01076105, 01076106

Object Oriented Programming

Object Oriented Programming Project

Object and Class

หลักการสำคัญของ Object Oriented

- Object Oriented Programming มีหลักการสำคัญอยู่ 4 ข้อ
  1. **Encapsulation** เป็นหลักการ modular คือแบ่งโปรแกรมเป็นส่วนย่อย เพื่อให้ความซับซ้อนโดยรวมลดลง โดย OOP จะนำ data และ code ที่เกี่ยวกับเรื่องใดเรื่องหนึ่งมารวมไว้ด้วยกันโดยเรียกว่า object และป้องกันไม่ให้ผู้อื่นมายุ่งกับข้อมูล
  2. **Abstraction** คือ การแยกระหว่าง ส่วนที่ให้ผู้อื่นมองเห็น (Interface) กับการทำงาน (Implementation) ของส่วนนั้น ยกตัวอย่างเช่น บริการดึงข้อมูลอุณหภูมิ กับการทำงานของบริการนั้น กล่าวคือ ผู้ที่เรียกใช้บริการไม่จำเป็นต้องรู้ว่าบริการนั้นทำงานอย่างไร เพียงแต่รู้วิธีการเรียกใช้
  3. **Inheritance** คือ การถ่ายทอดคุณสมบัติของ object ที่มีความคล้ายคลึงกัน
  4. **Polymorphism** คือ การใช้ Interface ที่เหมือนกันกับ object ที่ต่างกัน

ประโยชน์ของ Object Oriented

- การ reuse code จะทำได้ดีกว่า
- การขยายหรือแก้ไขโปรแกรมจะทำได้ง่ายกว่า
- การพัฒนา การทดสอบ และ การดูแลรักษาทำได้ง่ายกว่า
- เหมาะสมกับการพัฒนา software ขนาดใหญ่มากกว่า
- การอธิบายการทำงานให้กับ non technical จะทำได้ง่ายกว่า

