Classes

Class Definition Syntax

```
class className :
    <statement -1>
    .
    .
    .
    <statement –N>
```

Example

```
class Student :

    pass
```

Constructor

```
class Student :
    def __init__(self):
        print “Hello World”
```

Class and Instance Variables

```
class Student :
    university = “silpakorn”
    def __init__(self, id, name):
        self.__id = id
        self.name = name
```

Private Variables

- Begin with two underscore characters
- Can not be directly accessed

Change to Private Variables 1

```
class Student :
    id = 10
    def __init__(self, id):
        self.id = id
```

```
class Student :
    __id = 10
    def __init__(self, id):
        Self.__id = id
```

Change to Private Variables 2

```
class Student :
    def __init__(self, id):
        self.id = id
```

```
class Student :
    def __init__(self, id):
        Self.__id = id
```

```
class Student :
    school = “silpakorn”
    def __init__(self, id, name):
        self.__id = id
        self.__name = name
    def getId(self):
        return self.__id
    def getName(self):
        return self.__name
    def setId(self, id)
        self.__id = id
    def setName(self, name)
        self.__name = name
```

Let's Try It

s1 = Student(

Implicit Inheritance

- the implicit actions that happen when you define a function in the parent, but *not* in the child.

Example

```
class People():
    def eat(self):
        print "People eat"
class Student(People):
    pass
teacher = People()
somchai = Student()

teacher.eat()
somchai.eat()
```

```
People eat
People eat
```

Override Explicitly

- Sometimes you want the child to behave differently.
- Define a function with the same name in subclass

Example

```
class People():
    def eat(self):
        print "People eat"
class Student(People):
    def eat(self):
        print "Student eat"
teacher = People()
somchai = Student()

teacher.eat()
somchai.eat()
```

```
People eat
Student eat
```

Objects

การสร้างอ็อบเจคหรือการอินสแตนซ์ของคลาส มีรูปแบบคำสั่งโดยการระบุชื่อคลาสและใส่อาร์กิวเมนต์ตรงกับเมท็อดที่เป็น คอนสตรักเตอร์ หรือตรงกับเมทธอดที่มี __init__()

```
std1 = Student ("Sukjai", 07550686)
std2 = Student ("Meesuk", 07550988)
std3 = Student ("Dekdee", 07550224)
```

```
class Student:
    'เป็นการประกาศสร้างคลาส Student เพื่อเก็บข้อมูลนักเรียน'
    stdCount = 0      //นับจำนวนนักเรียน
    def __init__(self, name, ID):
        self.name = name
        self.ID = ID
        Student.stdCount += 1
    def displayStudent(self):
        print "Name : ", self.name, "ID : ",self.ID
```

```
std1 = Student ("Sukjai", 07550686)
std2 = Student ("Meesuk", 07550988)
std3 = Student ("Dekdee", 07550224)
std1.displayStudent()
std2.displayStudent()
std3.displayStudent()
print "Total stdudent %d" % Student.stdCount
```

```
Name : Sukjai ,ID : 07550686
Name : Meesuk, ID : 07550988 Name
Dekdee, ID : 07550224
Total student 3
```

การเข้าถึงแอตทริบิวต์

การเข้าถึงแอตทริบิวต์มี2วิธีคือ

- ใช้อ็อบเจค
- ใช้ชื่อคลาส

```
std1.displayStudent()
std2.displayStudent()
std3.displayStudent()
```

ใช้อ็อบเจคในการเข้าถึงเมทธอดที่ต้องการเรียกใช้

```
print "Total employee %d" % Student.stdCount
```

→ ใช้ชื่อคลาสในการเข้าถึงตัวแปร

การลบอ็อบเจค

ภาษาไพธอนจะมีตัวจัดการที่เรียกว่า garbage collection เหมือนในภาษาจาวา เมื่อมีการประมวลผลคำสั่งจะมีการเข้าไปตรวจสอบอ็อบเจคนั้น ๆ และเพิ่มค่าให้กับอ็อบเจกต์ที่เรียกใช้ครั้งละ 1 และอ็อบเจกต์ใดที่ไม่ถูกเรียกใช้งานจะถูกลบไปครั้งละ 1 เช่นกัน เมื่อตรวจสอบว่าอ็อบเจคใดไม่ถูกใช้งานจนครบกำหนดที่ 0 แล้ว โปรแกรมจะลบอ็อบเจคนั้นออกจากหน่วยความจำโดยอัตโนมัติทันที

