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 авіаційно-транспортних систем»

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

ХАІ.301.173.320.03 ЛР

Виконав студент гр. _____320_____

_____Полякова Софія___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
_____ ас. Є. В. Пявка
(підпис, дата) (П.І.Б.)

2024

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 (записує 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 x,

− зсув по y.

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

3. Створити список з трьох точок, порахувати відстань між першою і третьою, пересунути другу на 15 вліво.

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_x; координата_y

ВИКОНАННЯ РОБОТИ

Завдання 1. Клас point_3

Алгоритм вирішення показано в табл 1.

Таблиця 1. — Діаграма класу

Клас	point_3
Атрибути	- x: float - y: float - instance_count: int
Методи	+__init__(self, x: float, y: float) +_validate_coordinate(self, value: float) +x(self) — float (getter) +x(self, value: float) (setter) +y(self) — float (getter) +y(self, value: float) (setter) +move(self, shift_x: float, shift_y: float) +__del__(self) +get_instance_count() — int

Лістинг коду вирішення задачі наведено в дод. А (стор. 6)

Завдання 2,3 та 4.

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

points — лист для точок з координатами x та y (три точки);

x — координата x для кожної точки; дійсний тип; $-100 \leq x \leq 100$;

y — координата y для кожної точки; дійсний тип; $-100 \leq x \leq 100$

Вихідні дані (ім’я, опис, тип):

distance — розрахунок дистанції між першою та третьою координатами; дійсний тип;

points[1] — зміненні координати для другої точки; дійсний тип;

points — точки, які вписані у файл points_coordinates.txt; дійсний тип.

Алгоритм вирішення показано в рис. 1 та допоміжні функції на рис. 2.

Лістинг коду вирішення задачі наведено в дод. Б (стор. 7). Екран роботи програми показаний на рис. В.1 - В.2.