Class

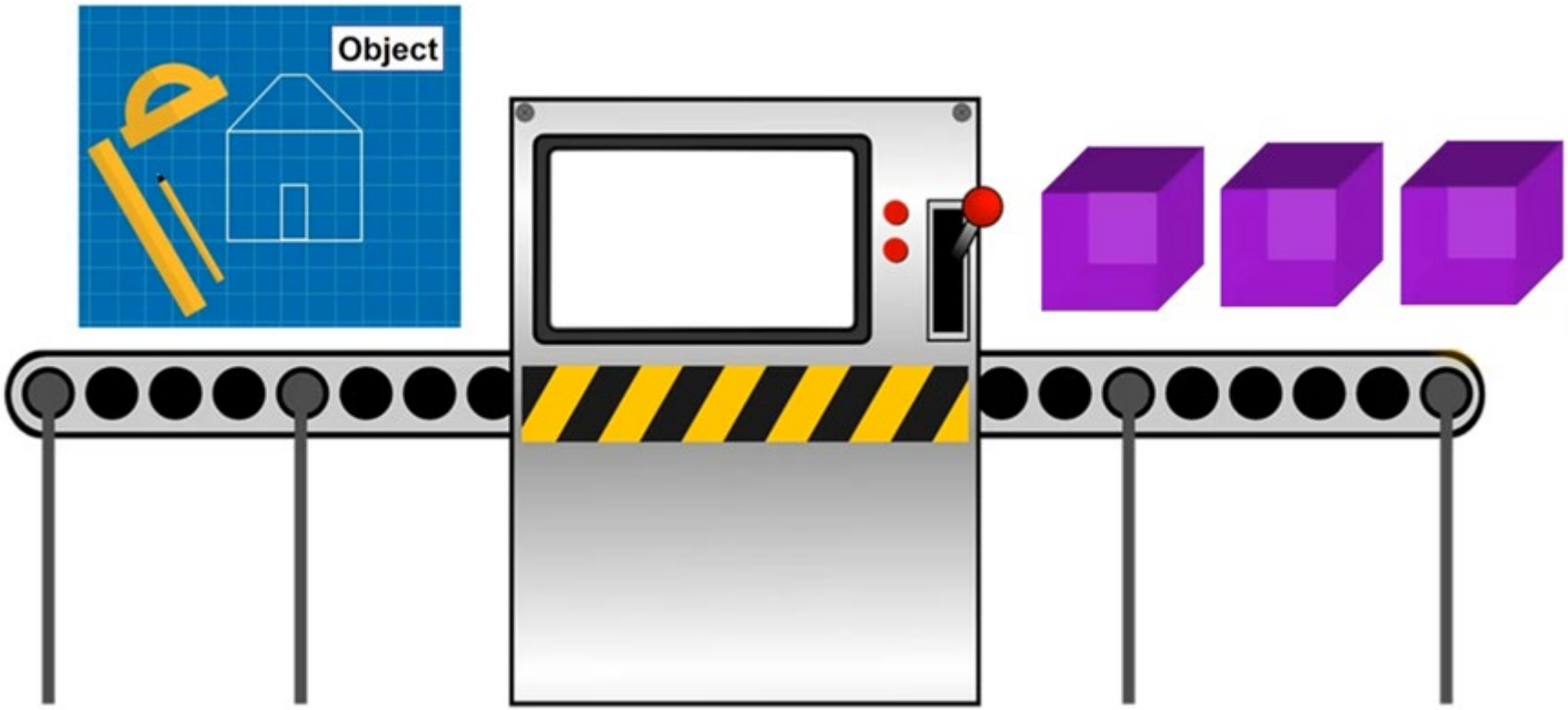
- การจะสร้าง object จะต้องมี class ก่อน

> A **blueprint** for creating objects.

- คลาสทำหน้าที่คล้าย “พิมพ์เขียว” ของ object โดยจะต้องกำหนดรายละเอียดของ class ก่อน จึงจะสามารถสร้าง object ได้ เหมือนแบบบ้านที่นำไปสร้างที่ไหน ก็จะออกมาเหมือนกัน
- ภายในคลาสจะประกอบด้วย attribute (คุณลักษณะ) ของ object นั้น และ behavior (พฤติกรรม) ที่ object นั้นสามารถทำได้

Class

- Class เป็นต้นแบบในการผลิต object



Class

- ตัวอย่างของ object ในระบบการลงทะเบียน เช่น
  - นักศึกษาเป็น object
  - อาจารย์เป็น object
  - รายวิชาเป็น object
  - ห้องเรียน เป็น object

Class

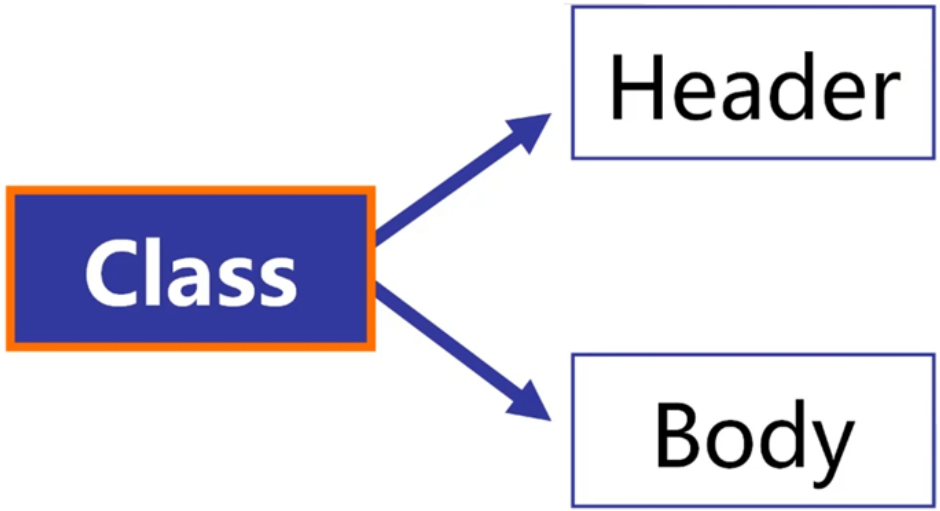
- Object ในบางระบบอาจรู้สึกว่าจับต้องได้ยากกว่า เช่น ตัวอย่างของ object ใน Facebook
  - ผู้ใช้ (user) เป็น object
  - page เป็น object
  - group เป็น object
  - post เป็น object
  - comment เป็น object
- แต่หากมีทั้ง attribute และ behavior ของสิ่งนั้น ก็สร้างเป็น object ได้

Activity #1 :

- จับกลุ่ม 2 คน กับเพื่อนที่นั่งข้างๆ แล้วเลือก กิจการ หรือ กิจกรรม หนึ่ง แล้วให้ list object ของกิจการ หรือ กิจกรรมนั้น (15 นาที)
  - ให้บอกชื่อ กิจการ หรือ กิจกรรม
  - ให้บอกชื่อของ object ของกิจการ หรือ กิจกรรม
  - ให้บอก attribute และ behavior ของ object ที่คาดว่าน่าจะมี

