09 การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming)

การเขียนโปรแกรมเชิงวัตถุคือการเขียนโปรแกรมที่แบ่งขอบเขตงานออกเป็นส่วนๆ เรียกว่า ออบเจ็ค (Object) โดยแต่ละส่วนไม่ขึ้นต่อกัน แต่จะมีการทำงานร่วมกัน โดยหลักของการพัฒนาโปรแกรมเชิงวัตถุคือ การนำกลับมาใช้ใหม่ได้ ทำให้ประหยัดเวลาในการพัฒนา และดูแลรักษาง่าย

คลาส (Class) คือ ต้นแบบหรือแม่แบบที่ใช้อธิบายลักษณะ และความสามารถของ object

ออบเจ็ค (Object) คือ สิ่งต่างๆที่มีคุณลักษณะและความสามารถในการทำงาน เช่น คน, รถยนต์, หนังสือ เป็นต้น โดย object จะถูกสร้างมาจากคลาสที่มีการกำหนด แอตทริบิวต์ (Attribute) และ เมธอด (Method) โดย แอตทริบิวก็คือสิ่งที่อธิบายคุณลักษณะของออบเจ็คต์ ส่วน เมธอดก็คือสิ่งที่ใช้อธิบายการทำงานของออบเจ็คต์

In [1]:

```
class MyClass:   #สร้าง Class ชื่อ MyClass
  x = 5          #กำหนด attribute x มีค่า = 5
```

In [2]:

```
p1 = MyClass()  #สร้าง object p1 จาก MyClass
print(p1.x)     #สั่งพิมพ์ค่าของ attribute X
```

```
5
```

ประเภทของแอตทริบิวต์

- Class Attribute คือ ข้อมูลหรือตัวแปรที่กำหนดไว้ในคลาส เปรียบเหมือนตัวแปรสาธารณะที่สามารถเข้าถึงได้จากทุกคลาส
- Instance Attribute คือ ข้อมูลหรือตัวแปรที่รับมาพร้อมกับการสร้างออบเจ็คต์โดยมีการกำหนดค่าเริ่มต้นให้กับข้อมูลหรือตัวแปรที่รับเข้ามาผ่านเมธอดที่เรียกว่า คอนสตรัคเตอร์ (Constructor) โดยสามารถสร้างได้จากเมธอด '*init*'

In [8]:

```
class Tax:                #สร้าง class tax
   taxrate = 0.07         #กำหนด ตัวแปร taxrate = 0.07 เป็น Class atribute

   def __init__(self,product_price,taxrate=None): #สร้างเมธอด _init_ เป็น คอนสตรัคเตอร์ รับข้อมูลราคาสินค้า และข้อมูล taxrate ซึ่งไม่มีก็ได้
      self.productprice =product_price  #กำหนดค่า product_price ให้กับ instance attribute คือ self.productprice
      if taxrate is None: #ตรวจสอบค่า taxrate ที่รับเข้า
         self.taxrate = Tax.taxrate #ถ้าไม่มีให้กำหนด Tax.taxrate ซึ่งเป็น Class Atrribute ให้กับ self.taxrate ซึ่งเป็น Instance Attribute
      else:
         self.taxrate = taxrate #ถ้ามีการรับค่าเข้ามาให้กำหนด self.taxrate = ค่าที่รับเข้ามา

   def caltax(self): #สร้างเมธอด caltax
      return self.taxrate * self.productprice #คำนวนภาษีสินค้าโดย คูณราคาสินค้ากับอัตราภาษี
```

In [10]:

```
myTax = Tax(200) #สร้าง object myTax จาก Class Tax และระบุราคาสินค้า 200
ans1 = myTax.caltax()  #เรียกใช้ เมธอด caltax ใน คลาส Tax เพื่อคำนวนภาษี คือค่าใส่ตัวแปร ans1
print(ans1) #แสดงผลตัวแปร ans1
```

14.000000000000002

In [13]:

```
myTax2 =Tax(450,0.10)
ans2 = myTax2.caltax()
print(ans2)
```

45.0

ประเภทของเมธอด

เมธอด คือ ชุดข้อมูลหรือชุดของคำสั่งที่ใช้อธิบายความสามารถของออบเจ็คต์ ซึ่งจะถูกกำหนดและสร้างไว้ในคลาสเช่นเดียวกับแอดทริบิวต์ โดยเมธอดจะนำข้อมูลแอตทริบิวต์มาใช้เพื่อการทำงานในชุดคำสั่งต่างๆ ด้วยก็ได้ แยก ได้ 2 ประเภท คือ