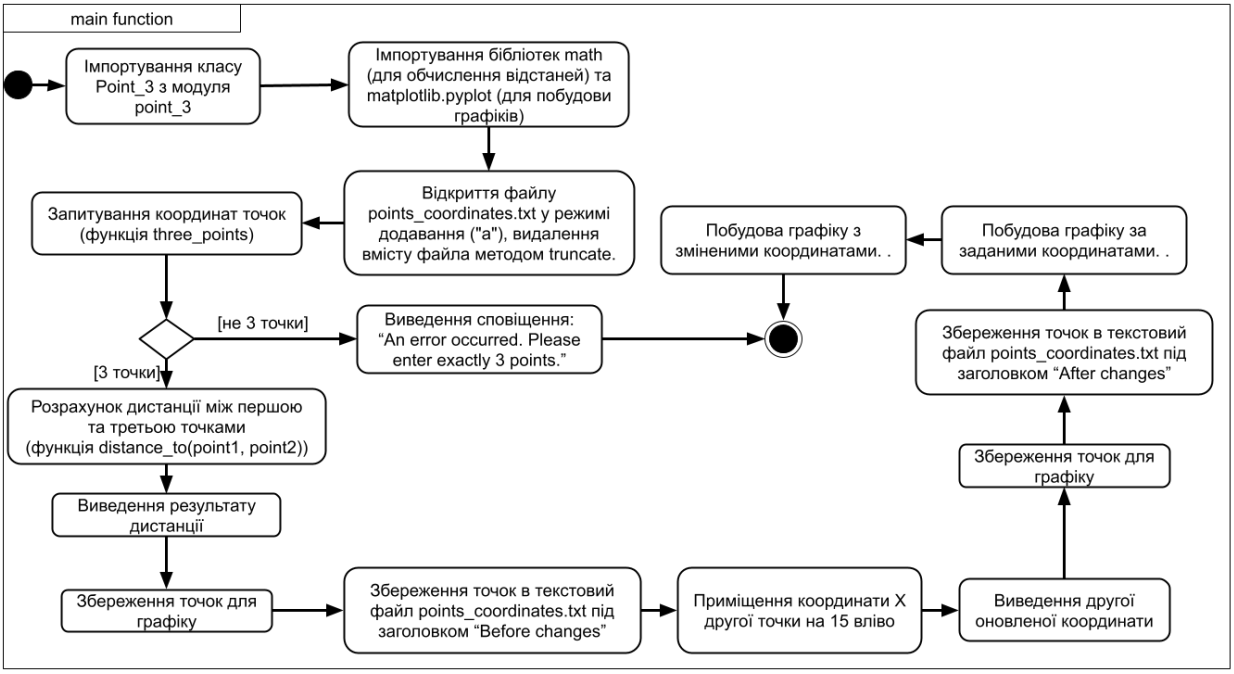

Рисунок 1 — Алгоритм вирішення задачі

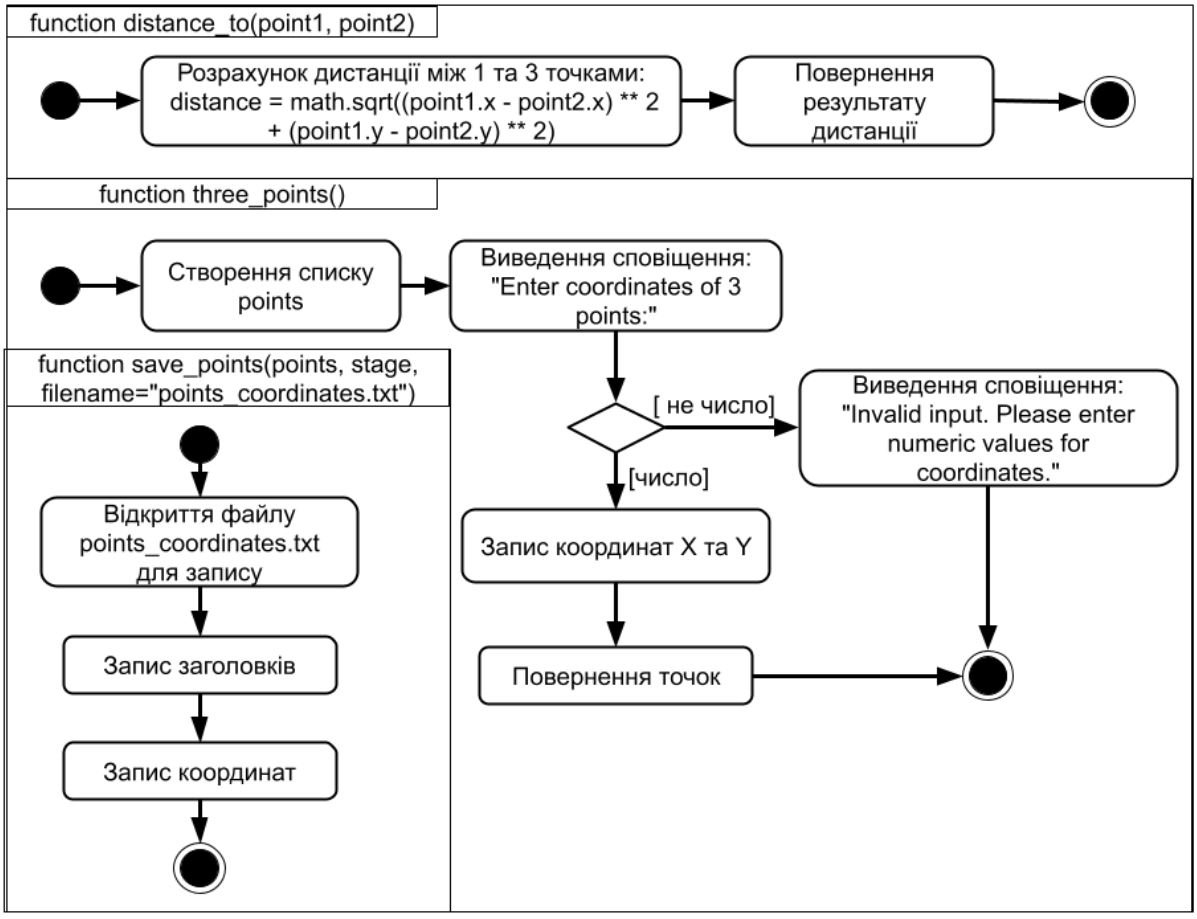

Рисунок 2 — Алгоритм вирішення допоміжних функцій

ВИСНОВКИ

В лабораторній роботі було вивчено теоретичний матеріал з основ програмування на мові Python з використанням об'єктів і класів. Відпрацьовано на коді використання бібліотеки для візуалізації масивів даних. Отримано навички створення класів. Закріплено на практиці будування графіків у середовищі програмування.

