# รายวิชา การเขียนโปรแกรมเชิงวัตถุ
# รหัสวิชา 4122309

## บทที่ 8 Python : Class and Object

**โดย ผู้ช่วยศาสตราจารย์ ดร. นัฐพงศ์ ส่งเนียม**
**สาขาวิชาวิทยาการคอมพิวเตอร์**
**คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏพระนคร**

# Agenda

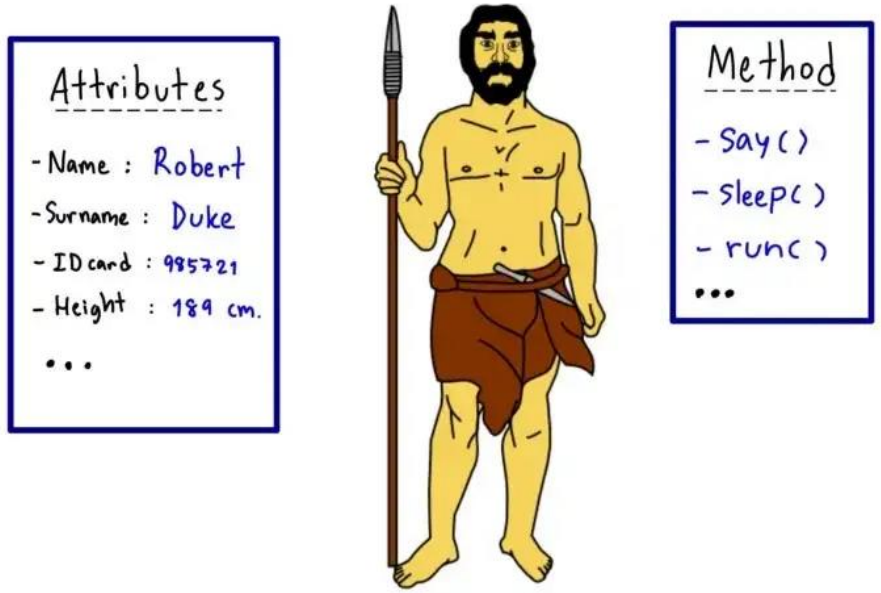
## Class and Object ในภาษาไพธอน

8.1 ความหมายของ OOP
8.2 การสร้าง class และ object
8.3 Constructor และ Destructor
8.4 Static variables และ Static methods
8.5 การลบ attribute
8.6 การลบวัตถุ
8.7 การใช้งาน pass statement

## 8.1 ความหมายของ OOP

OOP เป็นวิธีการเขียนโปรแกรมรูปแบบหนึ่ง โดยมองสิ่งต่าง ๆ ในระบบเป็นวัตถุ (Object) ชิ้นหนึ่งที่มีหน้าที่และความหมายในตัว โดยวัตถุ ๆ นั้น ก็มี คุณสมบัติ (Attributes) และ พฤติกรรม (Method,Behavior) หรือการกระทำของมัน เป็นการมองบนพื้นฐานความเป็นจริงมากขึ้น เช่น

**Object → มนุษย์ ประกอบด้วย คุณสมบัติ (Attributes) คือ ชื่อ,นามสกุล,รหัสบัตรประชาชน ,ส่วนสูง..... และมี พฤติกรรม (Method) เช่น พูด,นอน ,นั่ง,วิ่ง.....**



## 8.1 ความหมายของ OOP

**OOSE**

- OOA : Object Oriented Analysis
- OOD : Object Oriented Dsign
- OOP : Object Oriented Programming
- OOT : Object Oriented Testing

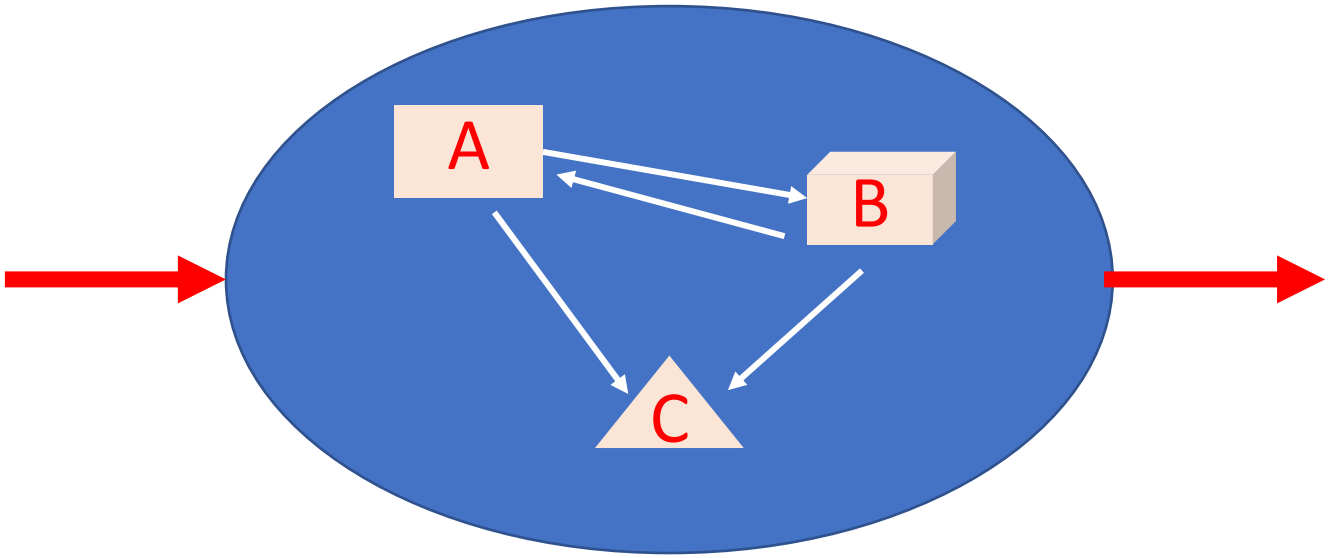
OOAD

OOP

## 8.1 ความหมายของ OOP

การเขียนโปรแกรมเชิงวัตถุ (อังกฤษ: Object-oriented programming, OOP) คือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ วัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้ วัตถุอื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป

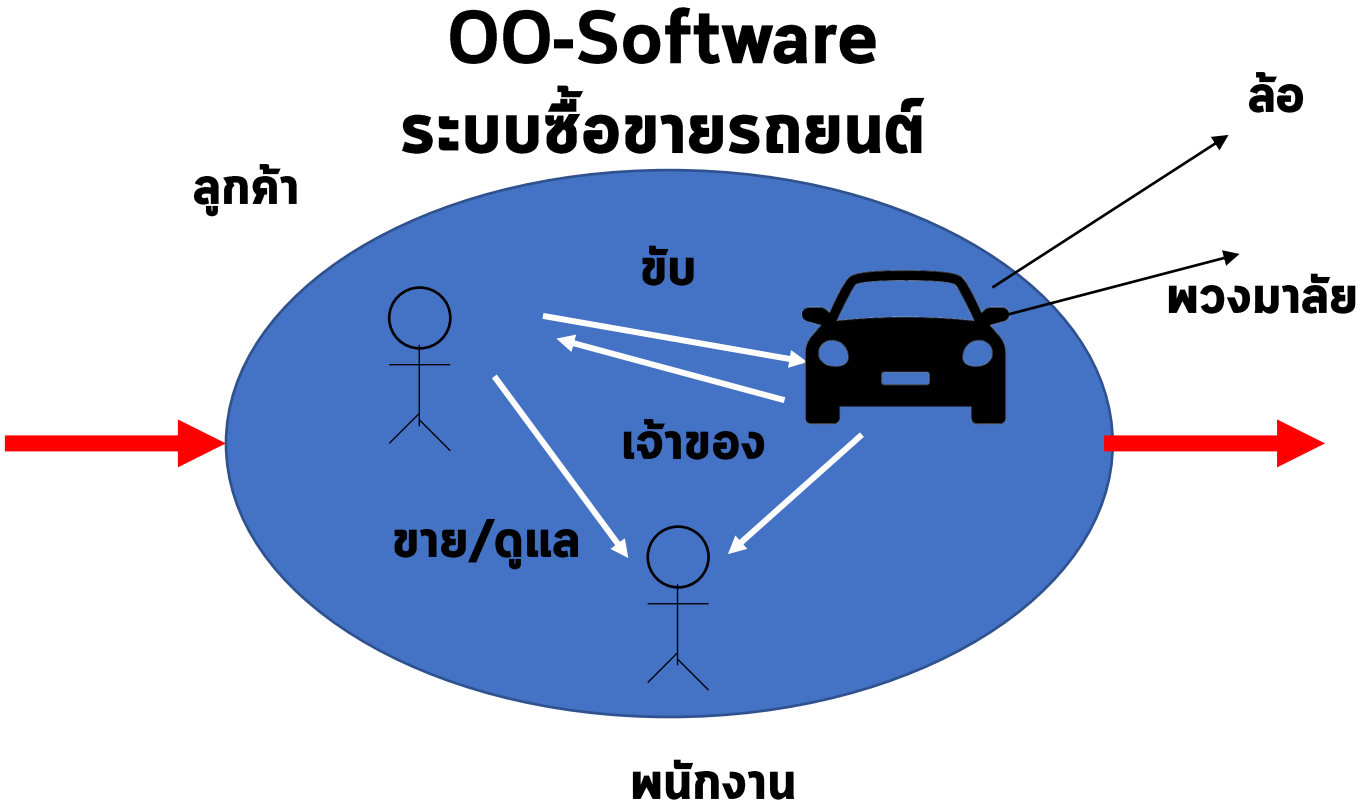
OO-Software




ที่มา : https://th.wikipedia.org/wiki/การเขียนโปรแกรมเชิงวัตถุ

## 8.1 ความหมายของ OOP

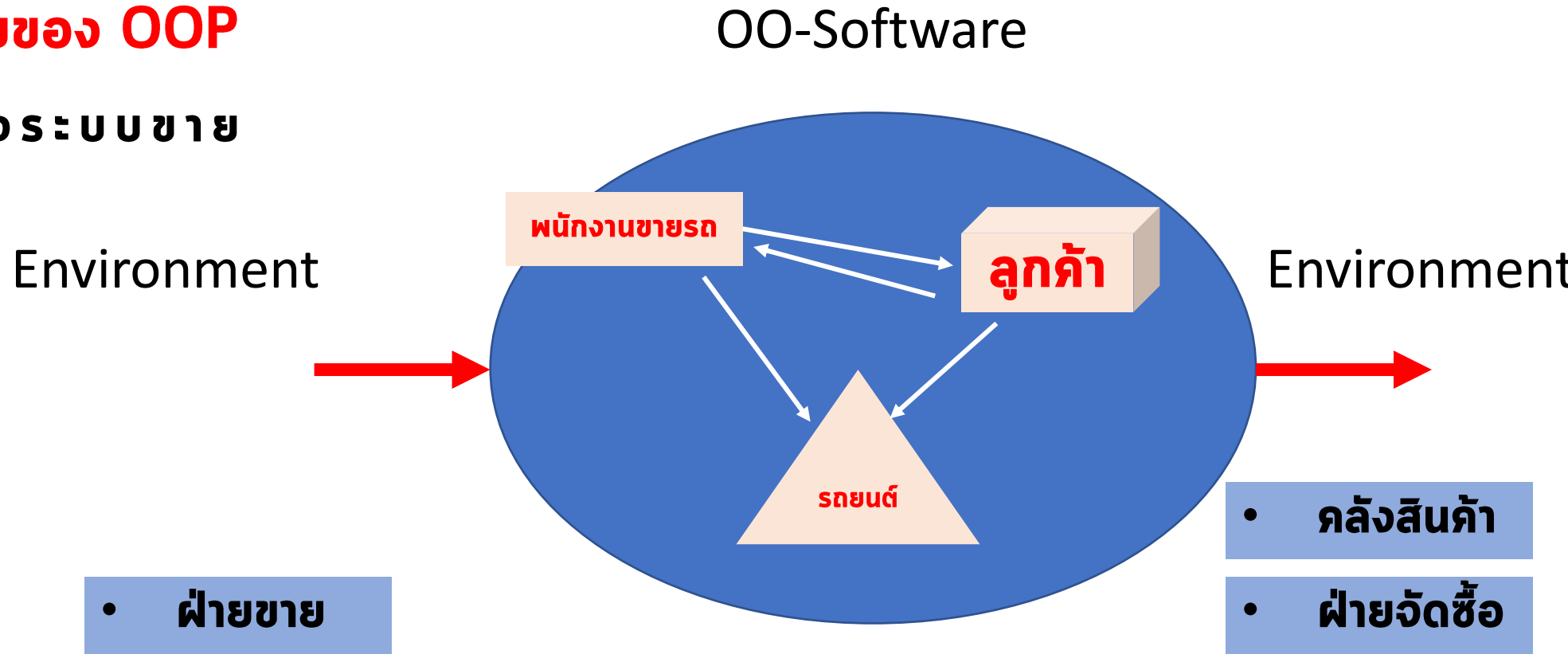
การเขียนโปรแกรมเชิงวัตถุ (อังกฤษ: Object-oriented programming, OOP) คือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ วัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้ วัตถุอื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป




