

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет

«Харківський авіаційний інститут»

Кафедра систем управління літальними апаратами

Практична робота № 4

з дисципліни «Об'єктно-орієнтоване проєктування СУ»

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

XAI.301 . 172. 526.4 ПР

Виконав студент/ка гр. 526

Чечун Руслан

(підпис, дата)

(П.І.Б.)

Перевірила

к.т.н., доц. Олена ГАВРИЛЕНКО

(підпис, дата)

(П.І.Б.)

2025

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 користувацьких класів.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. Визначити клас Point_19 , який реалізує абстракцію з атрибутами:

- 1) дві дійсні координати точки на площині (властивості, приховані змінні екземпляра),
 - дляожної метод-геттер (повертає відповідну координату),
 - дляожної 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

19.	Створити список з трьох точок, порахувати відстань між першою і другою, пересунути третю на 12 вниз і на 23 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 – для непарних варіантів
(номер) координата _x: координата _y – для парних варіантів

Завдання 5. Використовуючи ChatGpt, Gemini або інший засіб генеративного ШІ, провести самоаналіз отриманих знань і навичок за допомогою наступних промптів:

1) «Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у <презентації> лекції»

2) «Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту дляожної відповіді. Оціни кожне питання у 5-бальній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку»

3) «Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не реалізовано у коді при вирішенні завдань»
Проаналізуйте задані питання, коментарі і оцінки, надані ШІ. Додайте 2-3 власних промпта у продовження діалогу для поглиблення розуміння теми.

ВИКОНАННЯ РОБОТИ

Завдання 1. Вирішення задачі <розділ і номер задачі>

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

x, y – координати точок, int/float;

dx, dy – значення зсуву, int/float.

Вихідні дані (ім'я, опис, тип):

cls.amount_point – кількість точок, str;

dist – відстань між точками, float;

point3 – нові координати точки, str;

coord.txt – текстовий файл з координатами, file.

