การใช้คลาสในภาษาไพธอน

ไพธอน (Python) เป็นภาษาโปรแกรมเชิงวัตถุ (Object Oriented Programming Language) ภาษาหนึ่ง และเกือบทุกสิ่งในภาษาไพธอนเป็นออปเจ็ค (Object) โดยภายในออปเจ็คประกอบด้วย แอตทริบิวต์ (Attribute : คุณลักษณะ หรือข้อมูลภายในออปเจ็ค) และ เมธอด (Method) ส่วนคลาส (Class) เปรียบเสมือนต้นแบบที่ใช้ในการสร้างออปเจ็ค หรือเป็นพิมพ์เขียว (Blueprint) เพื่อใช้ในการสร้างออปเจ็ค

ตัวอย่าง ข้อมูลพื้นฐานที่เป็นคลาส

```
>>> print(type(3))
<class 'int'>
```

การสร้างคลาส

การสร้างคลาสจะใช้คีย์เวิร์ด class

ตัวอย่าง การสร้างคลาสชื่อ FirstClass เช่น

```
>>> class FirstClass:
        a = 10
```

ตัวอย่าง การใช้คลาสชื่อ FirstClass เพื่อสร้างออปเจ็ค เช่น

```
>>> c = FirstClass()
>>> print(c.a)
10
```

ฟังก์ชัน __init__()

ทุกคลาสมีฟังก์ชันที่ชื่อว่า __init__() ซึ่งจะทำงานทันทีเมื่อออปเจ็ค ถูกสร้างขึ้นมาแล้ว และส่วนมากจะถูกใช้ในการกำหนดค่าเริ่มต้น เช่น กำหนดค่าให้กับแอตทริบิวต์ของออปเจ็ค หรือการดำเนินการอื่น ๆ ที่จำเป็นเมื่อออปเจ็ค ถูกสร้างขึ้นมา

ตัวอย่าง การสร้างคลาส ชื่อ Person และใช้ฟังก์ชัน __init__ เพื่อกำหนดค่าให้กับแอตทริบิวต์ชื่อ name และ age เช่น

```
>>> class Person:
        def __init__(self,name,age):
            self.name = name
            self.age = age
```

หมายเหตุ ฟังก์ชัน __init__() จะถูกเรียกใช้งานทุกครั้งที่ใช้คลาสเพื่อสร้างออปเจ็ค ใหม่ขึ้นมา ทำหน้าที่เป็น Constructor ของคลาส และจะรับ Self มีเป็นพารามิเตอร์แรกเสมอ โดย Self จะใช้อ้างถึงออปเจ็ค ที่ถูกสร้างขึ้นจากคลาส

หมายเหตุ เมธอดที่มีเครื่องหมาย “__” (Double Under Score) ข้างหน้าและข้างหลัง จะเรียกว่า ดันเดอ เมธอด (Dunder Method) เช่น ชื่อออปเจ็ค .__len__() เราอาจเรียกเมธอดนี้ว่า ดันเดอ len ก็ได้ เป็นต้น และสำหรับการดู id ของตัวแปร จะใช้ print(id(a))

ตัวอย่าง การสร้างออปเจ็ค ของคลาส และการเรียกใช้งาน เช่น

```
>>> p1 = Person("Paiboon",45)
>>> print(p1.name)
Paiboon
>>> print(p1.age)
45
```

ฟังก์ชัน __str__()

ฟังก์ชัน __str__() ใช้ในกำหนดค่าสตริงของออปเจ็ค ที่ถูกส่งกลับ โดยถ้าไม่มีการกำหนดฟังก์ชัน __str__() ไว้ ก็จะส่งสตริงที่ใช้แทนออปเจ็ค กลับมา

ตัวอย่าง การส่งสตริงที่ใช้แทนออปเจ็ค ที่**ไม่มี**การกำหนดฟังก์ชัน __str__()

```
>>> class Person:
        def __init__(self, name, age):
            self.name = name
            self.age = age

>>> p1 = Person("Paiboon",48)
>>> print(p1)
<__main__.Person object at 0x000001998CA863D0>
```

