МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Лабораторна робота № 4

з дисципліни «Об'єктно-орієнтоване програмування СУ»

Тема: «Реалізація класу і робота з об'єктами»

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

Застосувати теоретичні знання з основ програмування на мові Python з

використанням об'єктів і класів, навички використання бібліотеки для

візуалізації масивів даних, і навчитися розробляти скрипти для роботи з

об'єктами призначених для користувача класів.

# ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Визначити клас Point\_n (n – номер варіанту), який реалізує

абстракцію з атрибутами:

1) дві дійсні координати точки на площині (властивості, приховані

змінні екземпляра),

− для кожної метод-геттер (повертає відповідну координату),

− для кожної метод-сеттер (записуює відповідну координату, якщо

вона у межах [-100, 100], інакше – дорівнює 0))

2) кількість створених екземплярів точки (змінна класу),

3) метод класу (повертає кількість створених примірників),

4) конструктор з двома параметрами (за замовчуванням),

5) деструктор, що виводить відповідне повідомлення,

6) метод, що змінює координати точки з двома вхідними дійсними

параметрами:

− зсув по х,

− зсув по у.

Завдання 2. Виконати операції з об'єктами даного класу відповідно до

варіанту (див. таб.1).

Завдання 3. Використовуючи пакет matplotlib, відобразити створені об'єкти

в графічному вікні до і після змін.

Завдання 4. Зберегти координати точок у текстовому файлі у форматі:

номер: координата\_х; координата\_у – для непарних варіантів

(номер) координата\_х:координата\_у – для парних варіантів

Код в лістингу програми повинен містити докладні коментарі !!!

У звіті повинно бути дві діаграми: Діаграма класів і діаграма активності

для основного сценарію роботи з об'єктами.

# ВИКОНАННЯ РОБОТИ

Завдання 1-4. Вирішення задачі 23

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

* x: x-координата точки
  + Опис: Координата x точки
  + Тип: Число з плаваючою точкою
  + Обмеження: Любе число
* y: y-координата точки
  + Опис: Координата y точки
  + Тип: Число з плаваючою точкою
  + Обмеження: Любе число

Вихідні дані (ім’я, опис, тип):

* count: Кількість створених екземплярів
  + Опис: Кількість створених екземплярів класу Point\_23
  + Тип: Ціле число
* x: x-координата точки
  + Опис: Координата x точки
  + Тип: Число з плаваючою точкою
* y: y-координата точки
  + Опис: Координата y точки
  + Тип: Число з плаваючою точкою
* distance\_before: Відстань між другою і третьою точками перед змінами
  + Опис: Відстань між другою і третьою точками до змін
  + Тип: Число з плаваючою точкою
* x\_values\_before: Координати x точок перед змінами
  + Опис: Список координат x точок перед змінами
  + Тип: Список чисел з плаваючою точкою
* y\_values\_before: Координати y точок перед змінами
  + Опис: Список координат y точок перед змінами
  + Тип: Список чисел з плаваючою точкою
* x\_values\_after: Координати x точок після змін
  + Опис: Список координат x точок після змін
  + Тип: Список чисел з плаваючою точкою
* y\_values\_after: Координати y точок після змін
  + Опис: Список координат y точок після змін
  + Тип: Список чисел з плаваючою точкою
* distance\_after: Відстань між другою і третьою точками після змін
  + Опис: Відстань між другою і третьою точками після змін
  + Тип: Число з плаваючою точкою
* filename: Назва файлу для збереження координат
  + Опис: Назва файлу, у якому зберігаються координати точок
  + Тип: Рядок (строка)

Алгоритм вирішення завдання:

1. Створити три об'єкти класу Point\_23, які представляють три точки.

2. Обчислити відстань між другою та третьою точкою за допомогою функції calculate\_distance().

3. Вивести відстань між другою та третьою точкою перед змінами.

4. Зсунути координати першої точки на 20 вниз і на 30 вправо за допомогою методу shift().

5. Вивести нові координати всіх точок.

6. Вивести відстань між другою та третьою точкою після змін.

7. Зберегти координати всіх точок у текстовому файлі "coordinates.txt".