ที่มา : https://th.wikipedia.org/wiki/การเขียนโปรแกรมเชิงวัตถุ

## 8.1 ความหมายของ OOP

**ซอฟต์แวร์ของระบบขายรถยนต์**




ที่มา : https://th.wikipedia.org/wiki/การเขียนโปรแกรมเชิงวัตถุ

## 8.1 ความหมายของ OOP

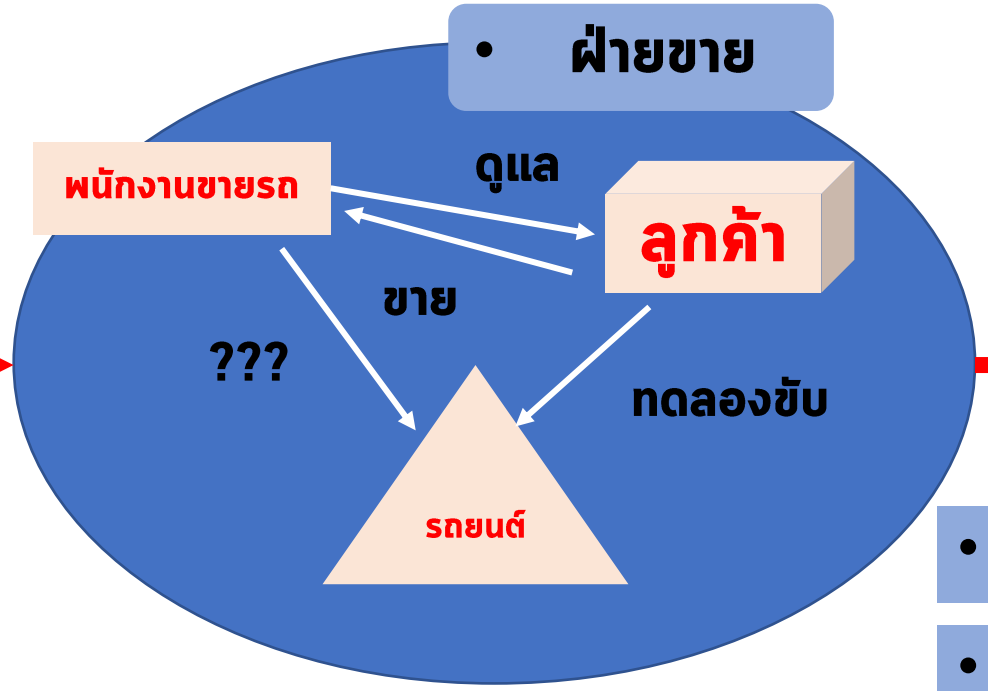
**ซอฟต์แวร์ของระบบขายรถยนต์**

Environment

**ระบบบริหารศูนย์จำหน่ายรถยนต์**

- ฝ่ายขาย
- คลังสินค้า
- ฝ่ายจัดซื้อ
- ฝ่ายบุคคล

OO-Software




Environment

- คลังสินค้า
- ฝ่ายจัดซื้อ

ที่มา : https://th.wikipedia.org/wiki/การเขียนโปรแกรมเชิงวัตถุ

## 8.1 ความหมายของ OOP

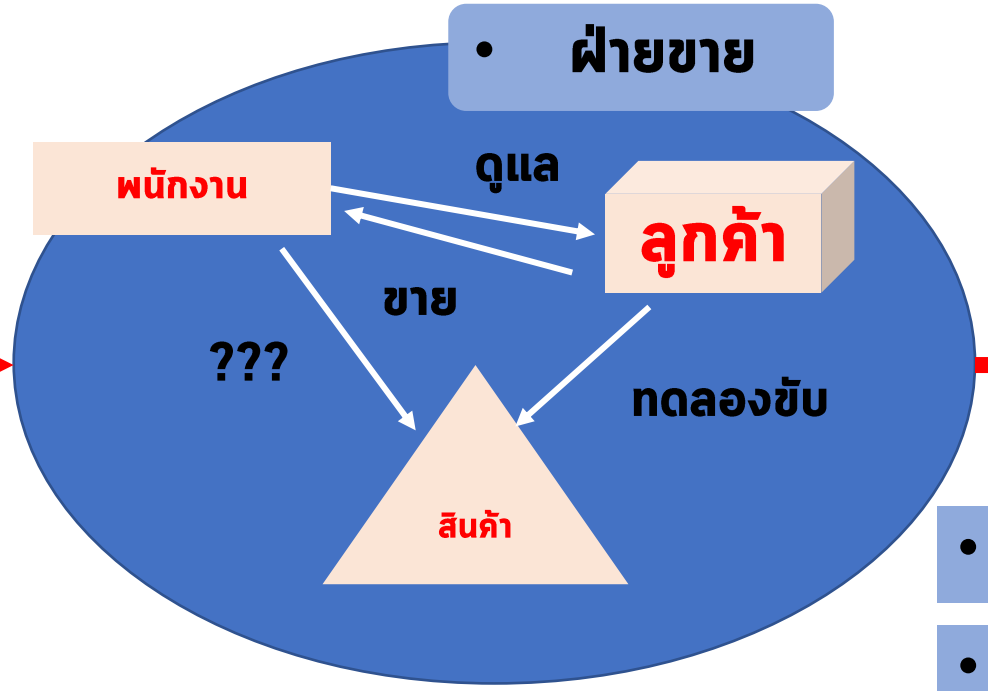
**ซอฟต์แวร์ของระบบร้านขายกาแฟออนไลน์**

Environment

**ร้านขายกาแฟออนไลน์**

- ฝ่ายขาย
- คลังสินค้า
- ฝ่ายจัดซื้อ
- ฝ่ายบุคคล

OO-Software



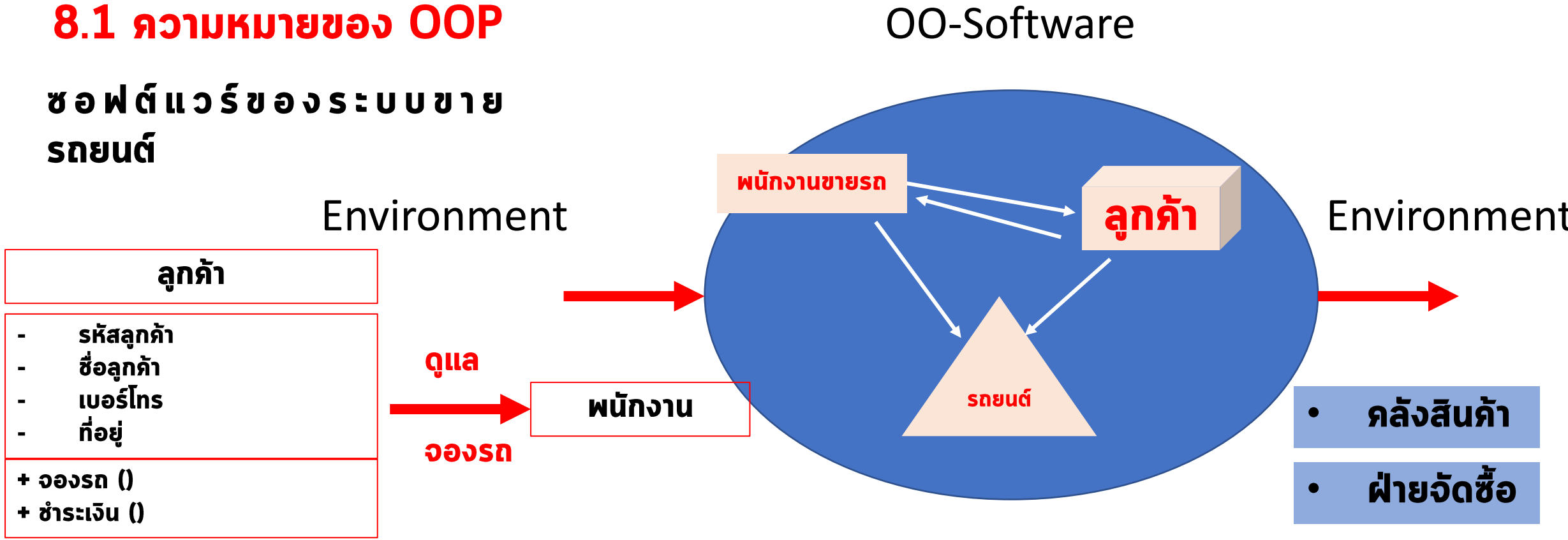

Environment

- คลังสินค้า
- ฝ่ายจัดซื้อ

ที่มา : https://th.wikipedia.org/wiki/การเขียนโปรแกรมเชิงวัตถุ

## 8.1 ความหมายของ OOP

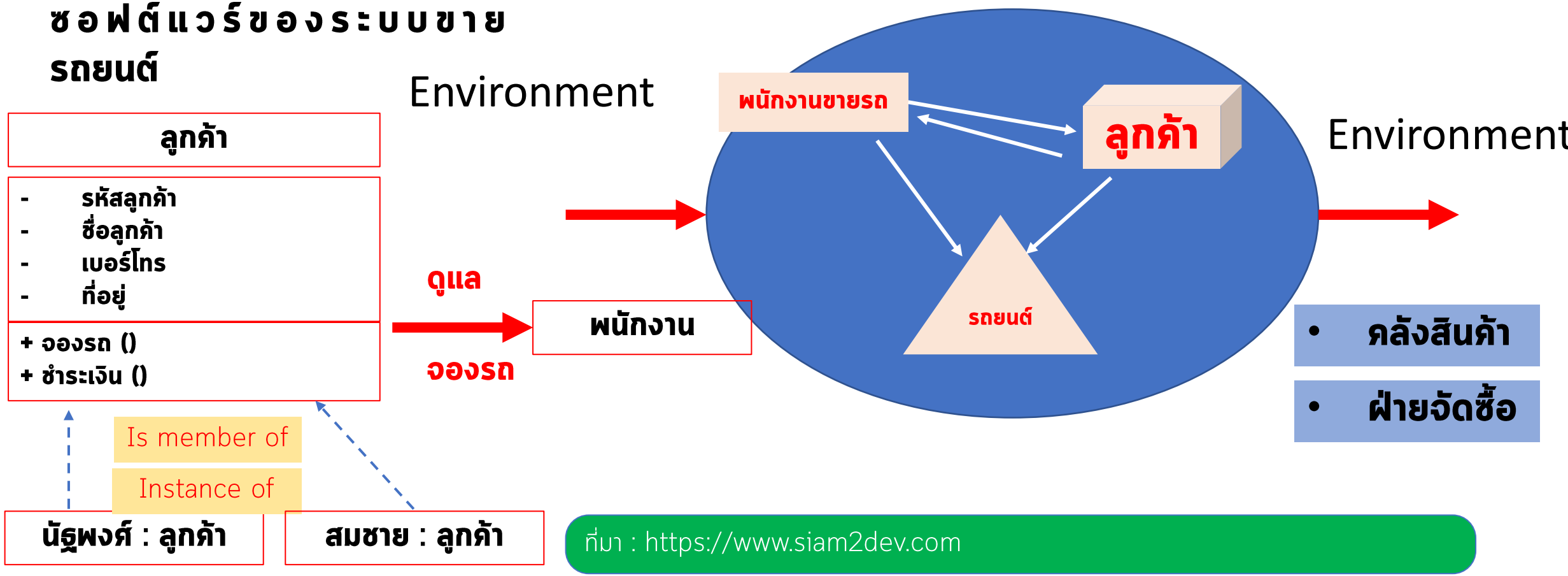
**ซอฟต์แวร์ของระบบขายรถยนต์**




ที่มา : https://www.siam2dev.com

## 8.1 ความหมายของ OOP

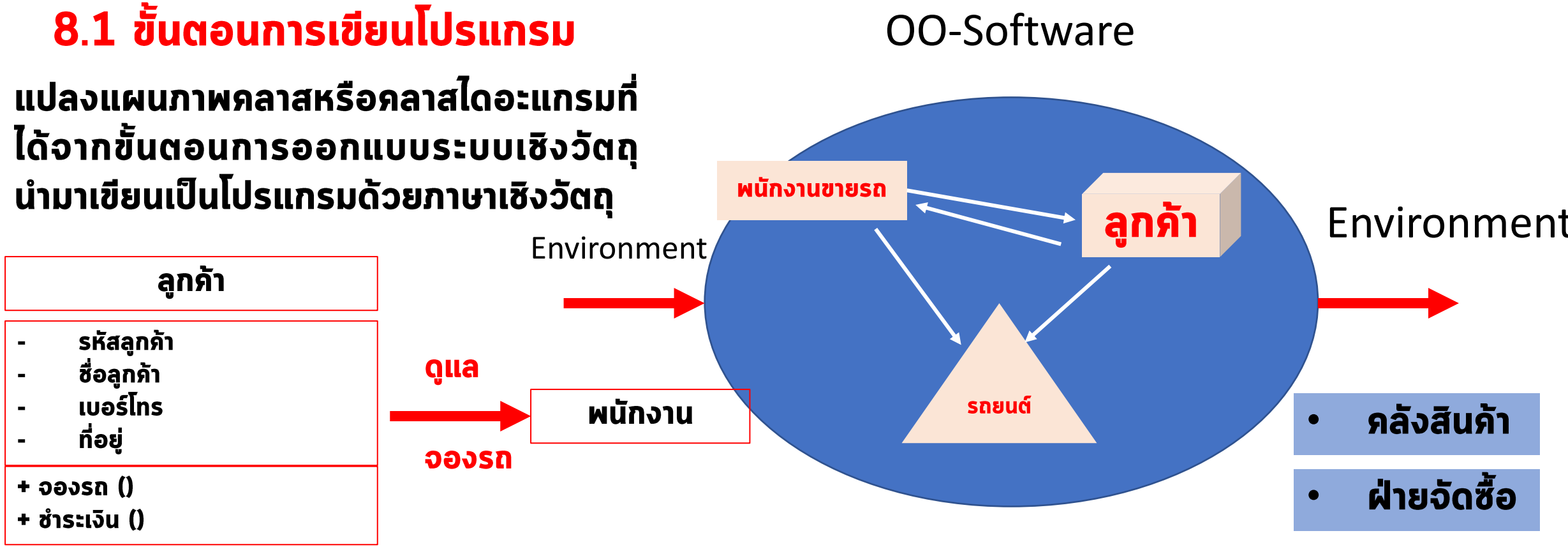
**ซอฟต์แวร์ของระบบขายรถยนต์**




ที่มา : https://www.siam2dev.com

## 8.1 ขั้นตอนการเขียนโปรแกรม

**แปลงแผนภาพคลาสหรือคลาสไดอะแกรมที่ได้จากขั้นตอนการออกแบบระบบเชิงวัตถุ นำมาเขียนเป็นโปรแกรมด้วยภาษาเชิงวัตถุ**




ที่มา : https://www.siam2dev.com

## 8.1 ขั้นตอนการเขียนโปรแกรม

**แปลงแผนภาพคลาสหรือคลาสไดอะแกรมที่ได้จากขั้นตอนการออกแบบระบบเชิงวัตถุ นำมาเขียนเป็นโปรแกรมด้วยภาษาเชิงวัตถุ**

| ลูกค้า |
|---|
| รหัสลูกค้า<br>ชื่อลูกค้า<br>เบอร์โทร<br>ที่อยู่ |
| + จองรถ ()<br>+ ชำระเงิน () |

ดูแล / จองรถ → พนักงาน

| ลูกค้า | ลูกค้า |
|---|---|
| - รหัสลูกค้า : Textual<br>- ชื่อลูกค้า : Textual<br>- เบอร์โทร : Textual<br>- ที่อยู่ : Textual | - รหัสลูกค้า : String<br>- ชื่อลูกค้า : String<br>- เบอร์โทร : String<br>- ที่อยู่ : String |
| + กำหนดชื่อลูกค้า (n : Textual)<br>+ บอกชื่อลูกค้า () : Textual<br>+ กำหนดรายได้ (income: floating-point)<br>+ จองรถ () :<br>+ ชำระเงิน () | + กำหนดชื่อลูกค้า (n : String)<br>+ บอกชื่อลูกค้า () : String<br>+ กำหนดรายได้ (income: double)<br>+ จองรถ () :<br>+ ชำระเงิน () |

**Refinement ปรับปรุงทำให้ละเอียด ชัดเจน ครอบคลุม**

Analysis Phase : What → Desing Phase : How → Inplement Phase

Analysis Phase : Behavior → Inplement Phase : Method

**Design Reviewer : นักทบทวนการออกแบบ**

## 8.1 ความหมายของ class

**คลาส คือสิ่งที่ใช้กำหนดรูปแบบของข้อมูล (Attributes) และเมธอด (Methods) การทำงานเข้าด้วยกัน การสร้างคลาส หมายถึง การสร้างประเภทของออบเจ็คขึ้นมา กล่าวอีกนัยหนึ่ง คลาสคือประเภทข้อมูลของออบเจ็คโดยคลาสนั้นสร้างขึ้นโดยผู้ใช้ (User-defined type) โดยปกติแล้ว ประเภทข้อมูลพื้นฐานในภาษา Python นั้นคือคลาส เมื่อคุณสร้างตัวแปรใดๆ ขึ้นมา ตัวแปรเหล่านั้นเป็นออบเจ็คของคลาส เพื่อทำให้คุณเข้าใจมากขึ้นมาดูตัวอย่างต่อไปนี้**

คลาส คือประเภทข้อมูลที่สร้างโดยผู้ใช้ โดยจะนำไปใช้สร้างออบเจ็ค กล่าวอีกนัยหนึ่ง คลาสคือประเภทข้อมูลของออบเจ็ค

ออบเจ็ค คือสิ่งที่สร้างมาจากคลาสหรือ class instances

แอตทริบิวต์ (instance attributes) คือข้อมูลที่เป็นสมาชิกของแต่ละออบเจ็ค โดยมักจะกำหนดไว้ในเมธอด __init__() ของคลาส

