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’єктивно-орієнтоване проектування програм для мобільних систем»

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

ХАІ.301. 174. 322. 4 ЛР

Виконав студент гр. 322
_____Діхтяренко М.Г.
(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. Вирішення задачі 13:

Створити список з трьох точок, порахувати відстань між першою і другою, пересунути третю на 40 вгору.

Алгоритм вирішення завдання:

1. **Початок**.

2. **Ініціалізація точок**:

   - Створити три об’єкти точок із заданими координатами:
     - Точка 1: (10, 20) ,
     - Точка 2: (-30, 40) ,
     - Точка 3: (50, -70) .

3. **Обчислення відстані між точками**:

   - Обчислити відстань між точкою 1 і точкою 2 за формулою
   - Вивести результат відстані у форматі:

“Відстань між точкою 1 і точкою 2

4. **Переміщення точки**:

   - Перемістити точку 3 на 40 одиниць вгору (додати 40 до її у -координати).
   - Вивести оновлені координати точки 3 у форматі:

“Точка 3 після переміщення: (x, y) ”.

5. **Підготовка даних для візуалізації**:

   - Створити два списки точок:
     - **До змін**: список із початковими координатами точок.
     - **Після змін**: список із оновленими координатами точок.

6. **Візуалізація точок**:

   - Відобразити точки на двох графіках:
     - Ліва частина: точки до змін.

    - Права частина: точки після змін.
- Налаштувати координатну сітку, вісь та підписи точок.

7. **Збереження точок у файл**:
    - Створити файл points.txt.
    - Записати координати точок зі списку “Після змін”.

Лістинг коду вирішення задачі наведено в дод. А (стор. 6). Екран роботи програми показаний в дод.Б (стор. 9).

ВИСНОВКИ

В результаті виконання Лабораторної роботи на практиці було застосовано теоретичні знання з основ програмування на мові Python з використанням об'єктів і класів, закріплено навички використання бібліотеки для візуалізації масивів даних, і закріплено навички розробки скриптів для роботи з об'єктами призначених для користувача класів.

ДОДАТОК А

Лістинг коду програми до задач

main.py

```
from Point_13 import Point_13
import matplotlib.pyplot as plt

def save_points_to_file(points, filename="points.txt"):
    """
    Зберігає список точок у файл.
    :param points: Список точок
    :param filename: Ім'я файлу
    """
    with open(filename, "w") as f:
        for i, point in enumerate(points, start=1):
            f.write(f"{i}: {point.x}; {point.y}\n")


def task():
    """Реалізація задачі варіанту 13"""
    # Створення трьох точок
    point1 = Point_13(10, 20)
    point2 = Point_13(-30, 40)
    point3 = Point_13(50, -70)

    # Обчислення відстані між першою та другою точками
    distance = point1.distance_to(point2)
     print(f"Відстань між точкою 1 {point1} і точкою 2 {point2}:
{distance:.2f}")

    # Переміщення третьої точки на 40 одиниць вгору
    point3.move(0, 40)
    print(f"Точка 3 після переміщення: {point3}")

    # Візуалізація точок до і після змін
    points_before = [point1, point2, Point_13(50, -70)]
    points_after = [point1, point2, point3]

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

    # До змін
    plt.subplot(1, 2, 1)
    for point in points_before:
        plt.scatter(point.x, point.y, label=f"{point}")
```

```
    plt.title("До змін")
    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)
    plt.grid()
    plt.legend()

    # Після змін
    plt.subplot(1, 2, 2)
    for point in points_after:
        plt.scatter(point.x, point.y, label=f"{point}")
    plt.title("Після змін")
    plt.axhline(0, color='black', linewidth=0.5)
    plt.axvline(0, color='black', linewidth=0.5)
    plt.grid()
    plt.legend()

    plt.show()

    # Збереження точок у файл
    save_points_to_file(points_after)
    print("Координати точок збережено у файл 'points.txt'.")


if __name__ == "__main__":
    task()
```

Point_13.py

```
import math

class Point_13:
    """Клас для опису точки на площині"""
    # Змінна класу для підрахунку кількості створених екземплярів
    instances_count = 0

    def __init__(self, x=0, y=0):
        """
        Ініціалізує точку з координатами x та y.
        Якщо координати не у межах [-100, 100], встановлює їх у 0.
        """
        self._x = x if -100 <= x <= 100 else 0
        self._y = y if -100 <= y <= 100 else 0
        Point_13.instances_count += 1
```

```
    def __del__(self):
        """Деструктор, що повідомляє про видалення екземпляра"""
        Point_13.instances_count -= 1
        print(f"Точка ({self._x}, {self._y}) видалена.")

    @property
    def x(self):
        """Геттер для координати x"""
        return self._x

    @x.setter
    def x(self, value):
        """Сеттер для координати x"""
        self._x = value if -100 <= value <= 100 else 0

    @property
    def y(self):
        """Геттер для координати y"""
        return self._y

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

    @classmethod
    def get_instances_count(cls):
        """Повертає кількість створених екземплярів"""
        return cls.instances_count

    def move(self, dx, dy):
        """Зсуває координати точки на dx по x і dy по y"""
        self.x += dx
        self.y += dy

    def distance_to(self, other):
        """Обчислює відстань до іншої точки"""
            return math.sqrt((self.x - other.x)**2 + (self.y -
other.y)**2)

    def __str__(self):
        """Повертає рядкове представлення точки"""
        return f"({self.x}, {self.y})"
```

ДОДАТОК Б

Скрін-шот вікна виконання програми