Лістинг коду вирішення задачі наведено в дод. А (стор. 6-8). Екран роботи програми показаний на рис. Б.1-4.

# ВИСНОВКИ

Під час виконання даного коду було вивчено принципи об'єктно-орієнтованого програмування на прикладі класу Point\_23, який визначає точку в двовимірному просторі. Було закріплено поняття конструктора, методів, властивостей та статичних методів. Практично відпрацьовано зсув координат точок, обчислення відстані між ними та збереження координат у текстовому файлі. В результаті роботи отримано навички роботи з класами, обробки помилок та використання бібліотеки Matplotlib для візуалізації даних.

ДОДАТОК А

Лістинг коду програми до задачі

<

import math

import matplotlib.pyplot as plt

**#Завдання 1**

class Point\_23:

# Змінна класу для зберігання кількості створених екземплярів

count = 0

def \_\_init\_\_(self, x=0.0, y=0.0):

"""

Конструктор класу Point\_23.

Параметри:

- x: x-координата (за замовчуванням 0.0)

- y: y-координата (за замовчуванням 0.0)

"""

# Перевірка і встановлення x та y координат

self.\_x = self.\_validate\_coordinate(x)

self.\_y = self.\_validate\_coordinate(y)

# Збільшення кількості створених екземплярів

Point\_23.count += 1

def \_\_del\_\_(self):

"""Деструктор, виводить відповідне повідомлення."""

print(f"Екземпляр Point\_23 з координатами ({self.\_x}, {self.\_y}) видалено.")

# Зменшення кількості створених екземплярів

Point\_23.count -= 1

def \_validate\_coordinate(self, coord):

"""Перевірка та встановлення координати в межах [-100, 100]."""

if -100 <= coord <= 100:

return coord

else:

return 0.0

@property

def x(self):

"""Метод для отримання x-координати."""

return self.\_x

@property

def y(self):

"""Метод для отримання y-координати."""

return self.\_y

@staticmethod

def get\_instance\_count():

"""Метод класу, який повертає кількість створених екземплярів."""

return Point\_23.count

def shift(self, dx, dy):

"""

Метод для зміни координат точки на dx вздовж x та dy вздовж y.

Параметри:

- dx: Зсув вздовж x

- dy: Зсув вздовж y

"""

# Збільшення або зменшення координат відповідно до зсувів

self.\_x += dx

self.\_y += dy

# Перевірка та встановлення нових координат в межах [-100, 100]

self.\_x = self.\_validate\_coordinate(self.\_x)

self.\_y = self.\_validate\_coordinate(self.\_y)

def calculate\_distance(point1, point2):

"""

Функція для обчислення відстані між двома точками.

Параметри:

- point1, point2: Екземпляри класу Point\_23

"""

return math.sqrt((point2.x - point1.x)\*\*2 + (point2.y - point1.y)\*\*2)

**# Завдання 2**

# Створити список із трьох точок

points\_list = [Point\_23(), Point\_23(10,15), Point\_23(-25,30)]

# Обчислити відстань між другою і третьою точками

distance\_before = calculate\_distance(points\_list[1], points\_list[2])

print(f"Відстань між другою і третьою точками перед змінами: {distance\_before}")

# Зсунути координати першої точки

points\_list[0].shift(30, -20)

**# Завдання 3**

# Відобразити точки до та після змін

x\_values\_before = [point.x for point in points\_list]

y\_values\_before = [point.y for point in points\_list]

# Змінити координати

points\_list[0].shift(30, -20)

x\_values\_after = [point.x for point in points\_list]

y\_values\_after = [point.y for point in points\_list]

# Відобразити на графіку

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.scatter(x\_values\_before, y\_values\_before, color='blue')

plt.title('Точки перед змінами')

plt.subplot(1, 2, 2)

plt.scatter(x\_values\_after, y\_values\_after, color='red')

plt.title('Точки після змін')

plt.show()

# Відобразити відстань після змін

distance\_after = calculate\_distance(points\_list[1], points\_list[2])

print(f"Відстань між другою і третьою точками після змін: {distance\_after}")

**# Завдання 4**

filename = "coordinates.txt"

with open(filename, "w") as file:

for i, point in enumerate(points\_list, start=1):

file.write(f"{i}: {point.x}; {point.y}\n")

print(f"Координати збережено у файлі {filename}.")