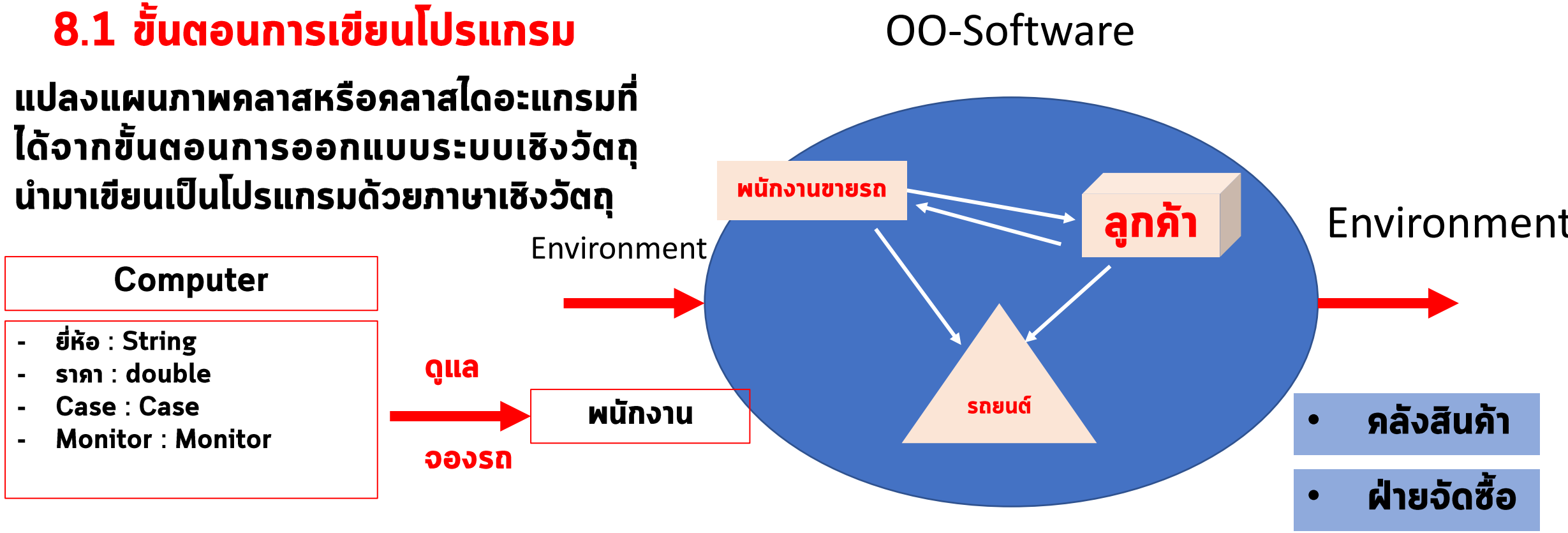
เมธอด คือฟังก์ชันการทำงานที่กำหนดไว้ในคลาส

คลาสแอตทริบิวต์ (class attributes) คือตัวแปรที่ประกาศไว้ในคลาส ซึ่งจะแชร์กับออบเจ็คทั้งหมดที่สร้างจากคลาสนั้นๆ

ที่มา : http://marcuscode.com/lang/python/classes-and-objects

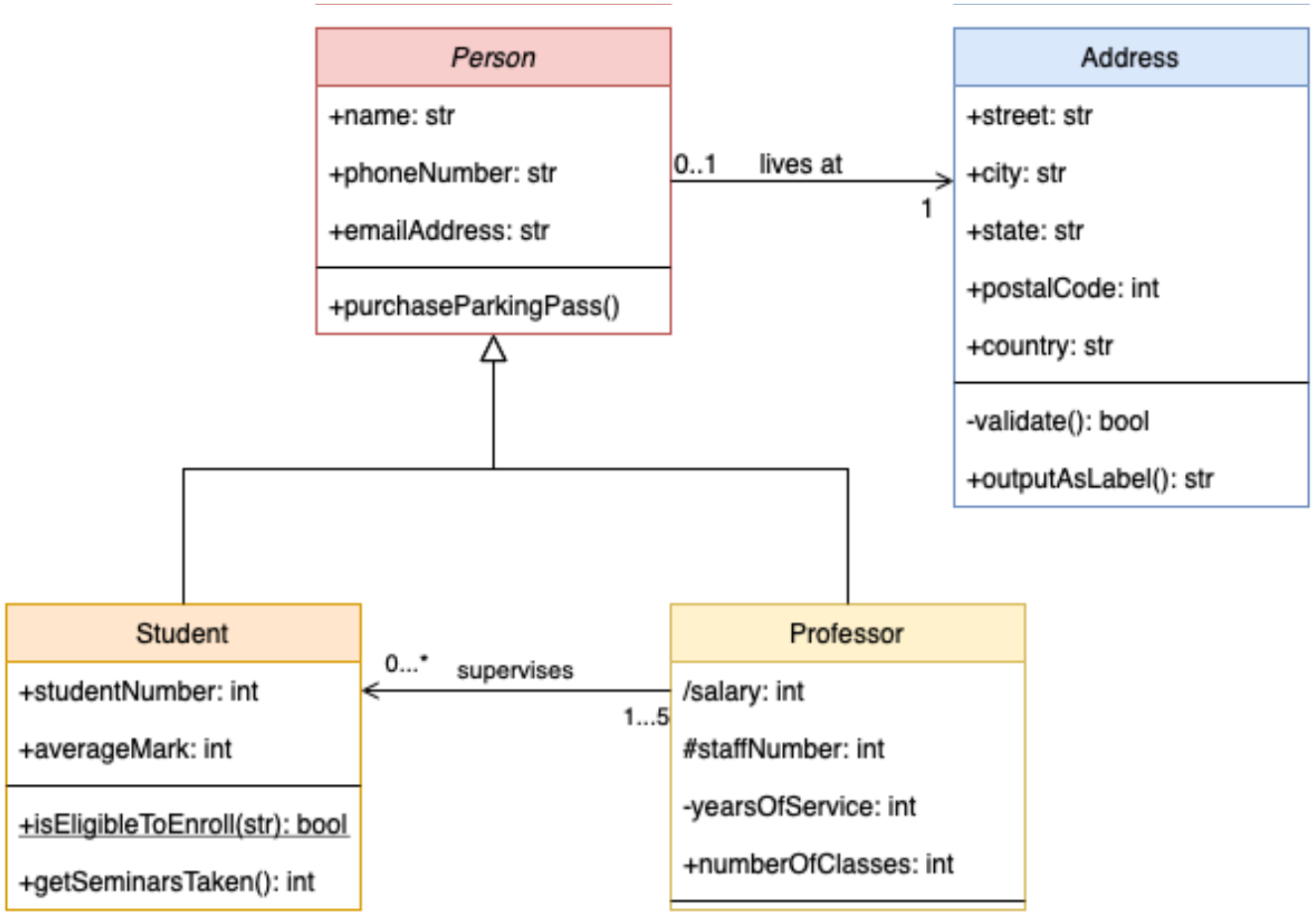
## 8.1 ขั้นตอนการเขียนโปรแกรม

**แปลงแผนภาพคลาสหรือคลาสไดอะแกรมที่ได้จากขั้นตอนการออกแบบระบบเชิงวัตถุ นำมาเขียนเป็นโปรแกรมด้วยภาษาเชิงวัตถุ**




ที่มา : https://www.siam2dev.com

## 8.1 ความหมายของ class




ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

## 8.1 ความหมายของ class

```
a = 1
b = 1.34
c = 'marcuscode.com'

print(type(a))
print(type(b))
print(type(c))
```

จากตัวอย่าง เป็นประกาศตัวแปรสามตัวคือ Integer Floating และ String ตามลำดับ ดังนั้นตัวแปรเหล่านี้ถือว่าเป็นออบเจ็คของคลาส ดังผลลัพธ์ข้างล่าง

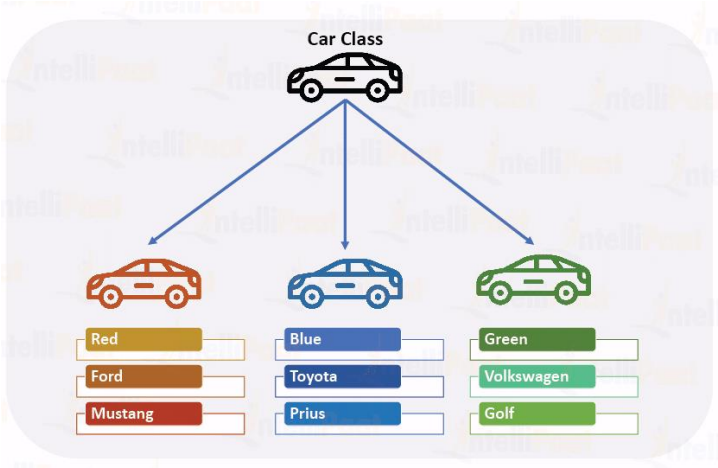
```
<class 'int'>
<class 'float'>
<class 'str'>
```

เหมือนที่คุณเห็น เราใช้ฟังก์ชัน type() เพื่อดูประเภทข้อมูลของออบเจ็คใดๆ หรือใช้สำหรับดูประเภทของคลาสที่มันสร้างมาจาก จากตัวอย่างนั้น เราเรียกคลาส int float และ str ว่า build-in type สำหรับในบทนี้ เรากำลังจะพูดเกี่ยวกับการสร้างคลาสซึ่งเป็น User-defined type นั่นเอง

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

## 8.1 ความหมายของ class

จากตัวอย่าง เป็นการสร้างรถยนต์ ยี่ห้อต่าง ๆ จากคลาสรถยนต์ โดยนักศึกษาสามารถสร้างคลาสจากหลักการ Classification Abstraction ที่ได้ศึกษามาแล้วในรายวิชาการวิเคราะห์และออกแบบระบบเชิงวัตถุ (OOAD)




ที่มา : https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/

## 8.2 การสร้างคลาสและออบเจ็ค

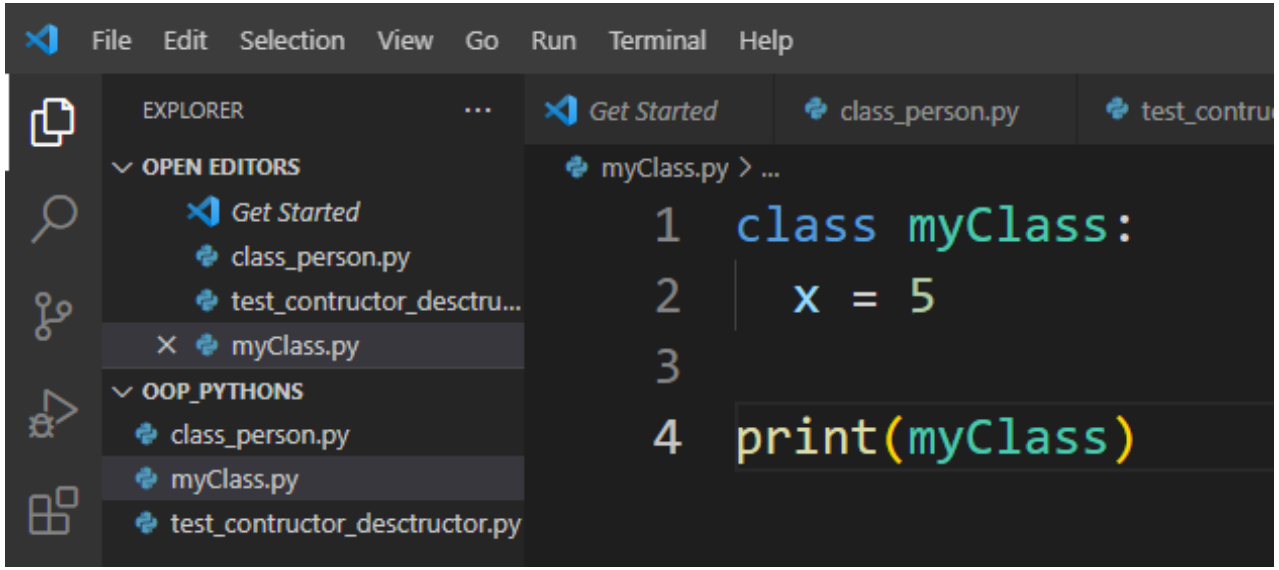
**รูปแบบ / ตัวอย่าง**

```
class ClassName:
    # statements
```

```
class MyClass:
  x = 5
print(MyClass)
```

```
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppD
<class '__main__.myClass'>
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

เราจะใช้คำสั่ง class สำหรับสร้างคลาสในภาษา Python และตามด้วยชื่อของคลาส ClassName ชื่อของคลาสควรจะขึ้นต้นด้วยตัวใหญ่และเป็นรูปแบบ camel case หลังจากนั้นเป็นคำสั่งในการกำหนดตัวแปรและเมธอดของคลาส ต่อไปมาดูตัวอย่างการสร้างคลาสในภาษา Python




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

## 8.2 การสร้างคลาสและออบเจ็ค

**การสร้างออบเจ็กต์**

**objectName = className()**

```
class MyClass:
  x = 5

p1 = MyClass()
print(p1.x)
```

การสร้างวัตถุ ทำได้โดยมีรูปแบบดังนี้

```
class MyClass:
  x = 5

p1 = MyClass()
print(p1.x)
```

ผลลัพธ์การทำงาน

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Us
5
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

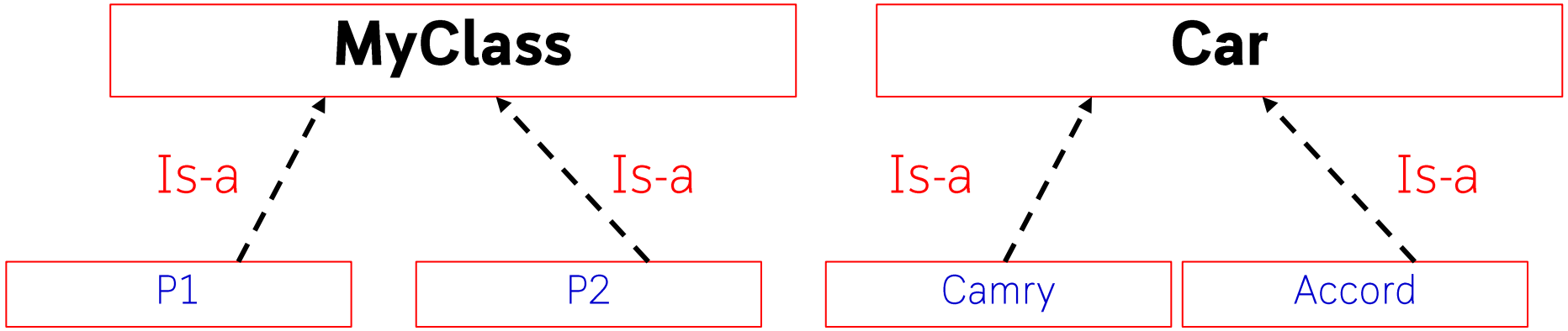
## 8.2 การสร้างคลาสและออบเจ็ค

**การสร้างออบเจ็กต์**

OO: Is member of

Implement: Is-a

```
P1 = MyClass()
```



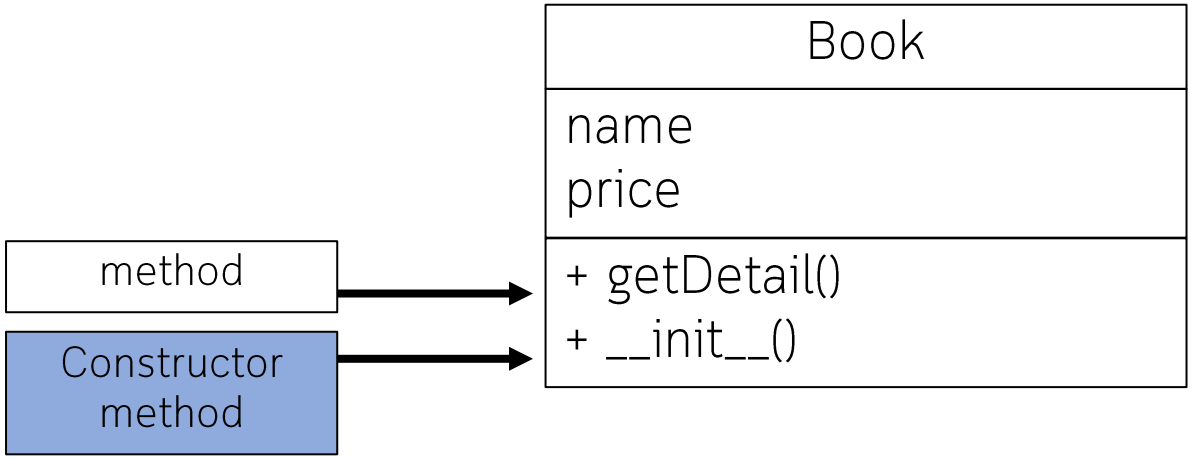

ที่มา : https://www.w3schools.com/python/python_classes.asp

## 8.2 การสร้างคลาสและออบเจ็ค

**ตัวอย่าง**