ตัวอย่าง การส่งสตริงของออปเจ็ค กลับมา เมื่อมีการกำหนดฟังก์ชัน __str__()

```
>>> class Person:
        def __init__(self, name, age):
            self.name = name
            self.age = age

        def __str__(self):
            return f"{self.name} ({self.age})"
>>> p1 = Person("Paiboon",48)
>>> print(p1)
Paiboon (48)
```

เมธอดของออปเจ็ค (Object Methods)

ในออปเจ็ค สามารถใส่เมธอด (Methods) ไว้ได้ ซึ่งเมธอดในออปเจ็ค ก็คือ ฟังก์ชันที่อยู่ภายในออปเจ็ค (ซึ่งจะต้องมีพารามิเตอร์ตัวแรกเป็น self เสมอ)

ตัวอย่าง การสร้างคลาส ชื่อ Person เช่น

```
>>> class Person:
        def __init__(self,name,age):
            self.name = name
            self.age = age
        def myFunc(self):
            print(f"Hello, my name is {self.name}")
```

ตัวอย่าง การใช้งาน เช่น

```
>>> p1 = Person("Paiboon",45)
>>> p1.myFunc()
Hello, my name is Paiboon
```

หมายเหตุ พารามิเตอร์ self ใช้อ้างอิงถึงอินสแตนซ์ (Instance) ปัจจุบันของคลาส และใช้ในการเข้าถึงแอตทริบิวต์และเมธอดต่าง ๆ ภายในคลาส

พารามิเตอร์ self

พารามิเตอร์ self ใช้อ้างอิงถึงอินสแตนซ์ (Instance) ปัจจุบันของคลาส และใช้ในการเข้าถึงแอตทริบิวต์และเมธอดต่าง ๆ ภายในคลาส

พารามิเตอร์ self ไม่จำเป็นต้องตั้งชื่อว่า self สามารถตั้งชื่อว่าอะไรก็ได้ แต่ต้องเป็น***พารามิเตอร์ตัวแรก*** (First Parameter) ของเมธอดที่อยู่ภายในคลาส

ตัวอย่าง การใช้ตัวแปรชื่ออื่น ๆ ที่ไม่ใช่ self เช่น

```
class Person:
    def __init__(sss,name,age):
        sss.name = name
        sss.age = age

    def pnt(aaa):
        print(f"Hello, my name is {aaa.name}")

p1 = Person("Paiboon", 45)
p1.pnt()
```

การแก้ไขค่าแอตทริบิวต์ของออปเจ็ค

ตัวอย่าง การแก้ไขค่าแอตทริบิวต์ของออปเจ็ค เช่น

```
class Person:
    def __init__(sss,name,age):
        sss.name = name
        sss.age = age

    def pnt(aaa):
        print(f"Hello, my name is {aaa.name}")

p1 = Person("Paiboon", 45)
p1.pnt()
print(p1.age)
p1.age = 46
print(p1.age)
```

การลบแอตทริบิวต์ของออปเจ็ค

การลบแอตทริบิวต์ของออปเจ็ค จะใช้คีย์เวิร์ด del

ตัวอย่าง การลบแอตทริบิวต์ของออปเจ็ค โดยใช้รูปแบบ ***del ชื่อออปเจ็ค .ชื่อแอตทริบิวต์*** เช่น

```
class Person:
    def __init__(sss,name,age):
        sss.name = name
        sss.age = age

    def pnt(aaa):
        print(f"Hello, my name is {aaa.name}")

p1 = Person("Paiboon", 45)
p1.pnt()
print(p1.age)
p1.age = 46
print(p1.age)
del p1.age
print(p1.age)
```

ผลลัพธ์ที่ได้

```
Hello, my name is Paiboon
45
46
Traceback (most recent call last):
  File "C:\Documents\Python\SourceCodes\Test03.py", line 15, in <module>
    print(p1.age)
AttributeError: 'Person' object has no attribute 'age'
```

ตัวอย่าง การลบออปเจ็ค ใช้รูปแบบ ***del ชื่อออปเจ็ค*** เช่น

```
class Person:
    def __init__(sss,name,age):
        sss.name = name
        sss.age = age

    def pnt(aaa):
        print(f"Hello, my name is {aaa.name}")

p1 = Person("Paiboon", 45)
p1.pnt()
del p1
p1.pnt()
```