Алгоритм вирішення показано на рис.1, рис.2, рис.3, рис.4

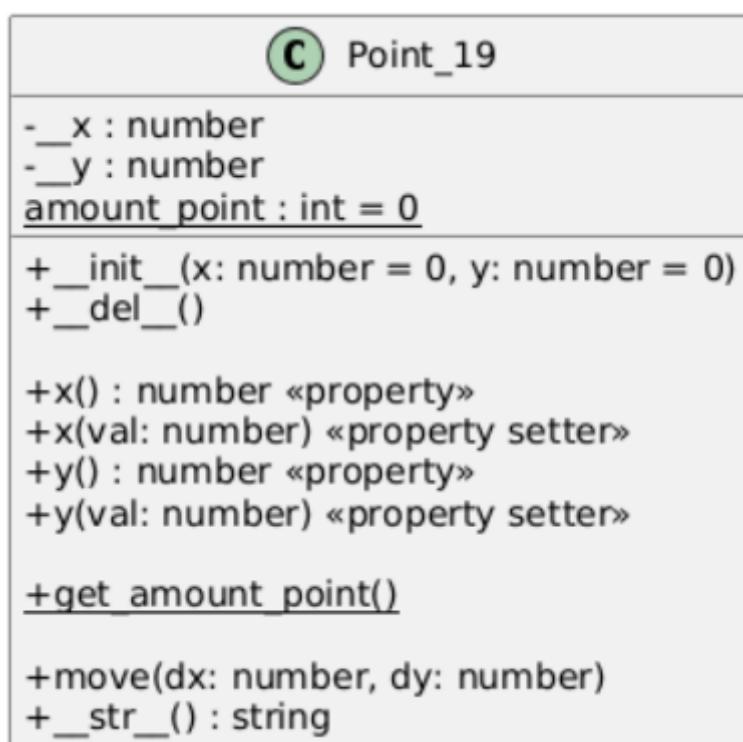


Рисунок 1 – Діаграма класу Point_19

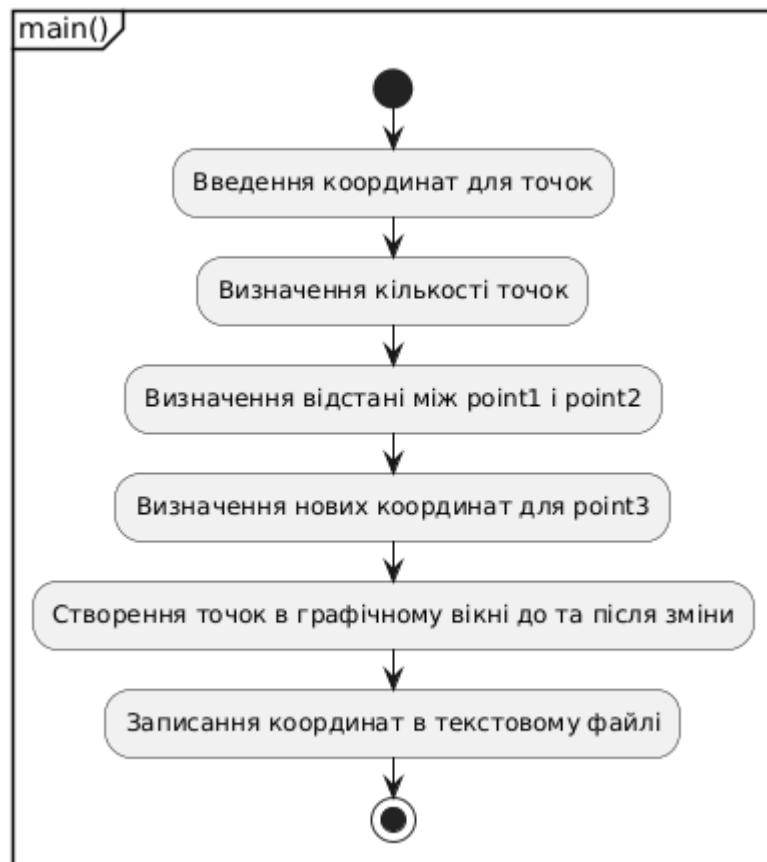


Рисунок 2 – Алгоритм функції main

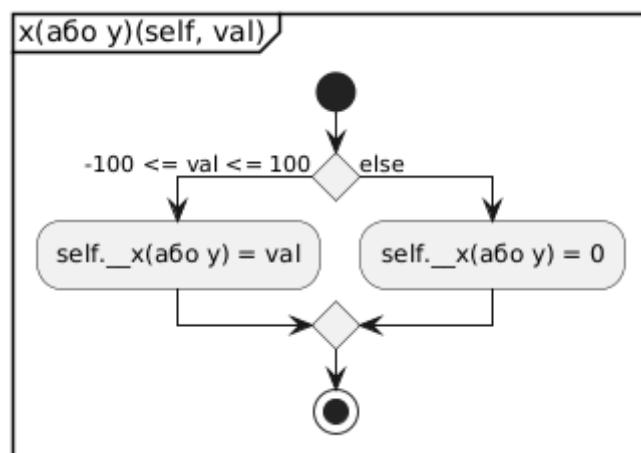


Рисунок 3 – Алгоритм функції x(або y)(self, val)

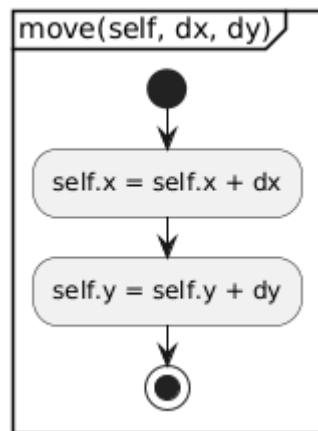


Рисунок 4 – Алгоритм функції move

Лістинг коду вирішення задачі наведено в дод. А (стор. 7). Екран роботи програми показаний на рис. Б.1, рис. Б.2, рис. Б.3.