```
class Book:

    def __init__(self, name, price):
        self.name = name
        self.price = price

    def getDetail(self):
        print('Name: %s' % self.name)
        print('Price: %d USD' % self.price)
```

เราจะใช้คำสั่ง class สำหรับสร้างคลาสในภาษา Python และตามด้วยชื่อของคลาส ClassName ชื่อของคลาสควรจะขึ้นต้นด้วยตัวใหญ่และเป็นรูปแบบ camel case หลังจากนั้นเป็นคำสั่งในการกำหนดตัวแปรและเมธอดของคลาส ต่อไปมาดูตัวอย่างการสร้างคลาสในภาษา Python




**ในตัวอย่าง เราได้สร้างคลาส Book และภายในมีเมธอด __init__() ซึ่งเป็นคอนสตรัคเตอร์ (Constructor) ซึ่งจะถูกเรียกอัตโนมัติเมื่อออบเจ็คถูกสร้างสำเร็จ พารามิเตอร์แรกของเมธอดจะเป็น self เสมอ นี่จะใช้เป็นตัวแปรในการอ้างถึงออบเจ็คปัจจุบัน คลาสนี้จะมีสองแอตทริบิวต์คือ name และ price ใช้สำหรับเก็บชื่อและราคาของหนังสือของแต่ละออบเจ็ค หลังจากที่เราได้สร้างคลาสเสร็จแล้ว ต่อไปเราจะนำมาสร้างออบเจ็ค**

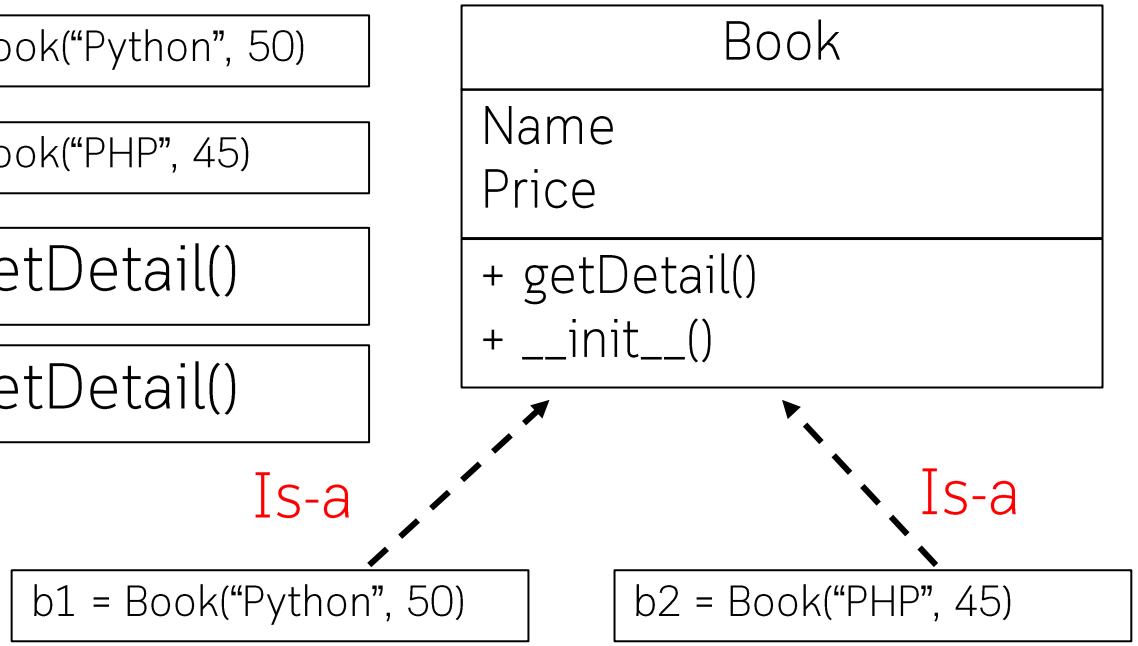
ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

## 8.2 การสร้างคลาสและออบเจ็ค

**ตัวอย่าง**

```
class Book:

    def __init__(self, name, price):
        self.name = name
        self.price = price

    def getDetail(self):
        print('Name: %s' % self.name)
        print('Price: %d USD' % self.price)
```

```
b1 = Book("Python", 50)
b2 = Book("PHP", 45)
b1.getDetail()
b2.getDetail()
```




ในตัวอย่าง เราได้สร้างคลาส Book และภายในมีเมธอด __init__() ซึ่งเป็นคอนสตรัคเตอร์ (Constructor) ซึ่งจะถูกเรียกอัตโนมัติเมื่อออบเจ็คถูกสร้างสำเร็จ พารามิเตอร์แรกของเมธอดจะเป็น self เสมอ นี่จะใช้เป็นตัวแปรในการอ้างถึงออบเจ็คปัจจุบัน คลาสนี้จะมีสองแอตทริบิวต์คือ name และ price ใช้สำหรับเก็บชื่อและราคาของหนังสือของแต่ละออบเจ็ค หลังจากที่เราได้สร้างคลาสเสร็จแล้ว ต่อไปเราจะนำมาสร้างออบเจ็ค

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.2 การสร้างคลาสและออบเจ็ค

## The __init__() Function

__init__ เป็นฟังก์ชันเริ่มต้นทำงานเมื่อมีการสร้างออบเจ็กต์ เรียกว่า เป็นคอนสตรักเตอร์ ฟังก์ชัน (constructor )

| Person |
| --- |
| Name<br>age |
| __init__() |

person.py > ...

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

p1 = Person("Nattapong Songneam", 46)

print(p1.name)
print(p1.age)
```

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

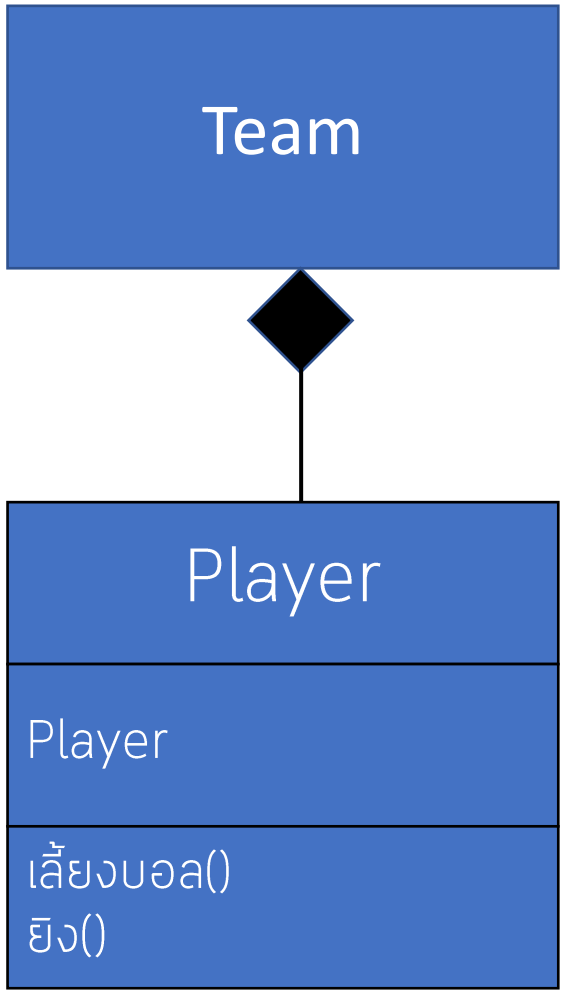
To understand the meaning of classes we have to understand the built-in __init__() function.

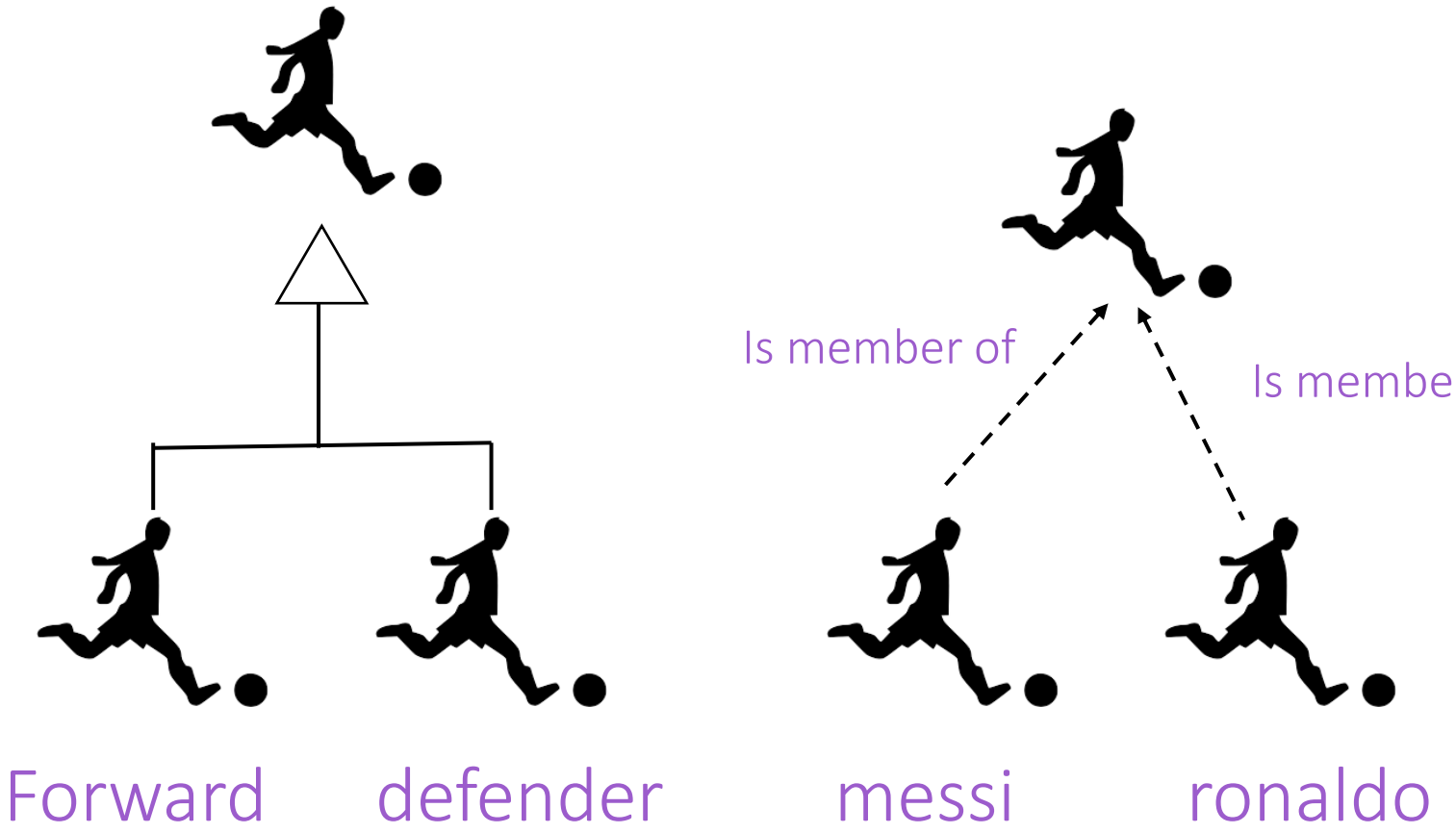
All classes have a function called __init__(), which is always executed when the class is being initiated.

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

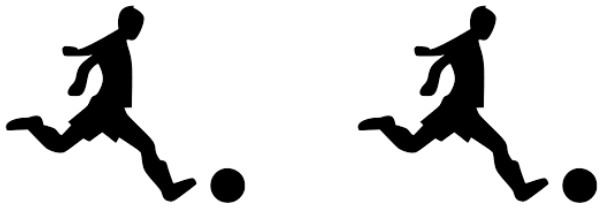
ที่มา : https://www.w3schools.com/python/python_classes.asp

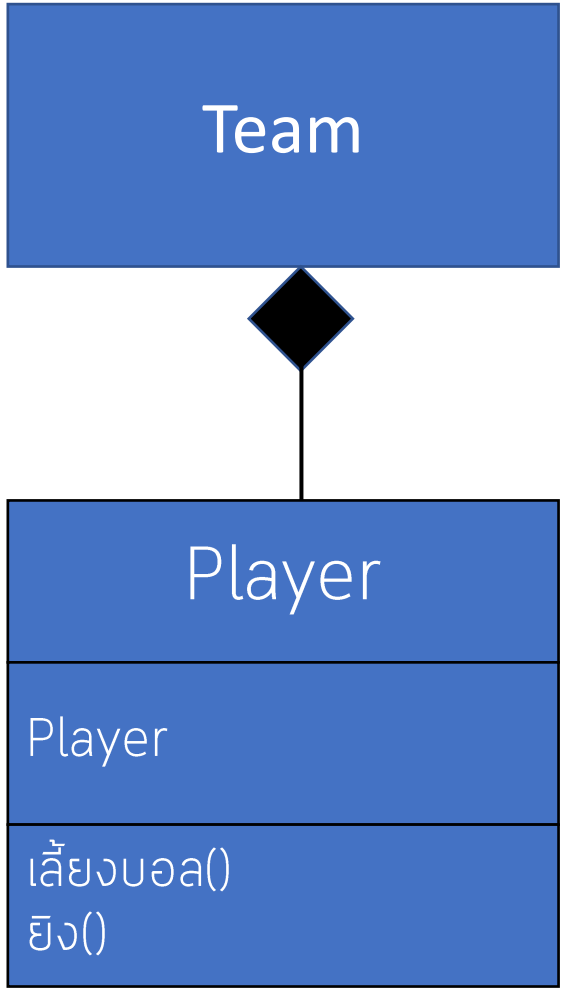
# Soccer Game


Team
Player
Player
เลี้ยงบอล()
ยิง()


Is member of
Is member of
Forward
defender
messi
ronaldo

# Soccer Game






lec08_SoccerGame.py > ...

```python
# ตัวอย่างคลาสและออบเจ็กต์ของเกมฟุตบอล

class Player:
    def __init__(self, name, position, shirt_number):
        self.name = name
        self.position = position
        self.shirt_number = shirt_number

    def dribble(self):
        print(f"{self.name} กำลังเดินเบาๆ ผ่านคู่แข่ง")

    def shoot(self):
        print(f"{self.name} ยิงลูกฟุตบอล!")

class Team:
    def __init__(self, name, coach):
        self.name = name
```

