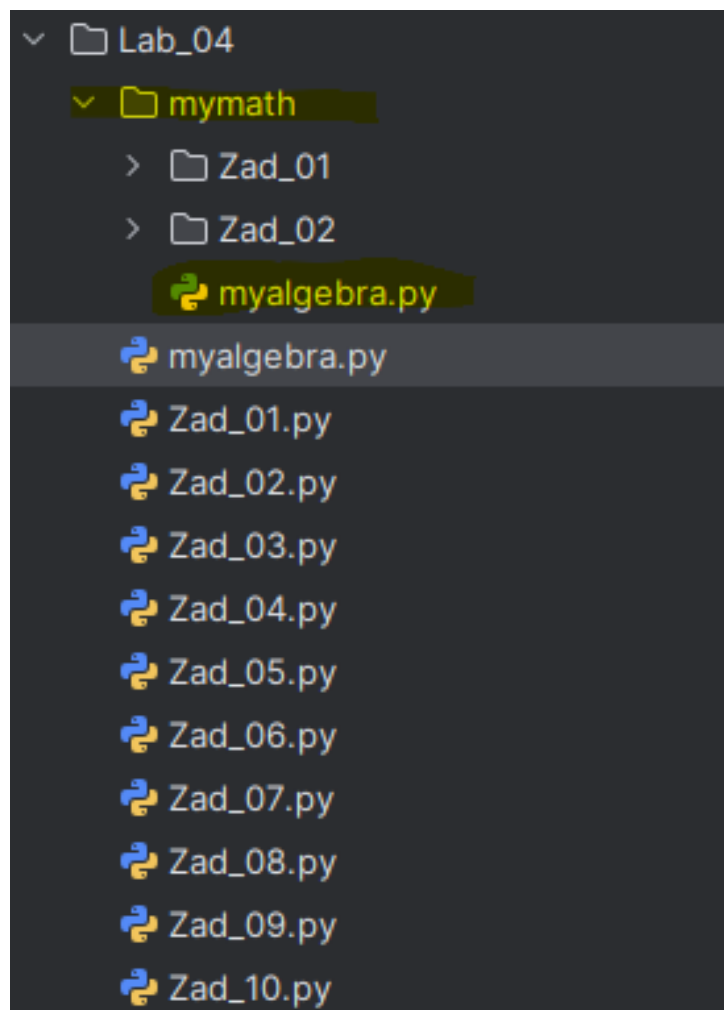
```
C:\Users\Deerion\Documents\GitHub\UniwersytetZielonogorski\Semestr_4\JęzykiSkryptowe\Laboratorium\.venv\Scripts\python.exe C:\Users\Deerion\Documents\GitHub\UniwersytetZielonogorski\Semestr_4\JęzykiSkryptowe\Laboratorium\Zad_04.py
Wynik dodawania macierzy [[1, 2, 3, 4], [4, 5, 6]] + [[7, 8, 9], [10, 11, 12]] = Nie można dodać macierzy o różnych wymiarach
Wynik odejmowania macierzy [[1, 2, 3, 4], [4, 5, 6]] - [[7, 8, 9], [10, 11, 12]] = Nie można odjąć macierzy o różnych wymiarach
Wynik mnożenia macierzy [[1, 2], [3, 4], [5, 6]] * [[7, 8, 9], [10, 11, 12]] = [[27, 30, 33], [61, 68, 75], [95, 106, 117]]
Process finished with exit code 0
```



