01076105, 01075106

Object Oriented Programming

Object Oriented Programming Project

Introduction to Object Oriented Concepts

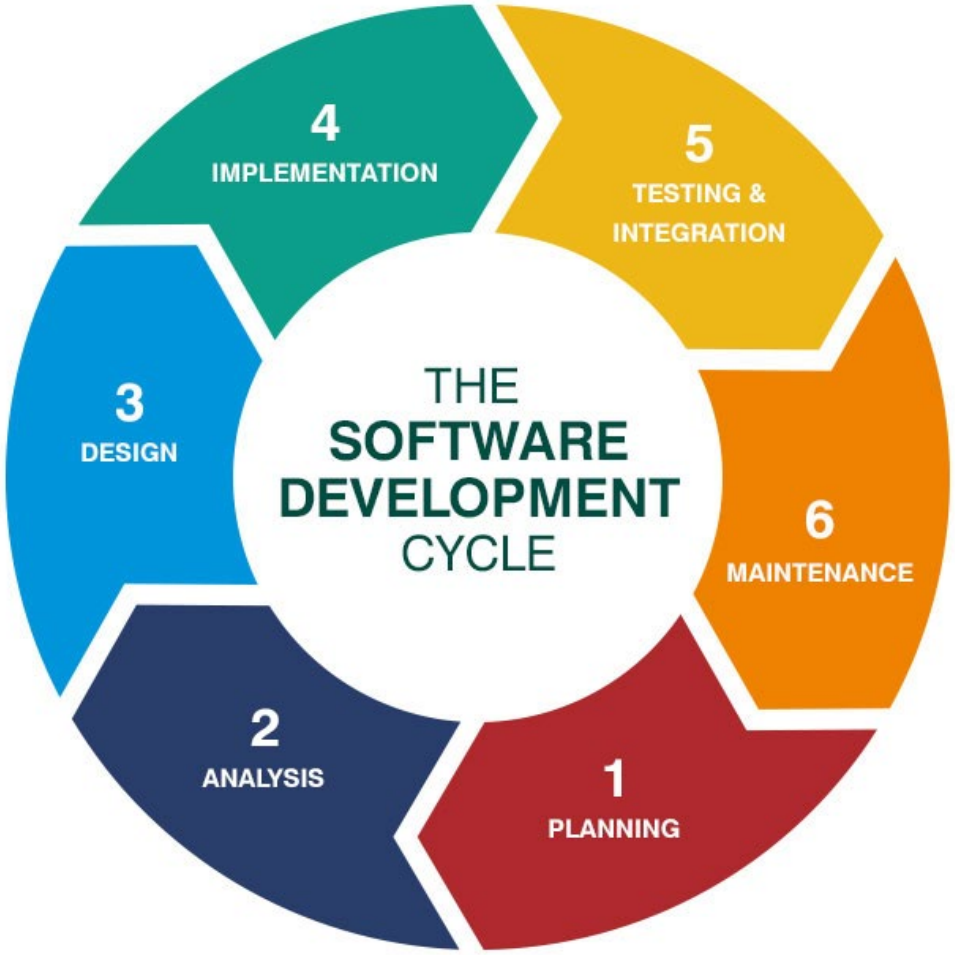
Introducing object-oriented

- Object คือ ชุดของข้อมูลและพฤติกรรมที่เกี่ยวข้อง (collection of data and associated behaviors)
- Object-oriented analysis (OOA) คือ กระบวนการที่มองไปที่ ปัญหา หรือ ระบบ หรือ งาน (ที่ต้องการใช้คอมพิวเตอร์) และระบุ Object ที่อยู่ในนั้น และความสัมพันธ์ระหว่าง Object
- ผลลัพธ์ ของ OOA คือ ชุดของความต้องการ (Requirement) ของระบบ หรือ งานนั้นๆ เช่น ถ้าเป็นเว็บขายสินค้า ก็จะเป็น
- ค้นหา เปรียบเทียบ และ สั่ง สินค้า (สีฟ้าหมายถึงการทำงาน สีแดงคือ Object) (ขั้นตอนนี้เป็นการกำหนดว่า ต้องการ อะไร)

Object-oriented design (OOD)

- OOD เป็นกระบวนการที่จะเปลี่ยนจาก Requirement ให้เป็นรายละเอียดของการพัฒนาโปรแกรม เช่น
  - การตั้งชื่อ Object
  - การกำหนดพฤติกรรม Behaviors ที่ Object จะทำงานกับ Object อื่นๆ อย่างไร
- ขั้นตอนนี้เป็นการกำหนดว่า ระบบจะทำงาน อย่างไร (How)
- ผลลัพธ์ของการออกแบบ คือ ข้อกำหนดในการ Implement
- จากนั้นจึงใช้ภาษาโปรแกรมที่เป็น Object-Oriented Programming พัฒนา

Software Development Life Cycle (SDLC)



Procedural vs OO Programming

- ถ้าเรามอง คน สัตว์ สิ่งของรอบตัว เราสามารถกำหนดคุณลักษณะของสิ่งของนั้นได้ 2 ประเภท
  - Data (Attribute) เช่น ถ้าเป็นมนุษย์ ก็คือ สีผิว อายุ ความสูง นน. ฯลฯ
  - Behaviors เช่น เดินได้ พูดได้ หายใจได้ ฯลฯ
- ในเบื้องต้นเราสามารถบอกได้ว่า Object is an entity that contains *both* attribute and behaviors.
- คำว่า both เป็นสิ่งสำคัญที่ทำหน้าที่แยกระหว่าง OO Programming กับภาษา Programming อื่นๆ

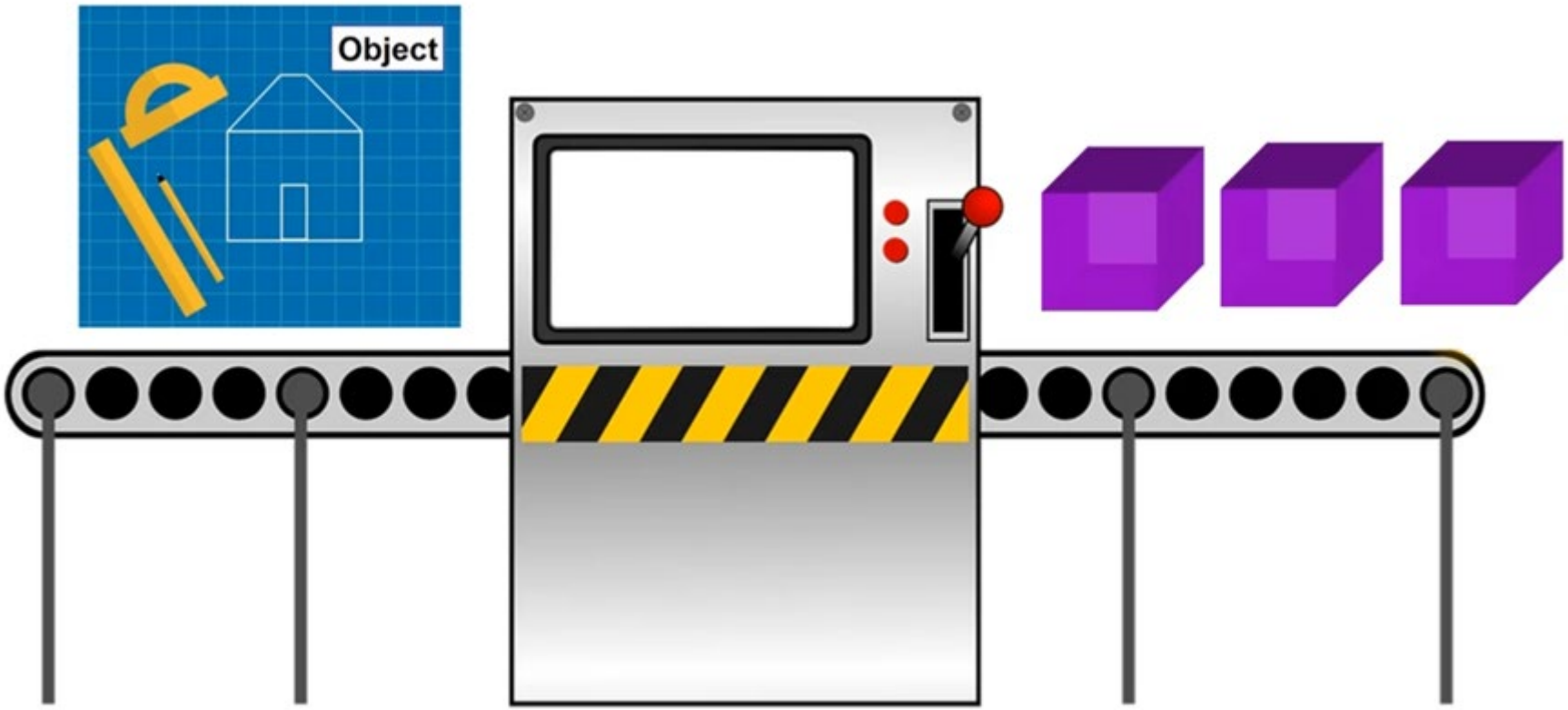
Procedural vs OO Programming

- ในภาษาที่เป็น procedural การทำงานจะเหมือนกับรูป คือ โปรแกรมที่เขียนจะเสมือนกับ Blackbox ที่รับข้อมูล input และแปลงเป็น output

- แต่ใน OO Programming นั้น จะต่างออกไป เพราะข้อมูลและการทำงานจะรวมอยู่ใน object เดียวกัน