Class

- โครงสร้าง Class ใน python ประกอบด้วย 2 ส่วน คือ Header และ Body
- Header เป็นส่วนหัวของ Class ประกอบด้วย ชื่อคลาส และโครงสร้าง
- Body บอกรายละเอียดภายใน Class



Class

- การตั้งชื่อ Class (header) มักจะใช้เป็นคำนาม เพราะคลาสเป็น object

```
class <ClassName>(object):
```

- การกำหนด Class จะเขียนคำว่า class เป็นตัวเล็ก และใช้ชื่อ class เป็น Pascal Case และปิดท้ายด้วยเครื่องหมาย :
- สำหรับ (object) ภาษา python ตั้งแต่ 3.0 ขึ้นไป ไม่ต้องเขียน (object) ก็ได้ เช่น
  - class Student:
  - class Teacher:

Class

- ในภาษา python เราจะใช้ Pascal Case (หรือ upper camel case) ในการกำหนดชื่อ Class
- การเขียนในแบบ Pascal Case คือ ให้ขึ้นต้นตัวแรกด้วยอักษรตัวใหญ่ ของแต่ละคำ เช่น
  - House
  - BankAccount
  - Student
  - Teacher

Class

- ส่วนของ body ส่วนที่ 1 คือ attribute ซึ่งทำหน้าที่บอกคุณลักษณะของ object เช่น object Student อาจมี attribute ดังนี้
  - รหัสนักศึกษา
  - ชื่อ
- สามารถเขียนเป็นโปรแกรมได้ดังนี้
- จะเห็นว่า stu1 และ stu2 สร้างจากต้นแบบหรือ class เดียวกัน ดังนั้นจึงมีคุณสมบัติ (attribute) เหมือนกันด้วย

```
class Student:
    id = ''
    name = ''

stu1 = Student()
stu1.id = '001'
stu1.name = "John"

stu2 =Student()
stu2.id = '002'
stu2.name = "Peter"

print(stu1.id)
print(stu2.id)
```

```
001
002
```

001
John

002
Peter

Class

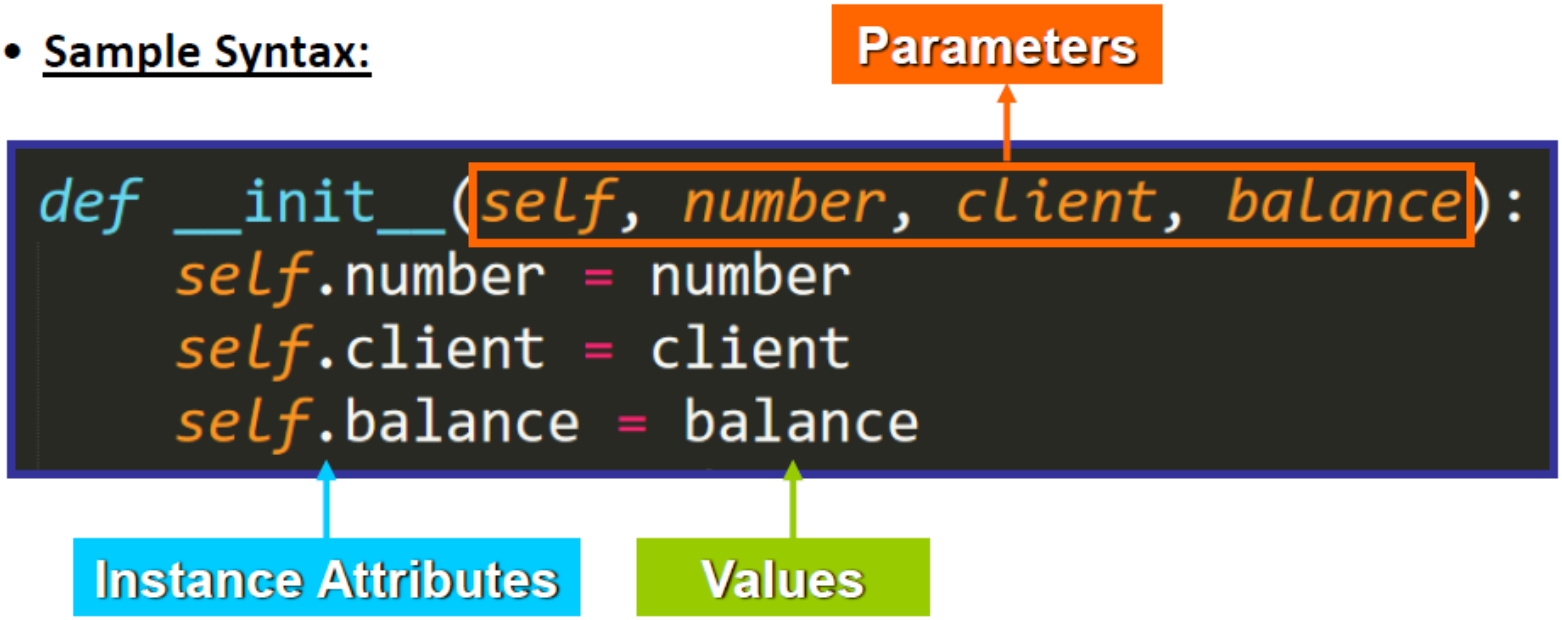
- สิ่งที่สร้างขึ้นมาจาก ต้นแบบ หรือ คลาส อาจเรียกว่า object ก็ได้ แต่ต่อไปจะขอเรียกว่า Instance เพราะคำว่า object อาจหมายถึงตัว class เองก็ได้
- การสร้าง instance ก่อนหน้า จะเห็นว่าเริ่มด้วยการสร้างคลาสก่อน จากนั้นจึงกำหนดค่าให้กับ attribute ของคลาส ซึ่งหาก attribute มีจำนวนมาก ก็จะไม่สะดวก
- ดังนั้นเพื่อความสะดวกจึงได้สร้างส่วน constructor ขึ้นมา เพื่อทำหน้าที่สร้าง Instance
- Constructor มีลักษณะคล้ายกับฟังก์ชัน แต่มีความพิเศษ คือ จะเรียกขึ้นมาทำงานโดยอัตโนมัติ เมื่อมีการประกาศคลาส