# Soccer Game

[] list หนึ่งทีมมีผู้เล่นหลายคน

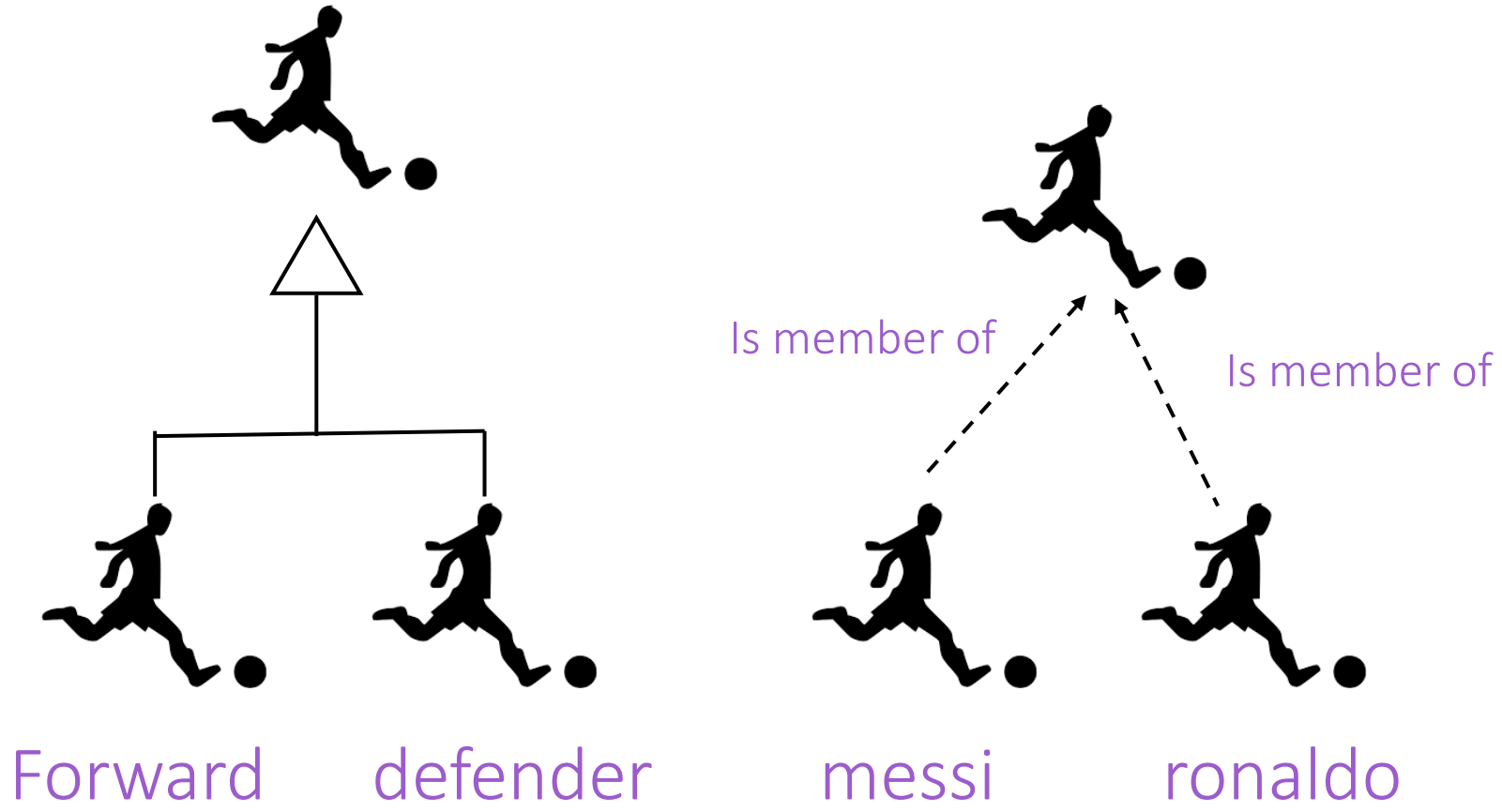
```python
class Team:
    def __init__(self, name, coach):
        self.name = name
        self.coach = coach
        self.players = []

    def add_player(self, player):
        self.players.append(player)

    def list_players(self):
        print(f"รายชื่อผู้เล่นในทีม {self.name}:")
        for player in self.players:
            print(f"{player.shirt_number}: {player.name} ({player.position})")
```

# Soccer Game

```python
# สร้างออบเจ็กต์ Player
player1 = Player("Cristiano Ronaldo", "Forward", 7)
player2 = Player("Lionel Messi", "Forward", 10)
player3 = Player("Sergio Ramos", "Defender", 4)

# สร้างออบเจ็กต์ Team
team1 = Team("Real Madrid", "Zinedine Zidane")
team2 = Team("Barcelona", "Ronald Koeman")

# เพิ่มผู้เล่นในทีม
team1.add_player(player1)
team1.add_player(player3)
team2.add_player(player2)
```

```python
# แสดงรายชื่อผู้เล่นในทีม
team1.list_players()
team2.list_players()

# เรียกใช้งานเมธอดของ Player
player1.dribble()
player2.shoot()
```

# Soccer Game




จากตัวอย่าง

เราสร้างคลาส Player และ Team สำหรับเกมฟุตบอล

ในคลาส Player, เรามีคุณสมบัติเช่นชื่อ (name), ตำแหน่ง (position), และหมายเลขเสื้อ (shirt_number) และเมธอดเช่น dribble และ shoot ที่เป็นการกระทำของผู้เล่น

ในคลาส Team, เรามีคุณสมบัติเช่นชื่อทีม (name) และโค้ช (coach) และเราสามารถเพิ่มผู้เล่นในทีมด้วยเมธอด add_player และแสดงรายชื่อผู้เล่นในทีมด้วยเมธอด list_players

เราสร้างออบเจ็กต์ของผู้เล่นและทีมแล้วเรียกใช้งานเมธอดที่เกี่ยวข้องกับผู้เล่นเพื่อแสดงข้อความในเอาท์พุต.

## The __str__() Function

### The string representation of an object WITHOUT the __str__() function:

| Person |
| --- |
| Name<br>age |
| ~~__str__()~~ |

The __str__() function controls what should be returned when the class object is represented as a string.
If the __str__() function is not set, the string representation of the object is returned:

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

p1 = Person(Asst. Prof. Dr. Nattapong Songneam", 46)

print(p1)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User
<__main__.Person object at 0x0000024E55F07D60>
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

## ตัวอย่างการสร้างวัตถุจากคลาสแบบไม่มี __str__ ฟังก์ชัน

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

## The __str__() Function

The string representation of an object WITHOUT the __str__() function:

| Person |
| --- |
| Name<br>age |
| ~~__str__()~~ |

| Rectangle |
| --- |
| width<br>Height |
| __str__() |

| Circle |
| --- |
| radius<br>color |
| __str__() |

## ตัวอย่างการสร้างวัตถุจากคลาสแบบไม่มี __str__ ฟังก์ชัน

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

## The __str__() Function

### The string representation of an object WITH the __str__() function:

| Person |
| --- |
| Name<br>age |
| __str__() |

The __str__() function controls what should be returned when the class object is represented as a string.
If the __str__() function is not set, the string representation of the object is returned:

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

  def __str__(self):
    return f"{self.name}({self.age})"

p1 = Person("Asst. Prof. Dr. Nattapong Songneam", 46)

print(p1)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppData/Lo
Asst. Prof. Dr. Nattapong Songneam(46)
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# Object Methods

Insert a function that prints a greeting, and execute it on the p1 object:

| Person |
| --- |
| name<br>age |
| + myfunc() |

Objects can also contain methods. Methods in objects are functions that belong to the object.
Let us create a method in the Person class:

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

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person"Asst. Prof. Dr. Nattapong Songneam", 46)
p1.myfunc()
```

## ผลลัพธ์การทำงานของโปรแกรม

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppDa
Hello my name is Asst. Prof. Dr. Nattapong Songneam
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# 8.2 การสร้างคลาสและออบเจ็ค

## การเรียกใช้งานเมธอด

Object.Method()

| Person |
| --- |
| name<br>age |
| + myfunc() |

**ดังตัวอย่าง**

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

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person"Asst. Prof. Dr. Nattapong Songneam", 46)
p1.myfunc()
```

## ผลลัพธ์การทำงานของโปรแกรม

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppDa
Hello my name is Asst. Prof. Dr. Nattapong Songneam
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# 8.2 การสร้างคลาสและออบเจ็ค

## The self Parameter

### Use the words mysillyobject and abc instead of self:

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.
It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

| Person |
| --- |
| name<br>age |
| + myfunc() |

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

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person"Asst. Prof. Dr. Nattapong Songneam", 46)
p1.myfunc()
```

## ผลลัพธ์การทำงานของโปรแกรม

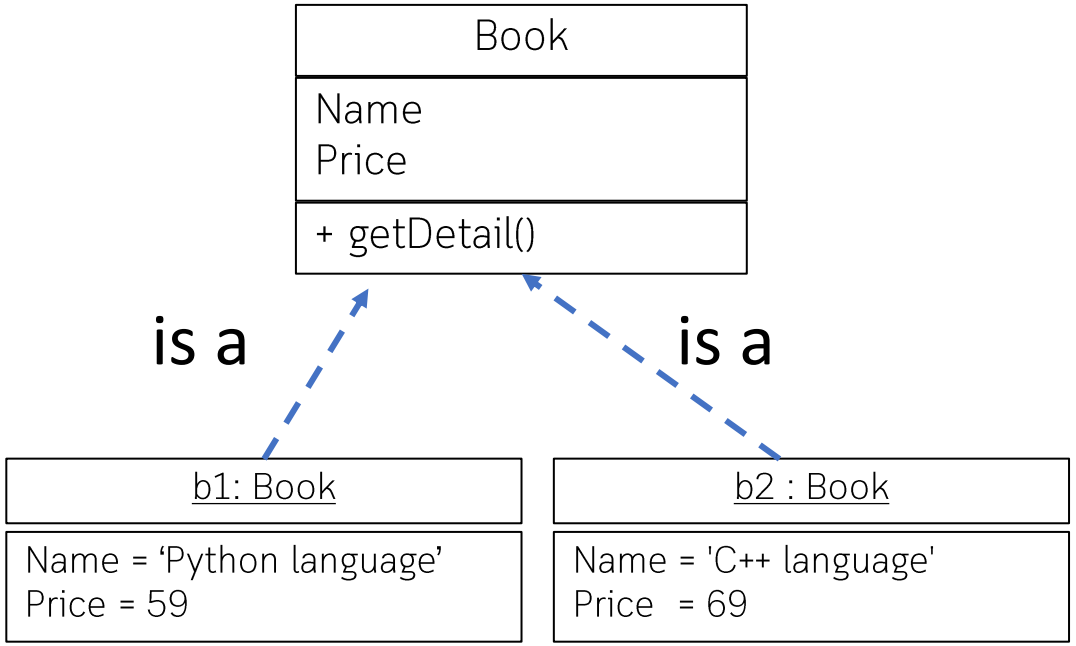
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppDa
Hello my name is Asst. Prof. Dr. Nattapong Songneam
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# 8.2 การสร้างคลาสและออบเจ็ค

**ตัวอย่าง**

```
class Book:

    def __init__(self, name, price):
        self.name = name
        self.price = price

    def getDetail(self):
        print('Name: %s' % self.name)
        print('Price: %d USD' % self.price)
```




ในตัวอย่าง เราได้สร้างคลาส Book และภายในมีเมธอด __init__() ซึ่งเป็นคอนสตรัคเตอร์ (Constructor) ซึ่งจะถูกเรียกอัตโนมัติเมื่อออบเจ็คถูกสร้างสำเร็จ พารามิเตอร์แรกของเมธอดจะเป็น self เสมอ นี่จะใช้เป็นตัวแปรในการอ้างถึงออบเจ็คปัจจุบัน คลาสนี้จะมีสองแอตทริบิวต์คือ name และ price ใช้สำหรับเก็บชื่อและราคาของหนังสือของแต่ละออบเจ็ค หลังจากที่เราได้สร้างคลาสเสร็จแล้ว ต่อไปเราจะนำมาสร้างออบเจ็ค

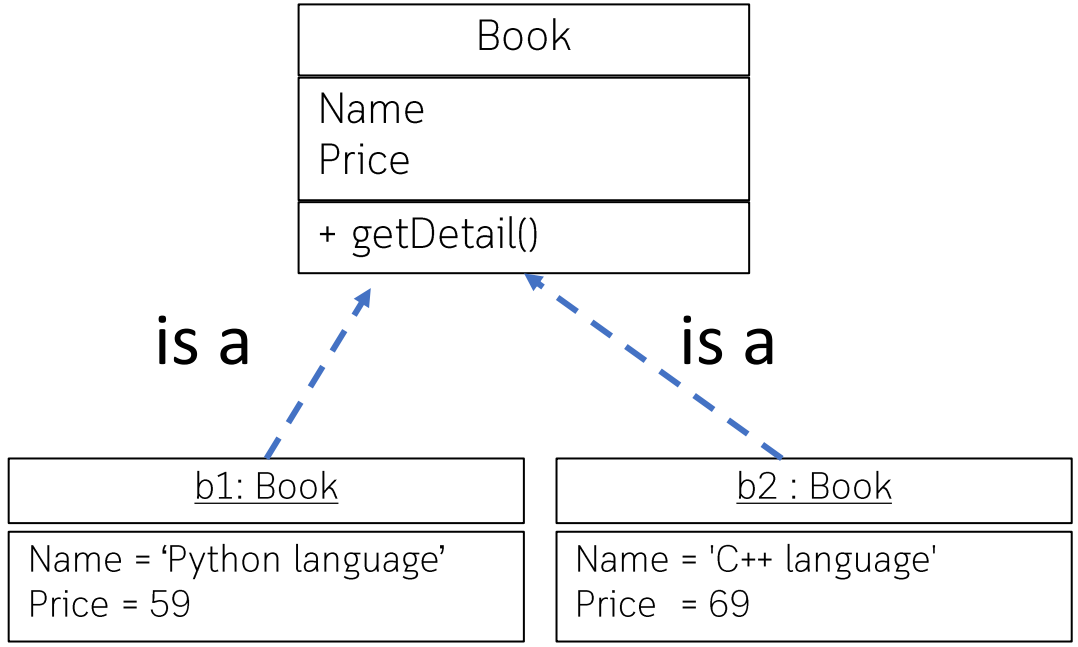
ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.2 การสร้างคลาสและออบเจ็ค

เราจะใช้คำสั่ง class สำหรับสร้างคลาสในภาษา Python และตามด้วยชื่อของคลาส ClassName ชื่อของคลาสควรจะขึ้นต้นด้วยตัวใหญ่และเป็นรูปแบบ camel case หลังจากนั้นเป็นคำสั่งในการกำหนดตัวแปรและเมธอดของคลาส ต่อไปมาดูตัวอย่างการสร้างคลาสในภาษา Python

## ตัวอย่าง การสร้างวัตถุ

```
b1 = Book('Python language', 59)
b2 = Book('C++ language', 69)

b1.getDetail()
b2.getDetail()

b1.price = 99

b1.getDetail()
```




**ในตัวอย่าง เราได้สร้างคลาส Book และภายในมีเมธอด __init__() ซึ่งเป็นคอนสตรัคเตอร์ (Constructor) ซึ่งจะถูกเรียกอัตโนมัติเมื่อออบเจ็คถูกสร้างสำเร็จ พารามิเตอร์แรกของเมธอดจะเป็น self เสมอ นี่จะใช้เป็นตัวแปรในการอ้างถึงออบเจ็คปัจจุบัน คลาสนี้จะมีสองแอตทริบิวต์คือ name และ price ใช้สำหรับเก็บชื่อและราคาของหนังสือของแต่ละออบเจ็ค หลังจากที่เราได้สร้างคลาสเสร็จแล้ว ต่อไปเราจะนำมาสร้างออบเจ็ค**

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.2 การสร้างคลาสและออบเจ็ค

**ตัวอย่าง**

```
Name: Python language
Price: 59 USD
Name: C++ language
Price: 69 USD
Name: Python language
Price: 99 USD
```

ในตัวอย่าง เราได้สร้างตัวแปรออบเจ็คจากคลาส Book สองตัวแปร คือ b1 และ b2 สำหรับพารามิเตอร์แรกในคอนสตรัคเตอร์นั้นเราได้ละเว้นไป เพราะ Python จะใส่เป็นออบเจ็คปัจจุบันให้อัตโนมัติ ดังนั้นพารามิเตอร์ที่เราจะต้องใส่คือชื่อและราคาของหนังสือ

| Book |
| --- |
| Name<br>Price |
| + getDetail() |

**หลังจากที่ออบเจ็คถูกสร้างแล้ว ทั้ง b1 และ b2 จะมีแอตทริบิวต์ name price และเมธอด getDetail() เป็นของตัวเองที่ไม่เกี่ยวข้องกัน ในการเข้าถึงสมาชิกภายในออบเจ็คจะใช้เครื่องหมายจุด (.) ดังนั้นเมื่อเราเรียก b1.getDetail() จะเป็นการแสดงรายละเอียดข้อมูลในออบเจ็ค b1 ซึ่ง b2 ก็เช่นเดียวกัน และต่อมาเราได้เปลี่ยนค่า price ของออบเจ็ค b1 ให้มีค่าเป็น 99 และแสดงรายละเอียดอีกครั้ง และนี่เป็นผลลัพธ์การทำงานของโปรแกรม**

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.2 การสร้างคลาสและออบเจ็ค

**ตัวอย่าง**

