Programming Fundamentals I Lab.

9. Classes and Objects

ชื่อ_____ รหัสนิสิต_____

ในปฏิบัติการนี้ คุณจะได้รู้จักแนวคิดพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ ได้แก่คลาส (class) และวัตถุ (object)

9.1 คลาสและวัตถุ

สร้างไฟล์ `.py` และพิมพ์โค้ดดังนี้

```
class Pet:
    def say_hi(self):
        print('Hi there.')
```

อธิบาย โค้ดดังกล่าวทำหน้าที่สร้างชนิดข้อมูลขึ้นใหม่โดยมีชื่อว่า `Pet` โดยมีการประกาศฟังก์ชันไว้ภายในชนิดข้อมูลนี้หนึ่งฟังก์ชันชื่อว่า `say_hi()` เราเรียกชนิดข้อมูลที่ประกาศขึ้นด้วยคำสั่ง `class` ว่าเป็นคลาส และเรียกฟังก์ชันที่ประกาศภายในคลาสว่า เมธอด (method)

ให้คุณทดลองรันโปรแกรม จะไม่มีสิ่งใดปรากฏขึ้น เนื่องจากเราแค่ประกาศชนิดข้อมูลใหม่ขึ้นเท่านั้น ไม่ได้สั่งให้คอมพิวเตอร์ทำอะไรต่อ

หลังจากรันโปรแกรม ลองพิมพ์คำสั่งต่อไปนี้ในหน้า Shell และบันทึกผลลัพธ์

```
>>> p = Pet()
>>> type(p)
```

อธิบาย คำสั่ง `p = Pet()` เป็นการสร้างตัวแปรชื่อ `p` ให้มีชนิดข้อมูลเป็นคลาส `Pet` เราจะเรียกตัวแปรของคลาสใด ๆ ว่าเป็นวัตถุ (object) ของคลาสนั้น ดังนั้น จากโค้ดเราจะได้ตัวแปร `p` เป็น object ของคลาส `Pet`

เนื่องจากคลาส `Pet` มีเมธอดชื่อ `say_hi()` ประกาศอยู่ นั่นหมายความว่า object ทุกตัวของคลาสนี้จะสามารถเรียกใช้เมธอดนี้ได้

ทดลองพิมพ์คำสั่งต่อใน Shell ดังนี้ และบันทึกผล

```
>>> p.say_hi()
```

หากคุณสังเกตให้ดี จะเห็นว่า ในตอนที่เราประกาศเมธอด `say_hi()` ในคลาส `Pet` นั้น เราประกาศให้เมธอดรับพารามิเตอร์หนึ่งตัวซึ่งตั้งชื่อรอรับว่า `self` แต่เราสามารถเรียกเมธอดได้โดยไม่ต้องส่งอะไรเข้าไปเป็นพารามิเตอร์ ที่จริงแล้ว ในการเรียก `p.say_hi()` นั้น มีการส่งพารามิเตอร์หนึ่งตัวไปอัตโนมัติ คือตัว `p` นั่นเอง

เราสามารถเรียกใช้เมธอด `say_hi()` ได้อีกวิธี (แต่ไม่นิยมใช้) เพื่อให้คุณเข้าใจกระบวนการทำงานมากขึ้น ให้ลองพิมพ์คำสั่งต่อไปนี้ใน Shell และบันทึกผล

```
>>> Pet.say_hi(p)
```

ทดลองแก้ไขไฟล์ `.py` เป็นดังนี้

```
class Pet:
    def say_hi(self):
        print('Hi there.')

    def play_with(self, owner):
        print('I am playing with ' + owner)
```

รันโปรแกรม และลองสั่ง

```
>>> p = Pet()
>>> p.play_with('Panther')
>>> p.say_hi()
```

บันทึกผล

9.2 Instance variable และ class variable

ในการออกแบบคลาส นอกจากเราสามารถประกาศเมธอดภายในคลาสได้แล้ว เรายังสามารถสร้างตัวแปรได้ด้วย โดยตัวแปรที่ใช้ในคลาสจะมีอยู่สองประเภทได้แก่ class variable และ instance variable (หรือ object variable)

ลองแก้ไขไฟล์ `.py` เป็นดังนี้

```
class Pet:
    tricks = []
    def say_hi(self):
        print('Hi there.')

    def add_trick(self, trick):
        Pet.tricks.append(trick)

p = Pet()
q = Pet()
p.add_trick('roll over')
q.add_trick('play dead')
print(p.tricks)
```

ตัวแปรที่มีการประกาศค่าอยู่ในคลาสเช่น `tricks` เราเรียกว่าเป็น class variable ให้คุณลองรันโปรแกรมและบันทึกผล

