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 проектування СУ»

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

ХАІ.301. 173. 3-91АВ/С. 4 ЛР

Виконав студент гр. _____3-91АВ/С_____

_____Опанасюк Олександр___
(підпис, дата) (П.І.Б.)

Перевірив
_____ к.т.н., доц. О. В. Гавриленко
(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

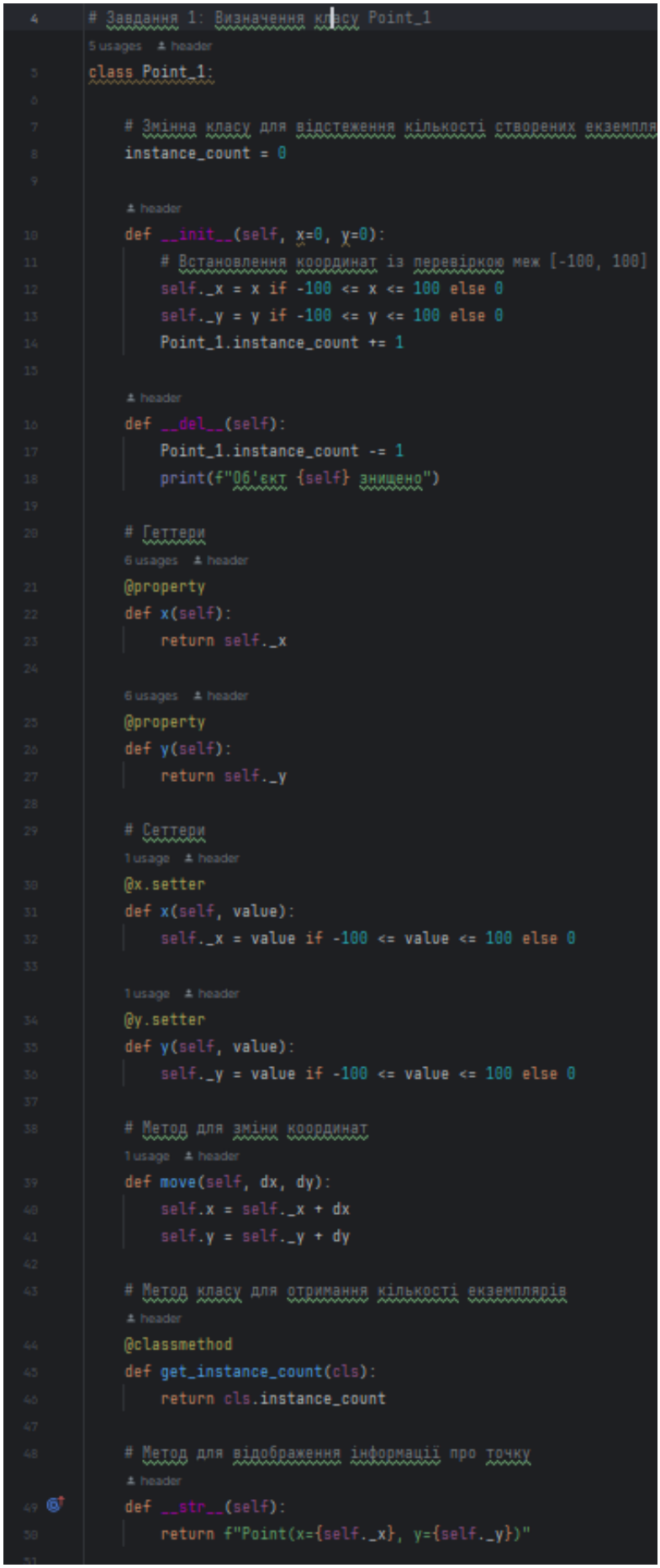
Завдання 1. Визначити клас Point_n (n – номер варіанту), який реалізує абстракцію з атрибутами:
1) дві дійсні координати точки на площині (властивості, приховані змінні екземпляра),
− для кожної метод-геттер (повертає відповідну координату),
− для кожної метод-сеттер (записує відповідну координату, якщо 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 (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_x; координата_y – для непарних варіантів
(номер) координата_x:координата_y – для парних варіантів

# ВИКОНАННЯ РОБОТИ

Завдання 1. – Створення та візуалізація точок на площині, №1
Вхідні дані:
x – координата точки по осі X; тип: float; допустимі значення: [-100, 100].
y – координата точки по осі Y; тип: float; допустимі значення: [-100, 100].
Вихідні дані:
x – координата точки по осі X після обробки; тип: float.
y – координата точки по осі Y після обробки; тип: float.
Текстові повідомлення – при створенні або знищенні об'єктів класу.

