01076105, 01075106

Object Oriented Programming

Object Oriented Programming Project

Methods

Object in Memory

- ใน Python ทุกอย่างเป็น Object

```python
print(object)   # Base Class of all Class

print(isinstance(5, object))

print(isinstance([1, 5, 2, 6], object))

print(isinstance((1, 5, 2, 6), object))
```

```
<class 'object'>
True
True
True
```

Object in Memory

```python
print(isinstance("Hello, World!", object))

print(isinstance({"a": 5, "b": 6}, object))

print(isinstance(False, object))

print(isinstance(True, object))
```

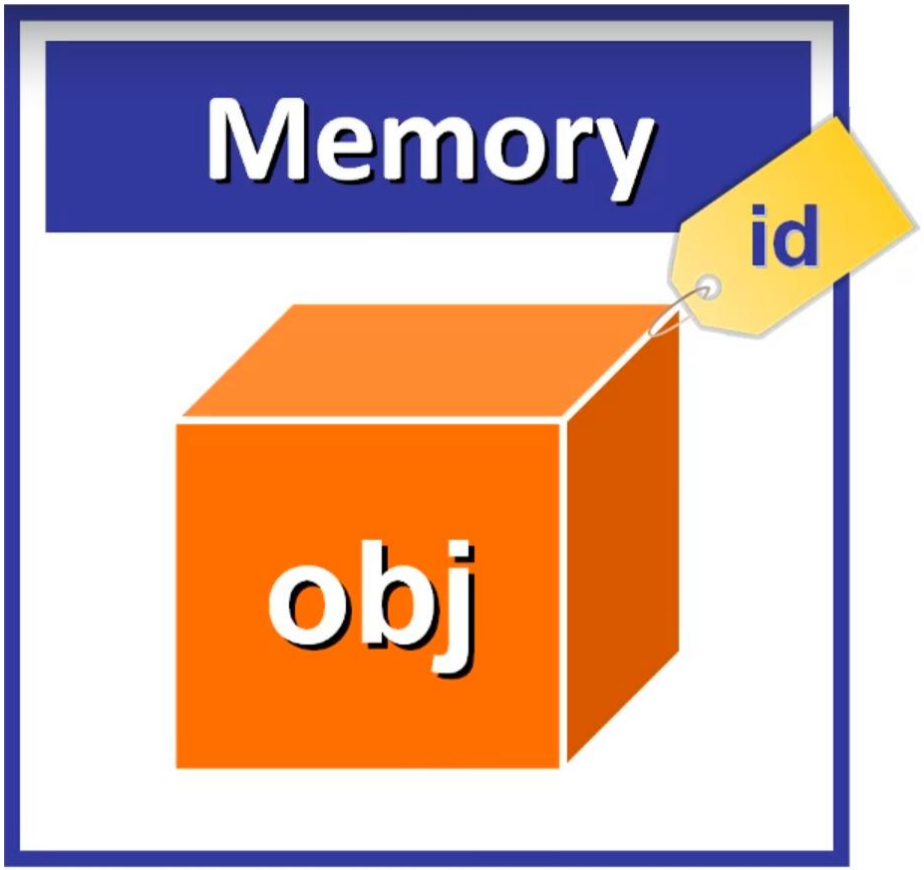
```
True
True
True
True
```

Object in Memory

```python
def f(x):
    return x * 2


print(isinstance(f, object))


class Movie:

    def __init__(self, title):
        self.title = title


print(isinstance(Movie, object))
```

Object in Memory

- แต่ละ object จะอยู่ในหน่วยความจำ ซึ่งสามารถระบุ object โดย id



Object in Memory

- เราสามารถใช้ฟังก์ชัน id() ในการหา id ของ object ได้ (address)

```
print(id(15))
print(id("Hello, World!"))
print(id([1, 2, 3, 4]))

a = [1, 2, 3, 4, 5]
b = [1, 2, 3, 4, 5]

print(id(a))
print(id(b))
```

```
2465587921584
2465589248240
2465589296640
2465589296640
2465589262592
```

