
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki <b>Zakład Systemów Teleinformatycznych</b>		
<b>Przedmiot</b>	Skryptowe języki programowania		
<b>Prowadzący</b>	mgr inż. Martyna Tarczewska		
<b>Temat</b>	Numpy		
<b>Student</b>	Paweł Jońca		
<b>Nr lab.</b>	9	<b>Data wykonania</b>	14.12.2024
<b>Ocena</b>		<b>Data oddania spr.</b>	14.12.2024

### Zad 1

```
import numpy as np

def replace_zeros(A, x):
    A = np.array(A)
    A[A == 0] = x
    return A

# Przykład użycia
mat = np.array([[0, 2, 0], [4, 0, 6], [0, 8, 9]])
x_val = 3
result = replace_zeros(mat, x_val)
print("Macierz po zamianie zer na", x_val, ":")
print(result)
```

```
Macierz po zamianie zer na 3 :
[[3 2 3]
 [4 3 6]
 [3 8 9]]
```

```
import numpy as np

def replace_zeros(A, x):
    A = np.array(A)
    A[A == 0] = x
    return A

# Przykład użycia
```

```
mat = np.array([[0, 2, 0], [4, 0, 6], [0, 8, 9]])
x_val = 3
result = replace_zeros(mat, x_val)
print("Macierz po zamianie zer na", x_val, ":")
print(result)
```

## Zad 2

```
def medianize(A): 1 usage new *
    if not A: # Sprawdzenie, czy lista nie jest pusta
        return []
    # Obliczenie średniej wartości tablicy
    average = sum(A) / len(A)
    # Odjęcie średniej od każdego elementu tablicy
    result = [x - average for x in A]
    return result
# Przykład użycia
A = [1, 2, 3, 4, 5]
print(medianize(A))
```

zad2 x

⋮

"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Do  
[-2.0, -1.0, 0.0, 1.0, 2.0]

```
def medianize(A):
    if not A: # Sprawdzenie, czy lista nie jest pusta
        return []
    # Obliczenie średniej wartości tablicy
    average = sum(A) / len(A)
    # Odjęcie średniej od każdego elementu tablicy
    result = [x - average for x in A]
    return result
# Przykład użycia
A = [1, 2, 3, 4, 5]
print(medianize(A))
```

### Zad 3

```
import numpy as np

def medianize(A): # usage new*
    if not A: # Sprawdzenie, czy lista nie jest pusta
        return []
    # Obliczenie średniej wartości tablicy
    average = sum(A) / len(A)
    # Odjęcie średniej od każdego elementu tablicy
    result = [x - average for x in A]
    return result

def matrix(): # usage new*
    # Generowanie macierzy 5x5 z liczbami naturalnymi mniejszymi od 100
    matrix = np.random.randint(low=0, high=100, size=(5, 5))
    print("Macierz:")
    print(matrix)
    # Największy element globalnie
    max_global = np.max(matrix)
    print("Największy element globalnie:", max_global)
    # Największy element w każdym wierszu
    max_rows = np.max(matrix, axis=1)
    print("Największy element w każdym wierszu:", max_rows)
    # Największy element w każdej kolumnie
    max_cols = np.max(matrix, axis=0)
    print("Największy element w każdej kolumnie:", max_cols)

# Przykład użycia
A = [1, 2, 3, 4, 5]
print(medianize(A))
matrix()
```

```
[-2.0, -1.0, 0.0, 1.0, 2.0]
Macierz:
[[31 41 96 41 67]
 [78 57 37 24 84]
 [25 60 14 43 30]
 [53 13 96 85 44]
 [99 30 31 60 96]]
Największy element globalnie: 99
Największy element w każdym wierszu: [96 84 60 96 99]
Największy element w każdej kolumnie: [99 60 96 85 96]
```

```
import numpy as np

def medianize(A):
    if not A: # Sprawdzenie, czy lista nie jest pusta
        return []
    # Obliczenie średniej wartości tablicy
    average = sum(A) / len(A)
    # Odjęcie średniej od każdego elementu tablicy
    result = [x - average for x in A]
    return result

def matrix():
    # Generowanie macierzy 5x5 z liczbami naturalnymi mniejszymi od 100
    matrix = np.random.randint(0, 100, (5, 5))
```

```

print("Macierz:")
print(matrix)
# Największy element globalnie
max_global = np.max(matrix)
print("Największy element globalnie:", max_global)
# Największy element w każdym wierszu
max_rows = np.max(matrix, axis=1)
print("Największy element w każdym wierszu:", max_rows)
# Największy element w każdej kolumnie
max_cols = np.max(matrix, axis=0)
print("Największy element w każdej kolumnie:", max_cols)
# Przykład użycia
A = [1, 2, 3, 4, 5]
print(medianize(A))
matrix()