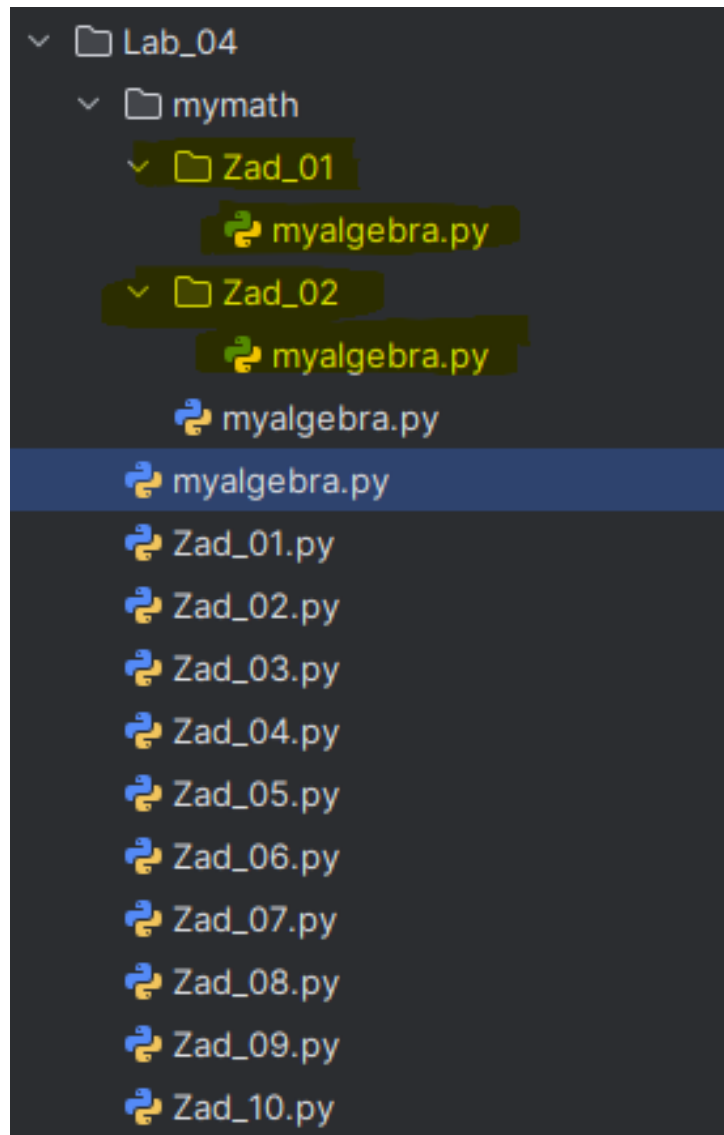
Sposoby importowania:

1. Importowanie całego modułu
-import math
2. Importowanie określonej funkcji z modułu
-from math import sqrt
3. Importowanie wszystkich funkcji z modułu
-from math import *
4. Importowanie z aliasem
-import numpy as np
5. Importowanie z aliasem dla funkcji
-from math import sqrt as s
6. Importowanie modułu z zagnieżdżonymi folderami
-from folder.subfolder import module
7. Importowanie tylko części modułu
-import module.submodule

Zadanie 5 Utworzyć pakiet mymath, umieścić w nim moduł myalgebra



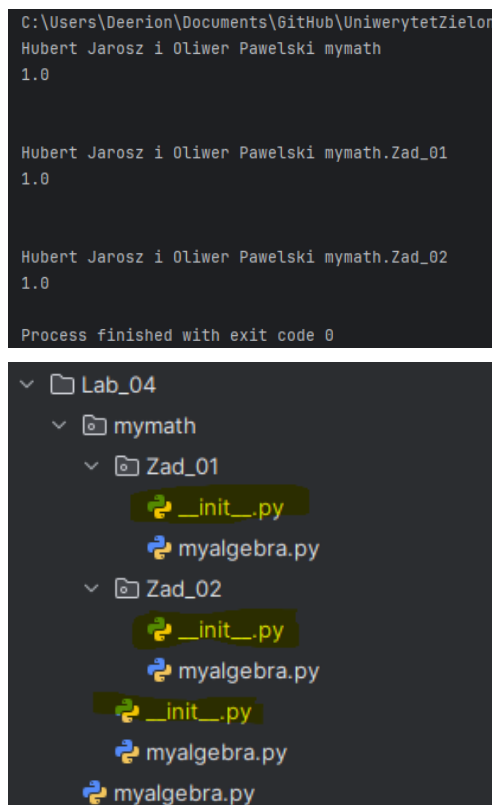
Zadanie 6 Do pakietu z zadania 8.5. dodać pakiety z funkcjami z zadań 8.1. i 8.2.