“is” operator

- Operator “is” จะใช้ในการตรวจสอบว่า 2 Object มี id เดียวกันหรือไม่ ก็คือ อยู่ใน memory ที่เดียวกันหรือไม่

```
a = [1, 6, 2, 6]
b = [1, 6, 2, 6]

print(a is b)
print(a == b)
```

```
False
True
```

“is” operator

```python
a = [5, 2, 1, 8, 3]
b = [6, 2, 8, 9, 3]
print(a is b)

a = [5, 2, 1, 8, 3]
b = a
print(a is b)

c = ("a", "b", "c")
d = ("e", "f")
print(c is d)
```

```
False
True
False
```

“is” operator

```
e = "Hello, World!"
f = "Hello, World!"

print(e is f)
print(f is e)

i = 1000
j = 1000
print(i is j)
```

```
True
True
True
```

```
>>> a = 1000
>>> b = 1000
>>> a is b
False
```

Object Passing

- การผ่านค่า (Pass) ใน Programming จะมี 2 แบบ
  - Pass by value คือ การผ่านค่า โดยการ copy เฉพาะข้อมูลไป ซึ่งผลก็คือ จะไม่ทำให้ข้อมูลเดิมมีการเปลี่ยนแปลง
  - Pass by reference คือ การผ่านค่า โดยการส่งตำแหน่งที่อยู่ของข้อมูลไป (Address) ซึ่งผลก็คือ อาจทำให้ข้อมูลเดิมมีการเปลี่ยนค่าได้
  - ปกติ Programmer ต้องระลึกและระวังในการเขียนโปรแกรม ว่ากำลังใช้การผ่านค่าแบบใด
- การผ่านค่า Object ใน Python จะเป็น Pass by reference ทั้งหมด

Object Passing

```python
my_list = [6, 2, 8, 2]


def print_data(seq):
    print("Inside the function:", id(seq))
    for elem in seq:
        print(elem)


print("Outside the function:", id(my_list))
print_data(my_list)
```

```
Outside the function: 1820905304576
Inside the function: 1820905304576
```

Object Passing

```python
my_list = [6, 2, 8, 2]


def multiply_by_two(seq):
    print("Inside the function:", id(my_list))
    for i in range(len(seq)):
        seq[i] *= 2


print("Outside the function:", id(my_list))
multiply_by_two(my_list)

print(my_list)
```

Aliasing in Python

- Alias ถ้าแปลแบบไทยๆ ก็อาจคล้าย ชื่อเล่น คือ เป็นอีกชื่อหนึ่งที่หมายถึงสิ่งเดียวกัน
- Alias หมายถึง อะไรก็ตามที่ชี้ไปยัง ข้อมูลในตำแหน่งเดียวกัน (Address)

```
a = [1, 2, 3, 4]
b = a
c = b
d = c

print(id(a))
print(id(b))
print(id(c))
print(id(d))

print(a is b is c is d)
```

Aliasing in Python

```
    def __init__(self, radius):
        self.radius = radius


my_circle = Circle(4)
your_circle = my_circle
print("Before:")
print(my_circle.radius)
print(your_circle.radius)

your_circle.radius = 18
print("After:")
print(my_circle.radius)
print(your_circle.radius)
```

Mutability and Immutability

- Mutable แปลว่า สามารถแก้ไขได้ หมายถึง Object ที่สามารถเปลี่ยนแปลงค่าได้ เช่น Lists, Set, Dictionary
- Immutable แปลว่า ไม่สามารถแก้ไขได้ หมายถึง Object ที่ไม่สามารถเปลี่ยนแปลงค่าได้ เช่น Tuple, Strings (หลังจาก assign ค่า)
- A = [1, 2, 3]

  A[1] = ‘4’
- A = (1, 2, 3)

  A[1] = ‘4’