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

stu1 = Student('001' ,'John')
stu2 =Student('002', 'Peter')

print(stu1.id)
print(stu2.id)
```

Class

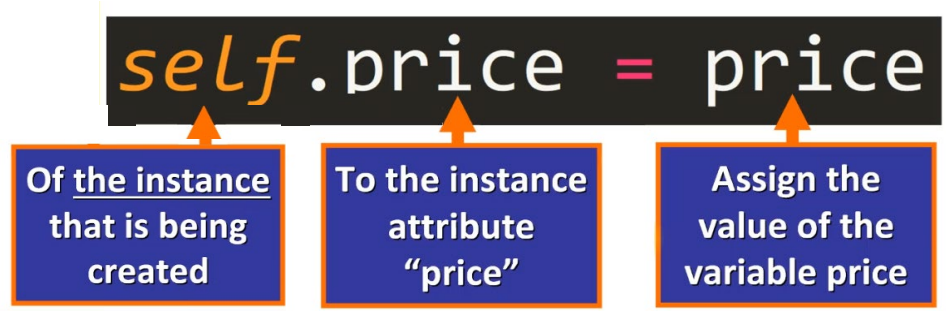
- constructor มีส่วนประกอบดังนี้
  - parameters เป็นข้อมูลที่ส่งมาเป็นค่าเริ่มต้นของ attribute ของ instance
  - instance attribute เป็นข้อมูลของคลาส อาจจะไม่เท่ากับ parameters ก็ได้
  - values เป็นค่าของ parameter ที่จะมาเป็นค่าเริ่มต้นให้กับ instance attribute

- **Sample Syntax:**



Class

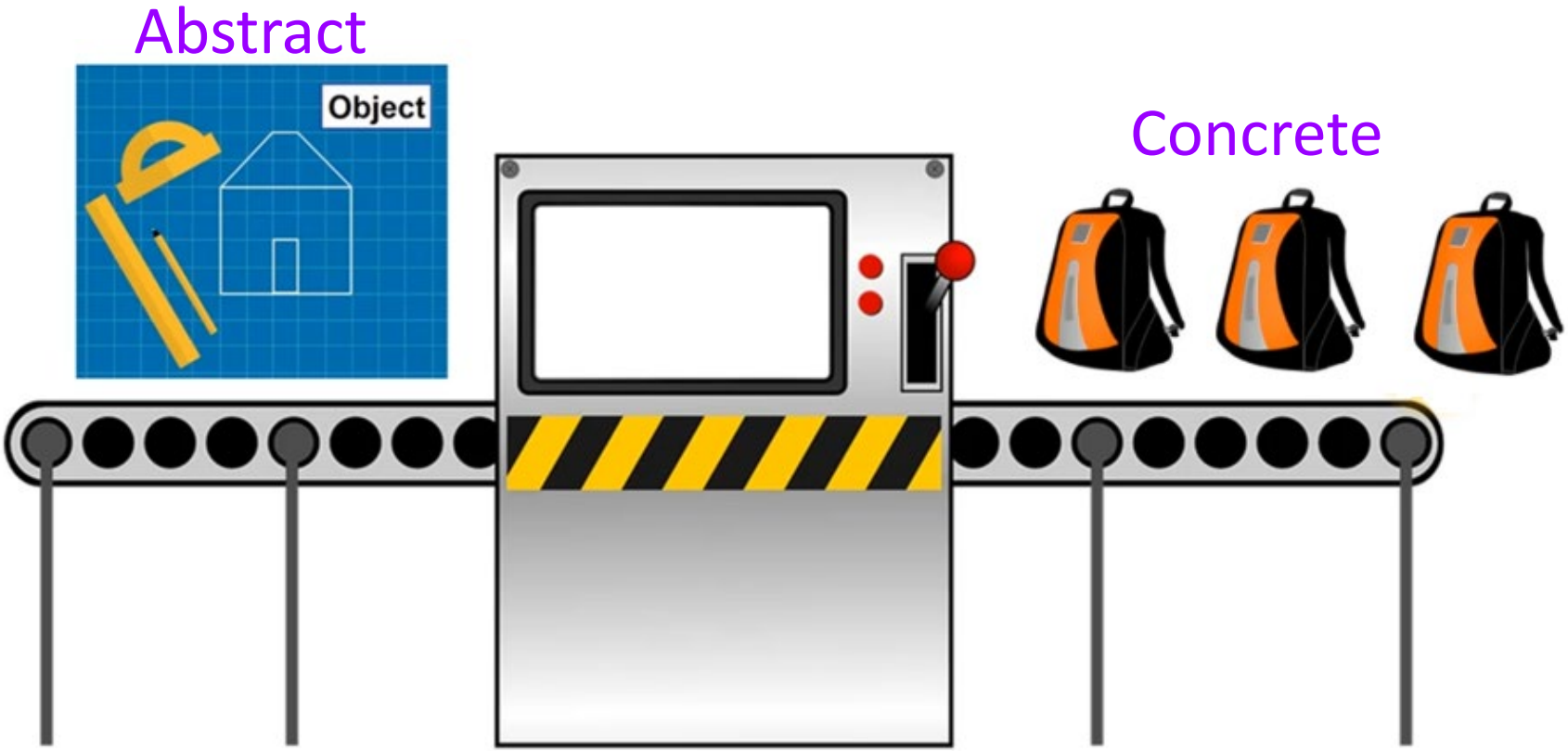
- รูปแบบทั่วไปของการกำหนดค่าให้กับ attribute มีดังนี้



