МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 3

з дисципліни «Об'єктно-орієнтоване програмування СУ»

Тема: «Структурування програм з використанням функцій»

ХАІ.301 . Авіоніка . група 3-92АВ(і). ЛР3

Виконав студент гр. \_\_\_\_\_\_3-92АВ(і)\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сидоренко В.В\_\_\_

(підпис, дата) (П.І.Б.)

Перевірив

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. О. В. Гавриленко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ас.  В. О. Білозерський

(підпис, дата) (П.І.Б.)

2023

# МЕТА РОБОТИ

Вивчити теоретичний матеріал із синтаксису визначення і виклику

функцій та особливостей послідовностей у Python, а також документацію

бібліотеки numpy; отримати навички реалізації бібліотеки функцій з

параметрами, що структурують вирішення завдань «згори – донизу».

# ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Описати функцію відповідно до варіанту. Для виклику функції

(друга частина задачі) описати іншу функцію, що на вході має список вхідних

даних і повертає список вихідних даних. Введення даних, виклик функції та

виведення результатів реалізувати в третій функції без параметрів. Завдання

наведено в табл.1.(Proc7)

Завдання 2. Розробити дві вкладені функції для вирішення задачі обробки двовимірних масивів відповідно до варіанту: зовнішня – без параметрів, внутрішня має на вході ім’я файлу з даними, на виході – підраховані параметри матриці (перша частина задачі) та перетворену матрицю (друга частина задачі). Для обробки масивів використати функції бібліотеки numpy. Завдання представлено в табл.2.(Matrix13)

*Всі завдання повинні бути реалізовані в одному скрипті у вигляді окремих*

*функцій без параметрів зі строкою документації. Запустити функції з*

*командного вікна.*

*При введенні даних і обчисленнях необхідно передбачити перевірку даних і*

*обробку виняткових ситуацій.*

*Код повинен містити коментарі!*

# ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі Proc7

Вхідні дані (ім’я, опис, тип, обмеження):

input\_data (list): Список цілих чисел.

* *Опис*: Список п'яти натуральних чисел.
* *Тип*: list
* *Обмеження*: Кожне число повинно бути натуральним і знаходитися в інтервалі (0 < K < 10000).

Вихідні дані (ім’я, опис, тип):

output\_data (list): Список цілих чисел.

* *Опис*: Список чисел з оберненим порядком цифр.
* *Тип*: list

Алгоритм вирішення

1.Створити порожній список для введених даних (input\_data).

2.Створити порожній список для виведених даних (output\_data).

3.Виконати цикл 5 разів (оскільки потрібно ввести 5 чисел):

a. Запитати користувача ввести натуральне число (K) та зберегти його у змінній temp.

b. Перевірити, чи число temp є натуральним та належить інтервалу (0 < K < 10000).

c. Якщо умова виконується, додати temp до списку input\_data.

d. В іншому випадку вивести повідомлення про помилку та повторити запитання.

4.Для кожного числа в input\_data:

a. Створити строкове представлення числа.

b. Перевернути рядок.

c. Перетворити отриманий рядок назад у ціле число.

d. Додати отримане число до списку output\_data.

5.Вивести повідомлення "Зворотний порядок цифр:" та вивести всі числа з output\_data.

Лістинг коду вирішення задачі наведено в дод. А (стор. 8-10). Екран роботи програми показаний на рис. Б.1(стор. 9).

Завдання 2. Вирішення задачі Matrix13

* file\_name (str): Ім'я файлу, що містить матрицю.
  + Опис: Змінна, яка містить ім'я файлу, введене користувачем.
  + Тип даних: Рядок (строка).
  + Діапазон допустимих значень: Будь-який рядок.
* determinant (float): Детермінант матриці.
  + Опис: Результат обчислення детермінанта матриці.
  + Тип даних: Дійсне число з плаваючою точкою (float).
  + Діапазон допустимих значень: Будь-яке дійсне число.
* inverse\_matrix (numpy.ndarray): Обернена матриця.
  + Опис: Результат обчислення оберненої матриці.
  + Тип даних: Numpy масив (numpy.ndarray).
  + Діапазон допустимих значень: Будь-який масив числів.

Вхідні дані:

* Введення користувача: file\_name (ім'я файлу).

Вихідні дані:

* Вивід на екран: determinant (детермінант матриці) та inverse\_matrix (обернена матриця).