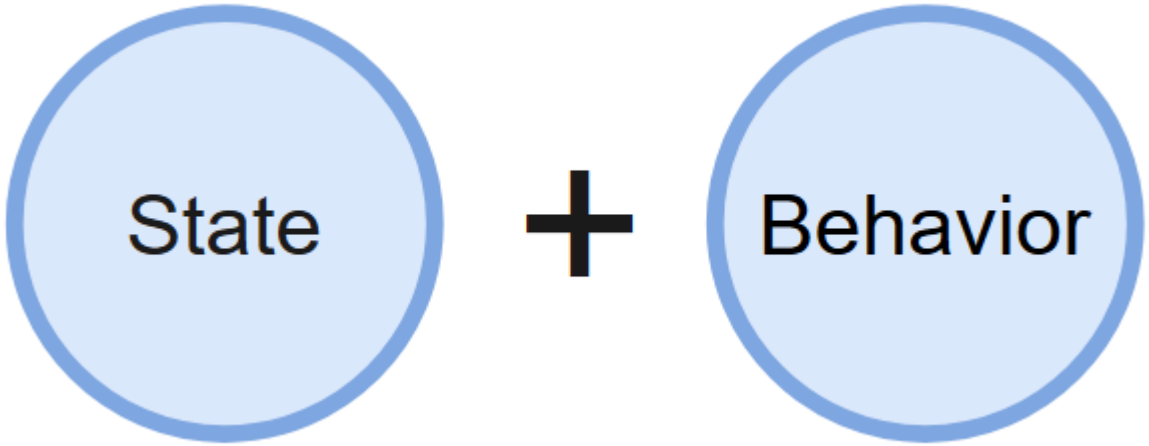
Objects and Classes

- Class จะเหมือนกับพิมพ์เขียวที่ใช้สร้าง Objects



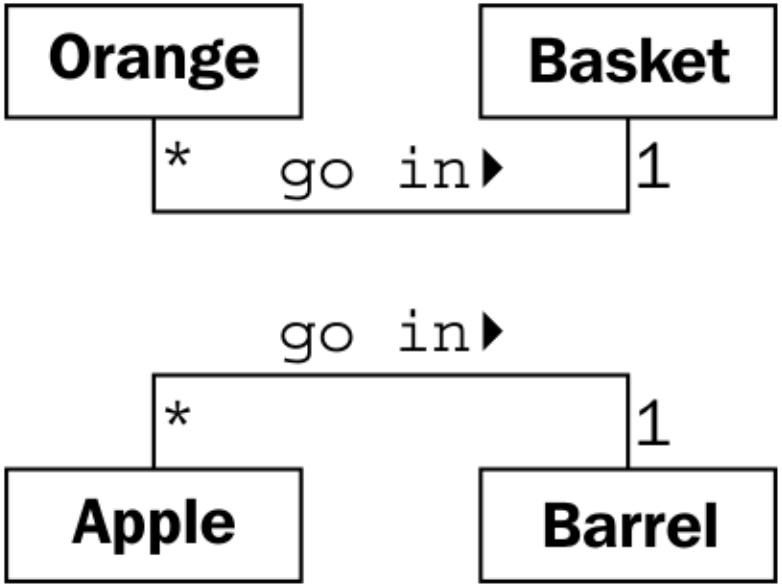
Objects and Classes

- Object คือ ชุดของข้อมูล และ พฤติกรรมที่เกี่ยวข้อง
- ข้อมูลที่บรรจุใน Object อาจมองเป็นสถานะของข้อมูลก็ได้



Objects and Classes

- การทำงานของ OOP คือ การส่งข้อมูล (message) เข้าไปใน Object เช่น

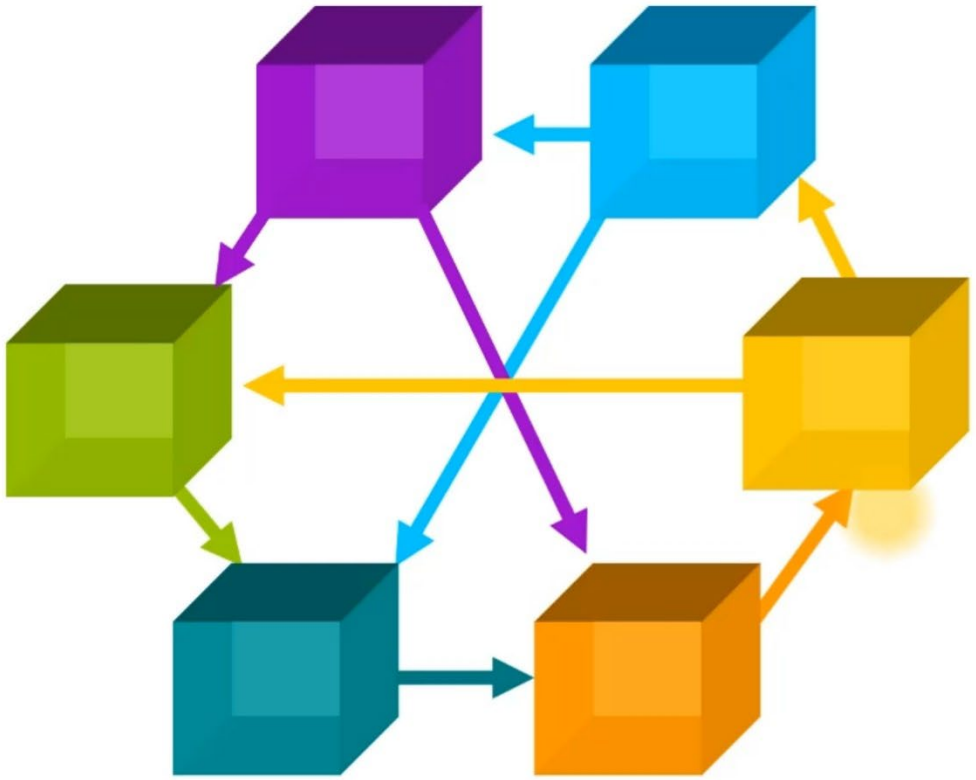


Objects and Classes

- เมื่อ Object ได้รับ message จะตอบสนอง (response) โดยการทำงานอย่างใดอย่างหนึ่ง
- จากตัวอย่างใน slide ก่อนหน้านี้ ใน Object **Basket** จะมีการนับว่าได้รับส้มเพิ่มขึ้นมา 1 ผล (Action)
- และใน Object ก็จะเก็บ **State** ของตัวเอง ผ่านข้อมูลหรือ Attribute เช่น ตัวอย่างก่อนหน้านี้ ใน Object **Basket** จะมีข้อมูลจำนวนส้มเก็บเอาไว้อยู่

Objects and Classes

- ดังนั้น OOP ก็คือ การทำงานของแต่ละ Object ที่มีการส่งข้อมูล และ ส่งให้เกิดพฤติกรรมที่ต้องการนั่นเอง



Advantage of OOP

- **Modularity** เนื่องจากแต่ละ Object มีการทำงานที่จบภายในตัวเอง ดังนั้น หากมีการเปลี่ยนแปลงใดๆ (ที่ไม่เปลี่ยน parameter) ก็จะไม่มีผลกระทบกับส่วนอื่นๆ ของโปรแกรม ทำให้โปรแกรม **ง่ายต่อการดูแล** แยกทดสอบได้
- **Reusability** การเขียนแบบ Object ทำให้การนำ code ไปใช้ซ้ำ ทำได้ง่ายกว่า เช่น ข้อมูลนักศึกษา แทนที่จะเก็บลงในตัวแปรจำนวนมาก ก็สามารถเก็บใน Object ซึ่งเป็นผลให้ **พัฒนาได้เร็ว** และ **มีต้นทุนที่น้อยกว่า**
- **Extensibility** โดยการใช้ Object เราสามารถรวมข้อมูลที่มีลักษณะคล้ายกัน แต่แตกต่างกันบางอย่างไว้ด้วยกันได้ แล้วเพิ่ม Attribute ใหม่ๆ หรือ Behavior ใหม่ๆ เข้าไปเพิ่ม

Quiz

- Q1: Object-Oriented Programming คือ
  1. ปรัชญาของการเขียนโปรแกรม
  2. กระบวนทัศน์ (Paradigm) ของการเขียนโปรแกรม
  3. แนวคิด (Idea) ของการเขียนโปรแกรม

Quiz

- Q2: Object-Oriented Programming organizes software design around…
  1. Functions
  2. Statements
  3. Objects

Quiz

- Q3: In Object-Oriented Programming, we define classes that act like blueprints. In these classes, we define the _____ and the _____ of the objects.

  1. Characteristics, Color
  2. State, Behavior
  3. Action, Language

Quiz

- Q 4: True or False:

  Programming paradigms are mutually exclusive, so you can't use different programming paradigms in the same program.