```
class Person:
    def __init__(self, firstName, lastName):
        self.firstName = firstName
        self.lastName = lastName
    def getName(self):
        return self.firstName + ' ' + self.lastName
p = Person('Chase', 'Rice')
p.career = 'Singer'
p.country = 'USA'
print('Name: ' + p.getName())
print('Career: ' + p.career)
print('Country: ' + p.country)
p2 = Person('Max', 'Graham')
p2.genres = ['Electronica', 'trance', 'tech house', 'techno']
print('Name: ' + p2.getName())
print('Genres: ', p2.genres)
```

**ในตอนนี้ คุณได้เห็นแล้วว่าคลาสหนึ่งคลาสสามารถนำไปสร้างเป็นออบเจ็คกี่อันก็ได้ ซึ่งนี่เองถือว่าเป็นแนวคิดที่สำคัญของการเขียนโปรแกรมเชิงวัตุในการนำโค้ดกลับมาใช้ซ้ำ**

| Person |
| --- |
| firstName<br>lastName |
| + getName() |

**หลังจากที่คุณได้เห็นการประกาศคลาสและสร้างออบเจ็คในเบื้องต้นแล้ว ต่อไปมาดูตัวอย่างเพิ่มเติมสำหรับการทำงานกับคลาสในภาษา Python**

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

Get Started | class_person.py ×

class_person.py > Person

```python
class Person:

    def __init__(self, firstName, lastName):
        self.firstName = firstName
        self.lastName = lastName

    def getName(self):
        return self.firstName + ' ' + self.lastName

p = Person('Chase', 'Rice')
p.career = 'Singer'
p.country = 'USA'

print('Name: ' + p.getName())
print('Career: ' + p.career)
print('Country: ' + p.country)

p2 = Person('Max', 'Graham')
p2.genres = ['Electronica', 'trance', 'tech house', 'techno']

print('Name: ' + p2.getName())
print('Genres: ', p2.genres)
```

| Person |
| --- |
| firstName<br>lastName |
| + getName() |

**นี่เป็นผลลัพธ์การทำงานของโปรแกรม คุณได้เห็นแล้วว่าการเขียนโปรแกรมในภาษา Python นั้นค่อนข้างยืดหยุ่น คุณไม่จำเป็นต้องประกาศแอตทริบิวต์ทั้งหมดไว้ในตอนแรกก็ได้ คุณอาจจะเพิ่มเข้ามาในภายหลังเฉพาะออบเจ็คที่ต้องการได้ เหมือนในออบเจ็ค p2 ไม่ได้ต้องการมีแอตทริบิวต์เหมือนกับออบเจ็ค p แต่ทั้งสองยังมีแอตทริบิวต์บางอย่างที่เหมือนกัน**

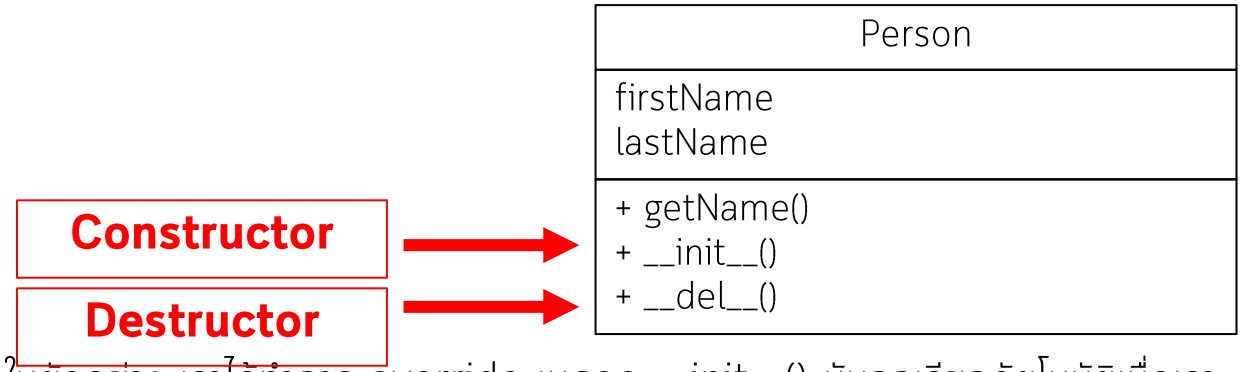
```
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppData/Local/Micr
Name: Chase Rice
Career: Singer
Country: USA
Name: Max Graham
Genres:  ['Electronica', 'trance', 'tech house', 'techno']
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

# 8.3 Constructor และ Destructor

**ตัวอย่าง**

```
class Person:

    def __init__(self, firstName, lastName):
        self.firstName = firstName
        self.lastName = lastName
        print('Object was created')

    def getName(self):
        return self.firstName + ' ' + self.lastName

    def __del__(self):
        print('Object was destroyed')

p = Person('Chase', 'Rice')
print(p.getName())
del p
```

ต่อมาเราจะพูดเกี่ยวกับการใช้งาน Constructor และ Destructor ในภาษา Python นั้นมีเมธอดพิเศษ (Special methods) ที่สร้างให้อัตโนมัติเมื่อคุณสร้างคลาสขึ้นมา เพื่อใช้งานเราจะต้อง override เมธอดเหล่านั้น เมธอดแรกคือ __init__() ซึ่งเมธอดนี้จะทำงานเมื่อออบเจ็คถูกสร้างสำเร็จ หรือเรียกว่า Constructor มันมักจะใช้ในการกำหนดแอตทริบิวต์และค่าเริ่มต้นให้กับออบเจ็ค ต่อมาคือเมธอด __del__() ซึ่งเมธอดนี้จะทำงานเมื่อออบเจ็คถูกทำลาย หรือเรียกว่า Destructor มาดูตัวอย่างการใช้งาน




ในตัวอย่าง เราได้ทำการ override เมธอด __init__() มันถูกเรียกอัตโนมัติเมื่อเราสร้างออบเจ็คสำเร็จ จากคำสั่ง Person('Chase', 'Rice') หลังจากนั้นเราได้ทำการ override เมธอด __del__() ซึ่งเมธอดนี้จะถูกเรียกใช้งานก่อนที่ออบเจ็คจะถูกทำลาย ในโค้ดเราได้ทำลายออบเจ็คด้วยคำสั่ง del p ซึ่งจะทำให้ออบเจ็คถูกลบออกไปจากหน่วยความจำ

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.3 Constructor และ Destructor

st_contructor_desctructor.py > ...

```python
class Person:

    def __init__(self, firstName, lastName):
        self.firstName = firstName
        self.lastName = lastName
        print('Object was created')

    def getName(self):
        return self.firstName + ' ' + self.lastName

    def __del__(self):
        print('Object was destroyed')

p = Person('Chase', 'Rice')
print(p.getName())
del p
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม เมื่อสร้างออบเจ็คสำเร็จเราได้แสดงข้อความในเมธอด __init__() ว่าสร้างออบเจ็คสำเร็จแล้ว และเมื่อเราลบออบเจ็คโปรแกรมแสดงข้อความในเมธอด __del__() ว่าออบเจ็คถูกทำลายไปแล้ว ซึ่งการทำงานเหล่านี้จะเกิดขึ้นอัตโนมัติ แน่นอนว่าในภาษา Python เรามักจะใช้คอนสตรัคเตอร์เสมอ แต่ว่า Destructor มักจะไม่ได้ใช้บ่อยนัก อย่างไรก็ตามยังมีเมธอดพิเศษอื่นๆ อีกที่เรายังไม่ได้พูดถึงในบทนี้

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users
Object was created
Chase Rice
Object was destroyed
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

# 8.3 Constructor และ Destructor

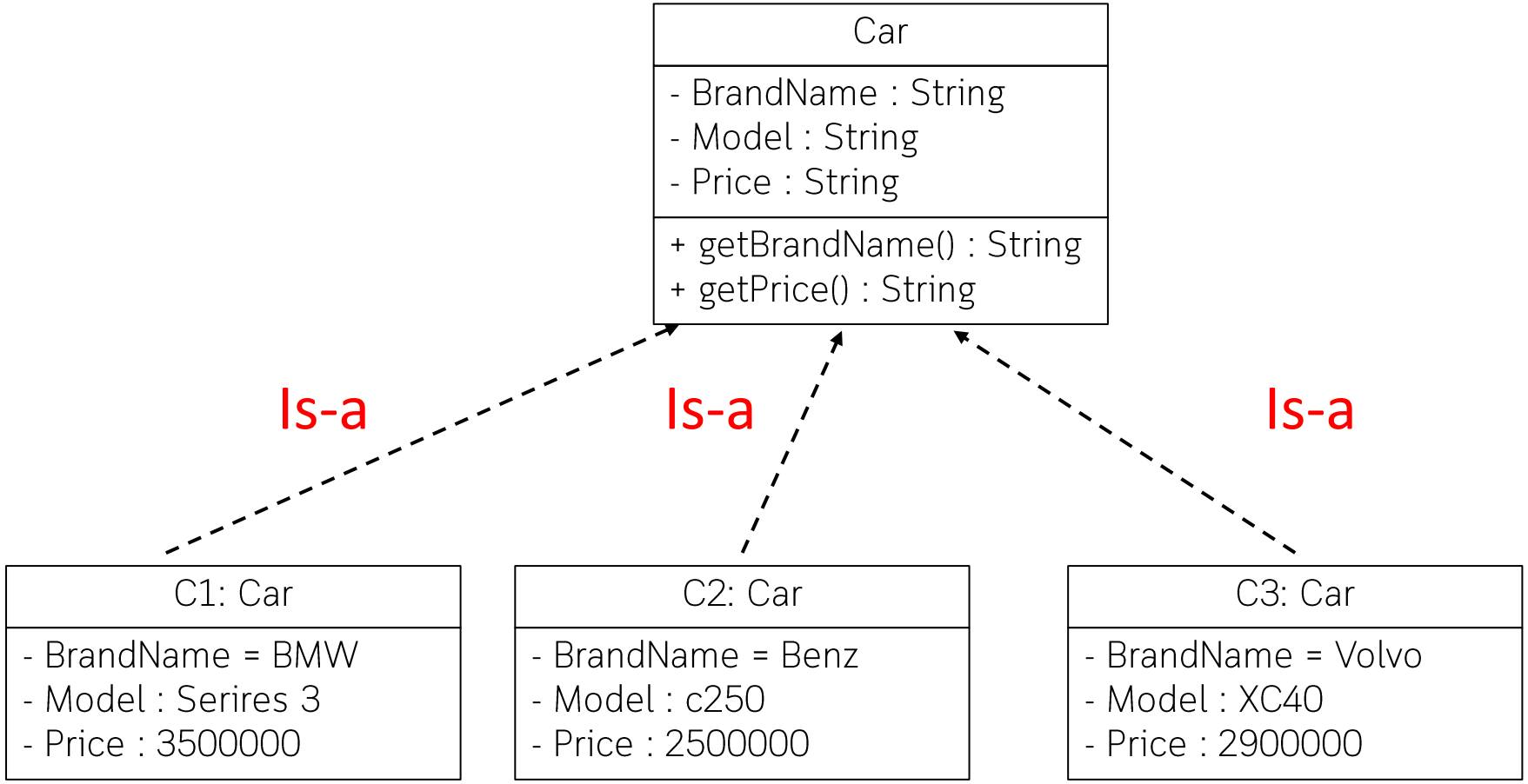
นี่เป็นผลลัพธ์การทำงานของโปรแกรม เมื่อสร้างออบเจ็คสำเร็จเราได้แสดงข้อความในเมธอด __init__() ว่าสร้างออบเจ็คสำเร็จแล้ว และเมื่อเราลบออบเจ็คโปรแกรมแสดงข้อความในเมธอด __del__() ว่าออบเจ็คถูกทำลายไปแล้ว ซึ่งการทำงานเหล่านี้จะเกิดขึ้นอัตโนมัติ แน่นอนว่าในภาษา Python เรามักจะใช้คอนสตรัคเตอร์เสมอ แต่ว่า Destructor มักจะไม่ได้ใช้บ่อยนัก อย่างไรก็ตามยังมีเมธอดพิเศษอื่นๆ อีกที่เรายังไม่ได้พูดถึงในบทนี้

| Car |
| --- |
| - BrandName : String<br>- Model : String<br>- Price : String |
| + getBrandName() : String<br>+ getPrice() : String<br>+ __init__()<br>+ __del__() |

จงสร้างคลาส และวัตถุต่าง ๆ ต่อไปนี้พร้อมทั้ง Contructor และ Descturctor

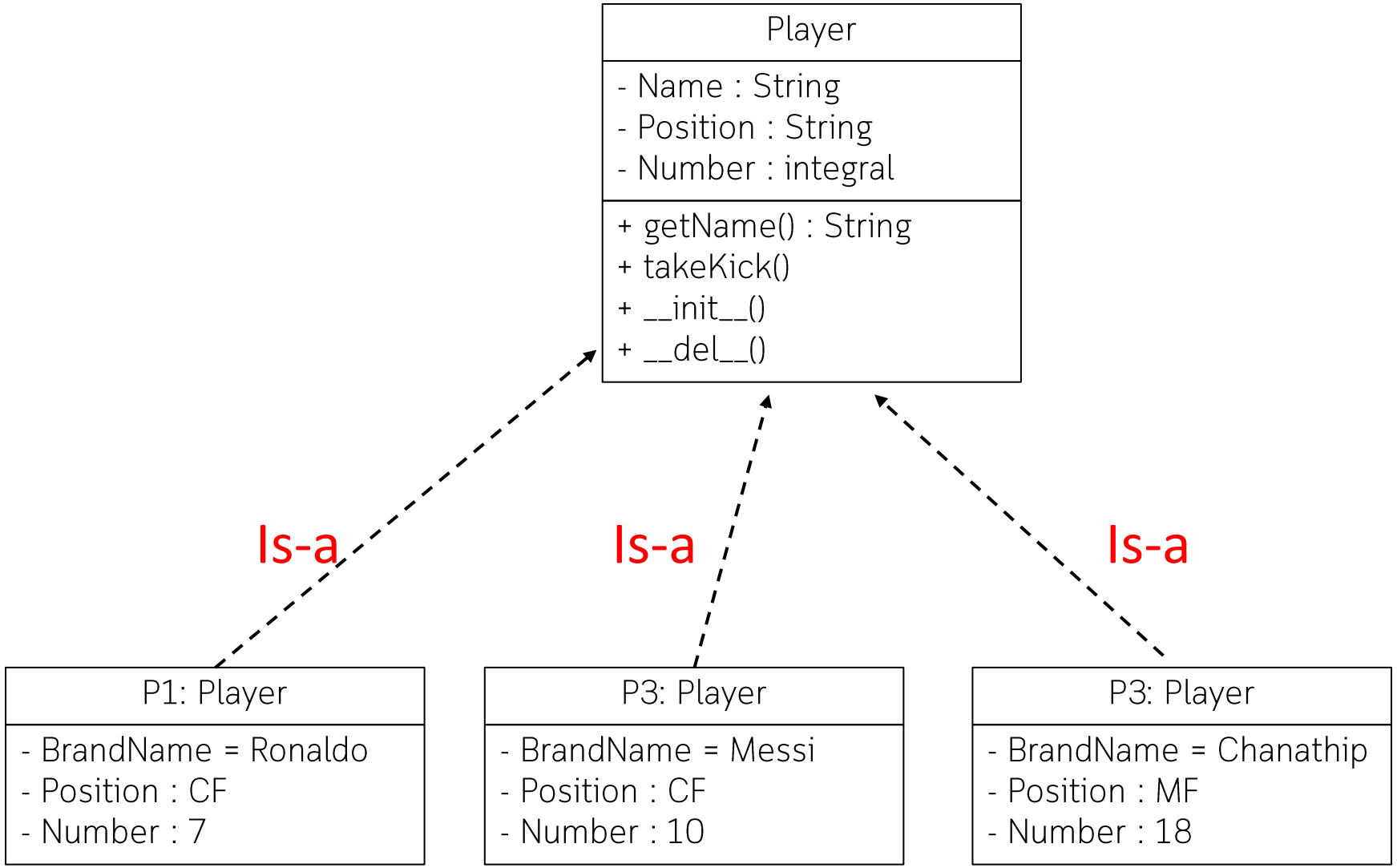
# 8.3 Constructor และ Destructor

จงสร้างคลาส และวัตถุต่าง ๆ ต่อไปนี้พร้อมทั้ง Contructor และ Descturctor



# 8.3 Constructor และ Destructor

จงสร้างคลาส และวัตถุต่าง ๆ ต่อไปนี้พร้อมทั้ง Contructor และ Descturctor



# 8.4 Static variables และ Static methods

## ตัวอย่าง