Алгоритм вирішення показано нижче на рис. 1.

Реалізовано клас Point_1 з наступними атрибутами:
- x та y – координати точки.
- Геттери та сеттери для роботи з координатами.
- Конструктор для ініціалізації об'єктів.
- Метод для зміни координат точки.
- Класовий метод для підрахунку екземплярів.
- Деструктор для очищення об'єктів.



```python
# Завдання 1: Визначення класу Point_1
class Point_1:

    # Змінна класу для відстеження кількості створених екземпля
    instance_count = 0

    def __init__(self, x=0, y=0):
        # Встановлення координат із перевіркою меж [-100, 100]
        self._x = x if -100 <= x <= 100 else 0
        self._y = y if -100 <= y <= 100 else 0
        Point_1.instance_count += 1

    def __del__(self):
        Point_1.instance_count -= 1
        print(f"Об'єкт {self} знищено")

    # Геттери
    @property
    def x(self):
        return self._x

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

    # Сеттери
    @x.setter
    def x(self, value):
        self._x = value if -100 <= value <= 100 else 0

    @y.setter
    def y(self, value):
        self._y = value if -100 <= value <= 100 else 0

    # Метод для зміни координат
    def move(self, dx, dy):
        self.x = self._x + dx
        self.y = self._y + dy

    # Метод класу для отримання кількості екземплярів
    @classmethod
    def get_instance_count(cls):
        return cls.instance_count

    # Метод для відображення інформації про точку
    def __str__(self):
        return f"Point(x={self._x}, y={self._y})"
```

Рисунок 1 – Алгоритм вирішення завдання 1

Завдання 2. Виконання операцій з об'єктами
Вхідні дані:
points – список об'єктів класу Point_1.

Вихідні дані:
distance – відстань між першою та другою точками; тип: float.
new_coordinates – нові координати точок після зміщення; тип: list[tuple[float, float]].

Алгоритм вирішення, показано на рис. 2:
- Створено три об'єкти точок з початковими координатами.
- Вирахувано відстань між першою та другою точками за формулою Евкліда.
- Третю точку зміщено на 10 одиниць вліво.

```
52  # Завдання 2: Робота з об'єктами
53  # Створення списку з трьох точок
54  point1 = Point_1( x: 20,  y: 30)
55  point2 = Point_1( x: 50,  y: 60)
56  point3 = Point_1(-90,  y: 0)
57
58  # Обчислення відстані між першою і другою точками
59  distance = math.sqrt((point2.x - point1.x) ** 2 + (point2.y - point1.y) ** 2)
60  print(f"Відстань між точками: {distance:.2f}")
61
62  # Пересування третьої точки на 10 вліво
63  point3.move(-10,  dy: 0)
64
65  # Відображення результатів
66  points_before = [(20, 30), (50, 60), (-90, 0)]
67  points_after = [(point1.x, point1.y), (point2.x, point2.y), (point3.x, point3.y)]
```

Рисунок 2 - Робота з об'єктами

Завдання 3. Візуалізація

Вхідні дані:
points_before – координати точок до змін; тип: list[tuple[float, float]].
points_after – координати точок після змін; тип: list[tuple[float, float]].

Вихідні дані:
Графічне зображення точок у вигляді діаграми до та після змін.