```

#### Zad 4

```

import numpy as np

# Przygotowanie przykładowej tablicy
array = np.arange(12) # Tablica z liczbami od 0 do 11

# Test 1: -1 jako pierwszy parametr
reshaped1 = array.reshape(-1, 3)
print("Test 1: -1 jako pierwszy parametr")
print(reshaped1)

# Test 2: -1 jako drugi parametr
reshaped2 = array.reshape(4, -1)
print("\nTest 2: -1 jako drugi parametr")
print(reshaped2)

```

```

Test 1: -1 jako pierwszy parametr
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

```

```

Test 2: -1 jako drugi parametr
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

```

```
import numpy as np

# Przygotowanie przykładowej tablicy
array = np.arange(12) # Tablica z liczbami od 0 do 11

# Test 1: -1 jako pierwszy parametr
reshaped1 = array.reshape(-1, 3)
print("Test 1: -1 jako pierwszy parametr")
print(reshaped1)

# Test 2: -1 jako drugi parametr
reshaped2 = array.reshape(4, -1)
print("\nTest 2: -1 jako drugi parametr")
print(reshaped2)
```

## Zad 5

```
import pandas as pd
# Wczytanie pliku CSV z separatorem tabulatora
df = pd.read_csv(filepath_or_buffer="oceny.csv", sep='\t')
# Sprawdzenie nazw kolumn i pierwszych wierszy
print("Nazwy kolumn w DataFrame:", df.columns)
print(df.head())
# Najniższa ocena z laboratoriów dla każdego studenta
df['Min_LAB'] = df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']].min(axis=1)
print("Najniższa ocena z laboratoriów dla każdego studenta:")
print(df[['Min_LAB']])
# Średnia ocena z egzaminu
average_exam = df['Exam'].mean()
print(f"Średnia ocena z egzaminu: {average_exam:.2f}")
# Liczba 2 z egzaminu
count_exam_2 = (df['Exam'] == 2).sum()
print(f"Liczba ocen 2 z egzaminu: {count_exam_2}")
# Czy jest student, który miał same 5 z laboratoriów
all_5_in_labs = any((df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']] == 5).all(axis=1))
print(f"Czy jest student, który miał same 5 z laboratoriów: {'Tak' if all_5_in_labs else 'Nie'}")
# Czy jest student, który miał 2 z Lab2 i Lab3
student_2_in_lab2_lab3 = any((df['Lab2'] == 2) & (df['Lab3'] == 2))
print(f"Czy jest student, który miał 2 z Lab2 i Lab3: {'Tak' if student_2_in_lab2_lab3 else 'Nie'}")
# Liczba studentów, którzy dostali wyższą ocenę z egzaminu niż średnia z laboratoriów
df['Average_LAB'] = df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']].mean(axis=1)
students_higher_exam = (df['Exam'] > df['Average_LAB']).sum()
print(f"Liczba studentów, którzy dostali wyższą ocenę z egzaminu niż średnia z laboratoriów: {students_higher_exam}")
# Liczba piątek uzyskanych przez studenta z największą liczbą 5
df['Count_5'] = (df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5', 'Exam']] == 5).sum(axis=1)
max_5_student = df['Count_5'].max()
print(f"Liczba piątek uzyskanych przez studenta z największą liczbą 5: {max_5_student}")
```

```

Nazwy kolumn w DataFrame: Index(['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5', 'Exam'], dtype='object')
  Lab1  Lab2  Lab3  Lab4  Lab5  Exam
0   3.0   3.5   3.5   4.0   5.0   4.0
1   5.0   4.5   3.5   5.0   4.5   5.0
2   3.5   3.5   3.5   3.5   3.5   3.0
3   5.0   5.0   5.0   5.0   5.0   4.5
4   2.0   3.0   3.5   4.0   2.0   2.0

Najniższa ocena z laboratoriów dla każdego studenta:
  Min_LAB
0       3.0
1       3.5
2       3.5
3       5.0
4       2.0
5       3.5
6       3.5
7       4.5
8       3.5
9       2.0

Średnia ocena z egzaminu: 3.45
Liczba ocen 2 z egzaminu: 2
Czy jest student, który miał same 5 z laboratoriów: Tak
Czy jest student, który miał 2 z Lab2 i Lab3: Nie
Liczba studentów, którzy dostali wyższą ocenę z egzaminu niż średnia z laboratoriów: 2
Liczba piątek uzyskanych przez studenta z największą liczbą 5: 5