Mutability and Immutability

- ข้อดีของ Object แบบ Mutable
  - ประหยัดหน่วยความจำมากกว่า เพราะสามารถใช้ข้อมูลเดิมได้ ไม่ต้องเพิ่มใหม่หากมีการแก้ไข
  - ตรงกับข้อมูลในโลกจริง ที่เปลี่ยนแปลงได้
- ข้อเสียของ Object แบบ Mutable
  - หากใช้ไม่ระวัง อาจมี Bug
  - โปรแกรมนี้มีปัญหาอย่างไร

```
def add_absolute_values(seq):
    for i in range(len(seq)):
        seq[i] = abs(seq[i])
    return sum(seq)
```

Mutability and Immutability

- โปรแกรมนี้ มีปัญหาที่ใด (และแก้ไขอย่างไร)

```
a = [1, 2, 3, 4]
b = a

b[0] = 15

print(a)
print(b)
```

Mutability and Immutability

```python
a = [1, 2, 3, 4]
b = a[:]

print(a)
print(b)

b[0] = 15

print(a)
print(b)
```

Mutability and Immutability

- ข้อดีของ Object แบบ Immutable
  - ไม่มี Bug
  - ง่ายต่อการทำความเข้าใจ เพราะไม่ต้องคิดเผื่อกรณีที่มีการเปลี่ยนแปลง
- ข้อเสียของ Object แบบ Immutable
  - ประสิทธิภาพต่ำกว่า เพราะเมื่อมีการเปลี่ยนแปลง ต้องสร้างข้อมูลใหม่

```
a = (1, 2, 3, 4)
print(id(a))
a = a[:2] + (7,) + a[2:]
print(a)
print(id(a))
```

```
2069842788576
(1, 2, 7, 3, 4)
2069838246224
```

Mutability and Immutability

- โปรแกรมนี้ทำงานได้หรือไม่

```python
def remove_even_values(dictionary):
    for key, value in dictionary.items():
        if value % 2 == 0:
            del dictionary[key]


my_dictionary = {"a": 1, "b": 2, "c": 3, "d": 4}

remove_even_values(my_dictionary)  # This throws an error.
```

Mutability and Immutability

- Code ต่อไปนี้ จะแสดงผลอะไร

```
class WaitingList:

    def __init__(self, clients=[]):  # The default argument is an empty list
        self.clients = clients

    def add_client(self, client):
        self.clients.append(client)


waiting_list1 = WaitingList()
waiting_list2 = WaitingList()
waiting_list1.add_client("Jake")

print(waiting_list1.clients)
print(waiting_list2.clients)
```