Алгоритм вирішення, показаний на рис. 3:
- Використано бібліотеку matplotlib для побудови графіків точок.
- Відображено точки до та після змін у двох окремих графіках.

```
69  # Завдання 3: Візуалізація точок
    2 usages  header
70  def plot_points(points, title):
71      x_coords, y_coords = zip(*points)
72      plt.scatter(x_coords, y_coords, color='blue', label='Точки')
73      for i, (x, y) in enumerate(points, 1):
74          plt.text(x, y, f'P{i}', fontsize=12, ha='right')
75      plt.title(title)
76      plt.xlabel('X')
77      plt.ylabel('Y')
78      plt.axhline(0, color='black',linewidth=0.5)
79      plt.axvline(0, color='black',linewidth=0.5)
80      plt.grid()
81      plt.legend()
82      plt.show()
83
84  # Відображення точок до змін
85  plot_points(points_before,  title: "Точки до змін")
86
87  # Відображення точок після змін
88  plot_points(points_after,  title: "Точки після змін")
```

Рисунок 3 – Візуалізація точок

Завдання 4. Збереження координат у файл

Вхідні дані:
points_after – координати точок після змін; тип: list[tuple[float, float]].

Вихідні дані:
Текстовий файл points.txt, у якому збережено координати точок у форматі:
номер: координата_x; координата_y.

Алгоритм вирішення, показаний на рис. 4:
- Відкрито файл для запису у текстовому режимі.
- Збережено нові координати точок у вказаному форматі.

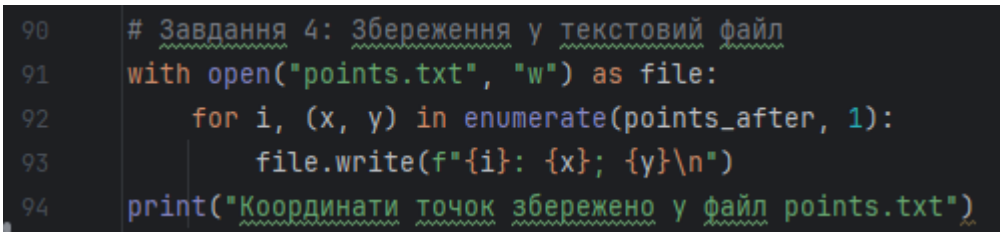



Рисунок 4 – Збереження результатів у текстовий файл

Також додаємо діаграму класів (рис. 5) та діаграму активності для основного сценарію роботи з об'єктами. (рис. 6)

