**อันแรกไม่มีไฟล์นับหนึ่งวิดีโอที่2**

1.สอนไพธอน Python OOP: การสร้างคลาสเบื้องต้น (simple class)

class Player:

def \_\_init\_\_(self): # dunder -> double underscores

self.fname = ""

self.lname = ""

self.number = ""

class Player2:

def \_\_init\_\_(self, fname, lname, number):

self.fname = fname

self.lname = lname

self.number = number

if \_\_name\_\_ == '\_\_main\_\_':

p1 = Player()

p1.fname = "Loris"

p1.lname = "Karius"

p1.number = 1

p2 = Player()

p2.fname = "Alex"

p2.lname = "Manninger"

p2.number = 13

p1a = Player2("Loris", "Karius", 1)

p2a = Player2("Alex", "Manninger", 13)

print(p1a.fname)

print(p2.lname)

p1t = ("Loris", "Karius", 1) # tuple

print(p1t)

print(p1t[0])

p1d = {"fname": "Loris", "lname": "Karius", "number": 1}

print(p1d)

print(p1d["lname"])

# 2.สอนไพธอน Python OOP: เปรียบเทียบการเก็บข้อมูลด้วย tuple, dict และ class instance

def demo\_tuple():

p12 = "Joe", "Gomez", 12

print(p12)

print(p12[1])

def demo\_dict():

p12 = {"fname": "Joe", "lname": "Gomez", "number": 12}

print(p12)

print(p12["lname"])

class Player:

pass

class Player2:

def \_\_init\_\_(self):

self.fname = ""

self.lname = ""

self.number = 0

class Player3:

def \_\_init\_\_(self, fname, lname, number):

self.fname = fname

self.lname = lname

self.number = number

def demo\_simple\_player\_class():

p12 = Player() # p12 -> instance

p12.fname = "Joe" # attribute/property

p12.lname = "Gomez"

p12.number = 12

print(p12.lname)

def demo\_simple\_player2\_class():

p12 = Player2()

p12.fname = "Joe"

p12.lname = "Gomez"

p12.number = 12

print(p12.lname)

def demo\_simple\_player3\_class():

p12 = Player3("Joe", "Gomez", 12)

print(p12.lname)

if \_\_name\_\_ == '\_\_main\_\_':

# demo\_tuple()

# demo\_dict()

# demo\_simple\_player\_class()

demo\_simple\_player3\_class()

# 3.สอนไพธอน Python OOP: รู้จักกับ \_\_init\_\_() หรือ constructor

class Person:

def \_\_init\_\_(self):

self.fname = ""

self.lname = ""

self.country = "Thailand"

# def \_\_init\_\_(self, fname, lname, country):

# self.fname = fname

# self.lname = lname

# self.country = country

class Person2:

def \_\_init\_\_(self, fname=None, lname=None, country="Thailand"):

self.fname = fname

self.lname = lname

self.country = country

def \_\_str\_\_(self):

return "fname: {} lname: {} country: {}".format(self.fname, self.lname, self.country)

if \_\_name\_\_ == '\_\_main\_\_':

# p1 = Person() # create new instance

# print(p1.fname)

# print(p1.country)

# p2 = Person("Prasert", "Kana", "Thailand")

# print(p2)

p1 = Person2()

print(p1.fname)

print(p1.country)

p2 = Person2(fname="Peter")

print(p2.fname, p2.country)

p3 = Person2("Peter", "Parker")

print(p3)

p4 = Person2(lname="Swift", fname="Taylor", country="USA")

print(p4)

# 4.สอนไพธอน Python OOP: การ import module/class แบบต่าง ๆ

class Medal:

def \_\_init\_\_(self, country, gold, silver, bronze):

self.country = country

self.gold = gold

self.silver = silver

self.bronze = bronze

def total(self): # instance method

return self.gold + self.silver + self.bronze

class Athlete:

def dummy(self):

return "hello"

class Foo:

def greeting(self):

return "hello all"

# 5.สอนไพธอน Python OOP: รู้จักกับ self

# self

class Medal:

def \_\_init\_\_(self, country, gold, silver, bronze):

self.country = country

self.gold = gold

self.silver = silver

self.bronze = bronze

def total(self): # instance method

return self.gold + self.silver + self.bronze

def dummy(self, a, b):

return a + b

if \_\_name\_\_ == '\_\_main\_\_':

th = Medal("Thailand", 5, 3, 2)

print(th.country)

print(th.total())

print(Medal.total(th))

print(th.dummy(1, 2))

print(Medal.dummy(th, 1, 2))

th.rank = 10

print(th.rank)