>

ДОДАТОК Б

Скрін-шоти вікна виконання програми

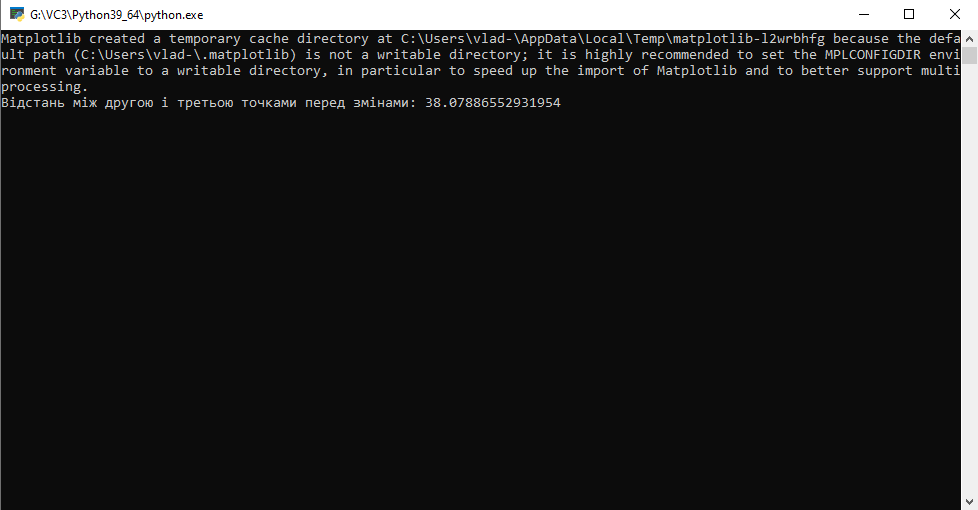


Рисунок Б.1 – Екран виконання програми для вирішення завдання

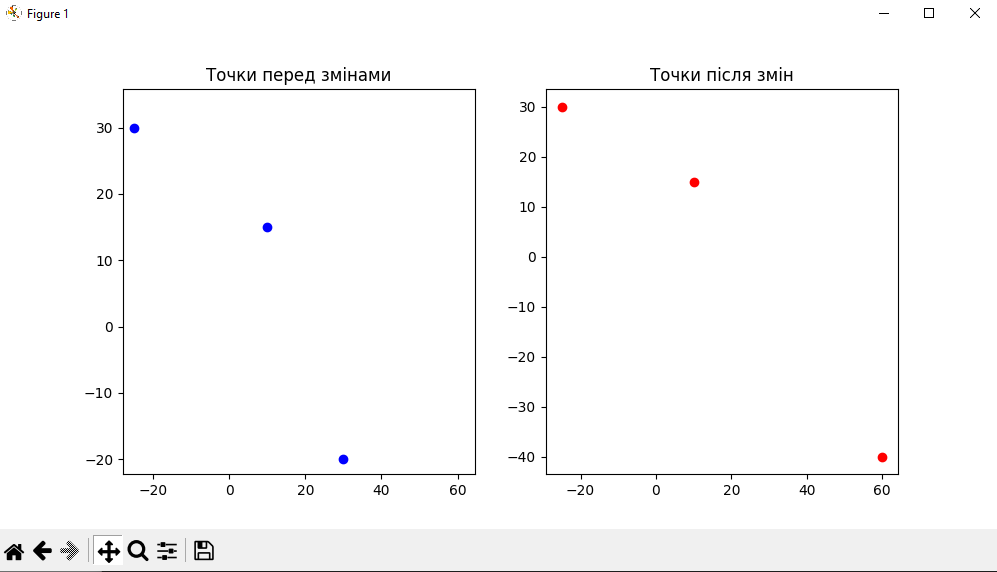


Рисунок Б.2 – Екран виконання програми для вирішення завдання

