Josh Learn Python

today we are learing Python with Khun Toy datarockie

```
print ("Hello World")
```

```
Hello World
```

```
print ("I'm learning python") #after this sign is a note, not code
```

```
I'm learning python
```

```
# 5 building blocks
# 1. variables
# 2. data types
# 3. data structures
# 4. function
# 5. control flow
# 6. OOP
```

1. variables

```
## assign a variable
my_name = "Josh"
age = 26
gpa = 3.52
movie_lover = True #False
```

```
print (my_name, age, gpa, movie_lover)
```

```
Josh 26 3.52 True
```

```
1
```

2. data types

```
# type hint
age: int = 34
ny_name: str = "Josh"
gpa: float = 3.52
seafood: bool = True
```

```
print (age, type(age))
```

```
34 <class 'int'>
```

```
print (my_name)
```

```
Josh
```

```
1
```

function

```
# function
print("hello", "world") #เอาstrมาต่อกัน
print (pow(5, 2), abs(-5)) #pow=เอาเลขมายกกำลัง, abs=เปลี่ยนเลขติดลบให้เป็นค่าบวก
```

```
hello world
25 5
```

```
# greeting()
def greeting(name="Josh", location="New York"):
    print("Hello " + name)
    print("He is in " + location)
```

```
greeting("Dream", "Paris")
```

```
Hello Dream
He is in Paris
```

```
def add_two_nums(num1, num2):
    print("hello world")
    print("Done!")
    return (num1 + num2) #returnคือการส่งค่าของฟังชั่นกลับมาและจะจบprocessเลย
```

```
result = add_two_nums(5, 15)
print(result)
```

```
hello world
Done!
20
```

```
def add_two_nums(a: int, b: int) -> int:
    return a + b
```

```
add_two_nums(15, 15)
```

```
30
```

```
# string template : fstrings
my_name = "Josh"
location = "Bangkok"

text  = f"Hi! ma name is {my_name} and I live in {location}"
```

```
print(text)
```

```
Hi! ma name is Josh and I live in Bangkok
```

```
text = "a duck walks into a bar"
print(text)
```

```
a duck walks into a bar
```

```
# slicing, index start with 0
text[22]
```

```
'r'
```

```
text[0], text[-1]
```

```
('a', 'r')
```

```
text
```

```
'a duck walks into a bar'
```

```
text[2:6]
```

```
'duck'
```

```
# string is immutable, str ไม่สามารถอัพเดทค่าได้
name = "Python" # -> Cython
name = "C" + name [1:]
print(name)
```

```
Cython
```

```
text = "a duck walks into a bar"
```

```
# function vs. method
# string methods ;ฟังชั่นที่ถูกสร้างมาเพื่อใช้กับstrเท่านั้นเรียกว่า string method
text.upper()
```

```
'A DUCK WALKS INTO A BAR'
```

```
print(text)
```

```
a duck walks into a bar
```

```
text = text.upper() #assignค่าใหม่ให้กับตัวแปรเดิมเป็นพิมพ์ใหญ่หมดโดยใช้string method text.upper
```

```
print(text)
```

```
A DUCK WALKS INTO A BAR
```

```
text = text.lower() #assignค่าใหม่ให้กับตัวแปรเดิมเป็นพิมพ์เล็กหมดโดยใช้string method text.lower
```

```
print(text)
```

```
a duck walks into a bar
```

```
text.replace("duck", "lion")
```

```
'a lion walks into a bar'
```

```
text
```

```
'a duck walks into a bar'
```

```
words = text.split(" ")
words
```

```
['a', 'duck', 'walks', 'into', 'a', 'bar']
```

```
words = " ".join(words)
```

```
words
```

```
'a duck walks into a bar'
```

```
1
```

data structures

```
# 1. list [] is mutable
# 2. tuple () is immutable ไม่สามารถแทนค่าได้
# 3. dictionary {} is mutable
# 4. set {unique}  จะเก็บค่าที่เป็นยูนีคเท่านั้น  ถ้าซ้ำจะลบทิ้งทั้งหมด
```

1. list [ ]