# 6.สอนไพธอน Python OOP: แปลงโค้ดที่เขียนแบบ procedural ให้เป็น class

**ไฟล์ที่ 1**

def bmi(w\_kg, h\_cm):

return w\_kg / (h\_cm / 100) \*\* 2

def category(w\_kg, h\_cm):

diag = ""

bmi\_value = bmi(w\_kg, h\_cm)

if bmi\_value < 18.5:

diag = "ต่ำกว่าเกณฑ์"

elif 18.5 <= bmi\_value <= 25:

diag = "ตามเกณฑ์"

elif 25 < bmi\_value <= 30:

diag = "เกินเกณฑ์"

elif bmi\_value > 30:

diag = "อ้วน"

return diag

if \_\_name\_\_ == '\_\_main\_\_':

w = 70

h = 170

print(bmi(w, h))

print(category(w, h))

**ไฟล์ที่2**

class Bmi:

def \_\_init\_\_(self, w\_kg, h\_cm):

self.w\_kg = w\_kg

self.h\_cm = h\_cm

def bmi(self):

return self.w\_kg / (self.h\_cm / 100) \*\* 2

def category(self):

diag = ""

if self.bmi() < 18.5:

diag = "ต่ำกว่าเกณฑ์"

elif 18.5 <= self.bmi() <= 25:

diag = "ตามเกณฑ์"

elif 25 < self.bmi() <= 30:

diag = "เกินเกณฑ์"

elif self.bmi() > 30:

diag = "อ้วน"

return diag

def \_\_str\_\_(self): # toString() in Java

return "BMI = {:.2f} ({})".format(self.bmi(), self.category())

if \_\_name\_\_ == '\_\_main\_\_':

a = Bmi(47, 168)

print(a)

# print(a.bmi())

# print(a.category())

# 7.สอนไพธอน Python OOP: ความแตกต่างระหว่าง \_\_str\_\_() และ \_\_repr\_\_()

# class Medal:

# def \_\_init\_\_(self, country, gold, silver, bronze):

# self.country = country

# self.gold = gold

# self.silver = silver

# self.bronze = bronze

# def total(self): # instance method

# return self.gold + self.silver + self.bronze

# # def \_\_str\_\_(self): # toString()

# # return "{:15} g: {:3} s: {:3} b: {:3} t: {:3}".format(self.country, self.gold, self.silver, self.bronze, self.total())

# def \_\_repr\_\_(self): # string representation

# return "({!r},{},{},{},{})".format(self.country, self.gold, self.silver,

# self.bronze, self.total())

# # return "{}{}".format(self.\_\_class\_\_.\_\_name\_\_, repr((self.country, self.gold, self.silver, self.bronze)))

# if \_\_name\_\_ == '\_\_main\_\_':

# # th = Medal("Thailand", 5, 6, 10)

# # print(th) -> print(str(th)) -> print(th.\_\_str\_\_()) -> print(th.\_\_repr\_\_())

# # print(th.country, "g", th.gold, "s", th.silver, "b", th.bronze, "t", th.total())

# m = [

# Medal("Thailand", 5, 6, 10),

# Medal("Japan", 15, 20, 10),

# Medal("China", 55, 40, 105)

# ]

# for c in m:

# print(c)

# 8.สอนไพธอน Python OOP: ใช้ vars() และ getattr() ในการแสดงชื่อ attribute และ value ของ instance

class Person:

def \_\_init\_\_(self, fname, lname, age):

self.fname = fname

self.lname = lname

self.age = age

def \_\_str\_\_(self):

# return "fname: {}, lname: {}, age: {}".format(self.fname, self.lname, self.age)

# a = vars(self)

# # print(a)

# s = ["{:10}: {}".format(k, v) for k, v in a.items()]

# return "\n".join(s)

attrs = ("fname", "lname", "age")

s = ["{:10}: {}".format(a, getattr(self, a)) for a in attrs]

return "\n".join(s)

if \_\_name\_\_ == '\_\_main\_\_':

p = Person("Peter", "Parker", 26)

print(p)