ในกรณีที่ต้องการลบอ็อบเจคด้วยตนเอง จะใช้เมท็อดพิเศษ คือ __del__() หรือที่เรียกอีกอย่างว่าdestructor แล้วเรียกใช้คำสั่ง del<ชื่ออ็อบเจค>

```
class Point:
    def __init__(self, x = 0, y = 0):
        self.x = x
        self.y = y
        print 'x, y = ',self.x, self.y

    def __del__(self):
        class_name = self.__class__.__name__
        print class_name, "destroyed"
```

```
point1 = Point(3,5)
point2 = Point(10,25)
point3 = Point(12,32)
print 'point1 id =', id(point1),\
        'point1 id =', id(point2),\
        'point3 id =', id(point3)
del point1
del point2
del point3
```

```
x, y =  3 ,5
x, y =  10 ,25
x, y =  12, 32
point1 id = 33245784 point1 id = 33247024
point3 id = 33287264
Point destroyed
Point destroyed
Point destroyed
```

แสดงผลค่า x, y ของแต่ละอ็อบเจค
แสดง id ของแต่ละอ็อบเจค
หลังจากนั้นมีการใช้คำสั่งลบอ็อบเจคจะไปเรียกเมทธอด __del__() ซึ่งจะลบอ็อบเจคพร้อมทั้งแสดงผลข้อความว่าอ็อบเจคถูกลบออกไป

การเพิ่ม แก้ไข และลบแอตทิบิวต์ของคลาสหรืออ็อบเจกต์สามารถกระทำได้ตลอดเวลา โดยใช้คำสั่งของฟังก์ชันที่ภาษาไพธอนมีมาให้

ชื่อฟังก์ชัน	หน้าที่
getattr(object, name [, default])	เพื่อเข้าถึงแอตทริบิวต์ของอ็อบเจ8
hasattr(object, name)	เพื่อตรวจสอบว่ามีแอตทริบิวต์นั้นอยู่ภายในอ็อบเจคหรือไม่
setattr(object, name, value)	เพื่อกำหนดค่าให้กับแอตทริบิวต์ ในกรณีที่ยังไม่มีแอตทริบิวต์จะสร้างขึ้นมาใหม่
delattr(object, name)	เพื่อลบแอตทริบิวต์

```
>>> hasattr(emp3,'age')
False
>>> setattr(emp1, 'age',25)
>>> setattr(emp2, 'age',30)
>>> setattr(emp3, 'age',28)
>>> getattr(emp3, 'age')
28
>>> delattr(emp3, ‘age')
>>> hasattr(emp3, 'gender')
False
```

การใช้ฟังก์ชันต่าง ๆ เพื่อจัดการแอตทริบิวต์ โดยที่

- คำสั่งที่ 1 hasattr() เพื่อตรวจสอบว่า อ็อบเจค emp3 มีแอตทริบิวต์ age หรือไม่ ผลลัพธ์เป็น False แสดงว่าไม่มีแอตทริบิวต์ age อยู่ในอ็อบเจค
- คำสั่งที่ 2 -4 setattr() เป็นการเพิ่มแอตทริบิวต์ age และค่าของแอตทริบิวต์ ให้กับอ็อบเจกต์ต่าง ๆ
- คำสั่งที่ 5 getattr() เป็นการเรียกดูข้อมูลของอ็อบเจค emp3 ผลลัพธ์ที่แสดง คือ 28
- คำสั่งที่ 6 delattr() เป็นการสั่งลบแอตทริบิวต์ age ของอ็อบเจค emp3 เป็นต้น

Inheritance

Syntax

```
class Subclass(Superclass):
    <statement -1>
    .
    .
    .
    <statement –N>
```

```
class People:
    fastfood="I have a bacon"
    lunch="I have a snausage"

class  Student(People):
    pass
```

```
>>>p = Student()
>>>p.fastfood
' I have a bacon '
>>>p.like2
' I have a snausage '
```

Override method and class variable

```
class People:
    def fastfood(self):
        print "I have a bacon"
    def dinner(self):
        print "With my parent"

class  Student(People):
    def fastfood(self):
        print "I have a toast "
```

```
>>>p = Student()
>>>p.fastfood()
' I have a toast '
>>>p.dinner()
' With my parent '
```

```
class People:
    fastfood="I have a bacon"
    lunch="I have a snausage"

class  Student(People):
    lunch= "I have a toast"
```

```
>>>p = Student()
>>>p.fastfood
' I have a bacon '
>>>p.like2
' I have a toast '
```

Multiple Inheritance

```
class Subclass(Superclass1,Superclass2,…):
    <statement -1>
    .
    .
    .
    <statement –N>
```

```
class Dad:
    fastfood="I have a bacon"

class Mom:
    lunch= "I have a toast"

class Child(Dad, Mom):
    pass
```

```
>>>c = Child()
>>>c.fastfood
' I have a bacon '
>>>c.lunch
' I have a toast '
```

อ้างอิงจาก

https://sites.google.com/site/dotpython/

https://docs.python.org/2.7/reference/index.html