  1. True
  2. False

Quiz

- Q 5: Select the main advantage(s) of Object-Oriented Programming:

  (A) Modularity

  (B) Extensibility

  (C) Reusability

1. A
2. A and B
3. B and C
4. A and B and C

Quiz

- 6: Select the **true** statement:
  1. Objects ไม่สามารถเพิ่ม attribute และ behavior ได้
  2. Object Oriented Programming สามารถพัฒนาได้เร็วกว่า ซอฟต์แวร์มีคุณภาพสูงกว่า และ มีต้นทุนน้อยกว่า
  3. Objects ไม่สามารถนำมาใช้ใหม่ (reuse) ในโครงการอื่นๆ ได้

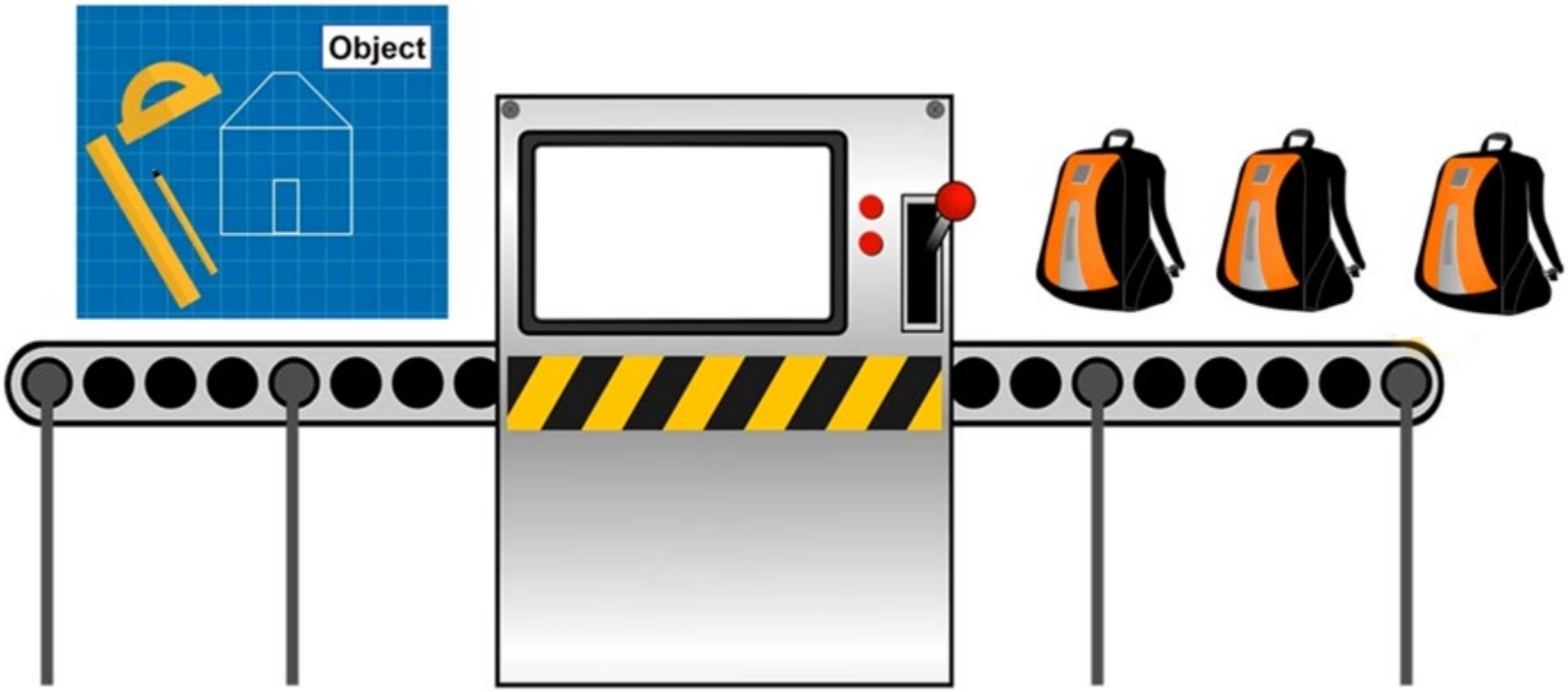
Class

A **blueprint** for creating objects.

- ทำหน้าที่คล้าย “พิมพ์เขียว” ของ Object โดยอธิบายถึง State (data) และ Behavior ของ Object ที่จะสร้างขึ้นมา
- ใช้แทน real world objects หรือ ส่วนของระบบ หรือ โปรแกรม เช่น สมุดบัญชี, พนักงาน, ลูกค้า หรือ ผลิตภัณฑ์ เป็นต้น

Class

- Class เป็นต้นแบบในการผลิต Object



Class

- ในภาษา Python เราจะใช้ Pascal Case (หรือ Upper camel case) ในการกำหนด Class
- การเขียนในแบบ Pascal Case คือ ให้ขึ้นต้นตัวแรกด้วยอักษรตัวใหญ่ ของแต่ละคำ เช่น
  - House
  - BackAccount
  - SchoolBackpack
  - LinkedList

Class

- โดยทั่วไป Class จะใช้เป็นคำนาม

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

- การกำหนด Class จะเขียนคำว่า class เป็นตัวเล็ก และใช้ชื่อ class เป็น Pascal Case และปิดท้ายด้วย : จากนั้นในเนื้อในของ class จะใช้ 1 indent

Class

- ในการเขียนโปรแกรม เราจะได้โจทย์ ซึ่งในที่นี้จะเรียกว่า Problem Statement
- สมมติได้รับโจทย์ดังนี้ : **ในร้านแห่งหนึ่ง ขาย fast food ประกอบด้วย pizza, burger และ hot dog นอกจากนั้นยังขาย น้ำเปล่า น้ำอัดลม และมันฝรั่งทอด ลูกค้าอาจจะซื้อสินค้า 1 ชิ้น หรือมากกว่าก็ได้ โดยหากซื้อมากกว่า 1 ชิ้น จะได้รับส่วนลด 20% ร้านค้าแห่งนี้มีพนักงาน 5 คน**
- **โจทย์ข้างต้น ให้หาว่าอะไรเป็น Class บ้าง**

Class

- อาจมี Class ดังนี้
- ในร้านแห่งหนึ่ง ขาย fast food ประกอบด้วย pizza, burger และ hot dog นอกจากนั้นยังขาย น้ำเปล่า น้ำอัดลม และมันฝรั่งทอด ลูกค้าอาจจะซื้อสินค้า 1 ชิ้น หรือมากกว่าก็ได้ โดยหากซื้อมากกว่า 1 ชิ้น จะได้รับส่วนลด 20% ร้านค้าแห่งนี้มีพนักงาน 5 คน
- มีแนวทางในการกำหนด Class อย่างไร?

Class

- Problem Statement
- ในร้านขายอุปกรณ์อิเล็กทรอนิคส์แห่งหนึ่ง ขาย Notebook, Desktop PC, Smartphone, Flash Drive, และหูฟัง นอกจากนั้นยังมีอุปกรณ์ประกอบอื่นๆ ร้านนี้มีพนักงาน 10 คน และ ผู้จัดการร้าน 1 คน ผู้ซื้อจะต้องมีการลงทะเบียนเมื่อมีการซื้อครั้งแรก
- จงหาว่าควรมี Object อะไรบ้าง

Class

- Problem Statement
- ในร้านขายอุปกรณ์อิเล็กทรอนิคส์แห่งหนึ่ง ขาย Notebook, Desktop PC, Smartphone, Flash Drive, และหูฟัง นอกจากนั้นยังมีอุปกรณ์ประกอบอื่นๆ ร้านนี้มีพนักงาน 10 คน และ ผู้จัดการร้าน 1 คน ผู้ซื้อจะต้องมีการลงทะเบียนเมื่อมีการซื้อครั้งแรก

Quiz

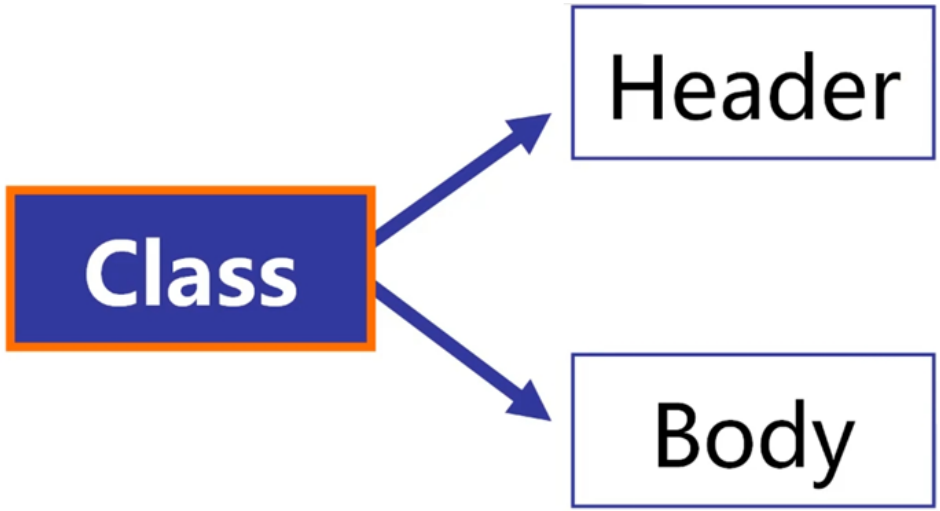
- Q 7: True or False:
  - Classes ทำหน้าที่คล้าย "blueprints" และสามารถใช้มันเป็นตัวแทนของ Object ใน Python program ได้
    - True
    - False

Quiz

- Q 8: ข้อใดผิด
  1. Class กำหนด state และ behavior ของ Object
  2. ชื่อของ Class เขียนในรูปแบบ Singular (เอกพจน์) เพราะแทนชนิดของ Object
  3. ชื่อของ Class ควรเป็นไปตามการตั้งชื่อแบบ snake_case ที่เขียนด้วยตัวเล็กและคั่นระหว่างคำด้วย _

Class in python

- Class ใน python ประกอบด้วย 2 ส่วน คือ Header และ Body
- Header ทำหน้าที่บอกข้อกำหนดของ Class
- Body บอกรายละเอียดภายใน Class



Class in python

- Header ประกอบด้วยชื่อ Class และบอกว่าสืบทอด (inherit) มาจากคลาสใด (ถ้ามี)
- รูปแบบการกำหนด ชื่อ Class มีดังนี้
  - ต้องมีคำว่า class (ตัวเล็ก)
  - ชื่อ class (เขียนในแบบ Pascal Case)
  - ปิดท้ายด้วย เครื่องหมาย :

```
class Backpack:
    pass
```

Class in python

- Class Body บรรจุรายละเอียดของ “blueprint” ซึ่งประกอบด้วย Attribute และ Behavior ของ Object
- ส่วนประกอบของ Class body มักจะประกอบด้วย
  - Class Attribute เป็นส่วนที่เก็บ Data ของ Class
  - __init__() เป็น Constructor คือ ส่วนที่จะถูกเรียกขึ้นมาทำงาน เมื่อมีการสร้าง Object ขึ้นมาจาก Class
  - Methods ทำหน้าที่อธิบาย behavior หรือ action ของ Class ที่จะมีในกรณีต่างๆ

```
class ClassName:

    # Class Attributes

    # __init__()

    # Methods
```