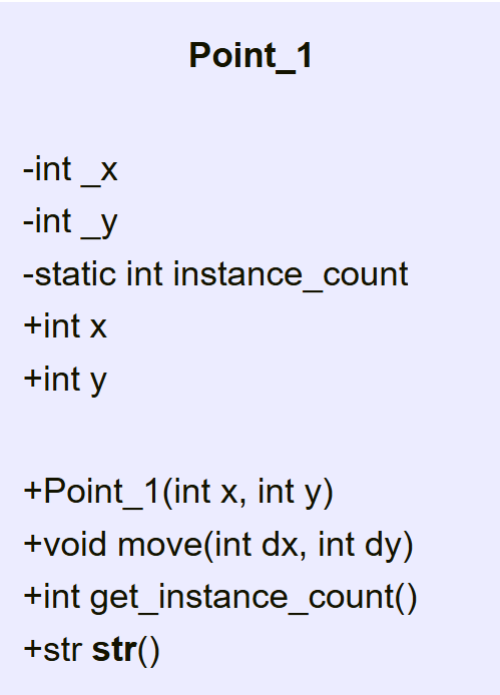



Рисунок 5 – Діграма класів

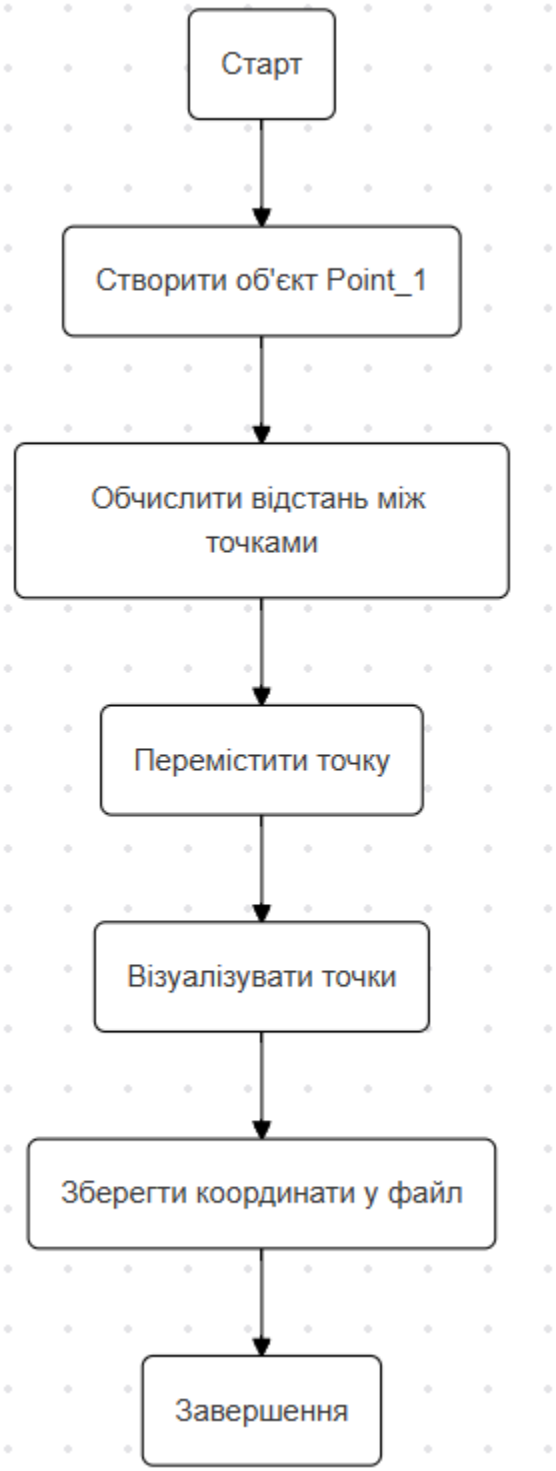



Рисунок 6 – Діаграма активності для основного сценарію роботи з об’єктами

# ВИСНОВКИ

Було вивчено основи об’єктно-орієнтованого програмування в Python, реалізовано клас із прихованими атрибутами, геттерами та сеттерами. Закріплено знання роботи з бібліотекою matplotlib для графічного представлення даних. Отримано навички роботи з текстовими файлами для збереження результатів обчислень.

# ДОДАТОК А

## Лістинг коду програми до задач №1: Визначення класу Point_1, №2: Робота з об'єктами, №3: Візуалізація точок, №4: Збереження у текстовий файл

```
<
import math  # Для обчислення відстаней між точками
import matplotlib.pyplot as plt  # Для візуалізації точок

# Завдання 1: Визначення класу Point_1
class Point_1:

    # Змінна класу для відстеження кількості створених екземплярів
    instance_count = 0

    def __init__(self, x=0, y=0):
        # Встановлення координат із перевіркою меж [-100, 100]
        self._x = x if -100 <= x <= 100 else 0
        self._y = y if -100 <= y <= 100 else 0
        Point_1.instance_count += 1

    def __del__(self):
        Point_1.instance_count -= 1
        print(f"Об'єкт {self} знищено")

    # Геттери
    @property
    def x(self):
        return self._x

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

    # Сеттери
    @x.setter
    def x(self, value):
        self._x = value if -100 <= value <= 100 else 0

    @y.setter
    def y(self, value):
        self._y = value if -100 <= value <= 100 else 0

    # Метод для зміни координат
    def move(self, dx, dy):
        self.x = self._x + dx
        self.y = self._y + dy

    # Метод класу для отримання кількості екземплярів
    @classmethod
```

```
    def get_instance_count(cls):
        return cls.instance_count

    # Метод для відображення інформації про точку
    def __str__(self):
        return f"Point(x={self._x}, y={self._y})"

# Завдання 2: Робота з об'єктами
# Створення списку з трьох точок
point1 = Point_1(20, 30)
point2 = Point_1(50, 60)
point3 = Point_1(-90, 0)

# Обчислення відстані між першою і другою точками
distance = math.sqrt((point2.x - point1.x) ** 2 + (point2.y - point1.y) ** 2)
print(f"Відстань між точками: {distance:.2f}")

# Пересування третьої точки на 10 вліво
point3.move(-10, 0)

# Відображення результатів
points_before = [(20, 30), (50, 60), (-90, 0)]
points_after = [(point1.x, point1.y), (point2.x, point2.y), (point3.x, point3.y)]

# Завдання 3: Візуалізація точок
def plot_points(points, title):
    x_coords, y_coords = zip(*points)
    plt.scatter(x_coords, y_coords, color='blue', label='Точки')
    for i, (x, y) in enumerate(points, 1):
        plt.text(x, y, f'P{i}', fontsize=12, ha='right')
    plt.title(title)
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.axhline(0, color='black',linewidth=0.5)
    plt.axvline(0, color='black',linewidth=0.5)
    plt.grid()
    plt.legend()
    plt.show()

# Відображення точок до змін
plot_points(points_before, "Точки до змін")

# Відображення точок після змін
plot_points(points_after, "Точки після змін")

# Завдання 4: Збереження у текстовий файл
with open("points.txt", "w") as file:
    for i, (x, y) in enumerate(points_after, 1):
        file.write(f"{i}: {x}; {y}\n")
print("Координати точок збережено у файл points.txt")
>
```