- price ทั้ง 2 ตัวต่างกัน ฝั่งขวาคือ พารามิเตอร์ที่ส่งเข้ามา ฝั่งซ้ายคือ instance attribute
- attribute ของ คลาส จะขึ้นต้นด้วย self เสมอ (ซึ่งจะต่างจากการกำหนดตัวแปรทั่วไป) โดย self จะหมายถึง address ของ instance นั้น ดังนั้น stu1.id จึงไม่เท่ากับ stu2.id เพราะอยู่คนละตำแหน่ง แม้จะมีชื่อเดียวกัน

Instance

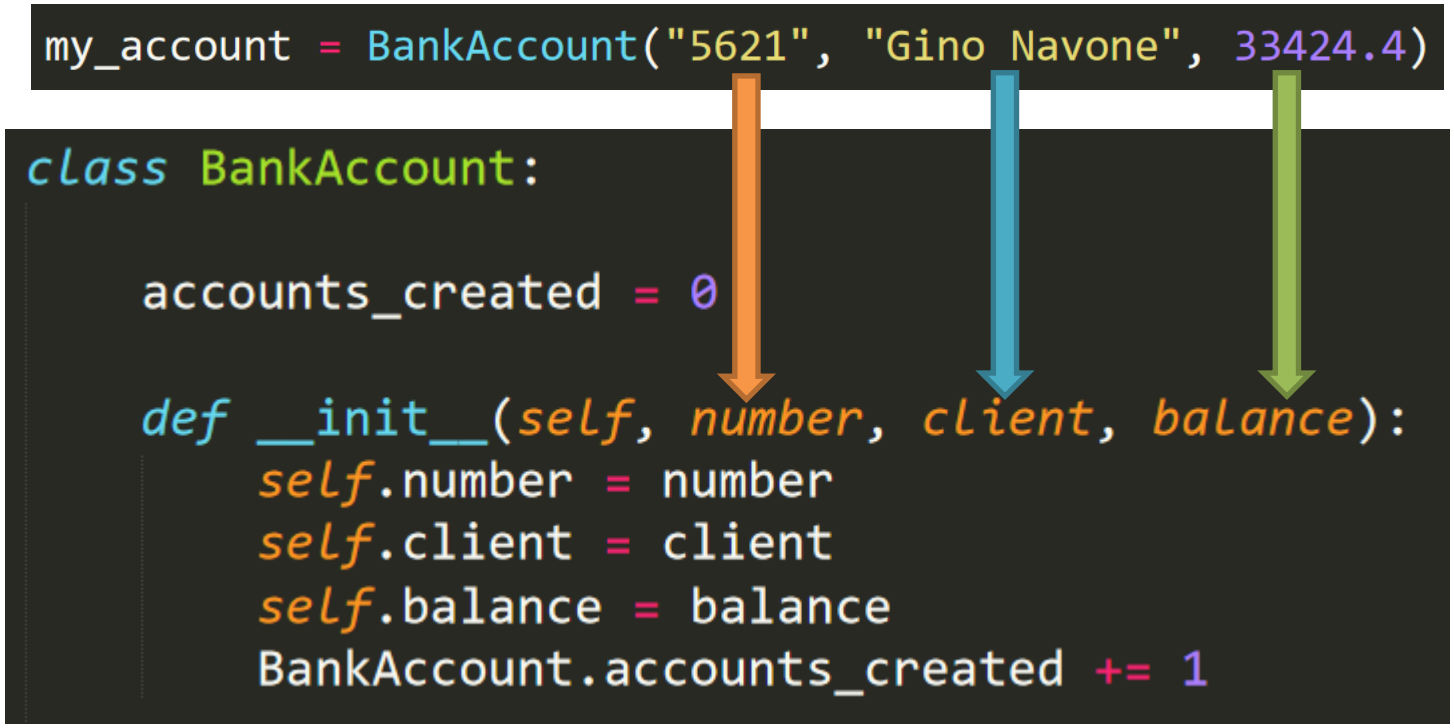
- สิ่งที่สร้างมาจาก Class จะเรียกว่า Instance ซึ่งจะมีที่อยู่แน่นอนในหน่วยความจำ ของแต่ละ Instance แยกกันออกไป ดังนั้นแต่ละ Instance จึงเป็นตัวของตัวเอง เพียงแต่มาจากต้นแบบเดียวกัน



Instance

- รูปแบบทั่วไปของการสร้าง instance คือ