Алгоритм вирішення

* 1.Введення ім'я файлу:
  + Запитати користувача про ім'я файлу з даними матриці (змінна file\_name).
* 2.Читання матриці з файлу:
  + Спробувати зчитати матрицю з файла, використовуючи бібліотеку NumPy та функцію np.loadtxt(file\_name).
* 3.Обчислення суми рядків:
  + Обчислити суму елементів в кожному рядку матриці, використовуючи matrix.sum(axis=1).
* 4.Обчислення середнього значення рядків:
  + Обчислити середнє значення рядків, поділивши суму рядка на кількість стовпців, використовуючи row\_sums / matrix.shape[1].
* 5.Транспонування матриці:
  + Транспонувати матрицю, використовуючи matrix.T.
* 6.Ділення транспонованої матриці на середні значення рядків:
  + Поділити транспоновану матрицю на середнє значення рядків, використовуючи transformed\_matrix = matrix.T / row\_means.
* 7.Обчислення детермінанту та оберненої матриці:
  + Спробувати обчислити детермінант та обернену матрицю за допомогою np.linalg.det(transformed\_matrix) та np.linalg.inv(transformed\_matrix).
* 8.Вивід результатів:
  + Вивести на екран обчислені значення детермінанту та оберненої матриці.
* 9.Обробка помилок:
  + Обробити можливі винятки, якщо щось пішло не так, та вивести повідомлення про помилку.

Лістинг коду вирішення задачі наведено в дод. А (стор. 8-10). Екран роботи програми показаний на рис. Б.2-3(стор. 11-12).

# ВИСНОВКИ

Під час виконання завдань було вивчено та закріплено практичні навички роботи з обертанням чисел та обробкою матриць в мові програмування Python. Код було вдосконалено та оптимізовано для ефективної обробки введених даних, що дозволяє отримувати коректні результати у різних ситуаціях.

ДОДАТОК А

Лістинг коду програми до задач 1-2 (Proc 7, Matrix 13)

Меню програми

<

import module

#Меню для вибору відповідного завдання

while True:

print("1. Task 1")

print("2. Task 2")

print("0. Exit")

choice = int(input("Choose a task (0-2): "))

if choice == 0:

break

elif choice == 1:

module.Task1()

elif choice == 2:

module.Task2()

else:

print("Невірний вибір. Виберіть ще раз.")

import module

>

Завдання 1-2 Proc7, Matrix13

<

import numpy as np

def InvDigits(K):

"""

Перетворює число K, обертаючи порядок його цифр.

Аргументи:

K (int): Ціле число, яке потрібно обернути.

Повертає:

int: Число K з оберненим порядком цифр.

Винятки:

ValueError: Генерується, якщо введені дані не є цілим числом або не можуть бути обернуті.

"""

try:

reversed\_K = int(str(K)[::-1])

return reversed\_K

except ValueError:

raise ValueError("Неправильні дані! Будь ласка, введіть додатне ціле число.")

def ReverseDigitsList(lst):

"""

Обертає порядок цифр усіх чисел у списку.

Аргументи:

lst (list): Список цілих чисел.

Повертає:

list: Список чисел з оберненим порядком цифр.

Винятки:

ValueError: Генерується, якщо обертання одного з чисел у списку неможливе.

"""

try:

reversed\_list = [InvDigits(num) for num in lst]

return reversed\_list

except ValueError as ve:

raise ValueError(f"Помилка при обертанні списку: {ve}")

def Task1():

"""

Виконує перше завдання: вводить п'ять чисел, обертає їх порядок цифр і виводить результат.

"""

input\_data = []

for \_ in range(5):

try:

temp = int(input("Введіть натуральне число (0 < K < 10000): "))

input\_data.append(temp)

except ValueError:

print("Неправильний формат числа. Спробуйте ще раз.")

try:

output\_data = ReverseDigitsList(input\_data)

print("Зворотний порядок цифр: ", output\_data)

except ValueError as ve:

print(f"Помилка: {ve}")

return

def ProcessMatrix(file\_name):

"""

Обробляє матрицю з файла.

Аргументи:

file\_name (str): Ім'я файлу, що містить матрицю.

Повертає:

tuple: Детермінант та обернена матриця.

Винятки:

FileNotFoundError: Генерується, якщо файл не знайдено.

ValueError: Генерується, якщо виникають проблеми з обробкою матриці.

"""

try:

matrix = np.loadtxt(file\_name)

row\_sums = matrix.sum(axis=1)

row\_means = row\_sums / matrix.shape[1]

transformed\_matrix = matrix.T / row\_means

determinant = np.linalg.det(transformed\_matrix)

inverse\_matrix = np.linalg.inv(transformed\_matrix)

return determinant, inverse\_matrix

except FileNotFoundError:

raise FileNotFoundError(f"Файл {file\_name} не знайдено.")

except Exception as e:

raise ValueError(f"Помилка при обробці матриці: {e}")

def Task2():

"""

Виконує друге завдання: обробляє матрицю з файла, обчислює детермінант та виводить обернену матрицю.

