Python 101

- variable
- data type
- data structures
- function
- control flow
- oop

```
1 print("Hello Word")
2 print("I am learning Python 101")
```

```
Hello Word
```

```
1 # comment
2 print(1+1)
3 2+2
4 3*2
5 # มันจะรันทุกบรรทัด แต่จะแสดงให้เห็นแค่บรรทัดสุดท้าย ถ้าต้องการแสดงทั้งหมดใส่ print
```

```
2
6
```

```
1 # basic calculation
2 1+1
3 print(7/2)
4 7//2 # floor division
```

```
3.5
3
```

```
1 pow(5,2) # power
```

```
25
```

```
1 abs(-999)
```

```
999
```

Modulo

```
1 print(5%2)
2 print(5%5)
```

```
1
0
```

5 building blocks

- variables
- data types
- data structures
- function
- control flow
- OOP

```
1 # assign a variavle
2 my_name = "toy"
3 age = 34
4 gpa = 3.41
5 movie_lover = True # False
6
```

```
1 # casr sensitive p!=P a!=A
2 print(age, gpa, movie_lover, my_name)
```

```
34 3.41 True toy
```

```
1 # over write a value การเขียนทับตัวแปรเดิม ต้องระวัง ชื่อตัวแปรอย่าพยายามซ้ำตัวแปรเดิม
2 age = 22
3 print(age)
```

```
22
```

```
1 s23_price = 29999
2 discount1 = 4999
3 new_s23_price1 = s23_price - discount1
4 discount2 = 0.15
5 new_s23_price2 = s23_price*(1-discount2)
6 print(new_s23_price1)
7 print(new_s23_price2)
```

```
25000
25499.149999999998
```

```
# remove variable
del new_s23_price
```

```
# count variable
age = 34
age = age +1
age += 1
age -= 2
age *= 2
age /= 2
print(age)
```

```
34.0
```

```
age = 34
gpa = 3.13
school = "Kasetsart"
movie_lover = True

```

```
#check data types
print(type(age))
print(type(gpa))
print(type(school))
print(type(movie_lover))
```

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

```
# convert type
x = 100
print(x , type(x))
x = str(x)
print(x , type(x))
```

```
100 <class 'int'>
100 <class 'str'>
```

```
y = False # t=1 f=0
y =int(y)
print(y , type(y))

```

```
0 <class 'int'>
```

```
1 z = 1
2 z = bool(z)
3 print(z, type(z))
```

```
True <class 'bool'>
```

```
1 age = 34
2 print(age+age, age*2, age/2)
```

```
68 68 17.0
```

```
1 text = "hello"
2 text + text
```

```
'hellohello'
```

```
1 # type hint ใบ้ หรือลองเปลี่ยน 34 เป็น "34" ก็รันออกแต่จะเป็น str เพราะเป็นแค่การใบ้
2 age: int  = 34
3 my_name: str = "Toy"
4 gpa: float = 3.41
5 seafood: bool = True
```

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

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

Function

```
1 print("Hello", "World")
2 print(pow(5,2))
3 print(abs(-5))
```

```
Hello World
25
5
```

```
1 # greeting()
2 def greeting(name, location = "London") :
3     print("Hello  " + name)
4     print("He is at  " + location)
```

```
1 greeting(location = "Japan",name="Toy")
```

```
Hello  Toy
He is at  Japan
```

```
def add_two_nums(num1, num2):
    print("Hi")
    print(num1 * num2)
    return num1 + num2 #หลังจาก return มันจะไม่นับอะไรเลย ถ้าจะให้แสดงอะไรใส่ก่อนl
```

```
result = add_two_nums(5 , 4)
print(result)
```

```
Hi
20
9
```

Type hints

```
def add_two_nums(a: int , b: int) -> int :  # แสดงให้รู้ว่า a,b เป็น  int และผลลัพก็จะเป็
    return a+b
```

```
add_two_nums(5,3)
```

```
8
```

String

```
# work with string
text = "hello world"

long_text = """ this is a
very long text
this is a new line """

print(text)
print(long_text)
```

```
hello world
this is a
very long text
this is a new line
```

```
# string template : fstrings
my_name = "John Wick"
location = "London"