จากการทดลอง สำหรับ class variable เช่น `tricks` นี้ object ทุกตัวของคลาส `Pet` เข้าถึงตัวแปรตัวเดียวกันหรือไม่

ลองแก้ไขไฟล์ `.py` ใหม่ดังนี้

```
class Pet:
    def say_hi(self):
        print('Hi there. My name is ' + self.name + '.')

    def set_name(self, name):
        self.name = name

p = Pet()
q = Pet()
p.set_name('Panther')
q.set_name('Cheetah')
p.say_hi()
q.say_hi()
```

ในโค้ดมีการเข้าถึงตัวแปร `self.name` โดยไม่มีการประกาศไว้ในคลาส เราเรียกตัวแปรแบบนี้ว่าเป็น instance variable หรือ object variable ให้คุณลองรันโปรแกรมและบันทึกผลที่ได้

จากการทดลอง สำหรับ instance variable นี้ object ของคลาส `Pet` แต่ละตัวเข้าถึงตัวแปรตัวเดียวกันหรือไม่

จงอธิบายความแตกต่างของ class variable กับ instance variable

9.3 Constructor

แก้ไขไฟล์ .py เป็นดังนี้

```
class Pet:
    def __init__(self):
        self.tricks = []

    def add_trick(self, trick):
        self.tricks.append(trick)

p = Pet()
q = Pet()

p.add_trick('jump')
q.add_trick('play dead')

print(p.tricks, q.tricks)
```

ทดลองรันและบันทึกผลที่ได้

จากการทดลอง ตัวแปร `tricks` ของ object `p` และ `q` เป็นตัวแปรตัวเดียวกันหรือไม่ สังเกตได้จากอะไร

ทดลองลบเมธอด `__init__()` ออกจากคลาส `Pet` และรันอีกครั้ง บันทึกผลที่ได้

คุณคิดว่าโปรแกรมเกิด error เพราะอะไร

นำเมธอด `__init__()` กลับมาใหม่ ทำการทดลองและสรุปผลว่าเมธอด `__init__()` ถูกเรียกให้ทำงานเมื่อใด

แก้ไขไฟล์ `.py` เป็นดังนี้

```
class Pet:
    number_of_pets = 0
    def __init__(self, name='', kind='pet', tricks=None):
        self.name = name
```

```
        self.kind = kind
        if tricks == None:
            self.tricks = []
        else:
            self.tricks = tricks
        Pet.number_of_pets += 1

    def say_hi(self):
        print('Hi there. My name is ' +
              self.name + '. I am a ' +
              self.kind + '.')

    def add_trick(self, trick):
        self.tricks.append(trick)

p = Pet(name='Panther',kind='cat')
q = Pet('Pluto', tricks=['play balls'])

p.say_hi()
print(p.tricks)
print(p.number_of_pets)

q.say_hi()
print(q.tricks)
print(q.number_of_pets)
```

object ของคลาส `Pet` มี instance variable ทั้งหมดกี่ตัว อะไรบ้าง และมี class variable กี่ตัว อะไรบ้าง

จากการทดลอง instance variable แต่ละตัวของ `p` มีค่าเป็นอะไร และของ `q` มีค่าเป็นอะไร

9.4 การเขียนโปรแกรมเชิงวัตถุเบื้องต้น

แก้ไขโค้ดในไฟล์ .py เป็นดังนี้

```
class RationalNumber:
    def __init__(self, nom, denom):
        self.nom = nom
        self.denom = denom

    def add(self, r):
        nom = self.nom * r.denom + r.nom * self.denom
        denom = self.denom * r.denom
        return RationalNumber(nom, denom)

    def show(self):
        print(str(self.nom)+'/'+str(self.denom))
```

ทดลองรันและพิมพ์คำสั่งใน Shell ตามลำดับดังนี้ บันทึกผลที่เกิดขึ้นทั้งหมดตามลำดับ

```
>>> r1 = RationalNumber(15, 50)
>>> r1.show()
>>> r2 = RationalNumber(6, 20)
>>> r2.show()
>>> r1.add(r2).show()
```

เราใช้คลาส `RationalNumber` ในการแทนข้อมูลประเภท *จำนวนตรรกยะ* หรือจำนวนที่สามารถเขียนในรูปเศษส่วนของจำนวนเต็มได้ โดยกำหนดให้มี instance variable สองตัว ได้แก่ `nom` แทนเศษ และ `denom` แทนส่วน