Class in python

- Exercise
  - กำหนดให้สถานสงเคราะห์สัตว์แห่งหนึ่ง ทำหน้าที่ดูแล สุนัข แมว นก กระรอก และ งู ในที่พักแห่งนี้มีเจ้าหน้าที่ 6 คน และมีอาสาสมัครจำนวนหนึ่ง โดยสถานสงเคราะห์มีรายได้จากการรับบริจาค โดยมีการบันทึกชื่อผู้บริจาค
  - ให้ระบุ Class และเขียน Class ที่จะใช้ในตัวอย่าง

Class in python

- Exercise
  - กำหนดให้สถานสงเคราะห์สัตว์แห่งหนึ่ง ทำหน้าที่ดูแล สุนัข แมว นก กระต่าย งู และ กิ้งก่า ในที่พักแห่งนี้มีเจ้าหน้าที่ 6 คน และมีอาสาสมัครจำนวนหนึ่ง โดยสถานสงเคราะห์มีรายได้จากการรับบริจาค โดยมีการบันทึกชื่อผู้บริจาค
  - ให้ระบุ Class และเขียน Class ที่จะใช้ในตัวอย่าง

Class in python

- เราสามารถแบ่ง Object ออกเป็น 2 ประเภท คือ สัตว์ และ มนุษย์

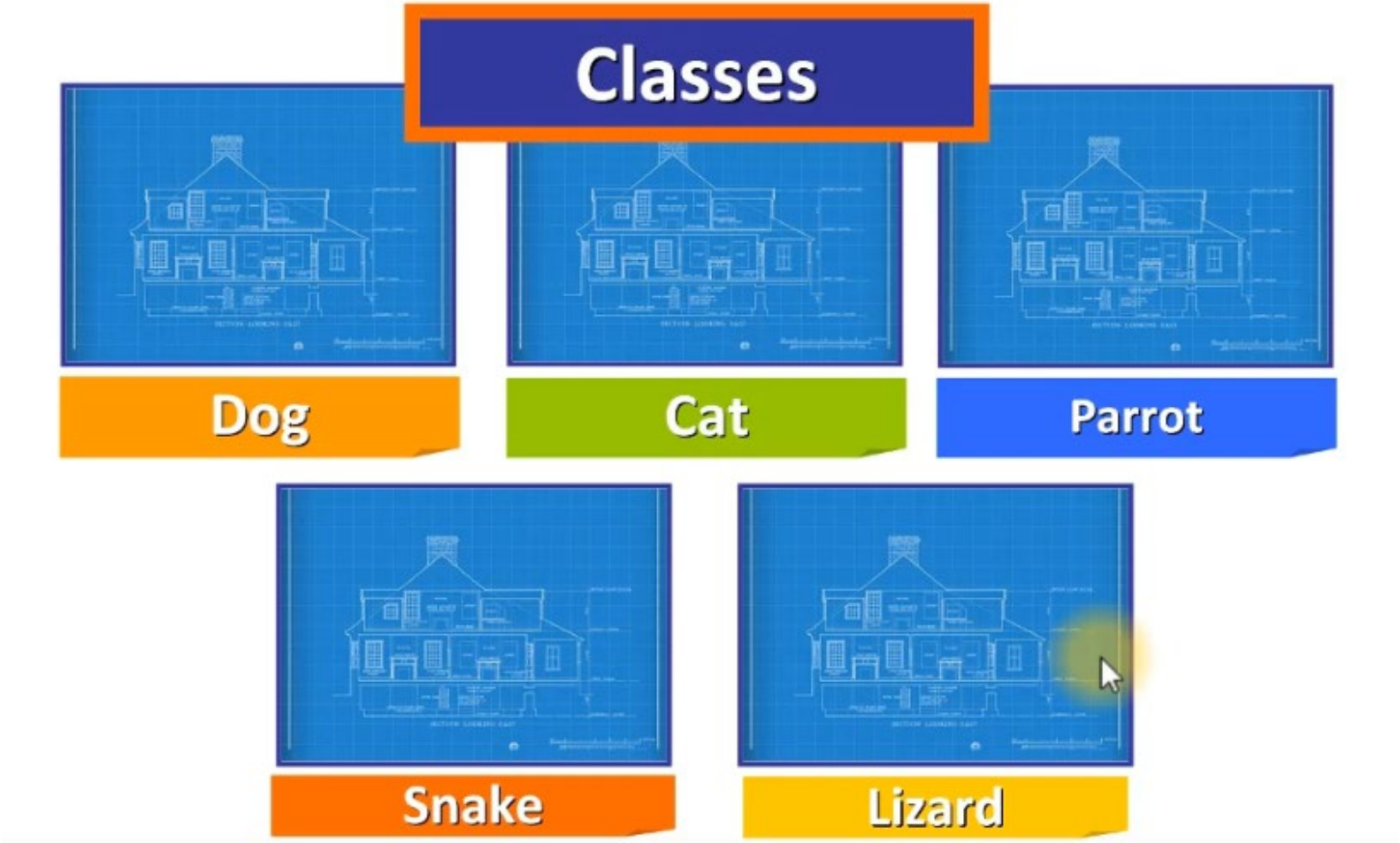
- **Animals:**
  - Dogs
  - Cats
  - Parrots
  - Lizards
  - Snakes

- **People:**
  - Employees
  - Volunteers
  - Donors

Class in python

- เราจะสร้าง Object ดังนี้ (เขียนในรูปแบบเอกพจน์)
  - กลุ่มสัตว์
    - Dog
    - Cat
    - Parrot
    - Snake
    - Lizard
  - กลุ่มมนุษย์
    - Employee
    - Volunteer
    - Donor