Mutability and Immutability

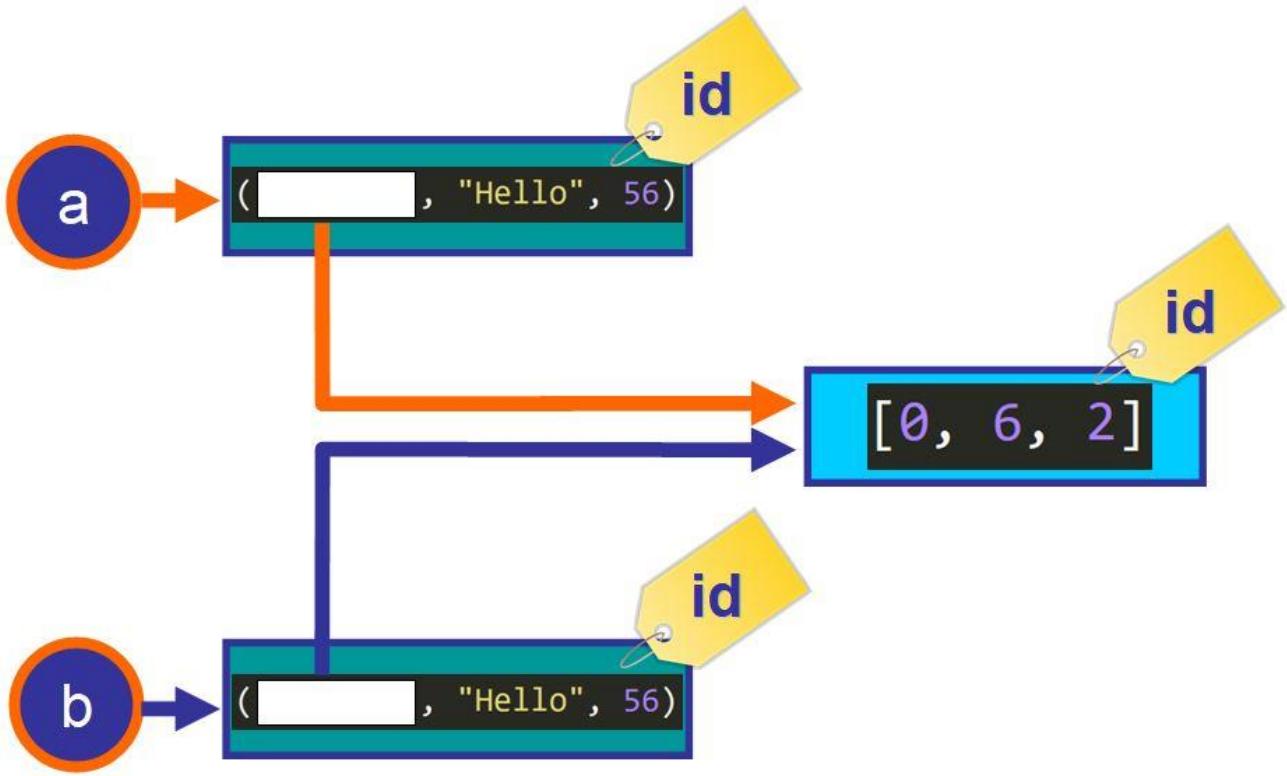
- คำว่า Immutable ก็ไม่ได้หมายความว่า เปลี่ยนแปลงไม่ได้
- ดูตามตัวอย่าง

```
a = ([0, 6, 2], "Hello", 56)
b = a[:]

a[0][1] = -5
print(a)
print(b)
```

```
([0, -5, 2], 'Hello', 56)
([0, -5, 2], 'Hello', 56)
```