ผลลัพธ์ที่ได้ คือ

```
Hello, my name is Paiboon
Traceback (most recent call last):
  File "C:\Documents\Python\SourceCodes\Test03.py", line 12, in <module>
    p1.pnt()
NameError: name 'p1' is not defined
```

การใช้ pass

ตัวอย่างการใช้คำสั่ง pass จะใช้ในกรณีที่ต้องการสร้างคลาสที่ว่างเปล่า ซึ่งปกติจะไม่สามารถสร้างคลาสที่ว่างเปล่าได้ และเพื่อหลีกเลี่ยงการเกิดข้อผิดพลาด (Error) ก็สามารถใส่ pass เพื่อป้องกันข้อผิดพลาดได้

ตัวอย่าง การใช้ pass ในคลาส

```
class Person:
    pass
```

การสืบทอดคลาสในภาษาไพธอน (Python Inheritance)

การสืบทอด (Inheritance) จะทำให้สามารถสร้างคลาสที่สืบทอด (Inherit) แอตทริบิวต์ (Attrubutes) และเมธอด (Methods) ทั้งหมดจากอีกคลาสหนึ่งได้

Parent Class หรือ**คลาสแม่** เป็นคลาสที่ถูกสืบทอด หรืออาจเรียกว่า Base Class

Child Class หรือ**คลาสลูก** เป็นคลาสที่สืบทอดจากอีกคลาสหนึ่ง สามารถเรียกว่า Derived Class

การสร้างคลาสแม่

คลาสใด ๆ ก็ตาม สามารถนำมาทำเป็นคลาสแม่ได้ทั้งหมด ดังนั้นในการสร้างคลาส ก็สามารถทำได้เหมือนกกับการสร้างคลาสทั่วไป

ตัวอย่าง การสร้างคลาส Person

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()
```

การสร้างคลาสลูก

การสร้างคลาสที่สืบทอดการทำงานต่าง ๆ จากคลาสอื่น สามารถทำได้โดยการใส่คลาสแม่เป็นพารามิเตอร์ในขณะที่สร้างคลาสลูก

ตัวอย่าง การสร้างคลาสชื่อ Student ซึ่งสืบทอดพรอพเพอร์ตีและเมธอดจากคลาส Person

```
class Student(Person):
    pass
```

หมายเหตุ ในที่นี้จะใส่คีย์เวิร์ด pass เพราะไม่ต้องการเพิ่มแอตทริบิวต์และเมธอดใด ๆ เข้าไปในคลาสอีก

และตอนนี้ คลาส Student ก็จะมีแอตทริบิวต์และเมธอดเหมือนกับคลาส Person

ตัวอย่าง การสร้างคลาส การสืบทอด และการเรียกใช้งาน

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

class Student(Person):
    pass

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()

s1 = Student("Meechai","Chaiyo")
s1.print_fullname()
```

ผลลัพธ์ที่ได้

```
Paiboon Pongwongtragull
Meechai Chaiyo
```

การเพิ่มฟังก์ชัน __init__()

เมื่อสร้างคลาสลูกที่สืบทอดแอตทริบิวต์และเมธอดจากคลาสแม่เสร็จแล้ว และต้องการที่จะเพิ่มฟังก์ชันชื่อ __init__() ให้กับคลาสลูก สามารถทำได้ดังนี้

ตัวอย่าง การเพิ่มฟังก์ชัน __init__()

```
class Student(Person):
    def __init__(self,fname,lname):
        # ใส่สิ่งต่าง ๆ ที่ต้องการเพิ่ม
        pass
```

หมายเหตุ เมื่อใส่ฟังก์ชัน __init__() เข้าไปในลักษณะนี้ จะทำให้คลาสลูกไม่สามารถสืบทอดฟังก์ชัน __init__() จาก Parent อีกต่อไป หมายความว่า ฟังก์ชัน __init__() ได้ทำการ Overides ฟังก์ชัน __init__() ของคลาสแม่

หากต้องการให้ฟังก์ชัน __init__() ที่สืบทอดจากคลาสแม่ยังคงใช้ได้อยู่ ให้เพิ่มการเรียกใช้งานฟังก์ชัน __init__() ของคลาสแม่ ดังตัวอย่างต่อไปนี้