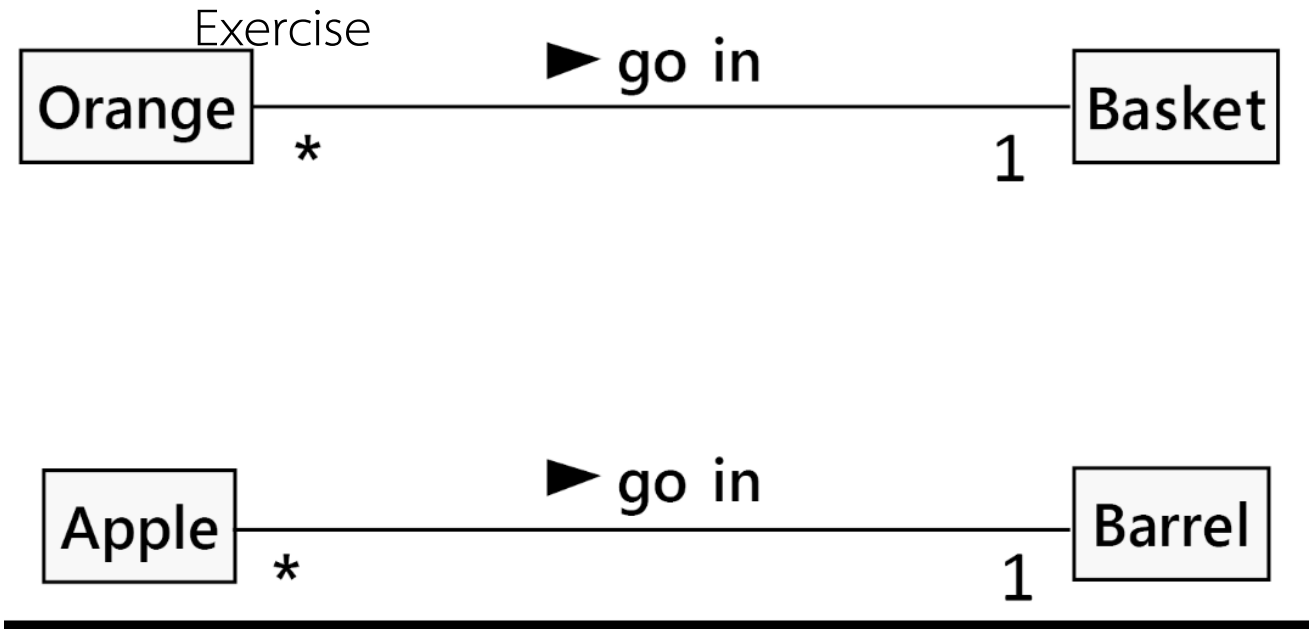
Class in python



Class in python

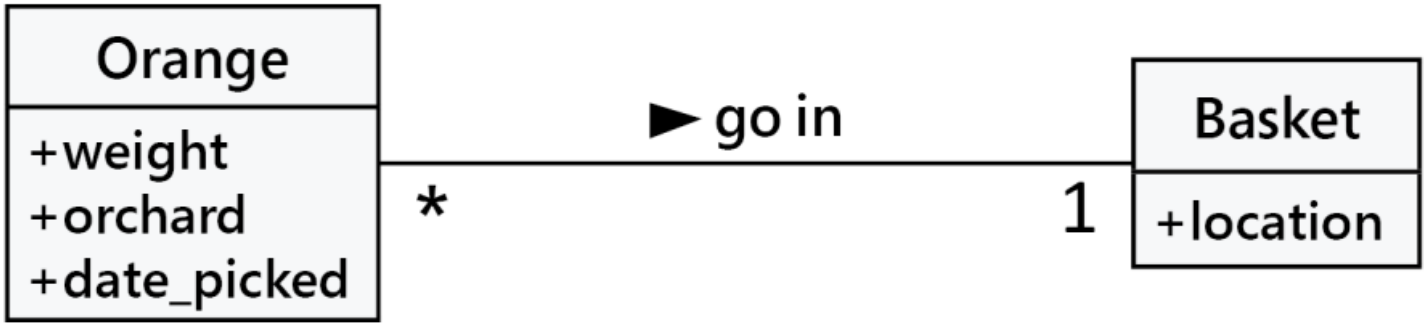
Class diagram

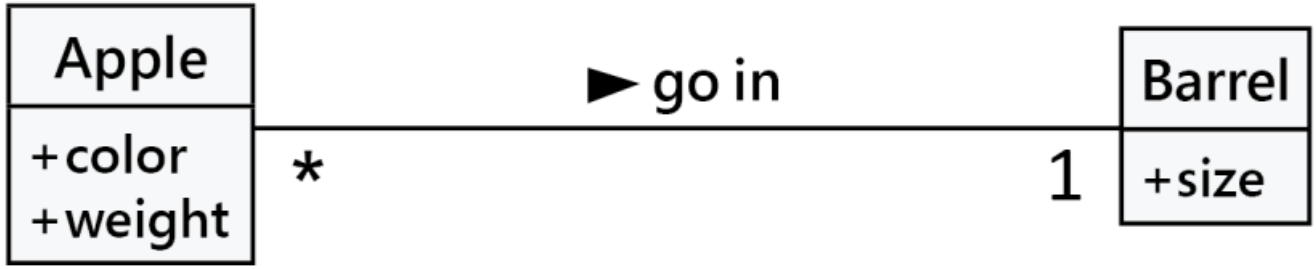
- ในกรณีที่มีหลาย Class เราสามารถแสดงความสัมพันธ์ระหว่าง Class โดยใช้ Class Diagram
- เช่น ส้ม เข้าไปอยู่ใน ตะกร้า และ แอปเปิล เข้าไปอยู่ใน ลัง (ถัง)
  (* หมายความว่าอาจมีหลาย Object)



Specifying attributes and behaviors

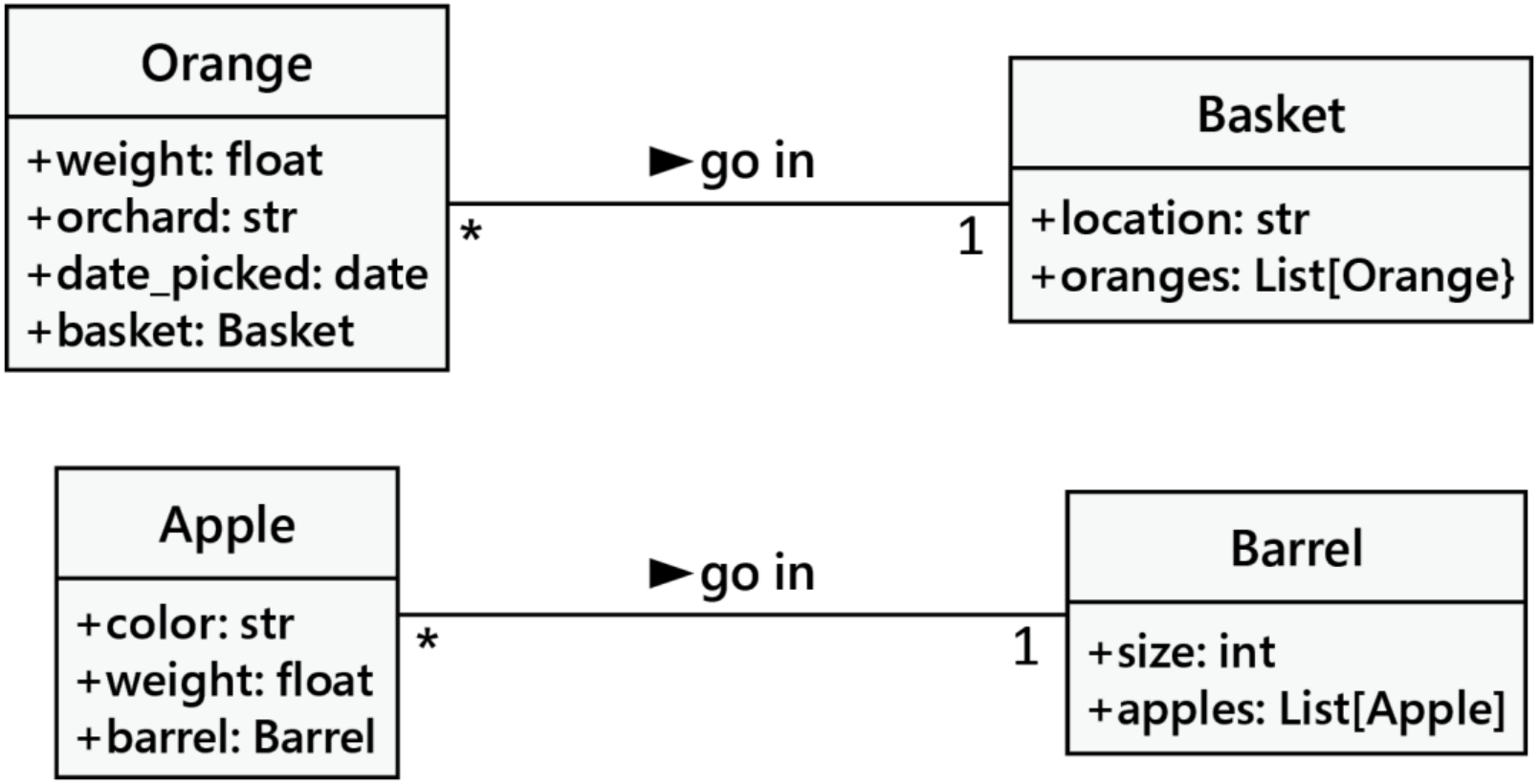
- หลังจากนั้นจะต้องระบุ attribute ของ Object ตามตัวอย่าง