Mutability and Immutability

```
a = ([0, 6, 2], "Hello", 56)
b = a[:]
```

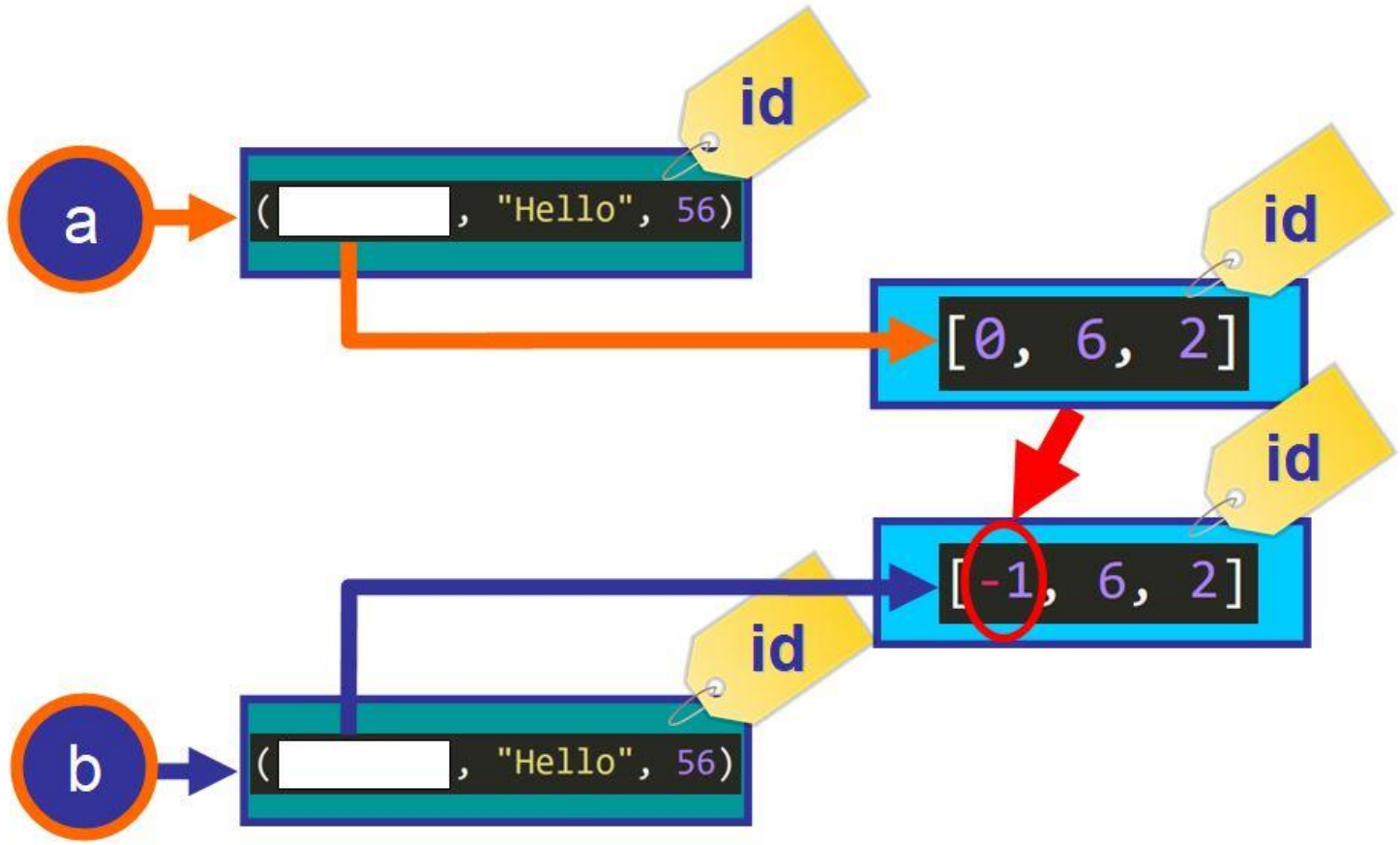


Mutability and Immutability

```python
import copy

a = ([5, 2, 6, 2], "Welcome", 67)
b = copy.deepcopy(a)
b[0][0] = -1

print(a)
print(b)
```

```
([5, 2, 6, 2], 'Welcome', 67)
([-1, 2, 6, 2], 'Welcome', 67)
```

Mutability and Immutability



Mutability and Immutability : Quiz

- Q1 . Dictionary a ถูกแก้ไขด้วยหรือไม่

```
>>> a = {"Nora": 55, "Gino": 30}
>>> b = a.copy()
>>> a is b
False
>>> b["Gino"] = 0

# Is the dictionary modified?
>>> a
```