Zadanie 7 W odpowiednich plikach init .py dodać zmienne przechowujące autora i wersję pakietu

```
1 import mymath
2
3 print(mymath.__author__)
4 print(mymath.__version__)
5 print("\n")
6
7 import mymath.Zad_01
8 print(mymath.Zad_01.__author__)
9 print(mymath.Zad_01.__version__)
10 print("\n")
11
12 import mymath.Zad_02
13 print(mymath.Zad_02.__author__)
14 print(mymath.Zad_02.__version__)
```

Listing 5: Kod do zadania 7



```
1 __author__ = "Hubert Jarosz i Oliwer Pawelski mymath"
2 __version__ = "1.0"
```

Listing 6: Kod do zadania 7

Zadanie 8 Zmienić zakres zmiennych z zad 8.7. na zmienne chronione i dodać funkcje pozwalające na wyświetlanie wersji oraz autora pakietu

```
1 __author__ = "Hubert Jarosz i Oliwer Pawelski mymath"
2 __version__ = "1.0"
3
4 def get__author__():
5     return __author__
6
7 def get__version__():
8     return __version__
```

Listing 7: Kod do zadania 8

```
1 import mymath
2
3 print(mymath.get__author__())
4 print(mymath.get__version__())
5 print("\n")
6
7 import mymath.Zad_01
8 print(mymath.Zad_01.get__author__())
9 print(mymath.Zad_01.get__version__())
10 print("\n")
11
12 import mymath.Zad_02
13 print(mymath.Zad_02.get__author__())
14 print(mymath.Zad_02.get__version__())
```

Listing 8: Kod do zadania 8

```
C:\Users\Deerion\Documents\GitHub\UniwersytetZielonogorski\
Hubert Jarosz i Oliwer Pawelski mymath
1.0

Hubert Jarosz i Oliwer Pawelski mymath.Zad_01
1.0

Hubert Jarosz i Oliwer Pawelski mymath.Zad_02
1.0

Process finished with exit code 0
```

Zadanie 9 Korzystając z mechanizmu przeładowania modułów napisać program pozwalający na modyfikację modułu myalgebra pomiędzy kolejnymi wywołaniami polecenia input() w pętli, które przyjmuje kolejne elementy wektora aż do wpisania litery K.

```
1 import importlib
2 from mymath.Zad_02 import myalgebra # Importujemy modul z właściwej ścieżki
3
4 vector = []
5
6 while True:
7     val = input("Podaj liczbę lub 'K', aby zakończyć: ")
8     if val.upper() == 'K':
9         break
10
11     try:
12         num = float(val)
13         vector.append(num)
14     except ValueError:
15         print("To nie jest liczba!")
16         continue # Pomijamy dalsze operacje przy błędnym wejściu
17
18 # Przeładuj modul myalgebra zmiany w pliku zostaną wczytane
19 importlib.reload(myalgebra)
20 print("Modul myalgebra został przeładowany!")
21
22 # Dla demonstracji: tworzymy drugi wektor 'd' – każdy element zwiększony o 1
23 c = vector
24 d = [x + 1 for x in vector]
25
26 print("Wynik dodawania wektorów:", c, "+", d, "=", myalgebra.dodaj(c, d))
27 print("Wynik odejmowania wektorów:", c, "-", d, "=", myalgebra.odejmij(c, d))
28 print("Wynik mnożenia wektorów:", c, "*", d, "=", myalgebra.pomnoz(c, d))
```