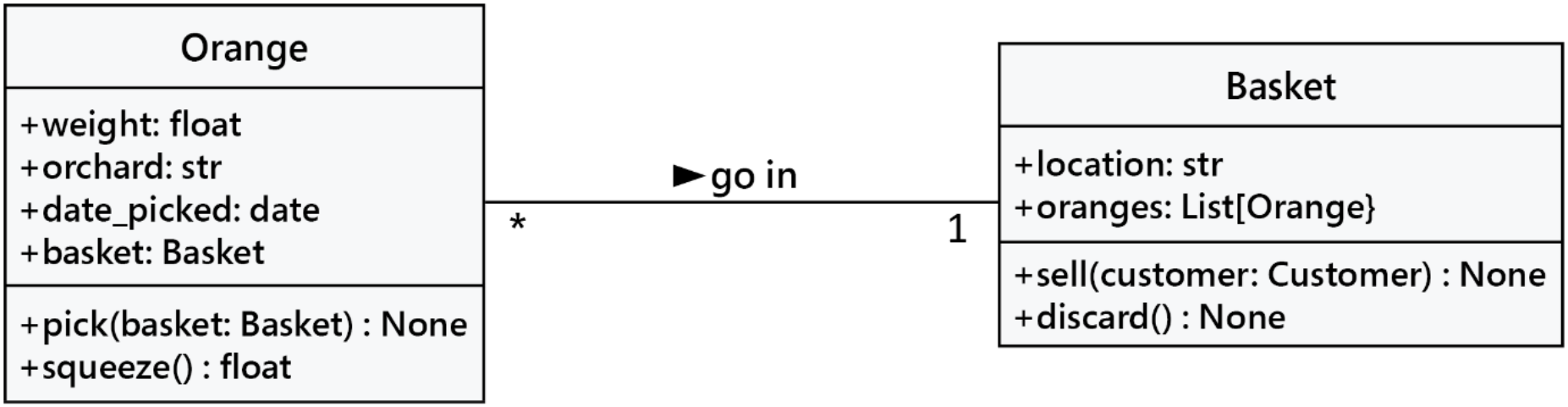
Specifying attributes and behaviors

- ขั้นตอนต่อไป คือ กำหนดชนิดข้อมูลให้กับ attribute ตามรูป



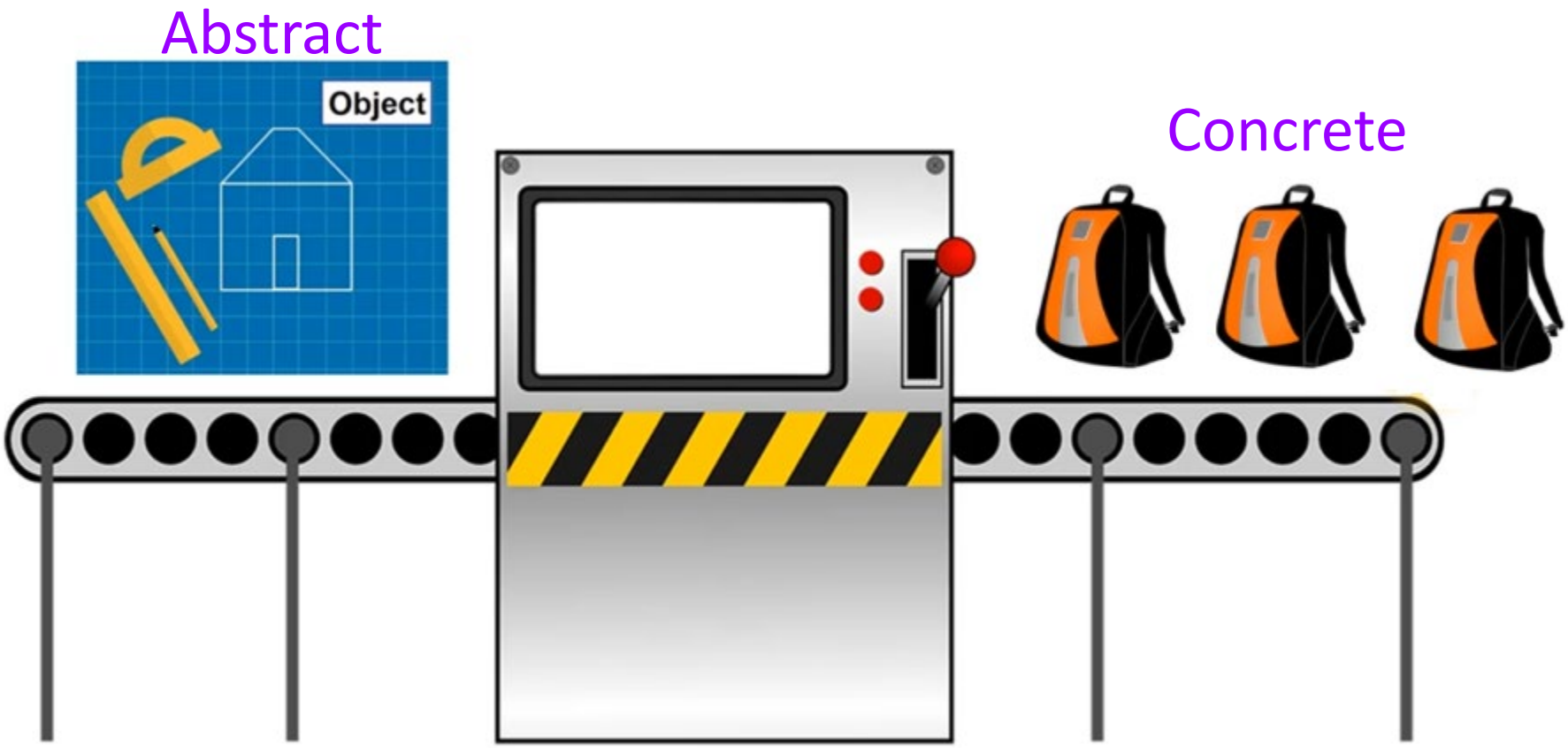
Specifying attributes and behaviors

- ขั้นตอนสุดท้ายในการกำหนด Class Diagram คือ การกำหนด method หรือ behavior ของ object นั้น



Instance

- สิ่งที่สร้างมาจาก Class เรียกว่า Instance (มีที่อยู่แน่นอนในหน่วยความจำ)



Instance

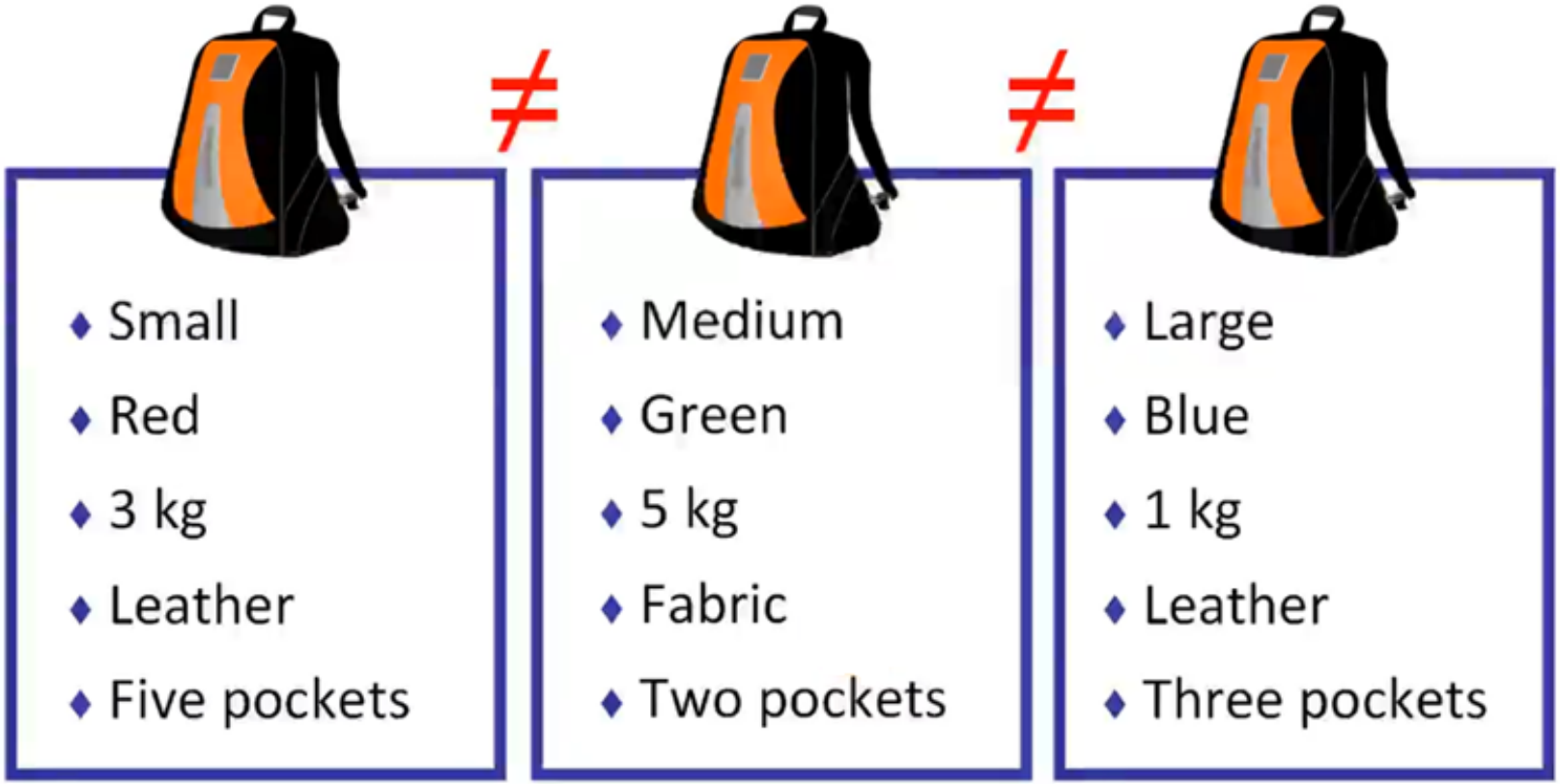
- Instance Attribute จะเป็นตัวแปรที่เป็นของแต่ละ Object แยกกันไปในแต่ละ Object เช่น กระเป๋า แต่ละใบอาจมี ขนาด สี น้ำหนัก วัสดุ ฯลฯ แตกต่างกันได้