- Static Method ชุดข้อมูลหรือชุดคำสั่งที่กำหนดให้มีการทำงานในคลาส โดยข้อมูลแอตทริบิวต์ที่ใช้อาจเป็น Class Attribute ที่สามารถเรียกใช้งานชื่อคลาส หรืออาจเป็นข้อมูลที่รับมากับการเรียกใช้เมธอด ในการสร้างเมธอดประเภทนี้จะกำหนดคำสั่ง @staticmethod ไว้ก่อนบรรทัดที่มีการสร้างเมธอดเสมอ และการเรียกใช้เมธอดนี้จะทำผ่านชื่อคลาสได้โดยไม่ต้องสร้างออบเจ็กต์
- Instance Method ชุดข้อมูลหรือชุดคำสั่งที่กำหนดให้มีการทำงานคลาส โดยข้อมูลแอตทริบิ้วต์ที่ใช้อาจจะเป็น Instance Attribute ที่สามารถเรียกใช้งานคีย์เวิร์ด self หรือ อาจเป็นข้อมูลที่รับมากับการเรียกใช้เมธอด การเรียกใช้งานเธอดประเภทนี้จะทำผ่านชื่อออบเจ็กต์ที่สร้างจากคลาสที่มีเมธอดที่ต้องการทุกครั้ง

In [14]:

```
class Tax:
    taxrate = 0.07

    def __init__(self): #สร้างเมธอด __init__ เป็นคอนสตรัคเตอร์แบบไม่มีการรับข้อมูล
        pass #ใช้คำสั่ง pass ผ่านการทำงาน คือไม่มีการกำหนดค่าใดๆให้กับ instance attribute

    @staticmethod #ประกาศคำสั่ง @staticmethod  เพื่อบอกให้รู้ว่าจะสร้าง static method
    def caltax(product_price): #สร้าง เมธอด caltax มีการรับข้อมูลราคาสินค้า เพื่อคำนวนภาษี
        return Tax.taxrate * product_price
```

In [15]:

```
price = 100
ans = Tax.caltax(price)
print(ans)
```

7.000000000000001

In [16]:

```
class Tax:                #สร้าง class tax
    taxrate = 0.07        #กำหนด ตัวแปร taxrate = 0.07 เป็น Class atribute

    def __init__(self,product_price): #สร้างเมธอด _init_ เป็น คอนสตรัคเตอร์ รับข้อมูลราคาสินค้า
        self.productprice =product_price  #กำหนดค่า product_price ให้กับ instance attribute คือ self.productpric
e
        self.taxrate = Tax.taxrate #กำหนด Tax.taxrate ซึ่งเป็น Class Atrribute ให้กับ self.taxrate ซึ่งเป็น Instance
Attribute


    def caltax(self): #สร้างเมธอด caltax
        return self.taxrate * self.productprice #คำนวนภาษีสินค้าโดย คูณราคาสินค้ากับอัตราภาษี

    def caltotal(self,tax):
        return self.productprice + tax
```

In [27]:

```
price = 200
myTax = Tax(price)
ans1 = myTax.caltax()
ans2 = myTax.caltotal(ans1)

print("จำนวนภาษีที่ต้องจ่าย {:.2f} บาท".format(ans1))
print("จำนวนเงินที่ต้องจ่าย {:.2f} บาท".format(ans2))
```

จำนวนภาษีที่ต้องจ่าย 14.00 บาท
จำนวนเงินที่ต้องจ่าย 214.00 บาท

ระดับการเข้าถึงข้อมูล (Access Modifier) Public, Private และ Protected

Access Modifier คือ คีย์เวิร์ดที่ใช้กำหนดระดับการเข้าถึงข้อมูล เป็นกลไกของการกำหนดระดับการเข้าใช้านสมาชิกของคลาส เพื่อรักษาความปลอดภัยและป้องกันการเปลี่ยนแปลงข้อมูลภายในคลาส โดยภาษา python จะมีการแบ่งระดับของ Access Modifier เป็น 3 ระดับ คือ

1. Public : มีขอบเขตเข้าใช้งานได้จากทุกคลาส
2. Private : มีขอบเขตการเข้าใช้งานภายในคลาสเดียวกัน
3. Protected : มีขอบเขตการใช้งานข้อมูลภายในคลาสเดียวกันและคลาสที่สืบทอดกัน

In [28]:

```
#public Attribute
class employee:
    def __init__(self, name, sal):
        self.name=name
        self.salary=sal
```

In [29]:

```
e1=employee("Kiran",10000)
e1.salary
```

Out[29]:

10000

In [30]:

```
e1.salary=20000
e1.salary
```

Out[30]:

20000

In [31]:

```
#protect attribute
class employee:
    def __init__(self, name, sal):
        self._name=name  # protected attribute
        self._salary=sal # protected attribute
```

In [32]:

```
e1=employee("Swati", 10000)
e1._salary
```

Out[32]:

10000

In [33]:

```
e1._salary=20000
e1._salary
```

Out[33]:

20000

In [34]:

```
#Private Attribute
class employee:
    def __init__(self, name, sal):
        self.__name=name  # private attribute
        self.__salary=sal # private attribute
```

In [35]:

```
e1=employee("Bill",10000)
e1.__salary
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-35-8530b5954e6a> in <module>
      1 e1=employee("Bill",10000)
----> 2 e1.__salary

AttributeError: 'employee' object has no attribute '__salary'
```