```

```
5 text = f"Hi my name is {my_name} and I live in {location}"  # แบบใหม่ที่ใช้ง่ายกว่าแ
6 print(text)
```

```
Hi my name is John Wick and I live in London
```

```
1 "Hi my name is {}, location: {}" .format(my_name, location) #วิธีแบบเก่า #ใน .fotma
```

```
'Hi my name is John Wick, location: London'
```

```
1 text = "a duck walks into a bar"
2 print(text)
3
4 print(len(text))
```

```
a duck walks into a bar
23
```

```
1 # slicing , index starts with 0
2
3 print(text[1], text[-1], text[22])
4
```

```
 r r
```

```
1 text
```

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

```
1 # up to but not include
2 print(text[2:5])
3 print(text[2:6])
4 print(text[7:12])
5 print(text[13:17])
6 print(text[7: ])
7 print(text[-3: ])
```

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

```
1 # String is immutable
2 name = "Python" # -> Cython
3 name = "C" + name[1:]
4 # name[0] = "C" ใช้แบบนี้ไม่ได้ Error
```

```
5 print(name)
6 # ใช้วิธีประกาศตัวแปรทับตัวเดิมได้
```

```
Cython
```

```
1 text = "a duck walks intp the bar"
2 len(text) #เป็น function
```

```
25
```

```
1 # function vs. method
2 ## function()
3 ## method คือfunction ที่ถูกออกแบบมาสำหรับ data type หรือ data structure นั้นๆ
4     # เช่น function ที่เกิดมาสำหรับ string จะเรียก String methods
```

```
1 text.upper()  # นี้คือ Methods
2 text = text.upper() # เขียนทับตัวแปรเดิมจากด้านบนคือให้เป็นตัวพิมพ์ใหญ่ทั้งหมด
3 print(text)
```

```
A DUCK WALKS INTP THE BAR
```

```
1 text.lower()
```

```
'a duck walks intp the bar'
```

```
1 text.title()
```

```
'A Duck Walks Intp The Bar'
```

```
1 text = text.lower()
2 print(text)
```

```
a duck walks intp the bar
```

```
1 # อยากเปลี่ยน duck ให้เป็น Lion
2 text.replace("duck","lion")
```

```
'a lion walks intp the bar'
```

```
1 # Split ซอยย่อยคำ ตัดคำ
2 text.split(" ") #ผลที่ได้จะออกมาเป็น list
```

```
['a', 'duck', 'walks', 'intp', 'the', 'bar']
```

```
1 words = text.split(" ")
2 print(words, type(words))
```

```
['a', 'duck', 'walks', 'intp', 'the', 'bar'] <class 'list'>
```

```
1 # join การรวมคำกลับมาเหมือนเดิม
2 " ".join(words)
```

```
'a duck walks intp the bar'
```

```
1 "-".join(words)
```

```
'a-duck-walks-intp-the-bar'
```

```
1 "_".join(words)
```

```
'a_duck_walks_intp_the_bar'
```

จบ Method

method = function สร้างขึ้ยมาสำหรับ object นั้นๆ

ตัวอย่าวของ String Methods

string is immutable (คือไม่สามารถอัพเดรตค่าด้านในได้ นอกสะจากประกาสตัวแปรขึ้นมาทับอันเดิม)

```
1
```

data structure

1. list[]
2. tuple()
3. dictionary{}
4. set {unique}

```
1 # list is mutable สามารถ อัพเดรตค่าได้ # สามารภ slide ได้
2 shopping_items = ["banana","egg","milk"]
3 print(shopping_items)
```