ДОДАТОК А
Лістинг коду програми до задачі Class

```
class Point_3:
  # class variable to count the number of instances (private)
  __instance_count = 0

def __init__(self, x, y):
   # private variables for coordinates
    self.x = x
    self.y = y
    Point_3.__instance_count += 1

# method for checking coordinates
def _validate_coordinate(self, value):
     return value if -100 <= value <= 100 else 0

# properties for x and y with validation
@property
def x(self):
     return self.__x

@x.setter
def x(self, value):
     self.__x = self._validate_coordinate(value)

@property
def y(self):
     return self.__y

@y.setter
def y(self, value):
     self.__y = self._validate_coordinate(value)

# class method to get the number of instances
@staticmethod
def get_instance_count():
     return Point_3.__instance_count

# method for changing coordinates with an offset
def move(self, shift_x, shift_y):
     self.x += shift_x
     self.y += shift_y

# destructor
def __del__(self):
     Point_3.__instance_count -= 1
```

ДОДАТОК А
Лістинг коду програми до задач 2,3 та 4

```
from point_3 import Point_3
import math
import matplotlib.pyplot as plt

    # clear the file
filename = "points_coordinates.txt"
with open(filename, "a") as file: # use "a" to append to file
    file.seek(0)  # move to the beginning of the file
    file.truncate()  # remove everything from the file

    # function for calculating the distance between two points
def distance_to(point1, point2):
        distance = math.sqrt((point1.x - point2.x) ** 2 + (point1.y - point2.y)
** 2)
        return round(distance, 2)

    # function for reading exactly three points from user input
def three_points():
        points = [] # list for coordinates
        print("Enter coordinates of 3 points:")
        try:
            for i in range(3): # entering coordinates
                x = float(input(f"Enter X of point {i + 1}: "))
                y = float(input(f"Enter Y of point {i + 1}: "))
                point = Point_3(x, y)
                points.append(point)
        except ValueError: # error notification
            print("Invalid input. Please enter numeric values for coordinates.")
        return points

    # function for saving point coordinates into the file
def save_points(points, stage, filename="points_coordinates.txt"):
        with open(filename, "a") as file:
            file.write(f"{stage}:\n")
            for i, point in enumerate(points, start=1):
                # write coordinates
                file.write(f" {i}: {point.x}; {point.y}\n")
            file.write("\n")

    # reading exactly three points from user input
points = three_points()

    # ensure exactly three points are provided
if len(points) == 3:
        # calculation of the distance between the first and third points
        distance = distance_to(points[0], points[2])
```

```
        print(f"Distance between point 1 and point 3: {distance}")

        # points with original coordinates
        initial_x = [p.x for p in points]
        initial_y = [p.y for p in points]

        # save the initial coordinates in a file labeled "Before changes"
        save_points(points, "Before changes")

        # move the second point 15 units to the left
        points[1].move(-15, 0)
        print(f"New coordinates of point 2: ({points[1].x}, {points[1].y})")

        # points with new coordinates
        updated_x = [p.x for p in points]
        updated_y = [p.y for p in points]

        # save the new coordinates in a file labeled "After changes"
        save_points(points, "After changes")

        # plotting
        plt.figure(figsize=(8, 6))

        # plot before changes
        plt.subplot(1, 2, 1)
        plt.scatter(initial_x, initial_y, color='green', label='Before')
        plt.plot(initial_x, initial_y, color='green', linestyle='--',
marker='o')
        plt.title("Before the changes")
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.legend()

        # plot after changes
        plt.subplot(1, 2, 2)
        plt.scatter(updated_x, updated_y, color='red', label='After')
        plt.plot(updated_x, updated_y, color='red', linestyle='--', marker='o')
        plt.title("After the changes")
        plt.xlabel("X")
        plt.ylabel("Y")
        plt.legend()

        # show both plots
        plt.show()
else:
        print("An error occurred. Please enter exactly 3 points.")
```

