# Essential Python 101

Today we are learning Python 101 for beginners.

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

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

```
Hello World
```

```
print("I am learining Python 101!")
```

```
I am learining Python 101!
```

```
# comment
# this is just a note
print(1+1)
print(2*2)
print(5*3)
```

```
2
4
15
```

```
# basic calculation
1 + 1
2 * 2
5 - 3
print(7 / 2)
print(7 // 2) # floor division
```

```
3.5
3
```

```
pow(5, 2)
```

```
25
```

```
pow(5, 3)
```

```
125
```

```
abs(-666)
```

```
666
```

```
# modulo -> return เศษ
5 % 3
```

```
2
```

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

```
# assign a variables
my_name = "gyu"
age = 23
gpa = 2.49
movie_lover = True # False
```

```
# case sensitive
print(age, gpa, movie_lover, my_name)
```

```
    23 2.49 True gyu
```

```
# over write a value
age = 23
new_age = age -12
print(age, new_age)
```

```
    23 11
```

```
s23_price = 29999
discount = 0.15 # percent
new_s23_price = s23_price * (1-discount)

print(new_s23_price)
```

```
    25499.149999999998
```

```
# remove variable
del s23_price
```

```
# count variable
age = 23
age += 1 # age + 1
age += 1
age += 1
age -= 2
age *= 2
age /= 2
print(age)
```

```
    24.0
```

```
# data types
# int float string bool
```

```
age = 23
gpa = 2.49
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
x = str(x)
print(x, type(x))
```

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

```
y = False #T=1, F=0
y = int(y)
print(y, type(y))
```

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

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

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

```
age = 23
print(age+age, age*2, age/2)
```

```
    46 46 11.5
```

```
text = "I'm learning Python"
text2 = '"hahahaha'
print(text, text2)
```

```
    I'm learning Python "hahahaha
```

```
text = "hello"
print(text + text + text + text, text*4)
```

```
    hellohellohellohello hellohellohellohello
```

```
5+5
```

```
    10
```

```
# type hint -> เป็นแค่"คำใบ้" แต่ไม่บังคับ
age: int = "23"
my_name: str = "Gyu"
gpa: float = 2.49
seafood: bool = True
```

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

```
    23 <class 'str'>
```

```
# function
print("hello", "world")
print(pow(5, 2), abs(-5))
```

```
    hello world
    25 5
```

```
# greeting()
def greeting(name="John", location="London"):
    print("Hello! " + name)
    print("He is in " + location)
```

```
greeting(location="Washington",
         name="Gyu")
```

```
    Hello! Gyu
    He is in Washington
```

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

```
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(5, 6)
```

```
    11
```

```
# 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: fstring
my_name = "John Wick"
location = "London"

text = f"Hi! my name is {my_name} and I leave in {location}."

print(text)
```

```
Hi! my name is John Wick and I leave in London.
```

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

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

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

```
a r r
```

```
text
```

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

```
# up tp but not include
text[-3: ]
```

```
'bar'
```

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

```
Cython
```

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

```
len(text)
```

```
23
```

```
# function vs method
# string methods
text = text.upper()
print(text)
```

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

```
text = text.lower()
text
```

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

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

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

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

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

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

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

```
# method = function สร้างขึ้นมาสำหรับ object นั้นๆ
# string methods
# string id immutable
```

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

```
# list is mutable
shopping_items = ["banana", "egg", "milk"]

shopping_items[0] = "pineappe"
shopping_items[1] = "ham cheese"

print(shopping_items)
```

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

```
# list methods
shopping_items.append("egg")
print(shopping_items)
```

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

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

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

```
# reuseable
def mean(scores):
    return sum(scores) / len(scores)
```

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

```
5 430 75 92 86.0
```

```
# remove last item
shopping_items.pop()
shopping_items
```

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

```
shopping_items.append("egg")
shopping_items
```

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

```
shopping_items = ["pineappe", "milk", "ham cheese", "egg"]
shopping_items.remove("milk")
shopping_items
```

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

```
# .insert()
shopping_items.insert(1, "milk")
```

```
shopping_items
```

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

```
# list + list
items1 = ["egg", "milk"]
items2 = ["banana", "bread"]

print(items1 + items2)
```

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

```
# tuple () is immutable
tup_items = ("egg", "bread", "pepsi", "egg", "egg")
tup_items
```