# ДОДАТОК Б

## Скрін-шоти вікна виконання програми

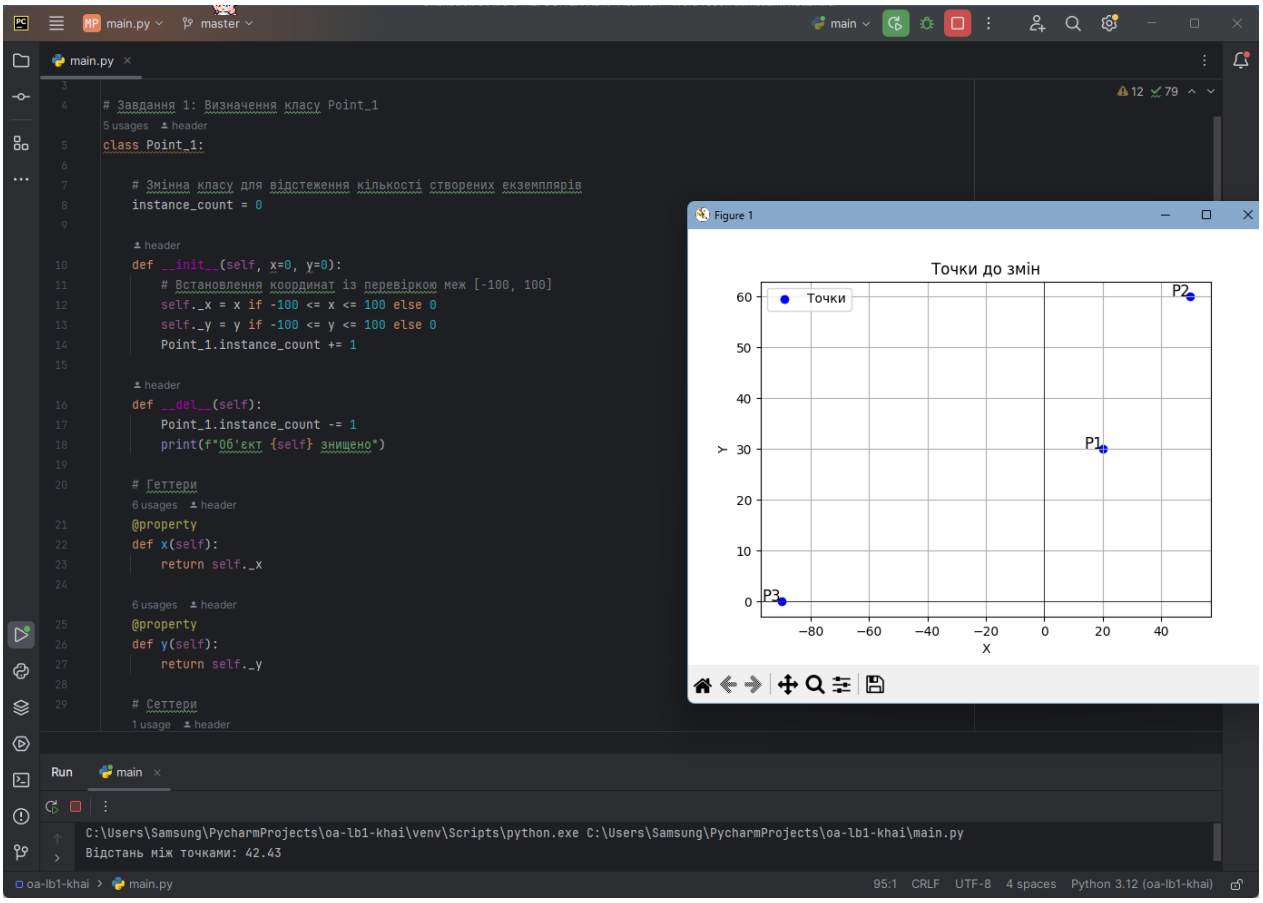



Рисунок Б.1 – Екран виконання програми для вирішення завдання Завдання 1: Визначення класу Point_1