# 9.สอนไพธอน Python OOP: เข้าใจความแตกต่างระหว่าง mutable และ immutable object

def immutable\_demo():

# n = 5

# print("id(n) = {}, n = {}".format(id(n), n))

# n = n + 4

# print("id(n) = {}, n = {}".format(id(n), n))

s = "rain"

print("id(s) = {}, s = {}".format(id(s), s))

s = s + "bow"

print("id(s) = {}, s = {}".format(id(s), s))

def mutable\_demo():

p = ["rain"]

print("id(p) = {}, p = {}".format(id(p), p))

p[0] = p[0] + "bow"

print("id(p) = {}, p = {}".format(id(p), p))

q = p

print("id(p) = {}, p = {}\nid(q) = {}, q = {}".format(id(p), p, id(q), q))

q.append("sunshine")

print("id(p) = {}, p = {}\nid(q) = {}, q = {}".format(id(p), p, id(q), q))

class Contact:

def \_\_init\_\_(self, name, phone):

self.name = name

self.phone = phone

def \_\_str\_\_(self):

return "id(self) = {}, name = {}, phone = {}".format(id(self), self.name, self.phone)

if \_\_name\_\_ == '\_\_main\_\_':

# immutable\_demo()

# mutable\_demo()

a = Contact("Fah", "0812223333")

print(a)

a.phone = "0992223333"

print(a)

b = a

print(b)

b.phone = "0773334444"

print("a = ", a)

print("b = ", b)

# 10.สอนไพธอน Python OOP: เข้าใจหลักการในการอ้างถึง Object/Instance reference

class Wallet:

def \_\_init\_\_(self):

self.amount = 0

def earn(self, a):

self.amount += a

def spend(self, a):

self.amount -= a

def \_\_str\_\_(self):

return "amount = {}".format(self.amount)

if \_\_name\_\_ == '\_\_main\_\_':

dad = Wallet()

dad.earn(100)

print("dad's wallet", dad)

mom = dad

print(mom is dad)

print("mom's wallet", mom)

mom.spend(20)

print("mom's wallet", mom)

print("dad's wallet", dad)

print(id(dad), id(mom))

kid = Wallet()

kid.earn(500)

mom = kid

print(id(dad), id(mom), id(kid))

mom.earn(40)

print("kid's wallet", kid)

# 11.สอนไพธอน Python 3: การใช้ \_ (ขีดล่าง single underscore)

**ไฟล์ที่ 1**

# import random

#

# for \_ in range(5):

# print("hello")

#

# for \_ in range(5):

# print(random.randint(1, 6))

\_\_all\_\_ = ["circle", "Alpha", "\_Beta", "\_rectangle"]

\_recipe = "a7b3c15" # private

vat = .07

def \_rectangle(w, h):

return w\*h

def circle(r):

return 22 / 7 \* r \* r

class Alpha:

@staticmethod

def foo():

print("hello")

class \_Beta:

@staticmethod

def bar():

print("wow")

**ไฟล์ที่2**

# import underscore\_demo

#

# print(underscore\_demo.\_recipe)

# print(underscore\_demo.vat)

# from underscore\_demo import Alpha, \_Beta, \_rectangle, \_recipe, vat, circle

from underscore\_demo import \*

# print(\_recipe)

# print(vat)

print(\_rectangle(3, 4))

# print(circle(10))

Alpha.foo()

\_Beta.bar()

# 12.สอนไพธอน Python 3: การใช้ \_\_ (ขีดล่างสองตัว double underscore หรือ dunder)

**ไฟล์ที่ 1**

# dunder -> double underscore

class Alpha:

normal = 1

\_lucky = 13

\_\_secret = 777 # name mangled -> change \_classname\_\_attri

def fx(self):

print("this is fx in A")

def \_fy(self):

print("this is fy in A")

def \_\_fz(self): # final method

print("this is fz in A")

class Beta(Alpha):

def \_\_init\_\_(self):

self.greeting = "hello from beta"

def \_\_foo\_\_(self):

pass

def \_\_fz(self):

print("THIS IS fz in B")

class \_\_Gamma():

def bar(self):

print("hello from Gamma")

if \_\_name\_\_ == '\_\_main\_\_':