จากตัวอย่างโค้ด จงวิเคราะห์ว่าเมธอด `add()` รับพารามิเตอร์เป็นชนิดข้อมูลใด และคืนค่าเป็นอะไร

ลองพิมพ์คำสั่งต่อไปนี้ใน Shell และบันทึกผลที่ได้

```
r = RationalNumber(5, 10).add(RationalNumber(4, 3)).add(RationalNumber(6, 30))
r.show()
```

จงอธิบายการทำงานของคำสั่ง

```
RationalNumber(5, 10).add(RationalNumber(4, 3)).add(RationalNumber(6, 30))
```

9.5 โจทย์ปัญหา

1. จงปรับปรุงคลาส `RationalNumber` ให้มีเมธอดชื่อ `normalize()` โดยมีหน้าที่ลดทอนเศษส่วนนั้นให้กลายเป็นเศษส่วนอย่างต่ำ ตัวอย่างการใช้งานเช่น

   ```
   >>> r = RationalNumber(15, 50)
   >>> r.normalize()
   >>> r.show()
   3/10
   ```

2. จงปรับปรุงเมธอด `add()` ของคลาส `RationalNumber` ให้คืนค่าเป็นเศษส่วนที่ลดทอนเป็นเศษส่วนอย่างต่ำเรียบร้อยแล้ว ตัวอย่างการใช้งานเช่น

   ```
   >>> RationalNumber(15, 50).add(RationalNumber(15, 50)).show()
   3/5
   ```

3. จงสร้างคลาสชื่อ `Set` ที่ใช้เก็บจำนวนเต็มไม่ซ้ำกัน และให้รองรับการเรียกด้วยคำสั่งต่อไปนี้

   - `add(k)` ทำการเพิ่มจำนวนเต็ม `k` เข้าไปในเซต หากในเซตมีจำนวนเต็ม `k` อยู่แล้วก็จะไม่เพิ่มเข้าไปซ้ำอีก
   - `remove(k)` ทำการลบจำนวนเต็ม `k` ออกจากเซต หากในเซตไม่มีจำนวนเต็ม `k` อยู่ก็จะไม่เกิดอะไรขึ้น
   - `union(another_set)` ทำการยูเนียนกับ object ของคลาส `Set` อีกตัวหนึ่ง และคืนค่าเป็น object ของคลาส `Set` ที่เป็นผลลัพธ์ของการยูเนียน
   - `intersec(another_set)` ทำการอินเตอร์เซคกับ object ของคลาส `Set` อีกตัวหนึ่ง และคืนค่าเป็น object ของคลาส `Set` ที่เป็นผลลัพธ์ของการอินเตอร์เซค
   - `diff(another_set)` ทำการลบเซตในตัวแปร `another_set` และคืนค่าเป็นผลลัพธ์ของการลบ
   - `count()` คืนค่าเป็นจำนวนสมาชิกในเซต
   - `show()` แสดงข้อมูลในเซต เรียงจากน้อยไปมาก

   ดูตัวอย่างการใช้งานดังนี้

```
>>> s1 = Set()
>>> s1.add(5)
>>> s1.add(3)
>>> s1.add(19)
>>> s2 = Set()
>>> s2.add(3)
>>> s2.add(9)
>>> s2.add(9)
>>> s1.show()
{3, 5, 19}
>>> s2.show()
{3, 9}
>>> r = s1.union(s2)
>>> r.show()
{3, 5, 9, 19}
>>> r.count()
4
```

4. จงจับคู่กับนิสิตในหมู่เรียนเดียวกัน คิดโครงงานซอฟต์แวร์หนึ่งชิ้นที่จะทำเป็นโปรเจคของรายวิชานำเสนอช่วงท้ายของภาคเรียน โดยอาจเป็นเกม หรือโปรแกรมเพื่ออำนวยความสะดวกต่าง ๆ ก็ได้ เกณฑ์การให้คะแนนขึ้นอยู่กับ

   (a) ความเหมาะสมของระดับความยากง่าย การเสนอหัวข้อชิ้นงานที่ง่ายเกินไป นอกจากจะได้คะแนนน้อย ยังแสดงถึงการดูถูกตัวเองอีกด้วย

   (b) ประโยชน์ของชิ้นงาน หากเป็นเกมจะสื่อถึงความน่าสนใจของเกม

   (c) การนำเสนอด้วยภาษาอังกฤษ

หมายเหตุ คุณยังไม่ต้องส่งข้อเสนอในสัปดาห์นี้ และยังสามารถปรับเปลี่ยนหัวข้อได้หากมีแนวคิดใหม่จากปฏิบัติการต่อ ๆ ไป จนกว่าจะถึงกำหนดส่งหัวข้อ