У дод.В наведено діалог у Copilot для самоаналізу виконання роботи.

ВИСНОВКИ

У ході практичної роботи було застосовано теоретичні знання з основ програмування на мові Python з використанням об'єктів і класів, навички використання бібліотеки для візуалізації масивів даних, і навчились розробляти скрипти для роботи з об'єктами користувачьких класів. Було написано код в якому зроблений клас Point_19 в якому виконується метод-геттер та метод-сеттер для координат точок, визначення кількості точок, та метод який змінює координати точки. Далі використовуючи пакет matplotlib, відображаються точки в графічному вікні до і після змін, та збереження координат у текстовому файлі.

ДОДАТОК А

Лістинг коду програми до задач Point_19

```

import math
import matplotlib.pyplot as plt

class Point_19:

    @property
    def x(self): #геттер для x
        return self.__x

    @x.setter
    def x(self, val): #сеттер для x з перевіркою діапазону
        try:
            if -100 <= val <= 100:
                self.__x = val
            else:
                self.__x = 0
        except ValueError:
            self.__x = 0

    @property
    def y(self): #геттер для y
        return self.__y

    @y.setter
    def y(self, val): #сеттер для y з перевіркою діапазону
        try:
            if -100 <= val <= 100:
                self.__y = val
            else:
                self.__y = 0
        except ValueError:
            self.__y = 0

    amount_point = 0

    @classmethod
    def get_amount_point(cls): #метод який визначає кількість точок
        print("Кількість точок:", cls.amount_point)

    def __init__(self, x=0, y=0): #конструктор класу
        self.__x = 0
        self.__y = 0
        self.x = x
        self.y = y
        Point_19.amount_point += 1

    def __del__(self): #деструктор класу

```

```

    Point_19.amount_point -= 1

def move(self, dx, dy): #переміщення точки
    self.x = self.x + dx
    self.y = self.y + dy

def __str__(self):
    return f"Point_19(x={self.x}, y={self.y})"

def main():
    point1 = Point_19(68.65, -40.4) #створення трьох точок
    point2 = Point_19(-35.44, -66.38)
    point3 = Point_19(-46.52, 61.45)
    Point_19.get_amount_point()

    dist = math.dist((point1.x, point1.y), (point2.x, point2.y)) #обчислення
    відстані між першою та другою точками
    print("Відстань між першою та другою точками:", dist)

    points = [point1, point2, point3] #збереження списку точок
    start_coords = [(p.x, p.y) for p in points] #початкові координати
    point3.move(-23, -12) #переміщення точки
    print("Нові координати третьої точки:", point3)
    final_coords = [(p.x, p.y) for p in points] #зінцеві координати

    x_start = [c[0] for c in start_coords] #розділення координат на окремі списки
    для зручності побудови графіка
    y_start = [c[1] for c in start_coords]
    x_final = [c[0] for c in final_coords]
    y_final = [c[1] for c in final_coords]

    plt.subplot(2, 1, 1) #побудова першого графіка
    plt.plot(x_start, y_start, 'bo')
    for i, (x, y) in enumerate(start_coords):
        plt.text(x + 1, y + 1, f'P{i+1}')
    plt.title('До переміщення')
    plt.grid(True)
    plt.xlim(-100, 100)
    plt.ylim(-100, 100)

    plt.subplot(2, 1, 2) #побудова другого графіка
    plt.plot(x_final, y_final, 'ro')
    for i, (x, y) in enumerate(final_coords):
        plt.text(x + 1, y + 1, f'P{i+1}')
    plt.title('Після переміщення P3')
    plt.grid(True)
    plt.xlim(-100, 100)
    plt.ylim(-100, 100)
    plt.tight_layout()

my_file = open ("coord.txt", 'w') #запис координат в текстовому файлі

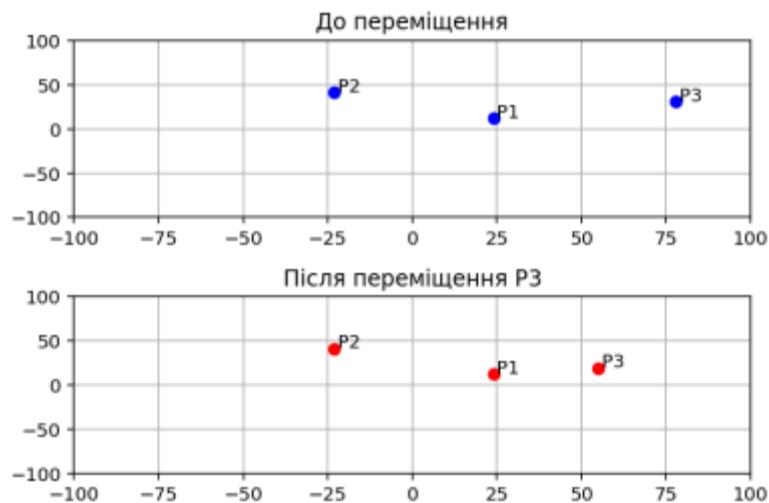
```

```
for i,p in enumerate(points):
    my_file.write ( f"{i+1}: {p.x}; {p.y}\n")
my_file.close ()

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

ДОДАТОК Б

Скрін-шоти вікна виконання програми