# print(Alpha.normal)

# print(Alpha.\_lucky)

# print(Alpha.\_\_secret)

# print(Alpha.\_Alpha\_\_secret)

# print(Alpha.\_\_dict\_\_)

# Alpha.normal = 99

# Alpha.\_lucky = 123

# Alpha.\_\_secret = 9999

# print(Alpha.normal)

# print(Alpha.\_lucky)

# print(Alpha.\_\_secret)

# print(Alpha.\_Alpha\_\_secret)

print(Alpha.\_\_dict\_\_)

a=Alpha()

a.fx()

a.\_fy()

# a.\_\_fz()

a.\_Alpha\_\_fz()

b = Beta()

b.\_Beta\_\_fz()

b.\_Alpha\_\_fz()

g = \_\_Gamma()

g.bar()

**ไฟล์ที่ 2**

# from dunder import \*

# from dunder import \_\_Gamma

import dunder

g = dunder.\_\_Gamma()

g.bar()

# 13.สอนไพธอน Python OOP: การใช้ @property decorator

class Student:

def \_\_init\_\_(self, s\_id, fname, lname):

self.s\_id = s\_id

self.fname = fname

self.lname = lname

# self.full\_name = "{} {}".format(self.fname, self.lname)

def full\_name(self): #getter in Java

return "{} {}".format(self.fname, self.lname)

def email(self):

return "{}.{}{}@cbs.chula.ac.th".format(self.fname, self.lname[:2], self.join\_yy)

@property

def full\_name2(self): #getter in Java

return "{} {}".format(self.fname, self.lname)

@property

def join\_yy(self):

return self.s\_id[:2]

@property

def degree(self):

return self.s\_id[2]

@property

def seq(self):

return self.s\_id[3:7]

@property

def check\_digit(self):

return self.s\_id[-3]

@property

def school(self):

return self.s\_id[-2:]

if \_\_name\_\_ == '\_\_main\_\_':

s = Student("5841234526", "Fah", "Puinoon")

print(s.full\_name())

print(s.full\_name2)

print(s.join\_yy)

print(s.s\_id[:2])

print(s.school)

print(s.email())

# s.full\_name = "Songkarn Fahsai"

# print(s.full\_name())

# 14.สอนไพธอน Python OOP: การสร้าง getter, setter ให้กับ attributes/properties (Java style)

class Student:

def \_\_init\_\_(self, fname, lname, blood):

self.fname = fname

self.lname = lname

self.blood = blood # A, B, AB, O

def \_\_str\_\_(self):

return "{} {}, blood group: {}".format(self.fname, self.lname, self.blood)

class Person:

def \_\_init\_\_(self, fname, lname, blood):

self.fname = fname

self.lname = lname

# self.blood = blood # A, B, AB, O

self.setBlood(blood)

def getBlood(self):

return self.\_\_blood

def setBlood(self, blood):

if blood.upper() in ["A", "B", "AB", "O"]:

self.\_\_blood = blood.upper()

else:

raise ValueError("invalid blood group.")

def \_\_str\_\_(self):

return "{} {}, blood group: {}".format(self.fname, self.lname, self.\_\_blood)

if \_\_name\_\_ == '\_\_main\_\_':

# s1 = Student("Peter", "Parker", "A")

# s1.blood = "Y"

# print(s1)

p1 = Person("Peter", "Parker", "A")

print(p1)

p2 = Person("Bruce", "Wayne", "Ab")

print(p2)

p2.setBlood("O")

print(p2)

p2.\_\_blood = "B"

print(p2)

print(p2.\_\_dict\_\_)

p2.\_Person\_\_blood = "A"

print(p2)

print(p2.getBlood())

# 15.สอนไพธอน Python OOP: การสร้าง getter, setter ให้กับ attributes/properties (Pythonic style)

class Person:

def \_\_init\_\_(self, fname, lname, blood):

self.fname = fname

self.lname = lname

self.blood = blood # A, B, AB, O

# self.setBlood(blood)

@property

def fname(self):

return self.\_\_fname

@fname.setter

def fname(self, fname):

self.\_\_fname = fname.strip().title()

@property

def blood(self): # getter

return self.\_\_blood

@blood.setter

def blood(self, blood): # setter

if blood.upper() in ["A", "B", "AB", "O"]:

self.\_\_blood = blood.upper()

else:

raise ValueError("invalid blood group.")