Backpack

- Size
- Color
- Weight
- Material
- Number of pockets
- Number of zippers

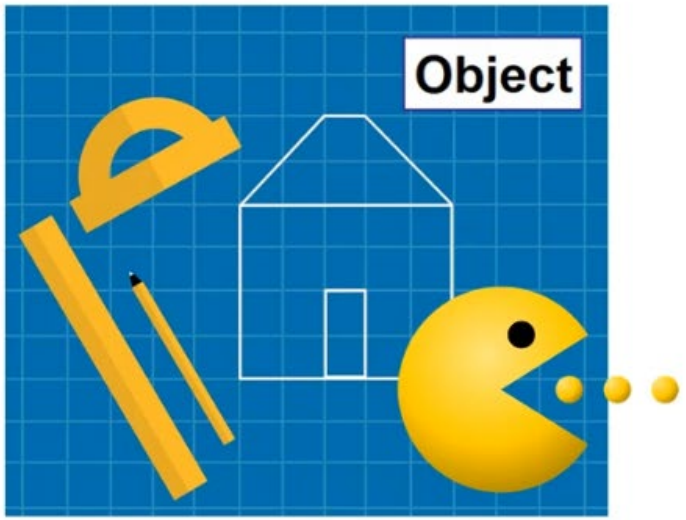
Instance

- แต่ละ Instance จะมีข้อมูลเฉพาะของตัวเอง แม้จะเป็น Class เดียวกัน



Instance

- อีกตัวอย่างของ Instance

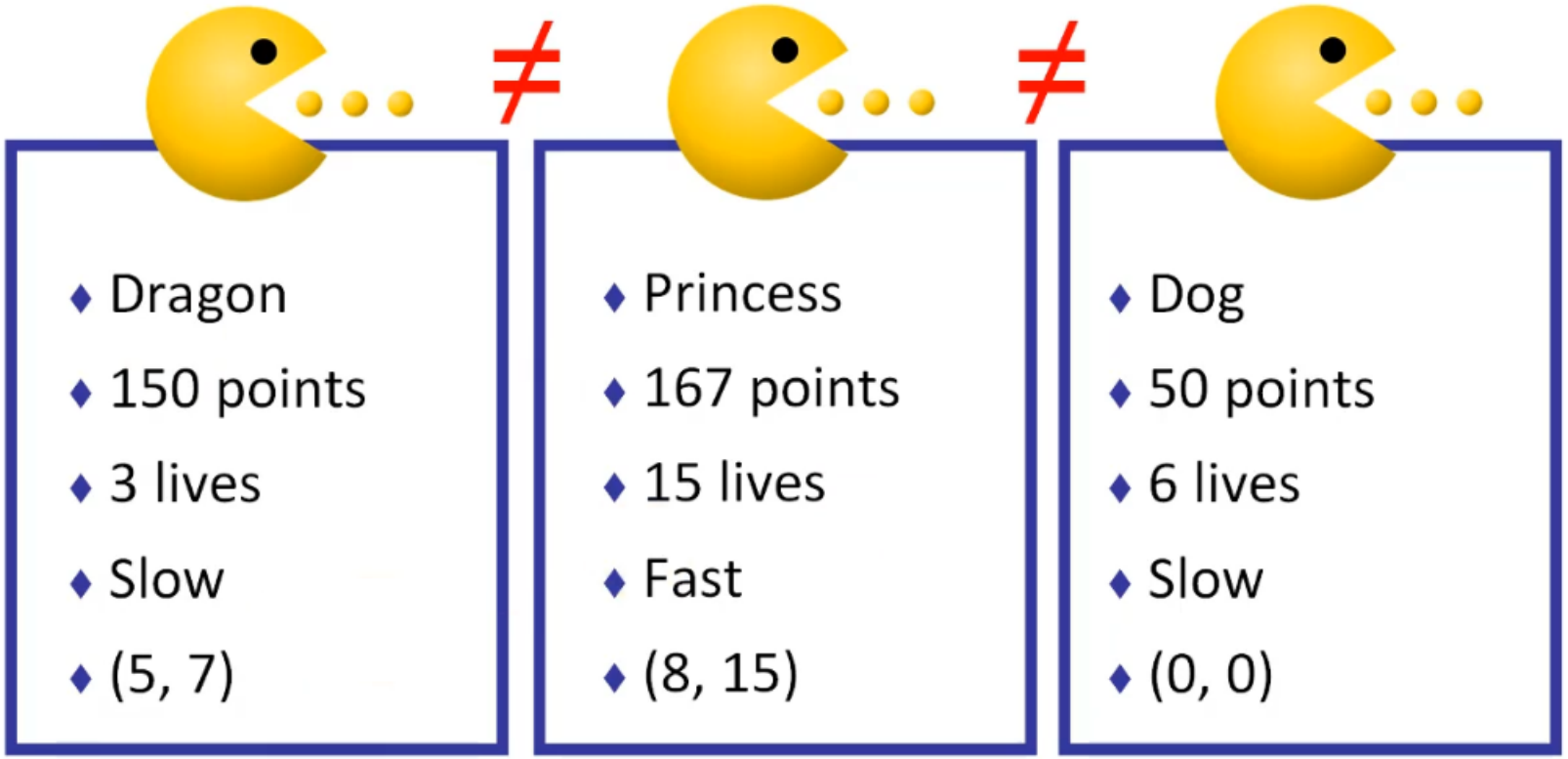


Player

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

Instance

- ในตัวละครของเกม เมื่อมีการเปลี่ยนตำแหน่ง หรือ เปลี่ยนความเร็ว ก็จะมีผลเฉพาะ Instance นั้น



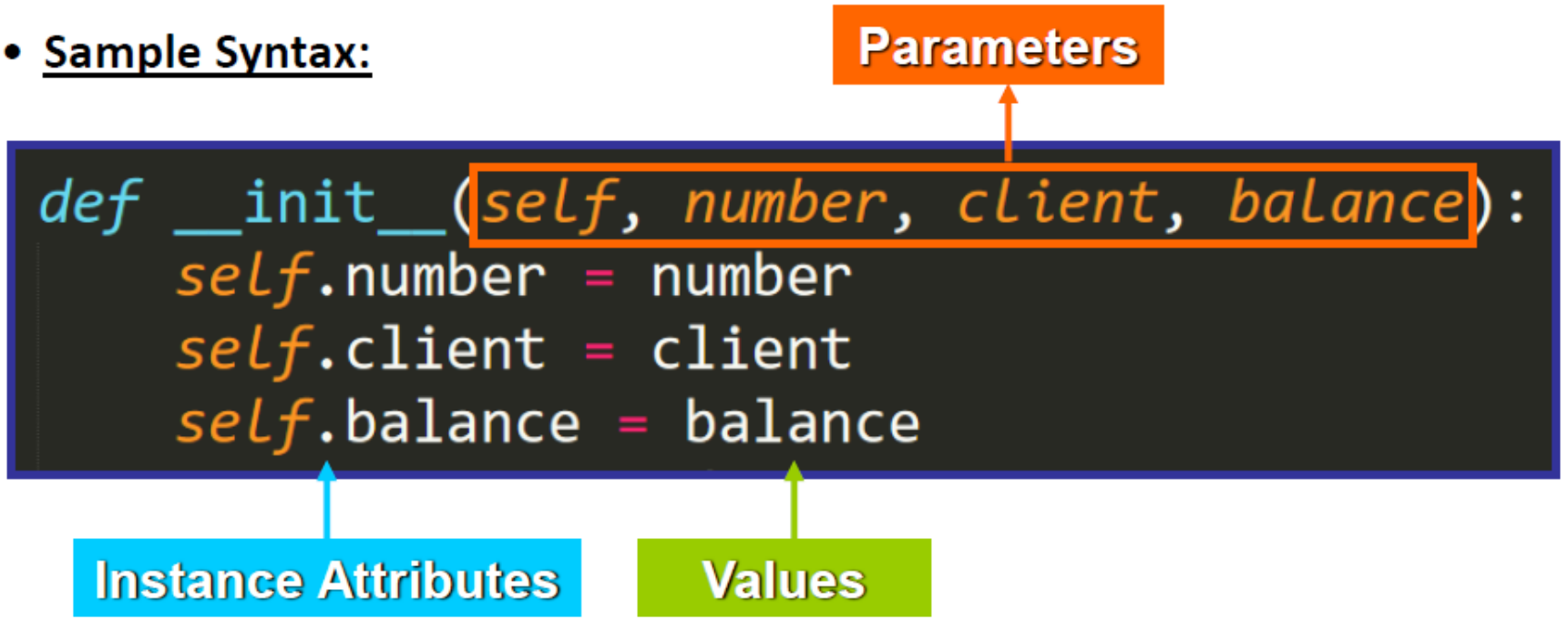
Quiz

- 9: เลือกข้อที่ถูก:
  - A. Instance เป็นอิสระจาก Instance อื่น เมื่อเรา update attributes ของ instance นั้น ใน Instance อื่นๆ จะไม่ได้รับผลกระทบ
  - B. Instance ไม่ได้เป็นอิสระจากอันอื่นโดยแท้จริง เมื่อเปลี่ยนค่าใน Instance ใดๆ ทุก Instance จะถูกแก้ไขตามไปด้วย
  - C. Instance สร้างจาก Class และ Class ก็เหมือนกับ "blueprints" ที่ทำหน้าที่กำหนด state และ behavior ของ Instance

Instance Attribute และ __init__()

- เนื่องจากใน Object แต่ละตัวอาจจะมี Attribute ต่างกัน ดังนั้นตอนที่สร้าง Object จึงต้องมีการกำหนดค่าเริ่มต้นให้กับ Object Attribute
- Method ที่ทำหน้าที่กำหนดค่าเริ่มต้นมีชื่อว่า __init__() ซึ่งเป็น method พิเศษที่มักจะมีในทุก Class โดยอาจจะเรียกว่า Constructor ก็ได้
- Method __init__() จะถูกเรียกขึ้นมาทำงานเมื่อมีการสร้าง Object

Instance Attribute และ __init__()



Instance Attribute และ __init__()

- Example : จาก Slide ที่ 46 สามารถสร้าง __init__() ดังนี้
- ใน parameter ต้องมีคำว่า self เสมอ หมายถึงตัว object เอง

```python
class Player:
    def __init__(self, name, x, y):
        self.name = name
        self.score = 0
        self.lives = 3
        self.speed = "Normal"
        self.position = (x, y)
```

Instance Attribute และ __init__()

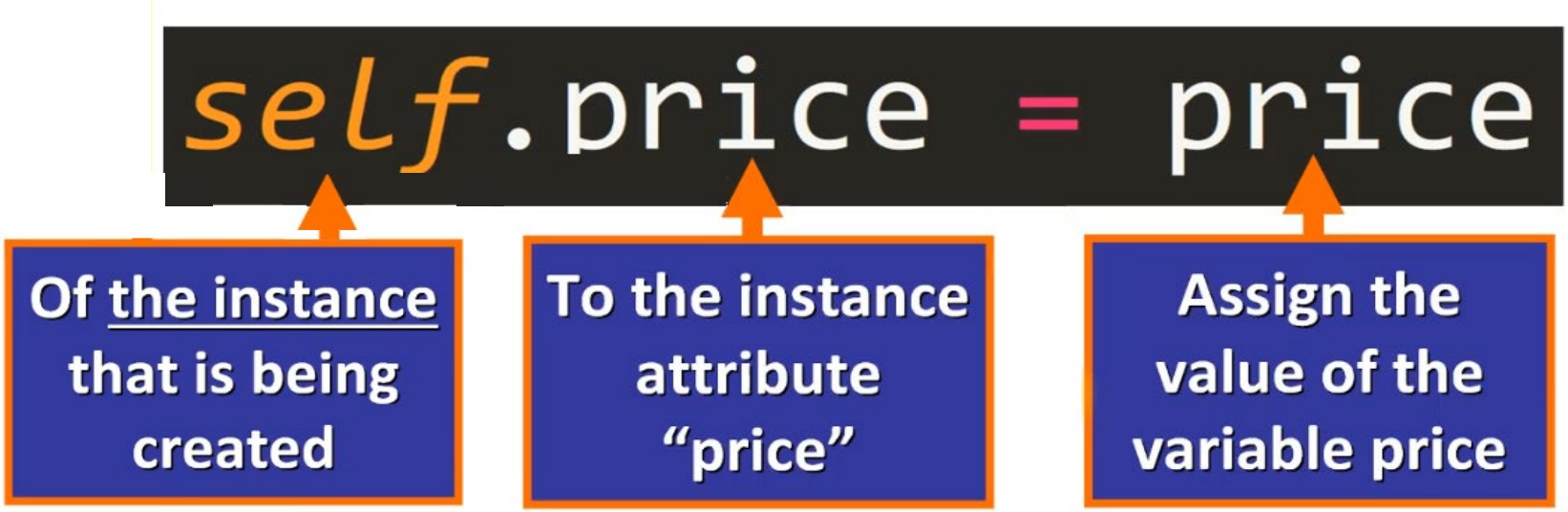
- ข้อผิดพลาดที่มักเกิดขึ้นกับ __init__()
  - ลืมเขียน def
  - ใช้เครื่องหมาย _ แค่ อันเดียว เช่น _init_()
  - ลืมเขียน self ใน parameter แม้ใน object ที่ไม่มี parameter เลยก็ต้องมี self เช่น
    - def __init__(width, height):
  - ลืมเขียน self.<attribute> ในการกำหนด instance attributes
  - PEP8 Style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