```python
# ตัวอย่างคลาส
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.speed = 0

    def accelerate(self, increment):
        self.speed += increment

    def brake(self, decrement):
        if self.speed - decrement >= 0:
            self.speed -= decrement
        else:
            print("ความเร็วเป็นลบไม่ได้")

  def honk(self):
        print(f"{self.make} {self.model} ทำเสียงนกฮูก")

# สร้างวัตถุจากคลาส Car
my_car = Car("Toyota", "Camry", 2022)

# เรียกใช้งานเมธอดในวัตถุ
print(f"รถของฉัน: {my_car.make} {my_car.model} ปี {my_car.year}")
my_car.accelerate(20)
print(f"ความเร็วปัจจุบัน: {my_car.speed} กิโลเมตรต่อชั่วโมง")
my_car.brake(10)
print(f"ความเร็วหลังเบรก: {my_car.speed} กิโลเมตรต่อชั่วโมง")
my_car.honk()
```

ในตัวอย่างนี้:

เราสร้างคลาส Car ที่มีคุณสมบัติและเมธอดต่าง ๆ สำหรับรถ เช่น accelerate เพื่อเพิ่มความเร็ว และ brake เพื่อลดความเร็ว

เราสร้างวัตถุ my_car จากคลาส Car และเรียกใช้เมธอดต่าง ๆ บนวัตถุ ผลลัพธ์ของการเรียกใช้งานเมธอดและคุณสมบัติของวัตถุ my_car จะแสดงค่าต่าง ๆ ที่เกี่ยวข้องกับรถของเรา และเมธอด honk จะพิมพ์ข้อความที่รถทำเสียงนกฮูกออกมาในเอาท์พุต.

| car |
| --- |
| Make<br>Model<br>year |
| +<br>+ |

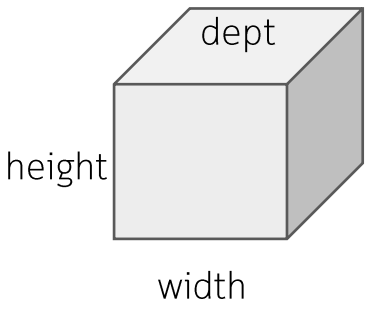
ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# 8.4 Static variables และ Static methods

## ตัวอย่าง

```
class Box:

    # shared variable for all object create by this class
    color = 'green'

    # class method for object
    def __init__(self, width, height, dept):
        self.width = width
        self.height = height
        self.dept = dept

     # class method for object
    def getVolume(self):
        return self.width * self.height * self.dept

    @staticmethod
    def compare(a, b):
        if a.getVolume() > b.getVolume():
            return 'greater than'
        elif a.getVolume() == b.getVolume():
            return 'equal'
        else:
            return 'less than'

a = Box(2, 3, 4)
b = Box(1, 2, 5)
Box.color = 'red'
print('Box a volume = %d' % a.getVolume())
print('Box b volume = %d' % b.getVolume())
print('Box a color = %s' % a.color)
print('Box b color = %s' % b.color)
print('Box a volume a is %s box b' % Box.compare(a, b))
```

สำหรับเรื่องสุดท้ายที่คุณจะได้เรียนในบทนี้ คือการใช้งาน static variable และ static method โดย static variable หรือคลาสแอตทริบิวต์ คือตัวแปรที่ประกาศภายในคลาสซึ่งตัวแปรนี้จะแชร์กับออบเจ็คทุกอันที่สร้างจากคลาสนี้ ส่วน static method เป็นเมธอดที่สร้างไว้ในคลาสแต่ไม่ได้มีส่วนเกี่ยวกับข้องกับการจัดการออบเจ็ค มาดูตัวอย่างการใช้งาน




ในตัวอย่าง เราได้ประกาศคลาส Box โดยคลาสนี้มีตัวแปรของคลาส คือ color นั่นหมายความว่าตัวแปรนี้จะถูกใช้งานร่วมกันกับทุกออบเจ็คที่สร้างจากคลาสนี้ ต่อมาภายในเมธอด __init__() เป็นการกำหนดข้อมูลให้กับออบเจ็คแต่ละอันที่จะมีความกว้าง ความยาว และความสูง และเมธอด getVolume() สำหรับรับปริมาตรของกล่อง

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

ในตัวอย่าง คือประกาศคลาส Box โดยคลาสนี้มีตัวแปรของคลาส คือ color นั่นหมายความว่าตัวแปรนี้จะถูกใช้งานร่วมกันกับทุกออบเจ็คที่สร้างจากคลาสนี้ ต่อมาภายในเมธอด __init__() เป็นการกำหนดข้อมูลให้กับออบเจ็คแต่ละอันที่จะมีความกว้าง ความยาว และความสูง และเมธอด getVolume() สำหรับรับปริมาตรของกล่อง

```
class Box:

    # shared variable for all object create by this class
    color = 'green'

    # class method for object
    def __init__(self, width, height, dept):
        self.width = width
        self.height = height
        self.dept = dept

     # class method for object
    def getVolume(self):
        return self.width * self.height * self.dept

    @staticmethod
    def compare(a, b):
        if a.getVolume() > b.getVolume():
            return 'greater than'
        elif a.getVolume() == b.getVolume():
            return 'equal'
        else:
            return 'less than'
```

```
a = Box(2, 3, 4)
b = Box(1, 2, 5)

Box.color = 'red'

print('Box a volume = %d' % a.getVolume())
print('Box b volume = %d' % b.getVolume())

print('Box a color = %s' % a.color)
print('Box b color = %s' % b.color)

print('Box a volume a is %s box b' % Box.compare(a, b))
```

ภายในคลาส Box เราได้สร้าง static method compare() โดยใช้ decorator @staticmethod นำหน้าก่อนหนึ่งบรรทัด เมธอดนี้ไม่ได้มีส่วนเกี่ยวข้องกับข้อมูลภายในคลาส สังเกตว่ามันไม่มีพารามิเตอร์ self ถูกส่งเข้ามาสำหรับการเรียกใช้งาน static variables หรือ static method นั้นจะไม่ขึ้นกับออบเจ็ค นั่นหมายความว่าเราจะใช้ชื่อของคลาสแทน ในคำสั่ง Box.color = 'red' เป็นการกำหนดสีให้กับกล่องทุกกล่องเป็นสีแดง และคำสั่ง Box.compare(a, b) เป็นการเปรียบเทียบปริมาตรของกล่องทางซ้ายว่ามากกว่าทางขวาหรือไม่

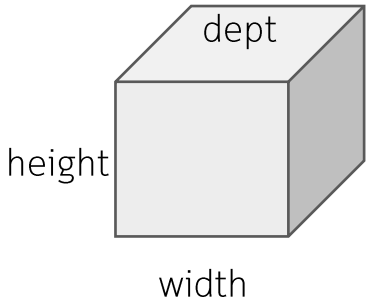
# 8.4 Static variables และ Static methods

## ผลลัพธ์การทำงาน

```
Box a volume = 24
Box b volume = 10
Box a color = red
Box b color = red
Box a volume a is greater than box b
```

นี่เป็นผลลัพธ์การทำงานของโปรแกรม จะเห็นว่าเราได้ทำการเปลี่ยนค่าสีของกล่องเป็นสีแดงในคำสั่งเดียว แต่มันถูกนำไปใช้กับทุกกล่อง และเมธอดสำหรับเปรียบเทียบปริมาตรของกล่องมันสามารถใช้ได้โดยไม่ขึ้นกับออบเจ็คใดๆ นั่นหมายความว่า static method อาจจะเป็นเมธอดที่ทำงานเกี่ยวกับออบเจ็คที่สร้างจากคลาสนี้ แต่ไม่ได้เป็นเมธอดสำหรับจัดการข้อมูลในคลาสโดยตรง

| Box |
| --- |
| Width<br>Height<br>Dept |
| + getVolumn()<br>+ compare() |




ภายในคลาส Box เราได้สร้าง static method compare() โดยใช้ decorator @staticmethod นำหน้าก่อนหนึ่งบรรทัด เมธอดนี้ไม่ได้มีส่วนเกี่ยวข้องกับข้อมูลภายในคลาส สังเกตว่ามันไม่มีพารามิเตอร์ self ถูกส่งเข้ามา สำหรับการเรียกใช้งาน static variables หรือ static method นั้นจะไม่ขึ้นกับออบเจ็ค นั่นหมายความว่าเราจะใช้ชื่อของคลาสแทน ในคำสั่ง Box.color = 'red' เป็นการกำหนดสีให้กับกล่องทุกกล่องเป็นสีแดง และคำสั่ง Box.compare(a, b) เป็นการเปรียบเทียบปริมาตรของกล่องทางซ้ายว่ามากกว่าทางขวาหรือไม่

ในบทนี้ ได้เรียนรู้เกี่ยวกับคลาสและออบเจ็คในภาษา Python ซึ่งเป็นสิ่งที่คุณควรจะทำความเข้าใจเพื่อที่จะใช้ประโยชน์จาก OOP ซึ่งมันช่วยให้การเขียนโปรแกรมมีประสิทธิภาพและรวดเร็วขึ้น นอกจากนี้เรายังพูดเกี่ยวกับเมธอดพิเศษ Constructor และ Destructor และสมาชิกที่เป็น static ของคลาส สำหรับเนื้อหาของ OOP ยังไม่จบเพียงเท่านี้ มันยังมีการสืบทอด (Inheritance) ซึ่งคุณจะได้เรียนในบทต่อไป

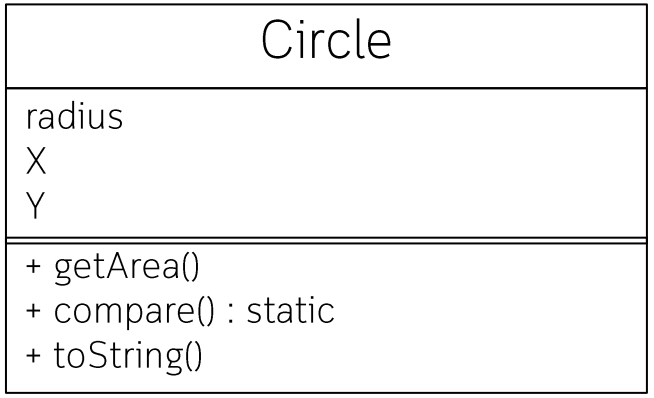
ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

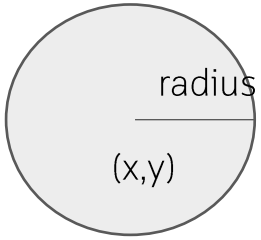
# 8.4 Static variables และ Static methods

## ตัวอย่าง

```
class Circle:
    # shared variable for all object create by this class
    color = 'pink'
    # class method for object
    def __init__(self, x, y, radius):
        self.X = x
        self.Y = y
        self.radius = radius
     # class method for object
    def getArea(self):
        return 3.14*self.radius * self.radius

    @staticmethod
    def compare(a, b):
        if a.getArea() > b.getArea():
            return 'greater than'
        elif a.getArea() == b.getArea():
            return 'equal'
        else:
            return 'less than'
a = Circle(2, 3, 10)
b = Circle(1, 2, 5)
Circle.color = 'red'
print('Circle a Area = %d' % a.getArea())
print('Circle b Area = %d' % b.getArea())
print('Circle a color = %s' % a.color)
print('Circle b color = %s' % b.color)
print('Circle a area a is %s Circle b' % Circle.compare(a, b)
```

สำหรับเรื่องสุดท้ายที่คุณจะได้เรียนในบทนี้ คือการใช้งาน static variable และ static method โดย static variable หรือคลาสแอตทริบิวต์ คือตัวแปรที่ประกาศภายในคลาสซึ่งตัวแปรนี้จะแชร์กับออบเจ็คทุกอันที่สร้างจากคลาสนี้ ส่วน static method เป็นเมธอดที่สร้างไว้ในคลาสแต่ไม่ได้มีส่วนเกี่ยวกับข้องกับการจัดการออบเจ็ค มาดูตัวอย่างการใช้งาน




Pi =3.14

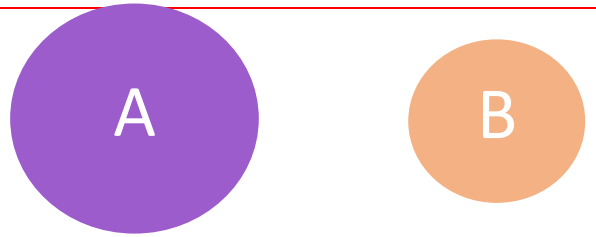
ในตัวอย่าง เราได้ประกาศคลาส Box โดยคลาสนี้มีตัวแปรของคลาส คือ color นั่นหมายความว่าตัวแปรนี้จะถูกใช้งานร่วมกันกับทุกออบเจ็คที่สร้างจากคลาสนี้ ต่อมาภายในเมธอด __init__() เป็นการกำหนดข้อมูลให้กับออบเจ็คแต่ละอันที่จะมีความกว้าง ความยาว และความสูง และเมธอด getVolume() สำหรับรับปริมาตรของกล่อง

ที่มา : https://www.diagrams.net/blog/uml-class-diagrams

# คลาส วงกลม (Circle)

```
class Circle:

    # shared variable for all object create by this class
    color = ‘pink'

    # class method for object
    def __init__(self, x, y, radius):
        self.X = x
        self.Y = y
        self.radius = radius

     # class method for object
    def getArea(self):
        return 3.14*self.radius * self.radius

    @staticmethod
    def compare(a, b):
        if a.getArea() > b.getArea():
            return 'greater than'
        elif a.getArea() == b.getArea():
            return 'equal'
        else:
            return 'less than'
```

```
a = Circle(2, 3, 10)
b = Circle(1, 2, 5)

Circle.color = 'red'

print(‘Circle a Area = %d' % a.getArea())
print(‘Circle b Area = %d' % b.getArea())

print(‘Circle a color = %s' % a.color)
print(‘Circle b color = %s' % b.color)

print(‘Circle a area a is %s Circle b’ % Circle.compare(a, b))
```




ภายในคลาส Circle เราได้สร้าง static method compare() โดยใช้ decorator @staticmethod นำหน้าก่อนหนึ่งบรรทัด เมธอดนี้ไม่ได้มีส่วนเกี่ยวข้องกับข้อมูลภายในคลาส สังเกตว่ามันไม่มีพารามิเตอร์ self ถูกส่งเข้ามา สำหรับการเรียกใช้งาน static variables หรือ static method นั้นจะไม่ขึ้นกับออบเจ็ค นั่นหมายความว่าเราจะใช้ชื่อของคลาสแทน ในคำสั่ง Circle.color = 'red' เป็นการกำหนดสีให้กับกล่องทุกกล่องเป็นสีแดง และคำสั่ง Circle.compare(a, b) เป็นการเปรียบเทียบปริมาตรของวงกลมทางซ้ายว่ามากกว่าทางขวาหรือไม่

# 8.5 Delete object

Delete the p1 object:

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

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)

del p1

print(p1)
```