def \_\_str\_\_(self):

return "{!r} {}, blood group: {}".format(self.fname, self.lname, self.blood)

if \_\_name\_\_ == '\_\_main\_\_':

p1 = Person(" Peter ", "Parker", "A")

print(p1)

p1.blood = "m"

print(p1)

print(p1.blood)

# 16.สอนไพธอน Python OOP: หลักการสืบทอด inheritance ตอนที่ 1

class Person:

def \_\_init\_\_(self, fname, lname):

self.fname = fname

self.lname = lname

def \_\_str\_\_(self):

return "{} {}".format(self.fname, self.lname)

class Student(Person):

def \_\_init\_\_(self, s\_id, fname, lname):

super().\_\_init\_\_(fname, lname)

self.s\_id = s\_id

# self.s\_id = s\_id

# self.fname = fname

# self.lname = lname

def \_\_str\_\_(self):

return "{} {}".format(self.s\_id, super().\_\_str\_\_())

class ExchangeStudent(Student):

def \_\_init\_\_(self, s\_id, fname, lname, partner\_univ):

super().\_\_init\_\_(s\_id, fname, lname)

self.partner\_univ = partner\_univ

if \_\_name\_\_ == '\_\_main\_\_':

s1 = Student("1234", "Fon", "Sairoong")

print(s1)

s2 = ExchangeStudent("7890", "Peter", "Parker", "ABC Univ.")

print(s2)

# 17.สอนไพธอน Python OOP: หลักการสืบทอด inheritance ตอนที่ 2

import re

class Person:

def \_\_init\_\_(self, fname, lname, sex):

self.fname = fname.strip().title()

self.lname = lname.strip().title()

self.sex = sex

def \_\_str\_\_(self):

return "{!r} {!r} sex: {}".format(self.fname, self.lname, self.sex)

class Student(Person):

def \_\_init\_\_(self, s\_id, fname, lname, sex):

super().\_\_init\_\_(fname, lname, sex)

self.s\_id = self.remove\_non\_digit(s\_id)

# self.s\_id = s\_id

# self.fname = fname

# self.lname = lname

def \_\_str\_\_(self):

return "{} {}".format(self.s\_id, super().\_\_str\_\_())

@staticmethod

def remove\_non\_digit(s):

return re.sub(r"[\D]", "", s)

def email(self):

return "{}.{}{}@cbs.chula.ac.th".format(self.fname, self.lname[:2], self.s\_id[:2])

class ExchangeStudent(Student):

def \_\_init\_\_(self, s\_id, fname, lname, sex, partner\_univ):

super().\_\_init\_\_(s\_id, fname, lname, sex)

self.partner\_univ = partner\_univ

def foo(self, s):

return self.remove\_non\_digit(s)

if \_\_name\_\_ == '\_\_main\_\_':

s1 = Student("584 12345-26", "fon ", "Sairoong", "F")

print(s1)

print(s1.email())

s2 = ExchangeStudent("584-12345(26)", " peter", "Parker", "M", "ABC Univ.")

print(s2)

print(s2.email())

# 18.สอนไพธอน Python OOP: การสร้าง abstract class

from abc import ABC, abstractmethod

# Abstract Base Class

class Member(ABC):

def \_\_init\_\_(self, m\_id, fname, lname):

self.m\_id = m\_id

self.fname = fname

self.lname = lname

def \_\_str\_\_(self):

return "ID: {} {} {}".format(self.m\_id, self.fname, self.lname)

@abstractmethod

def discount(self):

pass

def full\_name(self):

return "{} {}".format(self.fname, self.lname)

class Gold(Member):

def discount(self):

return .10

class Silver(Member):

def discount(self):

return .05

if \_\_name\_\_ == '\_\_main\_\_':

# m = Member("007", "James", "Bond")

# print(m)

g = Gold("123", "Peter", "Parker")

print(g)

print(g.discount())

print(g.full\_name())

s = Silver("888", "Jane", "Mark")

print(s)

print(s.full\_name())

# 19.สอนไพธอน Python OOP: การสืบทอดจากหลายคลาสพร้อมกัน (multiple inheritance)

class Camera:

def take\_photo(self):

print("take a photo")

def delete\_photo(self):

print("delete a photo")

def browse(self):

print("browse photo album")

class Phone:

def call(self, number):

print("calling {}".format(number))

def send\_sms(self, number, message):

print("sending {} to {}".format(message, number))

class MediaPlayer:

def play(self):

print("playing a song/video")

def browse(self):

print("browse media library")