Instance and self

- ก่อนอื่นต้องเข้าใจว่า Class ทำหน้าที่กำหนด state และ behavior ของ object ในลักษณะทั่วๆ ไป
- เมื่อ Code ที่อยู่ใน Class มีการทำงาน จะไม่ได้มีผลกับทุก Object แต่จะมีผลเฉพาะ Object นั้นๆ
- ดังนั้นคำว่า self จึงมีความหมายถึง Object ที่ code กำลังทำงานด้วย ไม่ใช่ Object อื่นๆ
- จึงมีความจำเป็นจะต้องระบุ self ไว้เป็น argument แรกของ Object เพื่อระบุตำแหน่งของ Object นั้น

Instance and self

- เป็นการกำหนดค่าของตัวแปร price ให้กับ attribute ของ instance ที่สร้างขึ้น
- price ทั้ง 2 ตัวต่างกัน ฝั่งขวาคือ พารามิเตอร์ที่ส่งเข้ามา ฝั่งซ้ายคือ instance attribute



Quiz

- Q10. จาก method __init__() ด้านล่างนี้ ให้บอกว่ามี instance attribute ใดบ้าง

```
1  def __init__(self, radius, magnitude, angle):
2      self.radius = radius
3      self.magnitude = magnitude
4      self.angle = angle
5      self.color = "blue"
```

Quiz

- Q11. ข้อใดถูกต้อง

○
```
def __init__(self, magnitude, latitude, longitude, scale):
    self.magnitude = magnitude
    self.latitude = latitude
    self.longitude = longitude
```

○
```
def __init__(self, magnitude, latitude, longitude, scale):
        self.magnitude = magnitude
        latitude = latitude
        self.longitude = longitude
        scale = scale
```

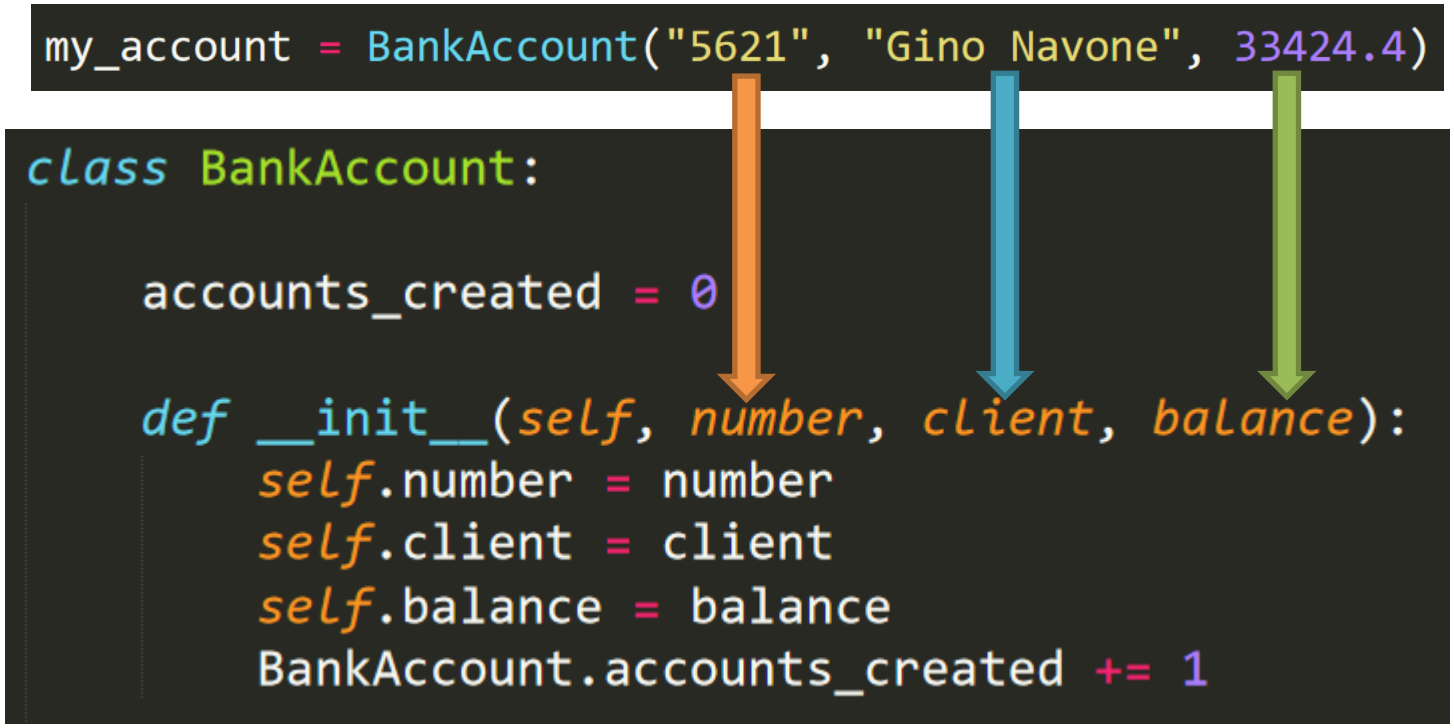
○
```
def __init__(self, magnitude, latitude, longitude, scale):
        self.magnitude = magnitude
        self.latitude = latitude
        self.longitude = longitude
        self.scale = scale
```

○
```
def __init__(magnitude, latitude, longitude, scale):
        self.magnitude = magnitude
        self.latitude = latitude
        self.longitude = longitude
        self.scale = scale
```

Create Instance

- รูปแบบทั่วไปของการสร้าง Instance คือ

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



Create Instance

- จาก Class Player สามารถสร้าง Object ดังนี้ จะเห็นว่าจะต้องใส่ parameter ให้เท่ากับที่ Class ต้องการ (ไม่รวม self)

```python
class Player:
    def __init__(self, name, x, y):
        self.name = name
        self.score = 0
        self.lives = 3
        self.speed = "Normal"
        self.position = (x, y)


player1 = Player("pacman", 50, 50)
```

Quiz

- Q12. จงเลือกรูปแบบที่ถูกต้องในการสร้าง Instance

- ○ `<variable> = <class>(<arguments>)`
- ○ `<class>(<arguments>)`
- ○ `<variable> = <instance>(<arguments)`

Quiz

- Q13. สมมติว่า method __init__ เป็นของ class Animal โดย attribute name กำหนดให้เป็น string และ age และ height ให้เป็น number

```
1  def __init__(self, name, age, height):
2          self.name = name
3          self.age = age
4          self.height = height
```

- ถ้าเราสร้าง Instance ตามคำสั่งนี้ ถูกหรือผิด

```
my_animal = Animal(6, "Noris", 50)
```

Access Instance Attribute

- การเข้าถึง attribute ของ object จากภายนอก object จะใช้รูปแบบที่เรียกว่า dot notation โดยการอ้างถึงชื่อของ object แล้วตามด้วย . จากนั้นจึงเป็นชื่อของ attribute
- สามารถเข้าถึงได้ทั้ง attribute และ method

Default Argument

- ในบางครั้งการสร้าง object อาจจะไม่ได้มีการส่งค่า attribute ไปหมดทุกตัว จึงมีการสร้าง default argument ซึ่งจะกำหนดค่าให้โดยอัตโนมัติ เมื่อไม่มีการส่งค่า
- จากตัวอย่าง เมื่อไม่ได้กำหนดตำแหน่ง x, y ก็จะเป็น 0,0 (การเขียนที่ถูกต้องคือไม่มี space) ที่สำคัญ คือ จะใช้ได้เฉพาะ last parameter เท่านั้น

```python
class Player:
    def __init__(self, name, x=0, y=0):
        self.name = name
        self.score = 0
        self.lives = 3
        self.speed = "Normal"
        self.position = (x, y)


player1 = Player("pacman")
```

Default Argument : Quiz

- Q14 : คลาสต่อไปนี้ เป็นไปตาม style guideline หรือไม่

```
1  class Doctor:
2      def __init__(self, specialty, num_children = 0):
3          self.specialty = specialty
4          self.num_children = num_children
```

Modify/Update Instance Attribute

- การแก้ไขหรือเปลี่ยนแปลง Instance Attribute ถือเป็นการเปลี่ยน state ของ Object
- วิธีการ คือ กำหนดค่าเข้าไปใน attribute โดยใช้ dot notation

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

- สำหรับกรณีที่ใช้ภายใน Class จะใช้ self แทนชื่อ Object

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

Quiz

- Q15 สมมติว่าเรามี Instance ชื่อ bookshelf โดยมี books เป็น attribute โดยมีรูปแบบดังนี้

```
books = ["Pride and Prejudice", "Metamorphosis"]
```

ให้เลือกวิธีที่ถูกต้องในการแก้ไขหนังสือเล่มที่ 2

- ○ `bookshelf.books[0] = "Romeo and Juliet"`
- ○ `bookshelf.books = "Romeo and Juliet"`
- ○ `bookshelf.books[1] = "Romeo and Juliet"`

Exercise

- ให้แก้ไข Bugs ของโปรแกรมต่อไปนี้ ให้ถูกต้อง

```
class Donut
        def _ini__(flavor, toppings, filling, size):
                self.flavor = flavor
                self.toppings = toppings
                self.filling = filling
                size = self.size
class Customer:
        def _init_(self, name, age address, favorite_dessert)
                self.name = name
                self.age = age
                self.address = address
                favorite_dessert= self.favorite_dessert

    Cake:
        __init__(self, flavor, price, quality):
                self.flavor = flavor
                self.price = price
                self.quality = quality
```

For your attention