italicized text

```
# list
# list is mutable
shopping_items = ["banana", "egg", "milk"]
print(shopping_items[0])
print(shopping_items[1])
print(shopping_items[2]) #[0],[1],[2]ใช้คอนเซปslicingในการเรียกดูlist
```

```
banana
egg
milk
```

```
print(shopping_items[0:])
```

```
['banana', 'egg', 'milk']
```

```
print(len(shopping_items))
```

```
3
```

```
shopping_items[0] = "durian" #อัพเดทค่า
print(shopping_items)
```

```
['durian', 'egg', 'milk']
```

```
# list methods
shopping_items.append("yokurt") #.append คือการเพิ่มค่าเข้าไปที่ด้านขวาสุด listสามารถอัพเดทค่าได้ ไม่ต้องassignค่าใหม่ให้กับตัวแปรนั้นเหมือนกับstr
print(shopping_items)
```

```
['durian', 'egg', 'milk', 'yokurt']
```

```
shopping_items.append("cabbage")
print(shopping_items)
```

```
['durian', 'egg', 'milk', 'yokurt', 'cabbage']
```

```
# sort items จะเรียงลำดับของitemใหม่ แบบAscendining order จากA-Z
shopping_items.sort()
print(shopping_items)
```

```
['cabbage', 'durian', 'egg', 'milk', 'yokurt']
```

```
shopping_items.sort(reverse=True) #จะเรียงลำดับของitemใหม่ แบบDescendining order จากZ-A
print(shopping_items)
```

```
['yokurt', 'milk', 'egg', 'durian', 'cabbage']
```

```
shopping_items.remove("cabbage")
shopping_items
```

```
['yokurt', 'milk', 'egg', 'durian']
```

```
shopping_items
```

```
['yokurt', 'milk', 'egg', 'durian']
```

2. tuple ( )

```
# tuple () is immutable ไม่สามารถแทนค่าได้ ประกาศแล้วเปลี่ยนค่าไม่ได้

# tuple unpacking
name, age, _ = ("Josh Wick", 26, 3.52)
print(name, age, _)
```

```
Josh Wick 26 3.52
```

3. dictionary { }

```
# dictionary จะใช้คอนเซปเท่ากับ key: value pairs , ซ้ายมือเป็น key ขวามือเป็น value
course = {
    "name": "Data Science Bootcamp",
    "duration": "4 months",
    "students": 150,
    "replay": True,
    "skills": ["Google Sheets", "SQL", "R", "Python", "Stata", "ML", "Dashboard", "Data Transformation"]
}

```

```
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 150,
 'replay': True,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stata',
  'ML',
  'Dashboard',
  'Data Transformation']}
```

```
course["skills"] #ถ้าจะดึงค่าออกมาต้องดึงโดยใช้ key
```

```
['Google Sheets',
 'SQL',
 'R',
 'Python',
 'Stata',
 'ML',
 'Dashboard',
 'Data Transformation']
```

```
course["start_time"] = "9 am" #แอด key เพิ่ม
```

```
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 150,
 'replay': True,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stata',
  'ML',
  'Dashboard',
  'Data Transformation'],
 'start_time': '9 am'}
```

```
# delete key
del course["start_time"]
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 150,
 'replay': True,
```

```
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stata',
  'ML',
  'Dashboard',
  'Data Transformation']}
```

```
course["replay"] = False
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 150,
 'replay': False,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stata',
  'ML',
  'Dashboard',
  'Data Transformation']}
```

```
# dictionary method
# .key ดูคีย์ทั้งหมด
# .values
# .item
# .get
```

```
course.keys() #ดึงคีย์ที่ประกาศค่าเอาไว้ออกมาดูว่ามีค่าอะไรบ้าง
```

```
dict_keys(['name', 'duration', 'students', 'replay', 'skills'])
```

```
list (course.keys()) #ดึงฟังชั่นlistมาใช้
```

```
['name', 'duration', 'students', 'replay', 'skills']
```

```
list (course.values())
```

```
['Data Science Bootcamp',
 '4 months',
 150,
 False,
 ['Google Sheets',
  'SQL',
  'R',
  'Python',
  'Stata',
  'ML',
  'Dashboard',
  'Data Transformation']]
```

```
list (course.items())
```

```
[('name', 'Data Science Bootcamp'),
 ('duration', '4 months'),
 ('students', 150),
 ('replay', False),
 ('skills',
  ['Google Sheets',
   'SQL',
   'R',
   'Python',
   'Stata',
   'ML',
   'Dashboard',
   'Data Transformation'])]
```

```
course.get("Replay") #ใช้เช็คโค้ดว่าจะรีเทิร์นค่าอะไรกลับมามั้ย ถ้าเราพิมชื่อตัวแปรผิด .get จะไม่รีเทิร์นค่าอะไรกลับมาและโค้ดเราจะไม่error
```

```
course.get("replay")
```

```
False
```

4. set {unique}

```
# set {unique} จะเก็บค่าที่เป็นยูนีคเท่านั้น ถ้าซ้ำจะลบทิ้งทั้งหมด
courses = ["Python", "Python", "R", "SQL", "SQL", "sql"]
```

```
set (courses)
```

```
{'Python', 'R', 'SQL', 'sql'}
```

```
1
```

OOP - Object Oriented Programing

```
# OOP - Object Oriented Programing
# OOP คือ คอนเซปในการสร้างobj ใหม่ที่ไพทอนยังไม่เคยมี
```

```
#Dog Class
class Dog:
    def __init__(self, name): # dunder = double under score
        self.name = name
```

```
dog1 = Dog("milo")
dog2 = Dog("coke")
dog3 = Dog("pepsi")
```

```
print (dog1.name,
       dog2.name,
       dog3.name)
```

```
milo coke pepsi
```

```
1
```