"""

file\_name = input("Введіть назву файлу з даними матриці: ")

try:

determinant, inverse\_matrix = ProcessMatrix(file\_name)

print(f"Детермінант: {determinant}")

print("Обернена матриця:")

print(inverse\_matrix)

except (FileNotFoundError, ValueError) as e:

print(f"Помилка: {e}")

>

ДОДАТОК Б

Скрін-шоти вікна виконання програми

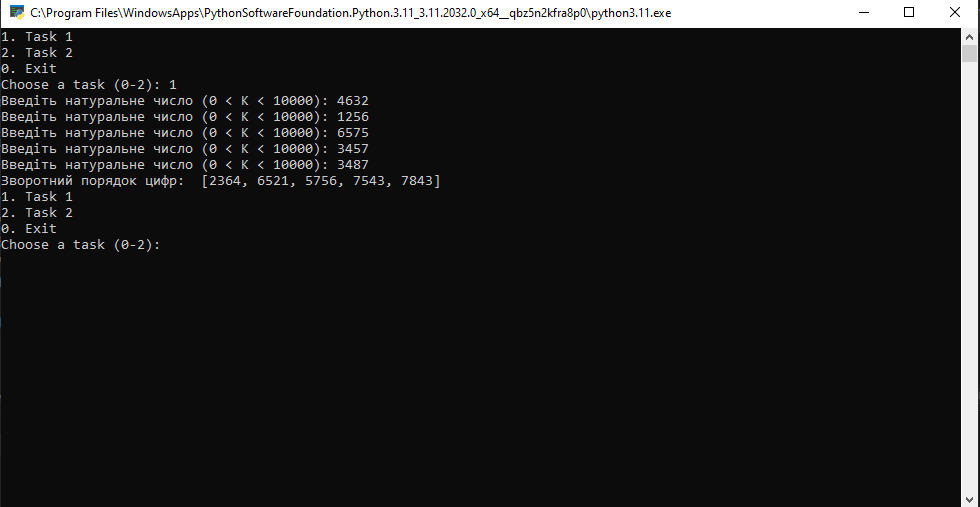


Рисунок Б.1 – Екран виконання програми для вирішення завдання   
 Proc7

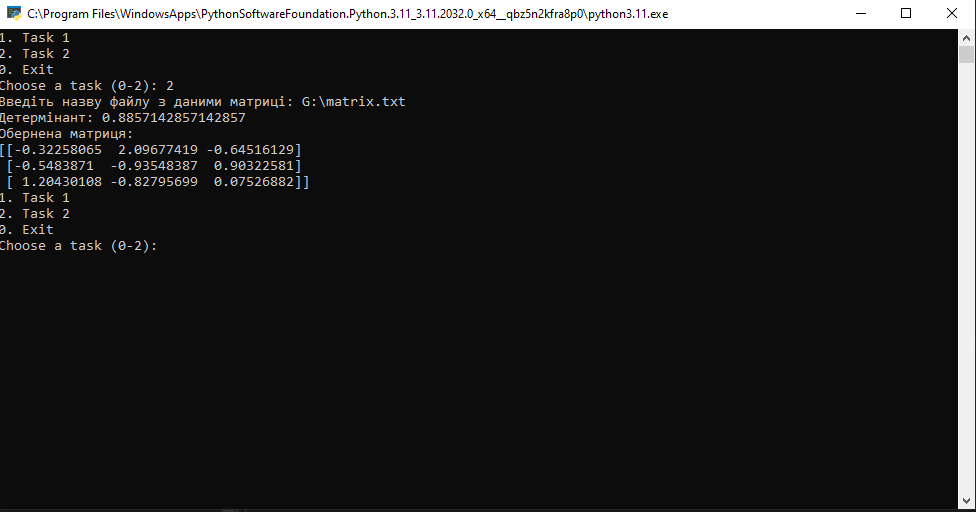


Рисунок Б.2 – Екран виконання програми для вирішення завдання   
Matrix13

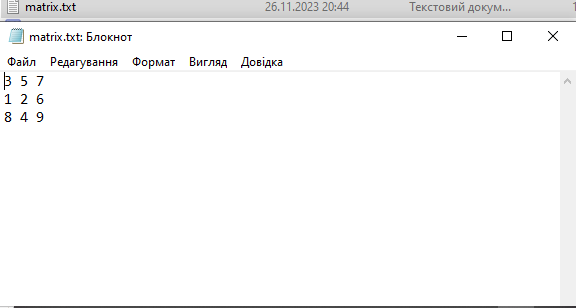


Рисунок Б.3 – Екран виконання програми для вирішення завдання   
Matrix13