Assignment 8 - Classes

1. *(class Point)*
Trong bài tập này ta sẽ xây dựng class `Point` để biểu diễn lớp các điểm trong mặt phẳng Oxy. Đoạn code (1) thể hiện một cách xây dựng class `Point` .

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
```

Listing 1: Xây dựng class `Point`

Với class `Point` như vậy, ta có thể khởi tạo hai instance của class `Point` như sau:

```
pointA = Point(0, 0)
pointB = Point(3, 4)
```

(a) Viết method `distance` thuộc class `Point` để tính khoảng cách Euclid (chuẩn $L2$) giữa hai instance của class `Point` . Method này cần tương thích với đoạn code (2).

```
dist = pointA.distance(pointB) # dist = 5
```

Listing 2: Cách sử dụng method `distance`

(b) Bổ sung tham số `metric` trong method `distance` của class `Point` để tính khoảng cách theo các metric khác nhau. Thay đổi cần tương thích với đoạn code (3).

```
dist = pointA.distance(pointB, metric = 'L1') # dist = 7
```

Listing 3: Bổ sung tham số `metric`

Trình bày class `Point` và **ví dụ thể hiện cách sử dụng** class `Point` trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_classPoint.ipynb`

2. *(class Line)*
Trong bài tập này ta sẽ xây dựng class `Line` để biểu diễn lớp các đường thẳng trong mặt phẳng Oxy bằng cách lần lượt trả lời câu hỏi / thực hiện yêu cầu sau đây.

(a) Vì sao phương trình đường thẳng có dạng $y = ax + b$ với a, b là hai tham số bất kỳ **không phải là** phương trình tổng quát của một đường thẳng trong mặt phẳng Oxy?

(b) Xét phương trình của đường thẳng $\mathcal{D}$ có dạng

$$ax + by + c = 0 \tag{1}$$

với a, b, c là ba tham số bất kỳ. Tìm điều kiện của ba tham số a, b, c để phương trình (1) là phương trình tổng quát của một đường thẳng trong mặt phẳng Oxy.

(c) Phần khởi tạo của class `Line` có thể được viết như sau

Copyright © by Optima at HUS