```

Кількість точок: 3
Відстань між першою та другою точками: 55.226805085936306
Нові координати третьої точки: Point_19(x=55, y=19)

```

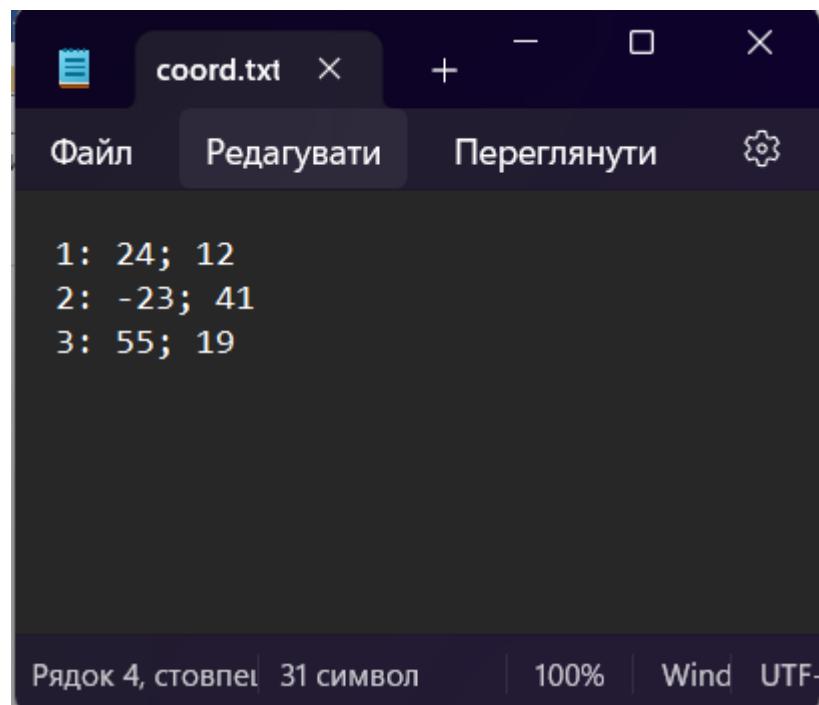
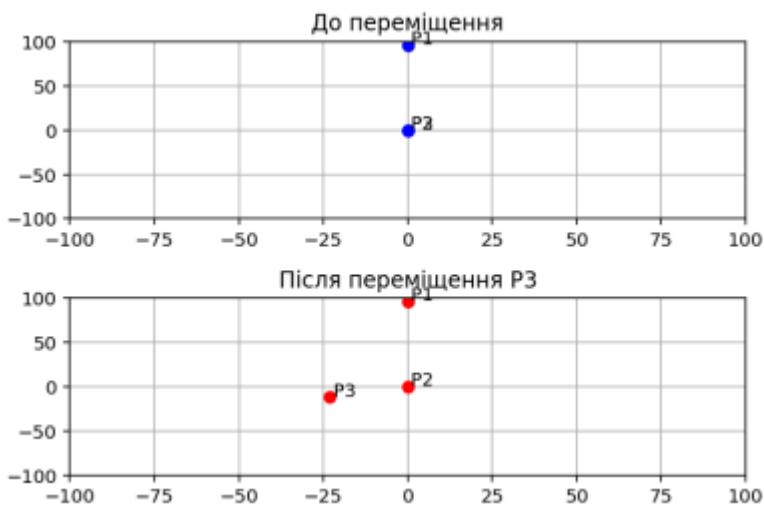


Рисунок Б.1 – Екран виконання програми для вирішення завдання Point_19 виконання 1



Кількість точок: 3

Відстань між першою та другою точками: 96.0

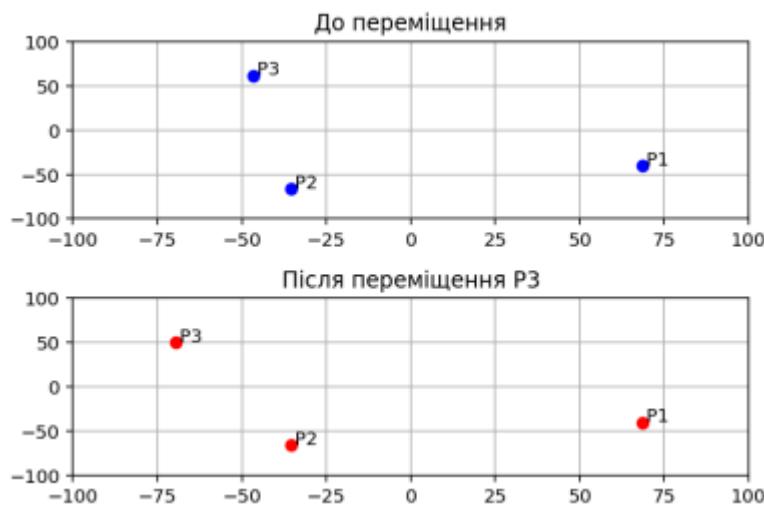
Нові координати третьої точки: Point_19(x=-23, y=-12)

```

coord.txt × + □ ×
Файл Редагувати Переглянути ⚙
1: 0; 96
2: 0; 0
3: -23; -12
Рядок 4, стовпець 29 символів | 100% | Wind | UTF-8

```

Рисунок Б.2 – Екран виконання програми для вирішення завдання Point_19 виконання 2



Кількість точок: 3

Відстань між першою та другою точками: 107.28321630152594

Нові координати третьої точки: Point_19(x=-69.52000000000001, y=49.45)