Listing 9: Kod do zadania 9

```
C:\Users\Deerion\Documents\GitHub\UniwersytetZielonogorski\Semestr_4\JęzykiSkryptowe
Podaj liczbę lub 'K', aby zakończyć: 1
Moduł myalgebra został przeładowany!
Wynik dodawania wektorów: [1.0] + [2.0] = [3.0]
Wynik odejmowania wektorów: [1.0] - [2.0] = [-1.0]
Wynik mnożenia wektorów: [1.0] * [2.0] = [2.0]
Podaj liczbę lub 'K', aby zakończyć: 2
Moduł myalgebra został przeładowany!
Wynik dodawania wektorów: [1.0, 2.0] + [2.0, 3.0] = [3.0, 5.0]
Wynik odejmowania wektorów: [1.0, 2.0] - [2.0, 3.0] = [-1.0, -1.0]
Wynik mnożenia wektorów: [1.0, 2.0] * [2.0, 3.0] = [2.0, 6.0]
Podaj liczbę lub 'K', aby zakończyć: 7
Moduł myalgebra został przeładowany!
Wynik dodawania wektorów: [1.0, 2.0, 7.0] + [2.0, 3.0, 8.0] = [2.0, 6.0, 56.0]
Wynik odejmowania wektorów: [1.0, 2.0, 7.0] - [2.0, 3.0, 8.0] = [-1.0, -1.0, -1.0]
Wynik mnożenia wektorów: [1.0, 2.0, 7.0] * [2.0, 3.0, 8.0] = [2.0, 6.0, 56.0]
Podaj liczbę lub 'K', aby zakończyć: K
Process finished with exit code 0
```

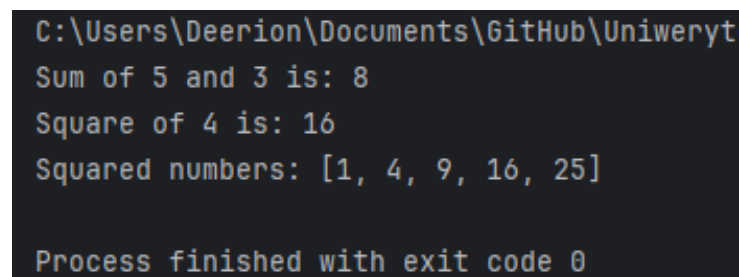
Zadanie 10 Wjaśnij na czym polega użycie funkcji lambda i kiedy warto ją stosować?

Funkcja lambda w Pythonie to sposób na zdefiniowanie anonimowej funkcji — czyli takiej, która nie ma nazwy. Używa się jej wtedy, gdy potrzebujesz szybko przekazać jakąś małą funkcję jako argument, np. do sortowania, filtrowania czy mapowania.

Przykład:

```
1 # Wyjasnij na czym polega uzycie funkcji lambda i kiedy warto ja stosowac?
2
3 def main():
4     # A lambda function to add two numbers
5     add = lambda x, y: x + y
6     print("Sum of 5 and 3 is:", add(5, 3))
7
8     # A lambda function to square a number
9     square = lambda x: x * x
10    print("Square of 4 is:", square(4))
11
12    # Using lambda with the map function
13    numbers = [1, 2, 3, 4, 5]
14    squared_numbers = list(map(lambda x: x * x, numbers))
15    print("Squared numbers:", squared_numbers)
16
17 if __name__ == "__main__":
18     main()
19
20 # — Krotkie i proste funkcje: Kiedy potrzebujesz malej funkcji przez krotki okres
21   # — Funkcje wyzszego rzędu: Kiedy przekazujesz prosta funkcje jako argument do
22   # — Definicje funkcji inline: Kiedy definiujesz funkcje inline bez potrzeby
23   # Funkcje lambda sa przydatne do zwiezlego i czytelnego kodu, szczegolnie w
   kontekstach programowania funkcyjnego.
```

Listing 10: Kod do zadania 10



```
C:\Users\Deerion\Documents\GitHub\Uniweryt
Sum of 5 and 3 is: 8
Square of 4 is: 16
Squared numbers: [1, 4, 9, 16, 25]

Process finished with exit code 0
```