```

```

import pandas as pd
# Wczytanie pliku CSV z separatorem tabulatora
df = pd.read_csv("oceny.csv", sep='\t')
# Sprawdzenie nazw kolumn i pierwszych wierszy
print("Nazwy kolumn w DataFrame:", df.columns)
print(df.head())
# Najniższa ocena z laboratoriów dla każdego studenta
df['Min_LAB'] = df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']].min(axis=1)
print("Najniższa ocena z laboratoriów dla każdego studenta:")
print(df[['Min_LAB']])
# Średnia ocena z egzaminu
average_exam = df['Exam'].mean()
print(f"Średnia ocena z egzaminu: {average_exam:.2f}")
# Liczba 2 z egzaminu
count_exam_2 = (df['Exam'] == 2).sum()
print(f"Liczba ocen 2 z egzaminu: {count_exam_2}")
# Czy jest student, który miał same 5 z laboratoriów
all_5_in_labs = any((df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']] == 5).all(axis=1))
print(f"Czy jest student, który miał same 5 z laboratoriów: {'Tak' if all_5_in_labs else 'Nie'}")
# Czy jest student, który miał 2 z Lab2 i Lab3
student_2_in_lab2_lab3 = any((df['Lab2'] == 2) & (df['Lab3'] == 2))
print(f"Czy jest student, który miał 2 z Lab2 i Lab3: {'Tak' if student_2_in_lab2_lab3 else 'Nie'}")
# Liczba studentów, którzy dostali wyższą ocenę z egzaminu niż średnia z laboratoriów
df['Average_LAB'] = df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5']].mean(axis=1)
students_higher_exam = (df['Exam'] > df['Average_LAB']).sum()
print(f"Liczba studentów, którzy dostali wyższą ocenę z egzaminu niż średnia z laboratoriów: {students_higher_exam}")
# Liczba piątek uzyskanych przez studenta z największą liczbą 5
df['Count_5'] = (df[['Lab1', 'Lab2', 'Lab3', 'Lab4', 'Lab5', 'Exam']] ==

```

```
5).sum(axis=1)
max_5_student = df['Count_5'].max()
print(f"Liczba piątek uzyskanych przez studenta z największą liczbą 5:
{max_5_student}")
```

## Zad 6

```
import numpy as np
# Wylosowanie 10 liczb do tablicy Numpy
array = np.random.rand(10) # Liczby losowe z zakresu [0, 1)
print("Oryginalna tablica:", array)
# Sortowanie rosnąco
sorted_array_ascending = np.sort(array)
print("Tablica posortowana rosnąco:", sorted_array_ascending)
# Sortowanie malejąco
sorted_array_descending = np.sort(array)[::-1] # Odwrócenie kolejności
print("Tablica posortowana malejąco:", sorted_array_descending)
```

```
Oryginalna tablica: [0.81449583 0.12216643 0.61988367 0.79572617 0.60115326 0.79654614
 0.28655845 0.57086114 0.03373808 0.85249082]
Tablica posortowana rosnąco: [0.03373808 0.12216643 0.28655845 0.57086114 0.60115326 0.61988367
 0.79572617 0.79654614 0.81449583 0.85249082]
Tablica posortowana malejąco: [0.85249082 0.81449583 0.79654614 0.79572617 0.61988367 0.60115326
 0.57086114 0.28655845 0.12216643 0.03373808]
```

```
import numpy as np
# Wylosowanie 10 liczb do tablicy Numpy
array = np.random.rand(10) # Liczby losowe z zakresu [0, 1)
print("Oryginalna tablica:", array)
# Sortowanie rosnąco
sorted_array_ascending = np.sort(array)
print("Tablica posortowana rosnąco:", sorted_array_ascending)
# Sortowanie malejąco
sorted_array_descending = np.sort(array)[::-1] # Odwrócenie kolejności
print("Tablica posortowana malejąco:", sorted_array_descending)
```