**ผลลัพธ์การทำงาน**

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppData/Local/Microsoft/Winc
Traceback (most recent call last):
  File "c:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons\oop_del_obj.py", line 13, in <module>
    print(p1)
NameError: name 'p1' is not defined
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# 8.6 Delete Object Properties

You can delete properties on objects by using the del keyword:

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

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)

del p1.age

print(p1.age)
```

**ผลลัพธ์การทำงาน**

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/User/AppData/Local/Microsoft/WindowsApps/python3.10.exe
Traceback (most recent call last):
  File "c:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons\oop_del_obj_attr.py", line 13, in <module>
    print(p1.age)
AttributeError: 'Person' object has no attribute 'age'
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

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

# 8.7 The pass Statement

**คลาสไม่สามารถเป็นคลาสว่างเปล่าได้**
**class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.**

```
class Person:
  pass

# having an empty class definition like this, would raise an error without the pass statement
```

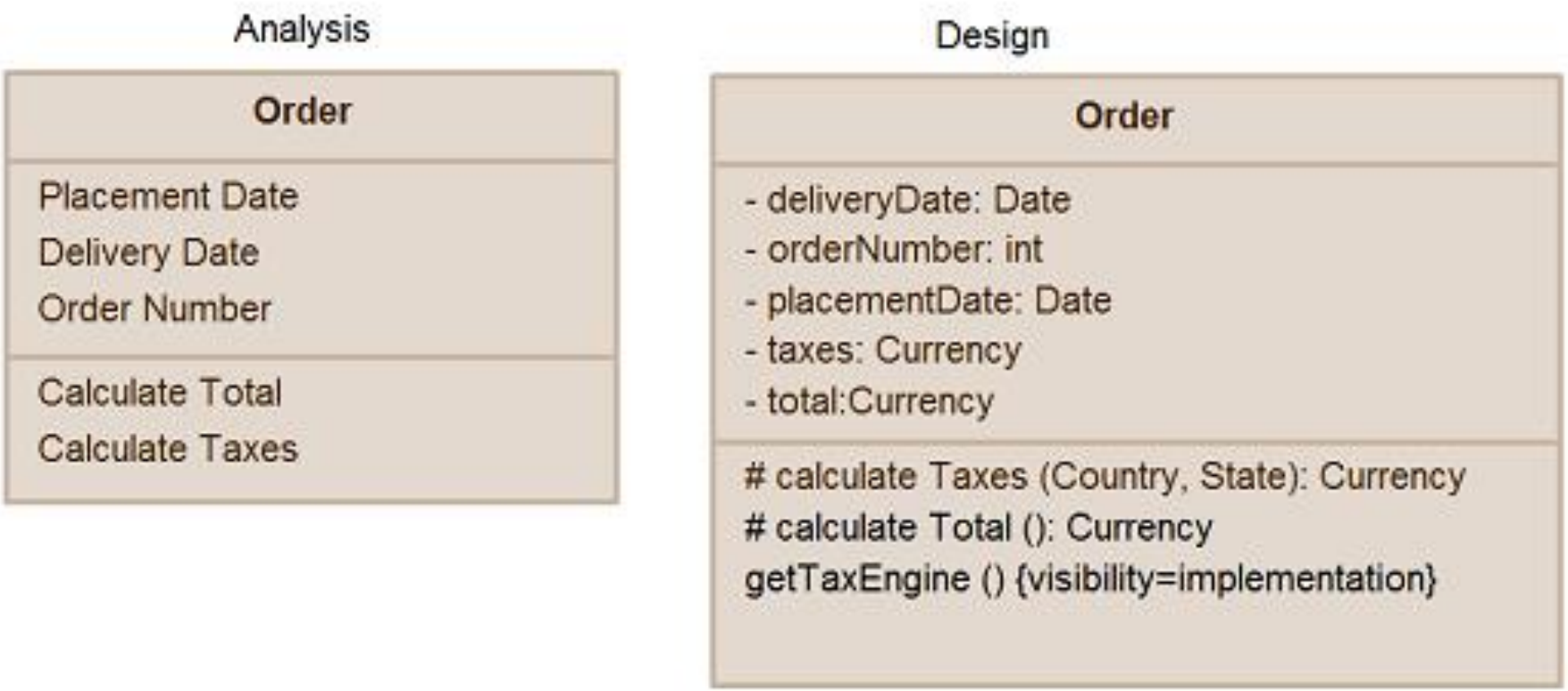
**ผลลัพธ์การทำงาน**

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

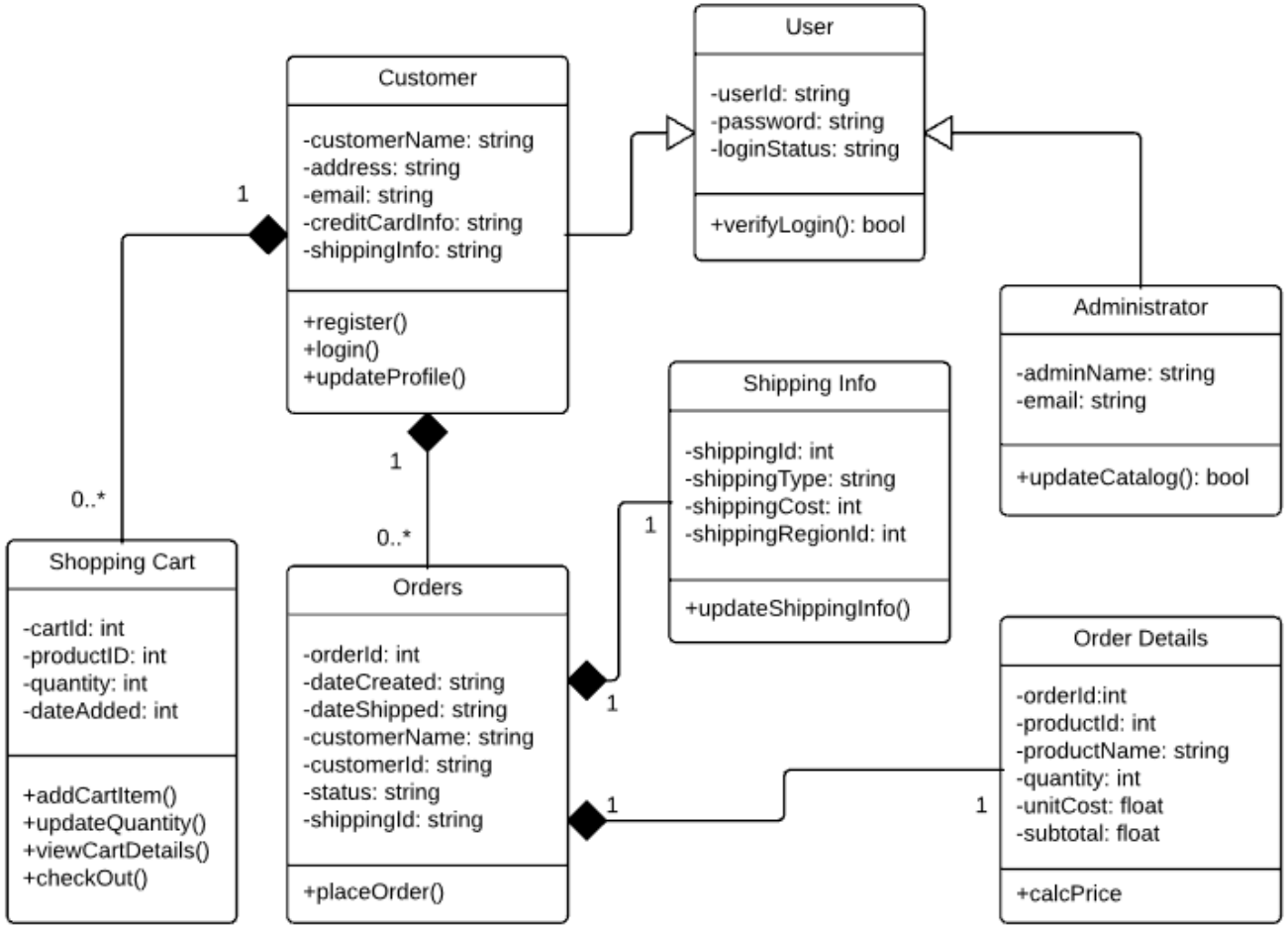
## การวิเคราะห์และออกแบบคลาส

- มุมมองการวิเคราะห์ และมุมมองการออกแบบคลาส

A - Analysis and design versions of a class



# Online shopping system class diagram example



# วันที่และเวลา ในภาษา Python

ในหัวข้อนี้ จะได้เรียนรู้เกี่ยวกับวันที่และเวลาในภาษา Python โดยจะพูดถึงการทำงานพื้นฐานเกี่ยวกับวันที่และเวลา เช่น การแสดงเวลาปัจจุบัน การบวกลบเวลา เขตเวลา และแนะนำคลาสและเมธอดเกี่ยวกับวันที่และเวลา

- วันที่และเวลาปัจจุบัน
- การสร้างออบเจ็ควันที่และเวลา
- Unix timestamp
- การบวกลบวันที่และเวลา
- คลาสและโมดูลเกี่ยวกับวันที่และเวลา

```
from datetime import datetime

# Get current date and time
now = datetime.today()

# Display using difference formats
print(now)
print(now.isoformat())
print(now.ctime())
```

ข้อมูลเพิ่มเติม :: https://docs.python.org/3/library/datetime.html

## การสร้างออบเจ็คของวันที่และเวลา

ในตัวอย่างก่อนหน้าเราได้พูดถึงการรับเอาวันที่และเวลาปัจจุบันด้วยเมธอด today() ในตัวอย่างนี้ เราจะพูดถึงการสร้างออบเจ็คจากวันที่และเวลาที่กำหนด โดยการใช้คอนสตรัคเตอร์ของคลาส datetime ซึ่งนี่อาจมีประโยชน์เมื่อเรามีค่าของวันที่อยู่แล้วอาจจากฐานข้อมูลหรือไฟล์ และต้องการนำมาสร้างเป็นออบเจ็ควันที่เพื่อใช้งานในโปรแกรม

ในตัวอย่างนี้แสดงการสร้างออบเจ็คของวันที่โดยการระบุส่วนของวันที่และเวลาเป็นตัวเลขด้วยคลาสคอนสตรัคเตอร์ datetime

**create_date.py**

```
from datetime import datetime

birthDay = datetime(1980, 3, 12, 10, 0, 0)
chrismasDay = datetime(2020, 12, 25)

print("My birth day: %s" % birthDay.ctime())
print("Christmas day: %s" % chrismasDay.ctime())
```

### ผลลัพธ์การทำงานของโปรแกรม

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/
My birth day: Wed Mar 12 10:00:00 1980
Christmas day: Fri Jan 27 00:00:00 2023
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

ข้อมูลเพิ่มเติม :: https://docs.python.org/3/library/datetime.html

## การสร้างออบเจ็คของวันที่และเวลา

ในตัวอย่างนี้ เป็นการสร้างออบเจ็คของวันที่จากคลาส datetime สำหรับเก็บค่าวันเกิดและวันคริสมาสต์โดยการใช้คลาสคอนสตรัคเตอร์ ในการสร้างวันที่ด้วยวิธีนี้ จะทำให้เราสามารถสร้างออบเจ็คของวันที่สำหรับช่วงเวลาใดๆ ก็ได้ ซึ่งคอนสตรัคเตอร์จะรับพารามิเตอร์โดยมีรูปแบบดังนี้

```
datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)
```

เราจะต้องระบุส่วนต่าง ๆ ของเวลาที่ต้องการสร้างโดยเรียงจาก ปี เดือน และวัน ตามลำดับ ซึ่งค่าของวันที่ในสามพารามิเตอร์แรกจะต้องถูกส่งเสมอ ส่วนพารามิเตอร์ที่เหลือสามาารถที่จะละเว้นได้ ถ้าหากเราไม่ส่งค่าเริ่มต้นที่เป็น 0 จะถูกใช้แทนเหมือนกับในออบเจ็ค chrismasDay

ในบางกรณี คุณอาจต้องการสร้างวันที่จากรูปแบบอื่นๆ นอกจากตัวเลข ยกตัวอย่างเช่น เราอาจมีค่าของวันที่ที่เป็น String ในรูปแบบของ ISO หรือค่าเวลา Timestamp ซึ่งคลาส datetime มีเมธอดสนับสนุนการสร้างวันที่จากรูปแบบดังกล่าว ยกตัวอย่างเช่น

ข้อมูลเพิ่มเติม :: https://docs.python.org/3/library/datetime.html

# การสร้างออบเจ็คของวันที่และเวลา

ในตัวอย่างก่อนหน้าเราได้พูดถึงการรับเอาวันที่และเวลาปัจจุบันด้วยเมธอด today() ในตัวอย่างนี้ เราจะพูดถึงการสร้างออบเจ็คจากวันที่และเวลาที่กำหนด โดยการใช้คอนสตรัคเตอร์ของคลาส datetime ซึ่งนี่อาจมีประโยชน์เมื่อเรามีค่าของวันที่อยู่แล้วอาจจากฐานข้อมูลหรือไฟล์ และต้องการนำมาสร้างเป็นออบเจ็ควันที่เพื่อใช้งานในโปรแกรม

ในตัวอย่างนี้แสดงการสร้างออบเจ็คของวันที่โดยการระบุส่วนของวันที่และเวลาเป็นตัวเลขด้วยคลาสคอนสตรัคเตอร์ datetime

## create_date2.py

```python
from datetime import datetime
birthDay = datetime.fromisoformat("1980-03-12T10:00:00")
chrismasDay = datetime.fromtimestamp(1608829200)
print("My birth day: %s" % birthDay.ctime())
print("Christmas day: %s" % chrismasDay.ctime())
```

## ผลลัพธ์การทำงานของโปรแกรม

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons> & C:/Users/Use
My birth day: Wed Mar 12 10:00:00 1980
Christmas day: Fri Dec 25 00:00:00 2020
PS C:\AppServ\www\siam2dev_net\E_Learning\OOP\OOP_Pythons>
```

ข้อมูลเพิ่มเติม :: https://docs.python.org/3/library/datetime.html

## 8.1 ความหมายของ OOP

การเขียนโปรแกรมเชิงวัตถุ (อังกฤษ: Object-oriented programming, OOP) คือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ วัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้ วัตถุ อื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป

ที่มา : http://marcuscode.com/lang/python/classes-and-objects

# แบบฝึกหัด บทที่ 8

**ข้อที่ 1 OOP คืออะไร**

**ข้อที่ 2 จงยกตัวอย่างการสร้างคลาสในภาษาไพธอน 5 ตัวอย่าง**

# อ้างอิง


1. เว็บไซต์ :: https://stackpython.medium.com/python-oop-ep-1-oop-คืออะไร-f85ba48591f3
2. เว็บไซต์ :: http://marcuscode.com/lang/python/inheritance
3. เว็บไซต์ :: http://marcuscode.com/lang/python/classes-and-objects
4. เว็บไซต์ :: https://www.tamemo.com/post/125/all-about-oop-3-polymorphism/
5. เว็บไซต์ :: https://www.w3schools.com/python/python_classes.asp
6. ภาษาไพธอน. online : http://marcuscode.com/lang/python, สืบค้นเมื่อ 2 ก.ค. 2565
7. https://www.9experttraining.com/articles/python-คืออะไร
8. https://medium.com/@moungsiri/โครงสร้างภาษา-python-53cc38a51462
9. http://marcuscode.com/lang/python/variables-and-types
10. http://cms576.bps.in.th/group11/introduction-to-computer-programming
11. https://www.mindphp.com/บทเรียนออนไลน์/83-python/2398-python-operator.html
12. oop :: https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/
13. ข้อมูลเพิ่มเติมเกี่ยวกับวันที่และเวลา :: https://docs.python.org/3/library/datetime.html