ДОДАТОК В

Скрін-шоти вікна виконання програми

```
Enter coordinates of 3 points:
Enter X of point 1: 76
Enter Y of point 1: 33
Enter X of point 2: 65
Enter Y of point 2: 24
Enter X of point 3: 23
Enter Y of point 3: 54
Distance between point 1 and point 3: 57.01
New coordinates of point 2: (50.0, 24.0)
```

Рисунок В.1 – Екран виконання програми для вирішення завдання

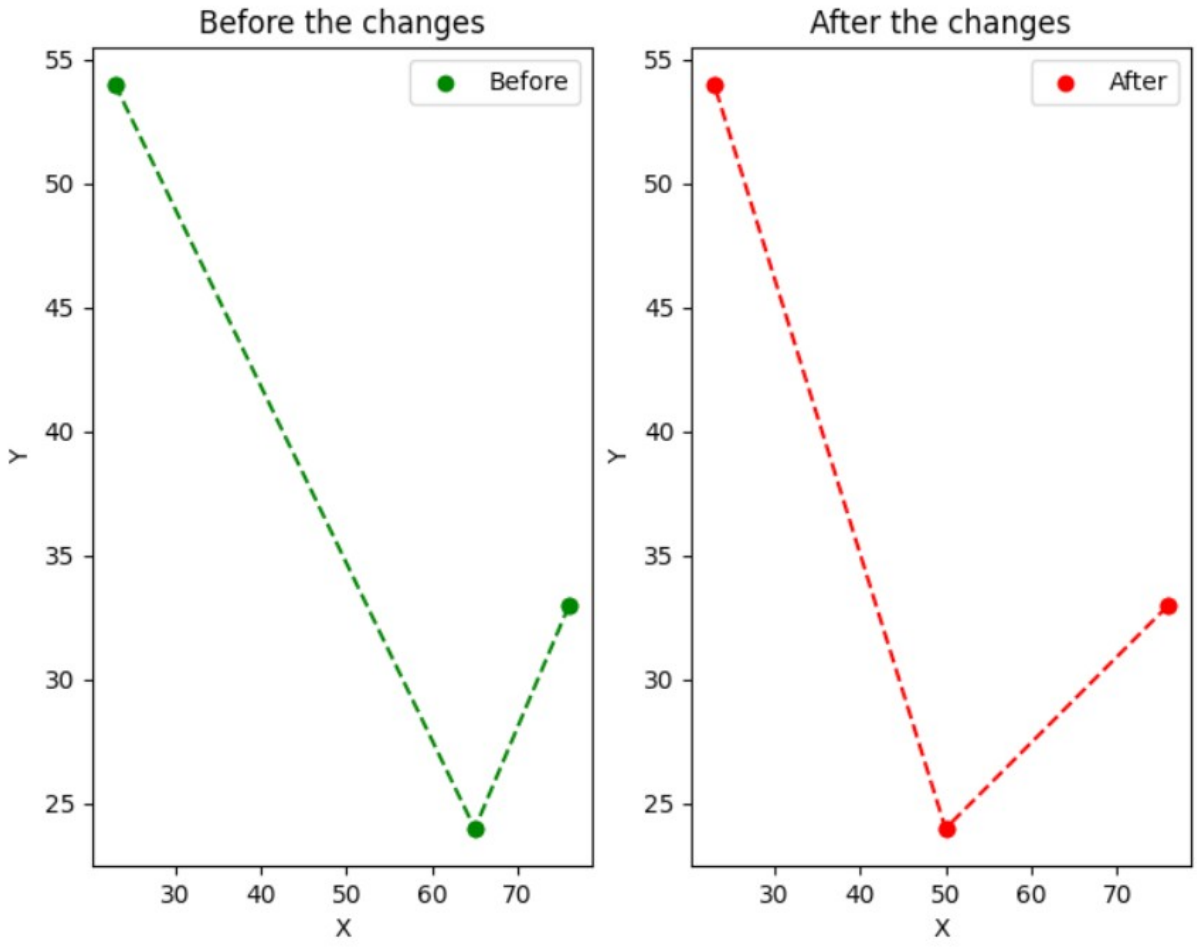

Рисунок В.2 – Графік виконання програми для вирішення завдання