## Zad 7

```
import numpy as np
# Tworzenie przykładowej tablicy 5x5
tablica = np.random.randint(low: 1, high: 10, size: (5, 5)) # Tablica z losowymi wartościami od 1 do 9
# Definiowanie wag
wagi = np.array([1, 2, 3, 2, 1])
# Obliczanie średniej ważonej dla każdego wiersza
srednie_wazone = np.dot(tablica, wagi) / sum(wagi)
# Wyświetlenie wyników
print("Tablica:")
print(tablica)
print("\nŚrednie ważne dla każdego wiersza:")
print(srednie_wazone)
```

Tablica:

```
[[1 8 9 5 1]
 [9 8 5 9 9]
 [5 9 3 8 9]
 [6 5 8 3 4]
 [3 9 3 7 6]]
```

Średnie ważne dla każdego wiersza:

```
[6.11111111 7.44444444 6.33333333 5.55555556 5.55555556]
```

```
import numpy as np
# Tworzenie przykładowej tablicy 5x5
tablica = np.random.randint(1, 10, (5, 5)) # Tablica z losowymi warto-
ściami od 1 do 9
# Definiowanie wag
wagi = np.array([1, 2, 3, 2, 1])
# Obliczanie średniej ważonej dla każdego wiersza
srednie_wazone = np.dot(tablica, wagi) / sum(wagi)
# Wyświetlenie wyników
print("Tablica:")
print(tablica)
print("\nŚrednie ważne dla każdego wiersza:")
print(srednie_wazone)
```



## Zad 8

```
import random
from collections import Counter
# Tworzenie tablicy 10x10 z losowymi liczbami
tablica = [[random.randint(a: 0, b: 10) for _ in range(10)] for _ in range(10)]
# Zliczanie wystąpień liczb w tablicy
flat_tablica = [element for row in tablica for element in row] # Spłaszczamy tablicę
liczba_wystapien = Counter(flat_tablica)
# Wypisanie tablicy
print("Tablica 10x10:")
for row in tablica:
    print(row)
# Wypisanie liczności wystąpień elementów
print("\nLiczność wystąpień elementów:")
for liczba, wystapienia in liczba_wystapien.items():
    print(f"Liczba {liczba}: {wystapienia} razy")
|
```

```
Tablica 10x10:
[6, 7, 3, 6, 3, 6, 7, 0, 10, 6]
[1, 0, 4, 9, 10, 7, 2, 8, 4, 2]
[3, 8, 8, 7, 8, 5, 8, 10, 10, 2]
[6, 9, 7, 7, 7, 5, 5, 10, 5, 7]
[3, 3, 7, 0, 3, 5, 5, 7, 9, 1]
[4, 5, 0, 10, 4, 9, 5, 1, 0, 9]
[6, 3, 0, 1, 2, 2, 3, 10, 0, 1]
[3, 3, 6, 1, 1, 0, 0, 7, 5, 1]
[4, 0, 0, 6, 3, 0, 3, 4, 1, 2]
[10, 4, 8, 1, 2, 1, 7, 5, 10, 1]
```

```
Liczność wystąpień elementów:
Liczba 6: 8 razy
Liczba 7: 12 razy
Liczba 3: 12 razy
Liczba 0: 12 razy
Liczba 10: 9 razy
Liczba 1: 12 razy
Liczba 4: 7 razy
Liczba 9: 5 razy
Liczba 2: 7 razy
Liczba 8: 6 razy
Liczba 5: 10 razy
```

```
import random
from collections import Counter
# Tworzenie tablicy 10x10 z losowymi liczbami
```

```

tablica = [[random.randint(0, 10) for _ in range(10)] for _ in range(10)]
# Zliczanie wystąpień liczb w tablicy
flat_tablica = [element for row in tablica for element in row] # Spłaszczamy tablicę
liczba_wystapien = Counter(flat_tablica)
# Wypisanie tablicy
print("Tablica 10x10:")
for row in tablica:
    print(row)
# Wypisanie liczności wystąpień elementów
print("\nLiczność wystąpień elementów:")
for liczba, wystapienia in liczba_wystapien.items():
    print(f"Liczba {liczba}: {wystapienia} razy")

```

## Zad 9

Przetestować operator \* oraz operator @. Do czego służą?