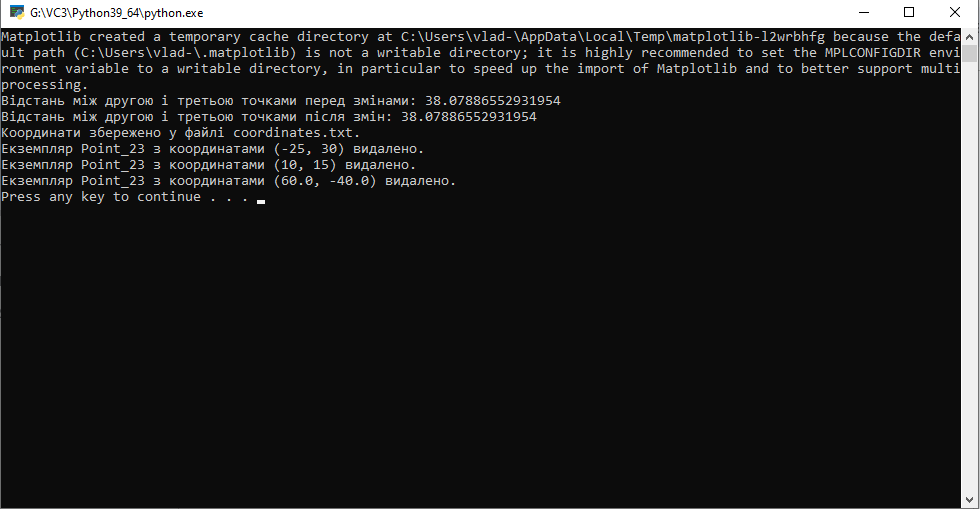


Рисунок Б.3 – Екран виконання програми для вирішення завдання

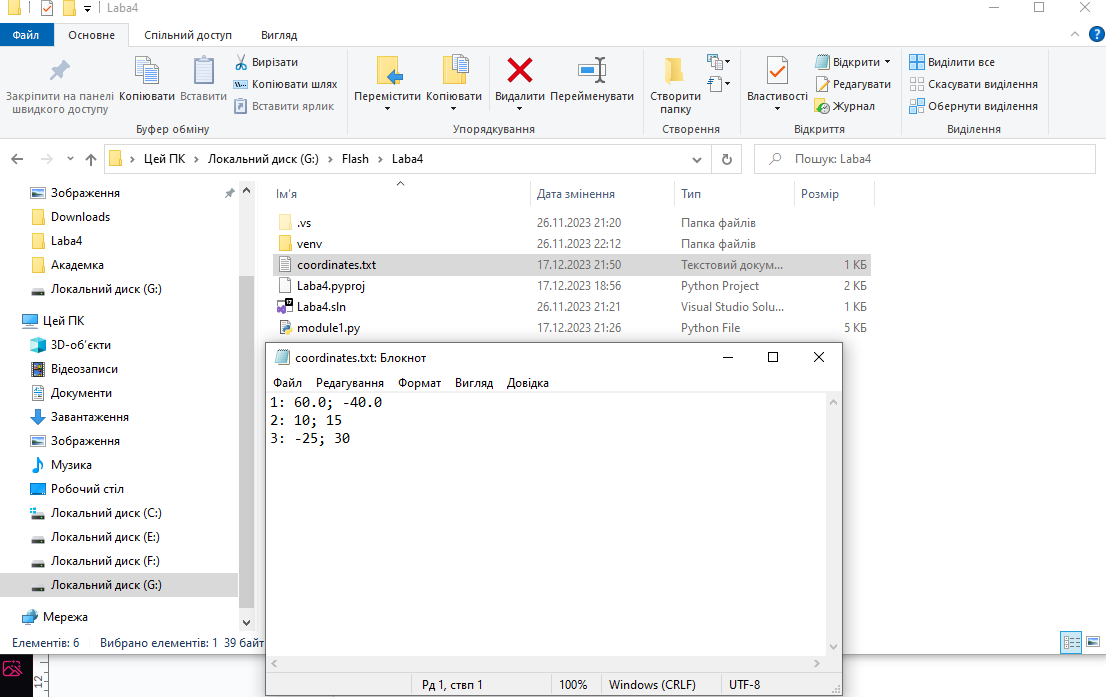


Рисунок Б.4 – Екран виконання програми для вирішення завдання