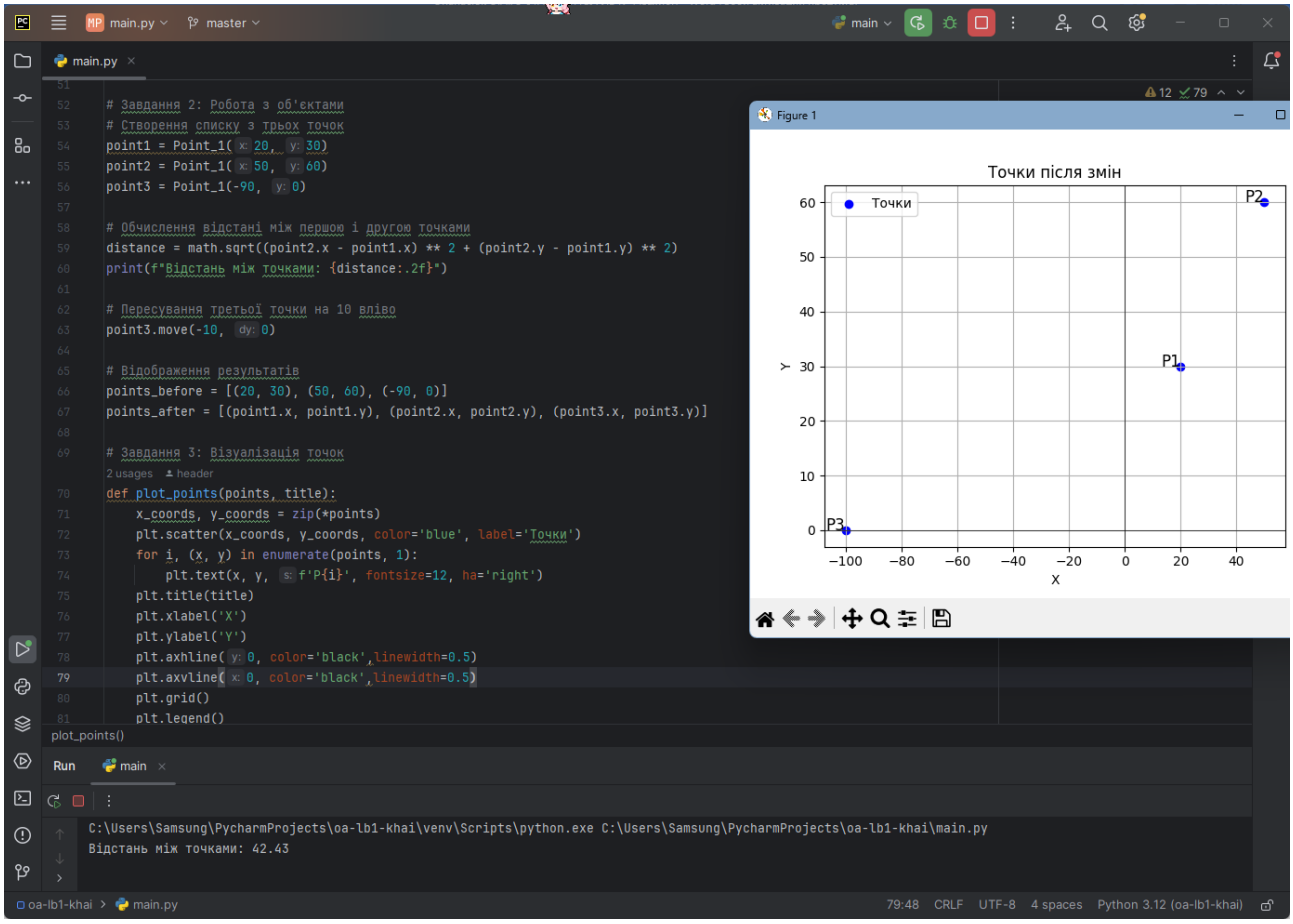



Рисунок Б.2 – Екран виконання програми для вирішення завдання Завдання 2: Робота з об'єктами