```
<variable> = <ClassName>(<arguments>)
```



Instance

- แต่ละ instance จะมีที่อยู่แยกกันในหน่วยความจำ แม้จะสร้างจากคลาสเดียวกัน แต่เมื่อสร้างขึ้นมาแล้ว จะเป็นข้อมูลที่แยกกันโดยเด็ดขาด

John	Peter
➢ id = '001' ➢ name = 'John'	➢ id = '002' ➢ name = 'Peter'

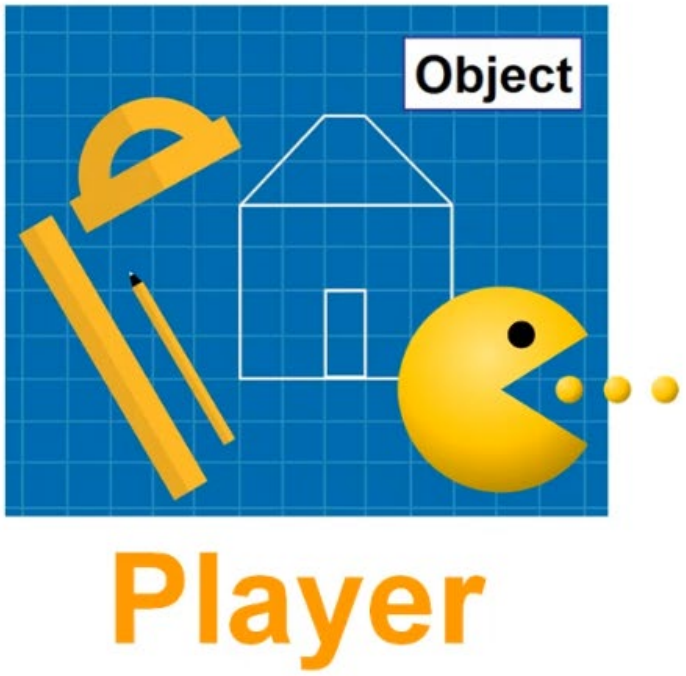
- หาก print ตำแหน่งของข้อมูลในแต่ละ instance ออกมาจะพบว่าไม่เท่ากัน