# class SmartPhone(MediaPlayer, Camera, Phone):

class SmartPhone(Camera, Phone, MediaPlayer):

def share(self):

print("share ...")

if \_\_name\_\_ == '\_\_main\_\_':

s = SmartPhone()

s.take\_photo()

s.send\_sms("1234123", "Hi")

s.play()

s.browse()

# 20.สอนไพธอน Python OOP: การออกแบบคลาสโดยใช้หลักการ Composition เบื้องต้น

from datetime import date

class Person:

def \_\_init\_\_(self, title, fname, lname, dob):

self.title = title

self.fname = fname

self.lname = lname

self.dob = dob

def \_\_str\_\_(self):

return "{} {} {} {}".format(self.title, self.fname, self.lname, self.dob)

class PersonV2:

def \_\_init\_\_(self, title, fname, lname, titleTh, fnameTh, lnameTh, dob):

self.title = title

self.fname = fname

self.lname = lname

self.titleTh = titleTh

self.fnameTh = fnameTh

self.lnameTh = lnameTh

self.dob = dob

def \_\_str\_\_(self):

return "{} {} {}\n{} {} {}\n{}" \

.format(self.title, self.fname, self.lname,

self.titleTh, self.fnameTh, self.lnameTh,

self.dob)

class PersonName:

def \_\_init\_\_(self, title, fname, lname):

self.title = title

self.fname = fname

self.lname = lname

def \_\_str\_\_(self):

return "{} {} {}".format(self.title, self.fname, self.lname)

class PersonV3:

def \_\_init\_\_(self, nameEn, nameTh, dob):

self.nameTh = nameTh

self.nameEn = nameEn

self.dob = dob

def \_\_str\_\_(self):

return "{}\n{}\n{}".format(self.nameEn, self.nameTh, self.dob)

if \_\_name\_\_ == '\_\_main\_\_':

p1 = Person("Mr.", "Peter", "Parker", date(1995, 10, 4))

print(p1)

print("\*" \* 40)

p2 = PersonV2("Mr.", "Peter", "Parker",

"นาย", "ปีเตอร์", "ปาร์คเกอร์",

date(1995, 10, 4))

print(p2)

print("\*" \* 40)

p3 = PersonV3(PersonName("Mr.", "Peter", "Parker"), PersonName("นาย", "ปีเตอร์", "ปาร์คเกอร์"), date(1995, 10, 4))

print(p3)

# 21.สอนไพธอน Python OOP: ประยุกต์ใช้หลักการออกแบบ Composition

class Printer:

def print\_page(self, data):

print("printing {}".format(data))

class Scanner:

def scan\_page(self):

print("scanning...")

class Fax:

def fax\_page(self, number):

print("faxing to {}".format(number))

class Aio: # all in one printer

def \_\_init\_\_(self, p, s, f):

self.p = p

self.s = s

self.f = f

if \_\_name\_\_ == '\_\_main\_\_':

a = Aio(Printer(), Scanner(), Fax())

a.p.print\_page("hello")

a.s.scan\_page()

a.f.fax\_page("022185765")

# 22.สอนไพธอน Python OOP: รู้จักกับ static method

class Student:

num\_students = 0

def \_\_init\_\_(self, fname, lname, w\_kg, h\_cm):

self.fname = fname

self.lname = lname

self.w\_kg = w\_kg

self.h\_cm = h\_cm

def \_\_str\_\_(self):

return "{} W: {}kg.({:.1f}lbs) H: {}cm.({:.1f}in)".format(self.fname, self.w\_kg, self.kg\_pound(self.w\_kg), self.h\_cm, Student.cm\_inch(self.h\_cm))

def bmi(self): # instance method

return self.w\_kg / (self.h\_cm / 100) \*\* 2

@staticmethod

def kg\_pound(kg):

return kg \* 2.20462

@staticmethod

def cm\_inch(cm):

return cm \* .393701

@staticmethod

def foo():

# self.num\_stuents

print(Student.num\_students)

# print(self.fname)

if \_\_name\_\_ == '\_\_main\_\_':

s = Student("Fah", "Sairoong", 50, 165)

print(s.bmi())

print(Student.kg\_pound(50))

print(s)