```python
class Line:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
```

Listing 4: Xây dựng class `Line`

Nhận xét về cách đặt tên trong đoạn code (4). Cải thiện đoạn code này nếu cần thiết.

(d) Viết một method trong class `Line` để xác định giao điểm của hai đường thẳng (nếu có), chú ý rằng giao điểm của hai đường thẳng nên là một instance của class `Point` .

(e) Viết một method trong class `Line` để xác định khoảng cách giữa một điểm và một đường thẳng.

(f) Biết rằng có duy nhất một đường thẳng đi qua hai điểm phân biệt cho trước. Viết một method trong class `Line` để khởi tạo một đường thẳng (instance của class `Line`) với tham số đầu vào là hai điểm (instance của class `Point`).

Gợi ý: sử dụng `@classmethod`

Trình bày câu trả lời, class `Line` , và **ví dụ thể hiện cách sử dụng** class `Line` trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_classLine.ipynb`

3. *(class Circle)*
Trong bài tập này ta sẽ xây dựng class `Circle` để biểu diễn lớp các đường tròn trong mặt phẳng *Oxy*. Cấu trúc của class `Circle` cần đáp ứng các tiêu chí sau đây:

(a) Các yếu tố để xác định (duy nhất) một đường tròn.

(b) Các thông số của một đường tròn (ví dụ như chu vi của đường tròn).

(c) Các method thể hiện mối quan hệ giữa hai instance của class `Circle` .

(d) Các method thể hiện mối quan hệ của class `Circle` với các class khác (ví dụ như một hàm kiểm tra tính tiếp xúc giữa một đường thẳng và một đường tròn).

Trình bày class `Circle` và **ví dụ thể hiện cách sử dụng** class `Circle` trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_classCircle.ipynb`

4. *(load objects from a Jupyter notebook)*
Nêu cách import một class/function/variable từ một file *Jupyter notebook* sang một file *Jupyter notebook* khác. Trình bày cách thực hiện bằng **hai file *Jupyter notebook*** với tên được đặt tên theo mẫu

`Assignment08_FullName_SourceNotebook.ipynb`

`Assignment08_FullName_DestinationNotebook.ipynb`

Gợi ý: sử dụng `ipynb package`.

Copyright © by Optima at HUS

Additional Problems

5. *(class Ellipse)*
Xây dựng class `Ellipse` sao cho class `Circle` có thể kế thừa từ class `Ellipse` . Trình bày class `Ellipse` và **ví dụ thể hiện cách sử dụng** class `Ellipse` trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_classEllipse.ipynb`

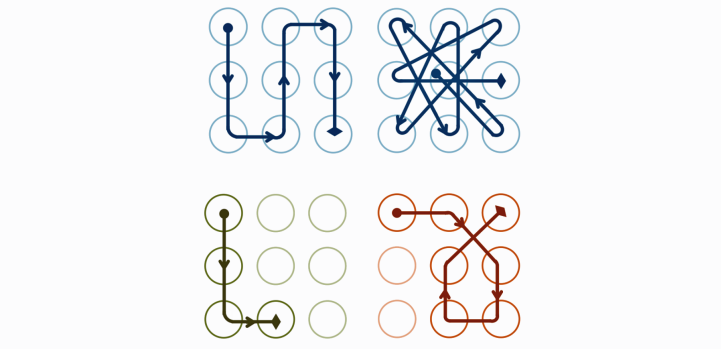
6. *(class Polynomial)*
Xây dựng class `Polynomial` sao cho có thể thực hiện các phép toán cộng $(+)$, trừ $(-)$, nhân $(*)$ giữa hai instance của class `Polynomial` .
Trình bày class `Polynomial` và **ví dụ thể hiện cách sử dụng** class `Polynomial` trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_classPolynomial.ipynb`

7. *(pretty representation)*
Bổ sung __repr__ method cho các Class được xây dựng trong những bài tập trên.

8. *(plotting method)*
Bổ sung hàm vẽ những đối tượng của các Class được xây dựng trong những bài tập trên.

9. *(cardinality of 4-to-9 points passwords)*
Hãy tính số mật khẩu đường gấp khúc khác nhau được tạo ra trên một lưới 9 điểm được xếp thành một hình vuông kích thước 3x3.



Hình 1: Một số mật khẩu đường gấp khúc

10. *(Three hands of a clock)*
Cho một đồng hồ kim với kim giờ, kim phút, kim giây có độ dài bằng nhau và bằng 1, được gắn với nhau tại tâm của đồng hồ. Gọi A, B, C là ba đầu mút còn lại của ba kim đồng hồ đó. Đặt $\mathcal{M} = AB + BC + CA$, ta đã biết một số giá trị của $\mathcal{M}$ tại một số thời điểm, ví dụ như:

(i) $\mathcal{M} = 0 + 0 + 0 = 0$ lúc 0h0'0".

(ii) $\mathcal{M} = 2 + 2 + 0 = 4$ lúc 6h0'0".

Hỏi trong thời gian hoạt động của đồng hồ từ 0h0'10" đến 11h59'50" cùng ngày, giá trị lớn nhất và giá trị bé nhất $\mathcal{M}$ có thể đạt được là bao nhiêu? $\mathcal{M}$ đạt những giá trị đó vào những thời điểm nào?

Gợi ý: $0 \leq \mathcal{M} \leq 3\sqrt{3}$

Copyright © by Optima at HUS

Journey to the Rectangle class

Trong một buổi seminar của nhóm Python Programming, bạn *A* được giao nhiệm vụ xây dựng một class biểu diễn lớp các hình chữ nhật (không xét đến vị trí của chúng trong mặt phẳng). *A* đề xuất hai cách tổ chức class `Rectangle` được trình bày ở (5) và (7). Ta sẽ đánh giá hai cách tiếp cận này.

Cách tiếp cận thứ nhất