Między liczbami \* służy do mnożenia dwóch liczb

Do rozpakowywania argumentów w funkcjach

```

def sum_numbers(*args):
    """usage new"""
    return sum(args)

result = sum_numbers(*args: 1, 2, 3) # wynik to 6

```

Rozpakowywanie w przypisaniach: Użycie \* w przypisaniach umożliwia rozpakowanie elementów z listy lub krotki.

```

a, *b = [1, 2, 3, 4]
# a = 1, b = [2, 3, 4]

```

Mnożenie elementów w iterowalnych obiektach (np. listach, ciągach):

```

repeated = [1] * 3 # wynik to [1, 1, 1]

```

W Pythonie operator @ jest używany w matematyce do mnożenia macierzy, zwłaszcza w bibliotece numpy.

```

import numpy as np
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6], [7, 8]])
result = a @ b
print(result)

```

```

[[19 22]
 [43 50]]

```

Operator @ jest również używany w Pythonie do dekorowania funkcji, tzn. przypisania funkcji dekorującej do innej funkcji.

### Zad 10 Wypróbować funkcję `set_printoptions`. Jakie daje możliwości?

Precision - Ustawia liczbę cyfr po przecinku dla liczb zmiennoprzecinkowych.

```
1 import numpy as np
2 np.set_printoptions(precision=3) # 3 miejsca po przecinku
3 print(np.array([1.123456, 2.345678]))
4
```

Run zad10 x

"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty" [1.123 2.346]

Suppress - Kontroluje, czy bardzo małe liczby (w notacji wykładniczej) mają być pokazywane w zwykłym formacie.

```
np.set_printoptions(suppress=True)
print(np.array([1.23e-10, 1.23e+10]))
```

Run zad10 x

"C:\Program Files\Python312\python.exe" "C:\Users" [1.23e-10 1.23e+10]

Threshold - Określa liczbę elementów tablicy, które mają być wyświetlane. Gdy liczba elementów przekracza próg, tablica jest skracana.

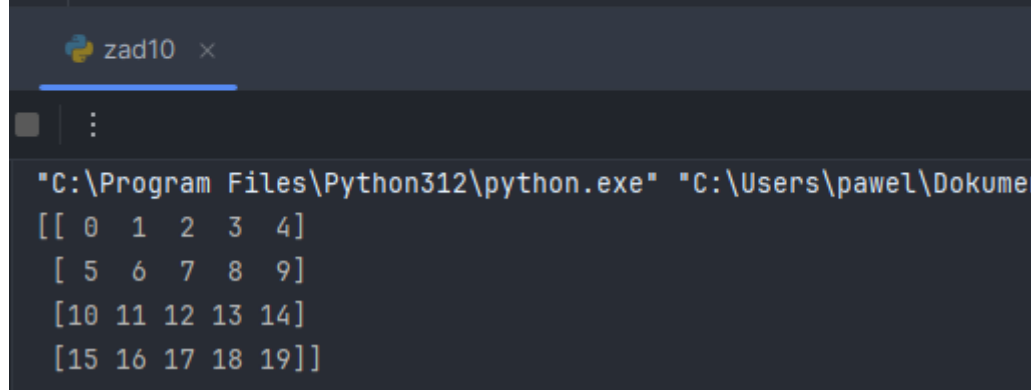
```
np.set_printoptions(threshold=5) # Pokaż maksymalnie 5 elementów
print(np.arange(10))
```

Run zad10 x

"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokumenty\03 -" [0 1 2 ... 7 8 9]

Linewidth – określa maksymalną szerokość linii wyświetlania tablicy

```
np.set_printoptions(linewidth=50)
print(np.arange(20).reshape(4, 5))
# Tablica będzie łamana w liniach o szerokości do 50 znaków
```



```
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokume
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

Pozwala zdefiniować własne funkcje formatowania dla różnych typów danych.

```
np.set_printoptions(formatter={'float': '{:0.1f}'.format})
print(np.array([1.2345, 2.3456]))
```



```
"C:\Program Files\Python312\python.exe" "C:\Users\pawel\Dokument
[1.2 2.3]
```

Wnioski: W trakcie realizacji zadań z modułu Numpy zyskałem praktyczne umiejętności w zakresie manipulacji danymi oraz wykonywania obliczeń numerycznych na macierzach. Użycie funkcji takich jak reshape, medianize oraz operacje na tablicach wielowymiarowych pozwoliło mi lepiej zrozumieć, jak efektywnie przetwarzać duże zbiory danych w Pythonie. Dodatkowo, nauczyłem się korzystać z wczytywania danych z plików oraz ich analizy, co jest niezbędne w praktycznych zastosowaniach programowania.