# 23.สอนไพธอน Python OOP: การสร้าง class method (@classmethod)

class Eyeglasses:

def \_\_init\_\_(self, eye, bridge, temple):

self.eye = eye

self.bridge = bridge

self.temple = temple

@classmethod # function decorator

def of(cls, frame\_string, sep="-"):

s = frame\_string.split(sep)

return cls(int(s[0]), int(s[1]), int(s[2]))

@classmethod

def of2(cls, frame\_with\_weight):

pass

@staticmethod

def gram\_oz(g):

return g \* 0.035274

def \_\_str\_\_(self):

return "{}-{}-{}".format(self.eye, self.bridge, self.temple)

if \_\_name\_\_ == '\_\_main\_\_':

f1 = Eyeglasses(55, 16, 140)

print(f1)

f2 = Eyeglasses.of("55,16,140", ",")

print(f2)

print(f2.eye, f2.bridge, f2.temple)

print(Eyeglasses.gram\_oz(20))

# 24.สอนไพธอน Python OOP: เปรียบเทียบว่า instance 2 ตัวเท่ากันหรือไม่ด้วย \_\_eq\_\_()

class Student:

def \_\_init\_\_(self, fname, score):

self.fname = fname

self.score = score

def \_\_str\_\_(self):

return "{}, score = {}".format(self.fname, self.score)

def \_\_eq\_\_(self, other):

return self.score == other.score

if \_\_name\_\_ == '\_\_main\_\_':

s1 = Student("Peter", 75)

s2 = Student("Taylor", 75)

s3 = Student("Taylor", 75)

s4 = Student("Ann", 60)

print(s1)

print(s2)

print(s1 == s2)

print(s2 == s3)

print(s1 == s4)

print(s1.\_\_eq\_\_(s4))

# print(id(s2), id(s3))

# print(id(s2) == id(s3))

# 25.สอนไพธอน Python OOP: การกำหนดลำดับการเรียงลำดับขึ้นใช้เอง (custom sort order)

class Athlete:

def \_\_init\_\_(self, name, sport, medal):

self.name = name

self.sport = sport

self.medal = medal

def \_\_str\_\_(self):

return "{:10} {:12} {}".format(self.name, self.sport, self.medal)

def \_\_lt\_\_(self, other):

d = {"gold": 1, "silver": 2, "bronze": 3}

if d[self.medal] == d[other.medal]:

return self.name < other.name

else:

return d[self.medal] < d[other.medal]

# return d[self.medal] < d[other.medal]

# return self.medal > other.medal

if \_\_name\_\_ == '\_\_main\_\_':

data = [

Athlete("Peter", "Judo", "silver"),

Athlete("Taylor", "Tennis", "gold"),

Athlete("John", "Tennis", "silver"),

Athlete("Sarah", "Badminton", "bronze"),

Athlete("Tony", "Boxing", "gold"),

Athlete("Bruce", "Judo", "gold"),

]

s = sorted(data)

[print(e) for e in s]

# 26.สอนไพธอน Python OOP: การทำ Operator overloading

class FootInch:

def \_\_init\_\_(self, foot, inch):

self.foot = foot

self.inch = inch

self.inches = self.foot \* 12 + self.inch

def \_\_str\_\_(self):

return "{}'{}\"".format(self.foot, self.inch)

def \_\_add\_\_(self, other): # overload + operator

x = self.inches + other.inches

f = x // 12

i = x % 12

return FootInch(f, i)

def \_\_sub\_\_(self, other): # overload - operator

x = self.inches - other.inches

f = x // 12

i = x % 12

return FootInch(f, i)

# def \_\_mul\_\_(self, other):

# x = self.inches \* other.inches

# f = x // 12

# i = x % 12

# return FootInch(f, i)

# def \_\_divmod\_\_(self, other):

# https://docs.python.org/3/library/functions.html#divmod

# x = self.inches \* other.inches

# f = x // 12

# i = x % 12

# return FootInch(f, i)

if \_\_name\_\_ == '\_\_main\_\_':

m1 = FootInch(2, 0)

m2 = FootInch(3, 0)

m = m1 + m2

ma = m1.\_\_add\_\_(m2)

m5 = m1 - m2

# m6 = m1 \* m2

print(m)

print(ma)

print(m5)

# print(m6)