ตัวอย่าง การเรียกใช้งานฟังก์ชัน __init__() ของคลาสแม่

```
class Student(Person):
    def __init__(self,fname,lname):
        Person.__init__(self,fname,lname)
```

ตัวอย่าง การเรียกใช้งานฟังก์ชัน __init__() ของคลาสแม่ฉบับเต็ม

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

class Student(Person):
    def __init__(self,fname,lname):
        Person.__init__(self,fname,lname)

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()

s1 = Student("Meechai","Chaiyo")
s1.print_fullname()
```

การใช้ฟังก์ชัน super()

ภาษาไพธอนมีฟังก์ชัน super() ที่ช่วยให้คลาสลูกสามารถเรียกเมธอดของคลาสแม่จากภายในคลาสลูกได้ ดังตัวอย่างต่อไปนี้

ตัวอย่าง การใช้ฟังก์ชัน super()

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

class Student(Person):
    def __init__(self,fname,lname):
        super().__init__(fname,lname)              #<------

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()

s1 = Student("Meechai","Chaiyo")
s1.print_fullname()
```

การใช้ฟังก์ชัน super() จะทำให้ไม่จำเป็นต้องใช้ชื่อของคลาสแม่ในการเรียกใช้งาน และไม่จำเป็นต้อง ส่งตัวแปร **self** เป็นอากิวเมนต์ให้กับฟังก์ชัน super()

การเพิ่มแอตทริบิวต์

การเพิ่มแอตทริบิวต์ให้กับคลาสลูกสามารถทำได้ดังตัวอย่างต่อไปนี้

ตัวอย่าง การเพิ่มแอตทริบิวต์ ชื่อ graduation_year ให้กับคลาส Student

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

class Student(Person):
    def __init__(self,fname,lname, gra_year):      #<------
        super().__init__(fname,lname)
        self.graduation_year = gra_year            #<------Add new attribute

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()

s1 = Student("Meechai","Chaiyo",2022)
s1.print_fullname()
print(s1.graduation_year)                          #<------ Displaying attribute
```

การเพิ่มเมธอด

ตัวอย่าง การเพิ่มเมธอด welcome ให้กับคลาส Student

```
class Person:
    def __init__(self,fname,lname):
        self.firstname = fname
        self.lastname = lname

    def print_fullname(self):
        print(self.firstname,self.lastname)

class Student(Person):
    def __init__(self,fname,lname, gra_year):
        super().__init__(fname,lname)
        self.graduation_year = gra_year

    def welcome(self):                                  #<------
        print(f"Welcome {self.firstname} {self.lastname} to the class of
{self.graduation_year}")

p1 = Person("Paiboon","Pongwongtragull")
p1.print_fullname()