Mutability and Immutability : Quiz

- Q2 คำสั่งต่อไปนี้ Error หรือไม่

```
1  a = ([5, 6, 2], 5)
2  a[0][0] = -1
```

- Q3 ผลลัพธ์ของโปรแกรม

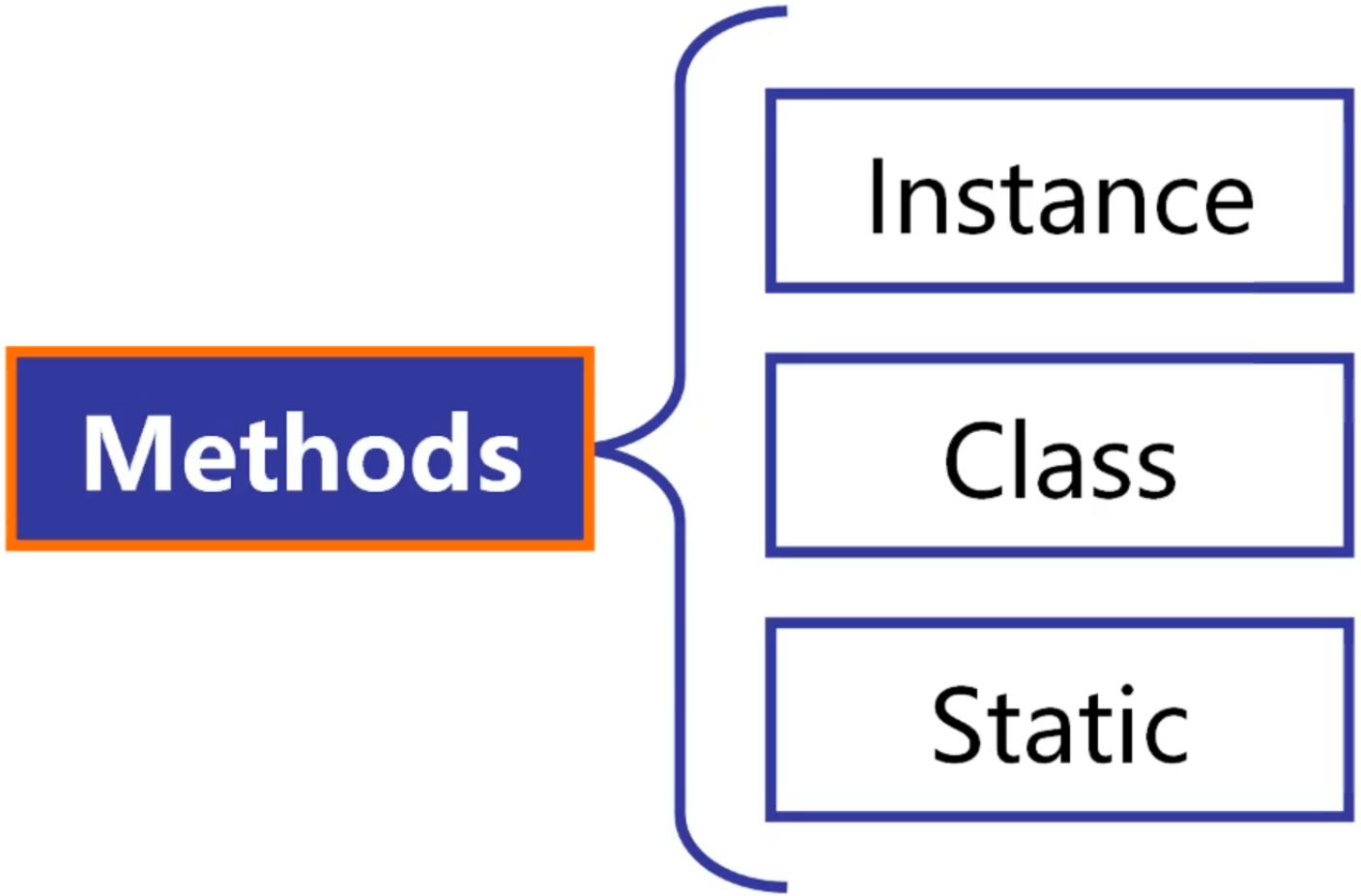
```
1  a = ([5, 2, 7], "Hello")
2  b = copy.deepcopy(a)
3  a[0][0] = -6
4
5  # Output?
6  print(b)
```

Exercise

- จากโปรแกรมจงแก้ไขปัญหา

```python
a = [7, 3, 6, 8, 2, 3, 7, 2, 6, 3, 6]
b = a
c = b
b = c


def remove_elem(data, target):
    for item in data:
        if item == target:
            data.remove(target)
    return data


def get_product(data):
    total = 1
    for i in range(len(data)):
        total *= data.pop()
    return total


remove_elem(c, 3)
print(get_product(b))
print(a)
```