```
print(id(stu1))
print(id(stu2))
print(id(stu1.id))
print(id(stu2.id))
```

```
140003980820288
140003980820192
140003980419312
140003980466032
```

Instance

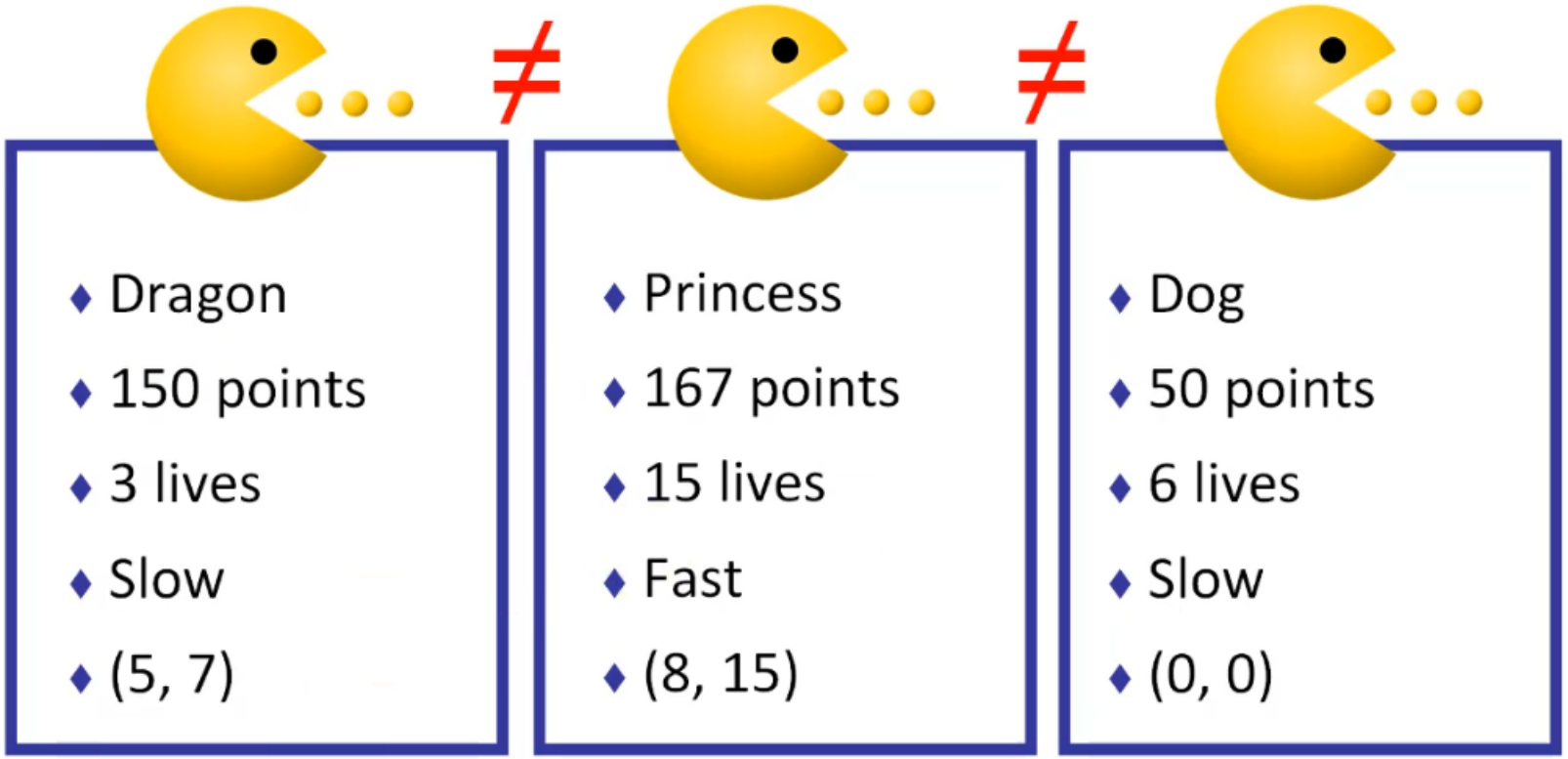
- อีกตัวอย่างของ instance



- Sprite
- Score
- Number of lives
- Speed
- X-coordinate
- Y-coordinate

Instance

- ในตัวละครของเกม เมื่อมีการเปลี่ยนตำแหน่ง หรือ เปลี่ยนความเร็ว ก็จะมีผลเฉพาะ Instance นั้น



Instance

- ข้อผิดพลาดที่มักเกิดขึ้นกับ __init__()
  - ลืมเขียน def
  - ใช้เครื่องหมาย _ แค่ อันเดียว เช่น _init_()
  - ลืมเขียน self ใน parameter แม้ใน object ที่ไม่มี parameter เลยก็ต้องมี self เช่น
    - def __init__(width, height):
  - ลืมเขียน self.<attribute> ในการกำหนด instance attributes
  - PEP8 Style ต้องวรรค 1 เคาะระหว่าง parameter เช่น
    - def __init__(self, name, age):

การเข้าถึง Instance Attribute

- เมื่อสร้าง Instance แล้ว หากต้องการเข้าถึงข้อมูล (attribute) ในแต่ละ Instance สามารถทำได้โดยใช้รูปแบบที่เรียกว่า dot notation
- โดยการอ้างถึงชื่อของ Instance แล้วตามด้วย . จากนั้นจึงเป็นชื่อของ attribute ตามตัวอย่าง

Modify/Update Instance Attribute

- การแก้ไขหรือเปลี่ยนแปลง Instance attribute ถือเป็นการเปลี่ยนข้อมูล (state) ของ Instance
- วิธีการ คือ กำหนดค่าทับเข้าไปใน attribute โดยใช้ dot notation เช่นกัน