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

```
print(shopping_items[0])
print(shopping_items[1])
print(shopping_items[2])
print(shopping_items[1: ])
print(len(shopping_items))
```

```
banana
egg
milk
['egg', 'milk']
3
```

```
shopping_items[0] = "pineapple"
shopping_items[1] = "ham cheese"
print(shopping_items)  # ทำแบบนี้กับ String ไม่ได้
```

```
['pineapple', 'ham cheese', 'milk']
```

```
# list methods #ปกติแล้วจะต้องแทนค่ากลับไปที่ตัวแปรเดิมแต่การทำแบบนี้ไม่ต้อง เป็นเหม
shopping_items.append("egg") # จะเพิ่มเขาไปใน list ทางขวามือ รัน2ครั้งก็เพิ่ม2 ครั้ง
shopping_items
```

```
['pineapple', 'ham cheese', 'egg']
```

```
# sort items (ascending order, A-Z)
shopping_items.sort()
print(shopping_items)
```

```
['egg', 'ham cheese', 'milk', 'pineapple']
```

```
shopping_items.sort(reverse=True) # descending order Z-A
print(shopping_items)
```

```
['pineapple', 'milk', 'ham cheese', 'egg']
```

```
# สร้าง function mean ขึ้นมาเอง
## reusable สามารถนำมาใช้งานได้เรื่อยๆ
def mean(scores):
    return sum(scores)/len(scores)
print(mean(scores))
```

```
86.0
```

```
scores = [90 , 88 , 85, 92, 75]
print(len(scores),sum(scores),min(scores),max(scores),mean(scores))
```

```
5 430 75 92 86.0
```

```
1 shopping_items
```

```
['pineapple', 'milk', 'ham cheese', 'egg']
```

```
1 # remove last item in list
2 shopping_items.pop() #ทุกการรันจะเป้นการลบ ตัวด้านขวาสุดทุกครั้งที่รัน
3 shopping_items
```

```
['pineapple', 'milk', 'ham cheese']
```

```
1 shopping_items.remove("milk")
2 shopping_items
```

```
['pineapple', 'ham cheese']
```

```
1 shopping_items
```

```
['pineapple', 'ham cheese', 'egg']
```

```
1 # .insert()
2 shopping_items.insert(1, "milk") # เป็นการเพิ่ม "milk" ไปในตำแหน่งที่1
```

```
1 shopping_items
```

```
['pineapple', 'milk', 'ham cheese', 'egg']
```

```
1 # list + list
2 items1 = ['egg','milk']
3 items2 = ["banaan","bread"]
4
5 print(items1+items2)
6
```

```
['egg', 'milk', 'banaan', 'bread']
```

Tuple () is immulable ไม่สามารถอัพเดทค่าได้

```
1 tup_items = ("egg", "bread", "pepsi","egg","egg")
2 tup_items
```

```
('egg', 'bread', 'pepsi', 'egg', 'egg')
```

```
# ไม่สามาทำแบบนี้ได้  กำลังพยายามเปลี่ยน egg เป็น coke
# tup_items[0] = "coke"
```

```
tup_items.count("egg")
```

```
3
```

mutable vs. immutable คือ การอัพเดทค่าได้ กับไม่สามารทำได้ ต้องประกาสตัวแปรทับเข้าไปใหม่

```
# ทำไมต้องมี tuble เอาไว้ใช้สำหรับเวลาที่ไม่อยากจะเปลี่ยนค่าบ่อยๆ
# username password
# student1, student2
s1 = ("id001", "123456")
s2 = ("id002", "654321")
user_pw = (s1 + s2 )

print(user_pw)

# เช่นตัวอย่างนี้เป็นต้น เพื่อไม่ให้สามารถอัพเดทค่า usernameได้
```

```
('id001', '123456', 'id002', '654321')
```

```
# tuble unpacking  -> กระจายค่า โดยการประกาศตัวแปร
username, password = s1
print(username, password)

```

```
id001 123456
```

```
# tuble unpacking 3 values  ต้องประกาศ3ตัวแปรเหมือนกับใน tuble  ถ้าตัวแปรไม่ตรงกันจะ
name, age , gpa = ("John Wick", 42, 3.98)  # =name, age, _= ("John Wick", 42, 3.9
print(name, age ,gpa)
print(age)
```

```
John Wick 42 3.98
42
```

set {unique}

```
courses = ["Python", "python","R", "SQL","SQL","sql"]
```

```
set(courses) #จะเก็บเฉพาะค่าที่ไม่ซ้ำกันเลยอยู่ใน set
```

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

dictionary key : value pairs => *mutable*

```
course = {
    "name" : "Data Science Bootcamp",
    "duration": "4 months",
    "students" : 200,
    "replay": True,
    "skills" : ["Google Sheet", "SQL", "R", "Python", "Stats", "ML", "Dashboard", "Data
}
```

```
course #ไม่สามารถใช้ course[0] ได้
```

```
{'name ': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': True,
 'skills': ['Google Sheet',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data Transformation']}
```

```
course["name"] # อยากดึง Value ไหนออกมา ดึงด้วย Key
```

```
'Data Science Bootcamp'
```

```
course["replay"]
```

```
True
```