Methods



Static method

- เป็น method ที่สามารถเรียกใช้ได้โดยไม่จำเป็นต้องสร้าง instant

```
class Student:
    def __init__(self, first_name, last_name, weight, height):
        self.first_name = first_name
        self.last_name = last_name
        self.weight = weight
        self.height = height

    def __str__(self):
        return "{} W: {}kg.({:.1f}lbs) H: {}cm.({:.1f}in)".format(self.first_name, self.weight,

    @staticmethod
    def kg_to_pound(kg):
        return kg * 2.20462

    @staticmethod
    def cm_to_inch(cm):
        return cm * .393701

s = Student("Fah", "Sairoong", 50, 165)
print(Student.kg_to_pound(50))
```

Class method

- เป็นที่ใช้ในการทำ Constructor แบบอื่นๆ ได้ (cls คือ constructor)

```python
class Point:
    def __init__(self, x, y):
        self._x = x
        self._y = y

    @classmethod
    def of(cls, point_string):
        s =  point_string.split("-")
        return cls(int(s[0]),int(s[1]))

p1 = Point(5, 5)
p2 = Point.of("10-10")
print(p2._x)
```

Import

- เป็นคำสั่งที่ขอเข้าถึง Code ในโมดูลอื่น
- การใช้ Import แบบมาตรฐาน
- กรณี Import หลาย Module ไม่ควรใช้ , (คือ ให้แยกคนละบรรทัด)
- เอาไว้ที่ส่วนหัวของไฟล์
- ลำดับการเรียง คือ Standard Mod., 3rd Party Mod, Local Mod. โดยเว้น 1 บรรทัดระหว่างกลุ่ม

Import

- ในการเขียนอาจเขียน ได้หลายแบบ เช่น

  import database (ทั้ง module)

  import math.pi (เฉพาะส่วน)

  from database import Database (เห็นว่าอะไรมาจากไหน)

  from database import Database as DB

  from database import Database, Query

  from database import * (ไม่ควรใช้)

- Module name ควรใช้ตัวเล็ก (อาจมี _)

Import

- ตัวอย่างกรณี Import ทั้ง Module

```python
import math


class Circle:

    def __init__(self, radius):
        self.radius = radius

    def find_area(self):
        return math.pi * (self.radius ** 2)
```

Import

- ตัวอย่างกรณี Import เฉพาะ element

```python
from math import pi


class Circle:

    def __init__(self, radius):
        self.radius = radius

    def find_area(self):
        return pi * (self.radius ** 2)
```

Import

- ตัวอย่างกรณี Import ทั้ง Module

```python
import random


class Die:

    def __init__(self, value):
        self.value = value

    def roll_die(self):
        random_value = random.randint(1, 6)
        self.value = random_value
```

Import

- ตัวอย่างกรณี Import เฉพาะ element

```python
from random import randint


class Die:

    def __init__(self, value):
        self.value = value

    def roll_die(self):
        random_value = randint(1, 6)
        self.value = random_value
```

Import

- กรณีที่เขียนโปรแกรมขนาดใหญ่ขึ้น นอกจากจะแยกแต่ละส่วนออกเป็นไฟล์แล้ว ยังอาจต้องแยกเป็น Directory ตามตัวอย่าง

```
parent_directory/
    main.py
    ecommerce/
        __init__.py
        database.py
        products.py
        payments/
            __init__.py
            square.py
            stripe.py
```

Import

- ในการ Import Module ที่แบ่งเป็น Directory จะมีด้วยกัน 2 แบบ
- Absolute Import

```
import ecommerce.products
product = ecommerce.products.Product()
```

- Relative Import

```
from .database import Database
from ..contact.email import send_mail
```

OOP Workshop #2

- เกม Tic-Tac-Toe หรือ เกม O-X
- เริ่มต้นเกมจะแสดงตำแหน่ง แล้วให้เลือกตำแหน่งของผู้ใช้
- จากนั้นคอมพิวเตอร์จะเลือกบ้าง
- และแสดงตำแหน่ง
- จะออกแบบ Class อย่างไร

```
**************************
  Welcome to Tic-Tac-Toe
**************************

Positions:
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Board:
|   |   |   |
|   |   |   |
|   |   |   |

Please enter your move (1-9): 4
```

For your attention