```
<object>.<attribute> = <new_value>
```

- สำหรับกรณีที่ใช้ภายใน Class เดียวกันสามารถใช้ self แทนชื่อของ instance ได้

```
self.<attribute> = <new_value>
```

ความสัมพันธ์ระหว่างคลาส

- class สามารถสร้างความสัมพันธ์ได้ เช่น ความสัมพันธ์ระหว่าง นักศึกษา กับ รายวิชา
- จากโปรแกรมมีการสร้างคลาส Subject ซึ่งเก็บข้อมูลรายวิชา และนักศึกษาที่ลงเรียนในรายวิชานั้น
- ในการ append student เข้าไปใน student_list จะเป็นการ append ทั้ง instance

```
[<__main__.Student object at 0
x7f663fc03fd0>, <__main__.Stud
ent object at 0x7f663fc03ee0>]
```

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

class Subject:
    def __init__(self, sub_id, sub_name):
        self.sub_id = sub_id
        self.sub_name = sub_name
        self.student_list = []

stu1 = Student('001' ,'John')
stu2 = Student('002', 'Peter')
sub1 = Subject('01076140', 'Calculus')

sub1.student_list.append(stu1)
sub1.student_list.append(stu2)
print(sub1.student_list)
```

Activity #2 :

- ให้เขียนโปรแกรม เพื่อสร้างคลาสต่อไปนี้
  - นักศึกษา (Student) โดยมี attribute : stu_id, name
  - รายวิชา (Subject) โดยมี attribute : subject_id, subject_name, section, credit
  - ผู้สอน (Teacher) โดยมี attribute : teacher_id, teacher_name
- ให้สร้าง Instance ของทุกคลาส และ สร้างความสัมพันธ์ (สามารถเพิ่ม attribute ได้)
  - ให้สร้าง instance ของนักศึกษา 5 คน
  - ให้สร้าง instance ของอาจารย์ 2 คน
  - ให้สร้าง instance ของวิชา oop ใน 2 section โดยแต่ละ section มีผู้สอนคนละคน และแต่ละ section มีคนเรียนอย่างน้อย 2 คน
- ให้เขียนโปรแกรมเพื่อค้นหาว่า เมื่อใส่ รหัสผู้สอน แล้วสามารถบอกได้ว่ามี นศ. คนไหนบ้างที่เรียนกับผู้สอนนี้ และ เมื่อใส่ รหัส นศ. แล้วบอกว่าเรียนวิชาอะไรบ้าง

Activity #3 :

- ให้แก้ไขคลาส student โดยเพิ่มข้อมูล พี่รหัส (student_menter)
- ให้สร้าง instance ของ นศ. 4 คน เช่น a เป็นปี 1, b เป็นปี 2 และเป็นพี่รหัสของ a c เป็นปี 3 และเป็นพี่รหัสของ b ส่วน d เป็นปี 4 และเป็นพี่รหัสของ c
- ให้เขียนโปรแกรมเพื่อตอบคำถามว่า
  - Student x มีพี่รหัส เป็นใครบ้าง
  - Student x และ student y เป็น พี่ หรือ น้องรหัส กันหรือไม่

For your attention