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

course["language"] = "Thai"
```

```
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': True,
 'skills': ['Google Sheet',
  'SQL',
```

```
 'R',
 'Python',
 'Stats',
 'ML',
 'Dashboard',
 'Data Transformation'],
'start_time': '9am',
'language': 'Thai'}
```

```
1 # remove Key ->  google -> "python remove key from dic"
2 # delete
3 del course["language"]
```

```
1 course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': True,
 'skills': ['Google Sheet',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data Transformation'],
 'start_time': '9am'}
```

```
1 # Update value
2 course["replay"] = False
3 course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': False,
 'skills': ['Google Sheet',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data Transformation'],
 'start_time': '9am'}
```

```
1 course["skills"][0:3] # sebset ได้ เอาสกิว 0ถึง3 ไม่เอา3 มาด้วย
```

```
['Google Sheet', 'SQL', 'R']
```

```
1 course["skills"][-3: ]
```

```
['ML', 'Dashboard', 'Data Transformation']
```

```
# dictionary Method
course.keys()
```

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

```
list(course.keys() )
```

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

```
course.values()
```

```
dict_values(['Data Science Bootcamp', '4 months', 200, False, ['Google Sheet', 'SQL', 'R', 'Python', 'Stats', 'ML', 'Dashboard', 'Data Transformation'], '9am'])
```

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

```
['Data Science Bootcamp',
 '4 months',
 200,
 False,
 ['Google Sheet',
  'SQL',
  'R',
  'Python',
  'Stats',
  'ML',
  'Dashboard',
  'Data Transformation'],
 '9am']
```

```
course.items()
```

```
dict_items([('name', 'Data Science Bootcamp'), ('duration', '4 months'), ('students', 200), ('replay', False), ('skills', ['Google Sheet', 'SQL', 'R', 'Python', 'Stats', 'ML', 'Dashboard', 'Data Transformation']), ('start_time', '9am')])
```

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

```
[('name', 'Data Science Bootcamp'),
 ('duration', '4 months'),
 ('students', 200),
 ('replay', False),
 ('skills',
  ['Google Sheet',
   'SQL',
   'R',
   'Python',
   'Stats',
   'ML',
   'Dashboard',
```

```
  'Data Transformation']),
 ('start_time', '9am')]
```

```
#.get ถ้าใส่ชื่อ key ผิด มันจะไม่ return ค่าอะไรกลับมาเลย
course.get("replays")
```

```
course.get("replay") #ใช้ในกรณีการเช็ค เพราะถ้าไม่ใช่ .get มันจะ Error ที่พิมพ์ผิด แต่ถ้าใ
```

```
False
```

Recap

list, dictionary = mutable -> จะใช้บ่อย และ update value ได้

tuble, string = immutable

```
1
```

Control Flow

- if
- for
- while

```
# final exam 150 questions, pass >= 120
score = 125
if score >= 120 :
    print("passed")
else :
    print("failed")

```

```
passed
```

```
# function
def grade(score) :
    if score >= 120 :
        return "passed"
    else :
        return "failed"
    #return None
```

```
# ใช้ return เพราะว่า จะได้นำค่านั้นๆ ไปใช้ต่อได้ในตัวอย่างจะเอาไปฝากค่าใน result

```

```
result = grade(144)
print(result)
# ทำไมมีค่า None -> https://www.youtube.com/watch?v=d6GUOcDbgoY
```

```
passed
None
```

```
# Example 2
def grade2(score):
    if score >= 120 :
        return "Excellent"
    elif score >= 100 :
        return "Good"
    elif scors >= 80 :
        return "Okey"
    else :
        return "Need to read more!"

```

```
result = grade2(130)
print(result)
```

```
Excellent
```

```
# use and , or in condition
# course == data science , score >= 80 passed
# course == english , score >= 70 passed
def grade3(course, score) :
    if course == "english" and score >= 70 :
        return "passed"
    elif course == "data science" and score >= 80 :
        return "passed"
    else :
        return "failed"
```