s1 = Student("Meechai","Chaiyo",2022)
s1.print_fullname()
print(s1.graduation_year)
s1.welcome()                                            #<------
```

คำศัพท์ที่ต้องทราบ

คลาส คือ องค์ประกอบพื้นฐานที่สุดของภาษาไพธอน ซึ่งเป็นส่วนสำคัญของการเขียนโปรแกรมเชิงวัตถุ ทุกอย่างในภาษาไพธอนล้วนเป็นออปเจ็ค เช่น เลขจำนวนเต็ม (Integer) ลิสก์ (List) ดิกชันนารี (Dictionary) หรือ ฟังก์ชัน (Functions) เป็นต้น ทุก ๆ ออปเจ็คจะมีชนิดของมันเอง และชนิดของออปเจ็คจะถูกสร้างจากคลาส ดังนั้น การสร้างคลาสสำหรับผู้ใช้ จึงเป็นการสร้างประเภทข้อมูลโดยผู้ใช้ (User-defined Type) และนำไปใช้ในการสร้าง ออปเจ็ค

อินสแตนซ์ (Instance) คือ ออปเจ็คที่สร้างจากคลาส ตัวอย่างเช่น ลิสก์เป็นคลาสอย่างหนึ่งในภาษาไพธอน และเมื่อสร้างลิสก์ ก็จะได้อินสแตนซ์ของคลาสลิสก์ สมมติว่า a ถูกสร้างมาจากคลาส Person เราอาจกล่าวได้ว่า a เป็นอินสแตนซ์ของคลาส Person

ออปเจ็ค (Object) คือ สิ่งที่สร้างมาจากคลาส (Class Instances) จากที่ผ่านมา เราจะเรียก a ว่าเป็น ออปเจ็ค a

แอตทริบิวต์ (Instance Attributes) คือ ข้อมูลที่เป็นสมาชิกของแต่ละออปเจ็ค โดยส่วนใหญ่จะกำหนด หรือสร้างไว้ในเมธอด __init__() ของคลาส

คลาสแอตทริบิวต์ (Class Attributes) คือ ตัวแปรที่ประกาศไว้ในคลาส ซึ่งจะแชร์กับออปเจ็ค ทั้งหมดที่สร้างจากคลาสนั้นๆ

เมธอด (Method) คือ ฟังก์ชันที่สร้างไว้ภายในคลาสและมีพารามิเตอร์ตัวแรกเป็น self

ข้อตกลงในการตั้งชื่อตัวแปร

ชื่อตัวแปร ชื่อฟังก์ชัน และชื่อเมธอด

ควรประกาศตัวแปรเป็นตัวอักษรพิมพ์เล็กทั้งหมด และควรตั้งชื่อโดยใช้ Snake Case เช่น i, x, y, my_variable, my_function, class_method, เป็นต้น

ชื่อคลาส

ควรตั้งชื่อโดยใช้ Pascal Case เช่น Model หรือ MyClass เป็นต้น

หมายเหตุ

Camel Case มีรูปแบบคือ แต่ละคำในชื่อตัวแปรยกเว้นคำแรก จะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่ เช่น myName เป็นต้น

Pascal Case มีรูปแบบคือ ทุกคำในชื่อตัวแปรจะขึ้นต้นด้วยตัวอักษรพิมพ์ใหญ่ทั้งหมด เช่น MyName หรือ MyPhoneNumber เป็นต้น

Snake Case มีรูปแบบคือ ทุกคำในชื่อตัวแปร จะคั่นด้วยเครื่องหมายขีดใต้ (_) (Underscore) เช่น my_name หรือ my_phone_number เป็นต้น

แบบฝึกหัดในชั้นเรียน

1. จงเขียนโปรแกรมเพื่อใช้ในการเก็บข้อมูลรูปสี่เหลี่ยม (Rectangle) สามเหลี่ยม (Triangle) และวงกลม (Circle) พร้อมทั้งสามารถแสดงพื้นที่ของแต่ละรูปทรงได้

ตัวอย่างการใช้งาน

```
>> s1 = Rectangle()
>> s1.set_size(20,30)          # กำหนดความกว้างและความยาวของสี่เหลี่ยม
>> s1.print_area()             # แสดงความกว้าง ความยาว และพื้นที่ของสี่เหลี่ยม
width = 20
height = 30
area = 600
```

2. จงสร้างคลาสเพื่อเก็บเงินโดยตั้งชื่อคลาสว่า PiggyBank ซึ่งมีความสามารถดังนี้

2.1เก็บเงิน เหรียญ หรือธนบัตรใดก็ได้ (0.25, 0.5, 1, 5, 10, 20, 50, 100, 500, 1000)

2.2 เช็คยอดเงินได้

2.3 เช็คยอดเหรียญ หรือธนบัตรได้

ตัวอย่างการใช้งาน

```
>> pig1 = PiggyBank()
>> pig1.add(0.25,5)            # เพิ่มเหรียญ 25 สตางค์ 5 เหรียญ
>> pig1.add(1000,2)            # เพิ่มธนบัตร 1000 บาท 2 ใบ
>> pig1.print_all_money()      # แสดงจำนวนเงินทั้งหมด
Coin 0.25 = 5
Bank 1000 = 2
Total = 2001.25
```

แหล่งข้อมูลอ้างอิง

https://www.w3schools.com/python/python_classes.asp

https://www.w3schools.com/python/python_inheritance.asp

https://expert-programming-tutor.com/tutorial/article/L61_PYTHON_CLASSSES_OBJECTS.php