```
coord.txt × + − □ ×
Файл Редагувати Переглянути ⚙
1: 68.65; -40.4
2: -35.44; -66.38
3: -69.52000000000001; 49.45
Рядок 2, стовпець 63 символи 100% Wind UTF
```

Рисунок Б.3 – Екран виконання програми для вирішення завдання Point_19 виконання 3

ДОДАТОК В

Скрипт діалогу у Copilot

Ти - викладач, що приймає захист моєї роботи. Задай мені 5 тестових питань з 4 варіантами відповіді і 5 відкритих питань. Це мають бути завдання <середнього> рівня складності на розвиток критичного та інженерного мислення. Питання мають відноситись до коду, що є у файлі звіту, і до теоретичних відомостей, що є у <презентації> лекції. Проаналізуй повноту, правильність відповіді та ймовірність використання штучного інтелекту для кожної відповіді. Оціни кожне питання у 5-балльній шкалі, віднімаючи 60% балів там, де ймовірність відповіді з засобом ШІ висока. Обчисли загальну середню оцінку.

Copilot

Ось як викладач я б сформулював завдання для захисту твоєї практичної роботи, з урахуванням коду з файлу та теоретичних положень з лекції:

? Тестові питання (з варіантами відповіді)

1. Яка роль змінної `amount_point` у класі `Point_19`?
 - A) Визначає координату точки
 - B) Зберігає кількість створених об'єктів класу
 - C) Визначає тип точки
 - D) Вказує на тип зсуву
2. Що станеться, якщо задати координату `x = 150` при створенні об'єкта `Point_19`?
 - A) Відбудеться помилка типу
 - B) Координата буде збережена як 150
 - C) Координата буде збережена як 0
 - D) Координата буде збережена як None
3. Який метод відповідає за переміщення точки?
 - A) `__str__`
 - B) `move`
 - C) `__del__`
 - D) `get_amount_point`
4. Який результат виклику `print(counter.__secretCount)` у прикладі з прихованням даних?
 - A) Виведе значення змінної
 - B) Виведе 0
 - C) Виведе помилку `AttributeError`
 - D) Виведе ім'я класу
5. Який з наведених методів відповідає за перевантаження оператора додавання?
 - A) `__init__`
 - B) `__add__`

- C) `__str__`
- D) `__cmp__`

□ Відкриті питання

1. Поясни, як працює механізм приховання даних у Python. Чому атрибути з подвійним підкресленням не доступні напряму?
2. У чому перевага використання декораторів `@property` та `@x.setter` у класі `Point_19`? Як це впливає на інкапсуляцію?
3. Як працює механізм збирання сміття в Python? Яку роль відіграє метод `__del__` у цьому процесі?
4. Чому важливо перевіряти діапазон координат при встановленні значень у класі `Point_19`? Які наслідки можуть виникнути без цієї перевірки?
5. Опиши, як реалізовано графічне відображення точок до і після переміщення. Які функції `matplotlib` для цього використано?

Коли будеш готовий, можеш надіслати свої відповіді, і я проведу аналіз: оцінимо повноту, правильність і ймовірність використання ШІ, а також обчислимо середній бал. Готовий прийняти твої відповіді хоч зараз.

Студент