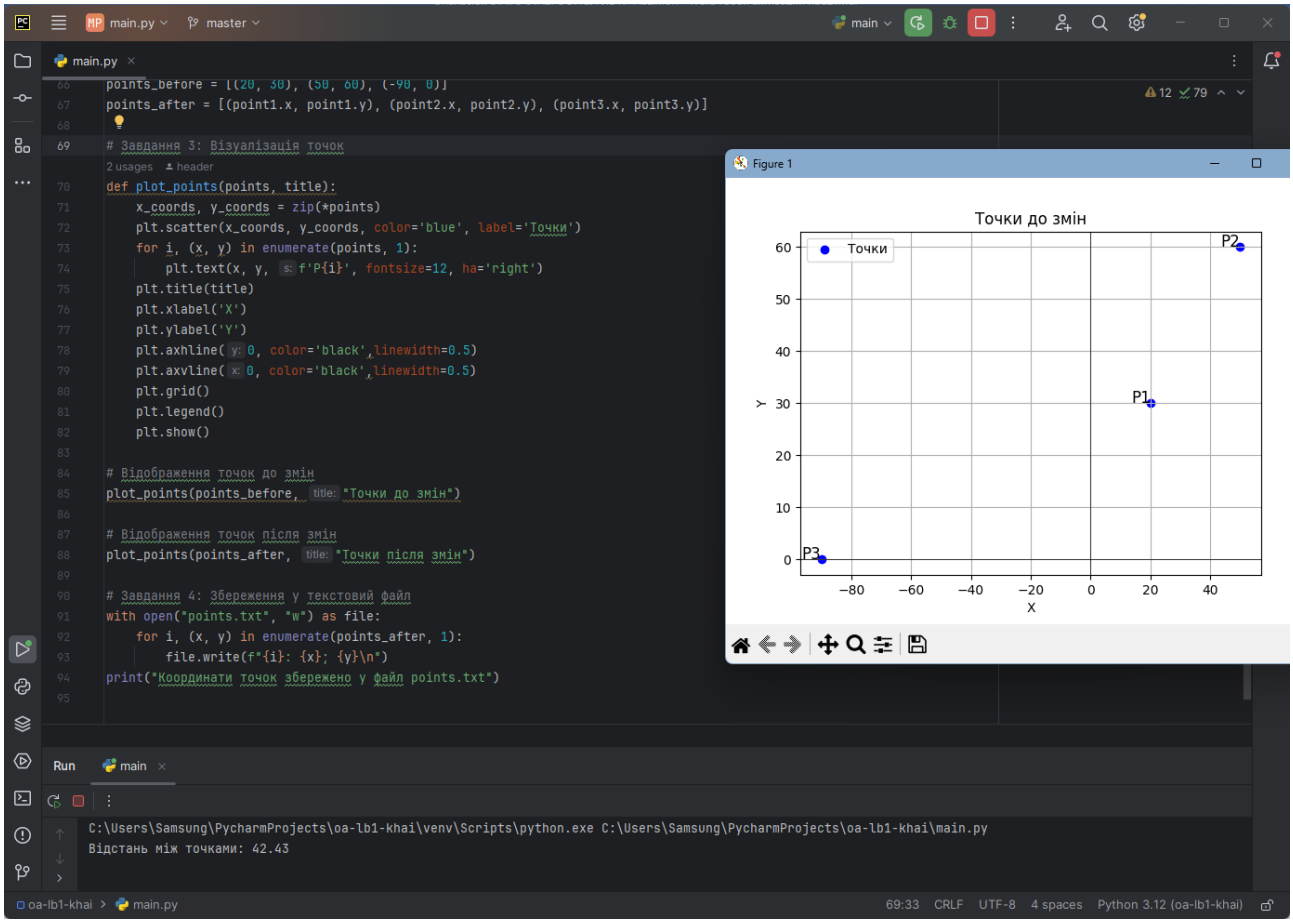



Рисунок Б.3 – Екран виконання програми для вирішення завдання Завдання 3: Візуалізація точок