```
grade3("english", 72)
```

```
'passed'
```

```
not True, not False
```

```
(False, True)
```

```
# for loop -> ทำทีละตัวทำที่ละ for-loop
# if score >= 80 , passed
scores = [88, 90, 75] #เวลาสร้างตัวแปรที่เป็นพหุนาม ที่มีหลายตัวเช่น list  score ก็จะมี s
```

```
for i in scores :
    print(i)
```

```
88
90
75
```

```
# สมมุติว่าต้องการลบคะแนน 2 คะแนนทุกคน
for score in scores :
    print(score-2)
```

```
86
88
73
```

```
new_scores = []
# การเอาค่าไปฝากไว้ใน list ตัวใหม่ที่อัพเดทค่าแล้ว
for score in scores :
    new_scores.append(score-2)

print(new_scores)
```

```
[86, 88, 73]
```

```
# for-loop function
def gradding_all(scores):
    new_scores = [] #สร้างไว้ก่อนเพื่อที่จะเอาvalue เข้ามาใส่
    for score in scores :
        new_scores.append(score+2)
    return new_scores
```

```
gradding_all([75, 88, 90, 95, 54])
```

```
[77, 90, 92, 97, 56]
```

list conprehension

การย่อรูป for loop แบบเต็มเหลือสั้นๆ แล้วส่งกลับมาเป็น list

```
1 scores = [75, 88, 90, 95, 54]
2 [s*2 for s in scores]     #### แนะนำแบบนี้
```

```
[150, 176, 180, 190, 108]
```

```
1 # หรือ แบบเก่า
2 for s in scores :
3     print (s*2)
```

```
150
176
180
190
108
```

```
1 # list comprehension
2 friends = ["toy", "ink", "bee", "zue", "yos"]
3
4 [f.upper() for f in friends]
5
6 #for f in friends :
7 #     print(f.upper())
```

```
['TOY', 'INK', 'BEE', 'ZUE', 'YOS']
```

```
1 # while loop -> run ไปเรื่อยๆจนกว่าจะมีเงื่อนไขเพื่อออกจาก loop
2 count = 0
3
4 while count < 5 :
5     print("hello")
6     count = count + 1 # count +=1
```

```
hello
hello
hello
hello
hello
```

chatbot for fruit order

```
1 #chatbot
2 user_name = input("what is your name? ")
```

```
what is your name? xo
```

งง ว่าทำไมย้ายไปตรงนั้นแล้วไม่ขึ้น exit ใน list

ใน live นาทีที่ 2.55 ชม.

+ โค้ด + ข้อความ

```
def chatbot():
    fruits = []
    while True :
        fruit = input("What fruit do you want to order?")
        fruits.append(fruit)
        if fruit == "exit":
            return fruits

```

```
def chatbot():
    fruits = []
    while True :
        fruit = input("What fruit do you want to order?")
        if fruit == "exit":
            return fruits
        fruits.append(fruit)
```

```
chatbot()
```

```
What fruit do you want to order?banana
What fruit do you want to order?exit
['banana']
```

HW01 - chatbot to order pizza

HW02 - pao ying chub

input รับ string

```
age = input("how old are ypu?")
```

```
how old are ypu?25
```

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

```
25 <class 'str'>
```

```
age1 = int( input("how old are ypu?") )
```

```
how old are ypu?43
```

```
1 print(age1, type(age1))
```

```
43 <class 'int'>
```

OOP = object Oriented Programming

```
1 # Dog class
2 class Dog :
3     pass
```

```
1 dog = Dog()
2 print(dog)
```

```
1 class Dog :
2     def __init__(self, name, age, breed) : #dunder
3         self.name = name
4         self.age = age
5         self.breed = breed
```

```
1 # สามารถมีได้มากกว่า 1ตัวแปร นอกจากชื่อ จะใส่ อายุ เพศ แล้วแต่
2 dog1 = Dog("ovaltine", 2, "chihuahua")
3 dog2 = Dog("milo", 3, "bulldog")
4 dog3 = Dog("pepsi", 3.5, "german shepherd")
```

```
1 print(dog1.name,dog1.age, dog1.breed)
2 print(dog2.name,dog2.age, dog2.breed)
3 print(dog3.name,dog3.age, dog3.breed)
```

```
ovaltine 2 chihuahua
milo 3 bulldog
pepsi 3.5 german shepherd
```