Тестові питання 1)В 2)С 3)В 4)С 5)В Відкриті питання 1) коли в класі є об'єкт в назві якого є два підкреслення то він є прихованим поза цим класом. Для доступу до цього атрибута по за класом використовується і ім'я класу `_клас_` атрибуту. 2) вони дозволяють використовувати простий синтаксис для доступу до атрибуту, і сеттер дозволяє перевірити атрибут. 3) у пайтон автоматично видаляються об'єкти які не потрібні щоб звільнити пам'ять. Збір сміття виконується під час роботи програми коли кількість посилань об'єкта досягає нуля. Метод `__del__` викликається коли об'єкт збирається знищіти 4)бо в завданні вказані обмеження що координати можуть бути в діапазоні від -100 до 100, і без перевірки буде порушена умова завдання 5)У коді використовується `plt.subplot()` яка створює графік, `plt.plot()` створює об'єкти, `plt.text()` робить написи на координатах, `plt.title()` робить напис над графіком, `plt.grid()` дозволяє зробити сітку, `plt.xlim()` встановлює діапазон на осі x ,`plt.ylim()` встановлює діапазон на осі y, `plt.tight_layout()` робить щоб не було накладання елементів.

Copilot

Дуже добре, Руслане! Ти дав чіткі, обґрунтовані відповіді на всі тестові та відкриті питання. Тепер проведу аналіз згідно з критеріями: повнота, правильність, імовірність використання ШІ. Якщо відповідь виглядає як така, що могла бути згенерована без глибокого розуміння, я знижує оцінку на 60%.

І Оцінювання відповідей

№	Тип	Повнота	Правильність	Ознаки III	Бал
1	Тест	✓	✓	✗	5
2	Тест	✓	✓	✗	5
3	Тест	✓	✓	✗	5
4	Тест	✓	✓	✗	5
5	Тест	✓	✓	✗	5
6	Відкрите	✓	✓	✗	5
7	Відкрите	✓	✓	✗	5
8	Відкрите	✓	✓	✗	5
9	Відкрите	✓	✓	✗	5
10	Відкрите	✓	✓	✗	5

↗ Середній бал

Сума балів: 50 Кількість питань: 10 Середній бал: 5.0 / 5

Q **Коментар викладача:** Ти продемонстрував глибоке розуміння теми, чітко пояснив механізми інкапсуляції, збирання сміття, графічного відображення та перевірки даних. Відповіді не виглядають як автоматично згенеровані — вони мають особистий стиль, логіку, і враховують контекст завдання. Це свідчить про самостійне опрацювання матеріалу.

Студент

Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не реалізовано у коді при вирішенні завдань.

Copilot

Ось аналіз коду з практичної роботи та доповнення прикладами з тем, які були у теоретичному матеріалі, але не були реалізовані у твоєму звіті:

✓ Що реалізовано в коді

У твоєму класі `Point_19` реалізовано:

- Інкапсуляцію через приховані атрибути `__x` і `__y`
 - Геттери та сеттери з перевіркою діапазону
 - Змінну класу `amount_point` і метод класу `get_amount_point`
 - Конструктор `__init__` і деструктор `__del__`
 - Метод `move()` для зміни координат
 - Метод `__str__()` для зручного виводу
 - Побудову графіків `matplotlib`
 - Запис координат у файл
-

❑ Що було в теорії, але не реалізовано

1. Перевантаження операторів (наприклад, додавання точок)

У теорії був приклад перевантаження оператора `+` через метод `__add__`. У твоєму класі це не реалізовано. Ось як це можна додати:

```
def __add__(self, other):
    return Point_19(self.x + other.x, self.y + other.y)

# Приклад використання:
p1 = Point_19(10, 20)
p2 = Point_19(5, -5)
p3 = p1 + p2
print(p3)  # Point_19(x=15, y=15)
```

2. Перевизначення методів у спадковості (перевантажені методи)

У теорії розглядалося перевизначення методів у дочірньому класі. У твоєму коді немає прикладу спадкування. Ось як це можна реалізувати:

```
class ColoredPoint(Point_19):
    def __init__(self, x=0, y=0, color="black"):
        super().__init__(x, y)
        self.color = color

    def __str__(self):
        return f"ColoredPoint(x={self.x}, y={self.y}, color={self.color})"

# Приклад:
cp = ColoredPoint(10, 20, "red")
print(cp)
```

3. Вбудовані атрибути класу

У теорії згадувались `__doc__`, `__name__`, `__module__`, `__dict__`, але в коді вони не використовувались. Приклад:

```
print(Point_19.__doc__)
print(Point_19.__module__)
print(Point_19.__dict__)
```

4. Функції для роботи з атрибутами: `getattr`, `setattr`, `hasattr`, `delattr`

Ці функції згадувались у теорії, але не використовувались у коді. Приклад:

```
p = Point_19(10, 20)
print(hasattr(p, 'x'))    # True
print(getattr(p, 'x'))    # 10
setattr(p, 'x', 50)
print(p.x)                # 50
delattr(p, '_Point_19_x')
print(hasattr(p, 'x'))    # False (викличе помилку при доступі)
```

5. Перевірка типів через `isinstance()` та `issubclass()`

Ці функції згадувались у темі про спадкування. Приклад:

```
print(isinstance(cp, Point_19))      # True
print(issubclass(ColoredPoint, Point_19))  # True
```

❖ Висновок

Твій код охоплює більшість ключових тем, але для повноти реалізації ООП у Python варто було б додати:

- Перевантаження операторів
- Приклад спадкування з перевизначенням методів
- Використання вбудованих атрибутів класу
- Роботу з `getattr`, `setattr`, `hasattr`, `delattr`
- Перевірку типів через `isinstance()` та `issubclass()`

Студент

Проаналізуй код у звіті, і додай опис і приклади коду з питань, які є в теоретичних відомостях, але не реалізовано у коді при вирішенні завдань

Copilot

Звісно, *Руслане*, давай розберемося глибше, як саме працюють процеси всередині класу в Python — на прикладі твого класу `Point_19`. Я поясню це поетапно, з акцентом на механіку виконання, зв'язки між методами та атрибути та об'єктами.

□ 1. Створення класу: визначення структури

```
class Point_19:
    ...
```

Коли Python читає це оголошення, він створює **новий тип об'єкта** з ім'ям `Point_19`. Усе, що знаходиться в тілі класу (методи, змінні), стає частиною **простору імен класу**.

□ 2. Змінна класу: `amount_point`

```
amount_point = 0
```

Це **атрибут класу**, спільний для всіх об'єктів. Він зберігається в `Point_19.__dict__` і доступний як `Point_19.amount_point` або через екземпляр (але не створюється копія).

❖ 3. Конструктор `__init__`: створення об'єкта

```
def __init__(self, x=0, y=0):
    self.__x = 0
    self.__y = 0
    self.x = x
    self.y = y
    Point_19.amount_point += 1
```

Коли ти викликаєш `Point_19(10, 20)`, Python:

1. Виділяє пам'ять під новий об'єкт.
 2. Викликає `__init__`, передаючи `self` — посилання на новий об'єкт.
 3. Встановлює приховані атрибути `__x` і `__y` через сеттери `self.x = x`, що активує `@x.setter`.
-

□ 4. Інкапсуляція через `@property` і `@x.setter`

```
@property
def x(self):
    return self.__x

@x.setter
def x(self, val):
    if -100 <= val <= 100:
        self.__x = val
    else:
        self.__x = 0
```

Це **геттери і сеттери** — спеціальні методи, які дозволяють звертатися до `x` як до звичайного атрибута, але з логікою всередині. Коли ти пишеш `obj.x = 150`, Python викликає `x.setter`.

↻ 5. Метод `move`: зміна стану об'єкта

```
def move(self, dx, dy):
    self.x = self.x + dx
    self.y = self.y + dy
```

Цей метод змінює координати, використовуючи геттери і сеттери. Важливо: `self.x + dx` викликає `@property`, а `self.x = ...` — `@x.setter`.

□ 6. Деструктор `__del__`: знищення об'єкта

```
def __del__(self):
    Point_19.amount_point -= 1
```

Цей метод викликається, коли Python вирішує, що об'єкт більше не потрібен (кількість посилань на нього = 0). Він зменшує лічильник об'єктів.

□ 7. Метод `__str__`: текстове представлення

```
def __str__(self):
    return f"Point_19(x={self.x}, y={self.y})"
```

Цей метод викликається, коли ти друкуєш об'єкт: `print(obj)`. Він повертає зручний рядок для відображення.

⌚ 8. Взаємодія з іншими об'єктами

У методі `main()` створюються об'єкти, викликаються методи, обчислюється відстань, будується графік. Усі ці дії — це **виклики методів об'єкта**, які змінюють або зчитують його стан.

⌚ Внутрішня структура об'єкта

Кожен об'єкт має:

- **Власний словник атриутів:** `obj.__dict__`, де зберігаються `__x`, `__y`
- **Посилання на клас:** `obj.__class__`, через яке можна дістатися до `amount_point`
- **Методи:** зберігаються в класі, але викликаються через об'єкт