```python
class Rectangle:
    def __init__(self, height, width):
        self.height = height
        self.width = width

    def calculate_area(self):
        return self.height * self.width
```

Listing 5: Cách tiếp cận thứ nhất

Khi nhắc đến một class biểu diễn lớp các hình chữ nhật, người dùng *thường* nghĩ rằng `length, width, area` là các thuộc tính (attribute) của một instance và sử dụng class `Rectangle` như trong đoạn code (6). Nói cách khác, việc sử dụng hàm `calculate_area` để có được thông tin về diện tích là *phản trực giác*.

```python
rect = Rectangle(10, 5)
print(f'Height of the rectangle: {rect.height}') # 10
print(f'Width of the rectangle : {rect.width}') # 5
print(f'Area of the rectangle  : {rect.calculate_area()}') # counter-intuitive
print(f'Area of the rectangle  : {rect.area}') # Attribute error
```

Listing 6: Cách người dùng sử dụng class `Rectangle`

Hãy thiết kế class `Rectangle` thỏa mãn hai yêu cầu:

(a) Diện tích hình chữ nhật được tính ngay tại thời điểm nó được nhắc đến (hàm `calculate_area` thỏa mãn yêu cầu này).

(b) Người dùng truy cập thông tin về diện tích thông qua thuộc tính `area` (hàm `calculate_area` không thỏa mãn yêu cầu này).

Gợi ý: sử dụng `@property`

Cách tiếp cận thứ hai

```python
class Rectangle:
    def __init__(self, height, width):
        self.height = height
        self.width = width
        self.area = height * width
```

Listing 7: Cách tiếp cận thứ hai

Cách tiếp cận ở (7) đã giải quyết được vấn đề về tính trực giác của chương trình khi tính (và lưu lại) diện tích hình chữ nhật ngay khi khởi tạo instance.
Tuy nhiên với cách thiết kế class `Rectangle` như vậy, sai sót sẽ xảy ra khi người dùng *vô tình* thay đổi các attribute cơ sở (trong trường hợp này là `height, width`). Đoạn code (8) thể hiện một ví dụ về lỗi này.

Copyright © by Optima at HUS

```python
rect = Rectangle(10, 5)
rect.height = 20
print(f'Height of the rectangle: {rect.height}') # 20
print(f'Width of the rectangle : {rect.width}') # 5
print(f'Area of the rectangle  : {rect.area}') # 50, it is supposed to be 100
```

Listing 8: Cách người dùng sử dụng class `Rectangle`

Hãy thiết kế class `Rectangle` thỏa mãn hai yêu cầu:

(a) Người dùng truy cập thông tin về diện tích thông qua thuộc tính `area` (cách tiếp cận trong (7) thỏa mãn yêu cầu này).

(b) Không cho phép (theo một nghĩa nào đó) người dùng thay đổi thuộc tính `height, width` sau khi đã khởi tạo instance của class `Rectangle` .

Gợi ý: sử dụng @property

Kết luận
Hãy tổng kết lại những ưu điểm, nhược điểm của hai cách xây dựng class `Rectangle` đã nêu ở trên. Tìm nhược điểm chung (nếu có) của cả hai cách xây dựng này và đưa ra cách cải thiện phù hợp. Trình bày câu trả lời trong một file *Jupyter notebook* với tên được đặt theo mẫu

`Assignment08_FullName_JourneyToTheRectangleClass.ipynb`

Related topics

- `@classmethod`, `@staticmethod`, and `@property` decorators.
- Differences between class attributes and instance attributes.
- Dunder methods (a.k.a methods with trailing double underscores) for "private" use.
- `__getattribute__, __getattr__, getattr, setattr` methods.
- Class Inheritance.
- Geometrical shapes in Sympy package.
- Having access to `class` through `self.__class__` variable.

Copyright © by Optima at HUS