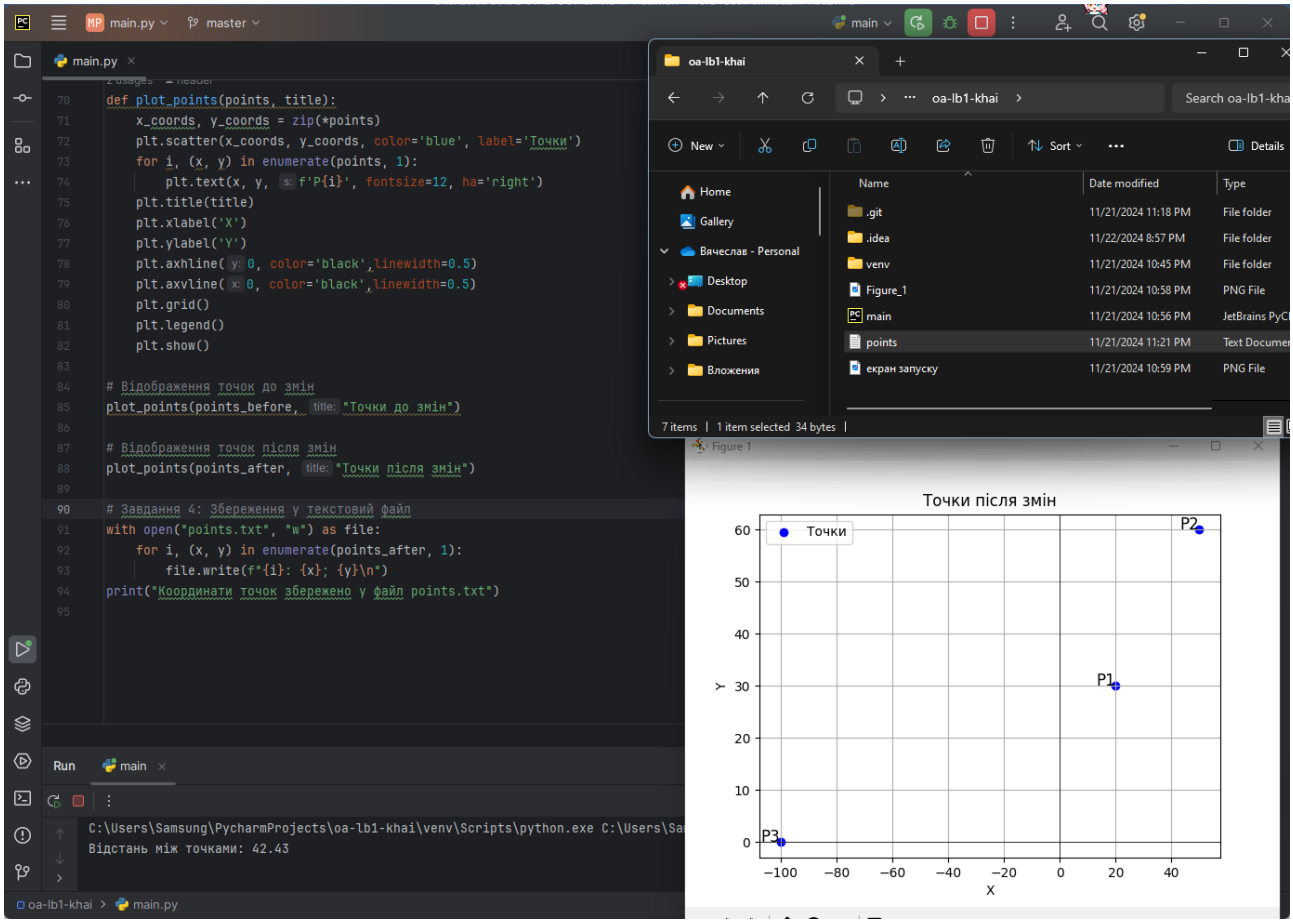



Рисунок Б.4 – Екран виконання програми для вирішення завдання Завдання 4: Збереження у текстовий файл