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

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

```
3
```

```
# username password
# student1, student2
s1 = ("id001", "123456")
s2 = ("id002", "654321")
usr_pw = (s1, s2)

print(usr_pw)
```

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

```
# tuple unpacking
username, password = s1

print(username, password)
```

```
id001 123456
```

```
# tuple unpacking 3 values
name, age, _ = ("John Wich", 42, 3.98)
print(name, age)
```

```
John Wich 42
```

```
# set {unique}
course = ["Python", "Python", "R", "SQL", "SQL", "SQL"]
```

```
set(course)
```

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

```
# dictionary key: value pairs
course = {
    "name": "Data Science Bootcamp",
    "duration": "4 months",
    "students": 200,
    "replay": True,
    "skills": ["Google Sheets", "SQL", "R", "Python",
               "Stats", "ML", "Dashboard", "Data Transformation"]
}
```

```
course
```

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

```
print(course["name"])
print(course["replay"])
```

```
Data Science Bootcamp
True
```

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

course["language"] = "Thai"

course
```

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

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

course
```

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

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

course
```

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

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

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

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

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

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

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

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

```
[('name', 'Data Science Bootcamp'),
 ('duration', '4 months'),
 ('students', 200),
 ('replay', False),
```

```
     ('skills',
      ['Google Sheets',
       'SQL',
       'R',
       'Python',
       'Stats',
       'ML',
       'Dashboard',
       'Data Transformation']),
     ('start_time', '9am')]
```

```
course.get("Replay") # นิยมใช้ .get แทนแบบนี้ course("Replay") -> ERROR เพราะ .get จะไม่ error
```

```
# Recap
# list, dictionary = mutable
# tuple, string = immutable
```

```
# control flow
# if for while
```

```
# final exam 150 questions, pass >= 120
def grade(score):
    if score >= 120:
        return "Excellent"
    elif score >= 100:
        return "Good"
    elif score >= 80:
        return "Okay"
    else:
        return "Need to read more!"
```

```
result = grade(95)
print(result)
```

```
Okay
```

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

```
grade("data science", 81)
```

```
'passsed'
```

```
# for loop -> ทำทีละตัวทีละitem ที่อยู่ใน data structure นั้นๆ
# if score >= 80, passed
def grading_all(scores):
    new_scores = []
    for score in scores:
        new_scores.append(score+2)
    return new_scores
```

```
grading_all([75, 88, 90, 95, 52])
```

```
[77, 90, 92, 97, 54]
```

```
# list comprehension
scores = [75, 88, 90, 95, 52]
```

```
new_scores = [s*2 for s in scores]
new_scores
```

```
[150, 176, 180, 190, 104]
```

```
# list comprehension
friends = ["gyu", "baiyok", "benz", "dear", "pun"]
[f.upper() for f in friends]
```

```
['GYU', 'BAIYOK', 'BENZ', 'DEAR', 'PUN']
```

```
# while loop
count = 0

while count < 5:
    print("hello")
    count += 1
```

```
hello
hello
hello
hello
hello
```

```
# chatbot fot fruit order
user_name = input("What is your name? ")
```

```
What is your name? John Wick
```

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

```
chatbot()
```

```
What fruit do you want to order? banana
What fruit do you want to order? orange
What fruit do you want to order? grape
What fruit do you want to order? strawberry
What fruit do you want to order? exit
['banana', 'orange', 'grape', 'strawberry', 'exit']
```

```
# chtbot แบบไม่มีคำว่า exit ใน list
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? milo
What fruit do you want to order? valtin
What fruit do you want to order? pepsi
What fruit do you want to order? coke
What fruit do you want to order? exit
['milo', 'valtin', 'pepsi', 'coke']
```

```
# HW01 - chatbot to order pizza
# HW02 - pao ying chub
```

```
age = int( input("How old are you? ") )
```

```
How old are you? 23
```

```
type(age)
```

```
int
```

```
# OOP - Object Oriented Programming
# Dog Class
```

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

```
dog1 = Dog("ovaltin", 2, "chihuahua")
dog2 = Dog("milo", 3, "bulldog")
dog3 = Dog("pepsi", 3.5, "german shepherd")
```

```
print(dog1.name, dog1.age, dog1.breed)
print(dog2.name, dog2.age, dog2.breed)
```

```
ovaltin 2 chihuahua
milo 3 bulldog
```

```
dog4 = Dog("Wick", 4, "assasin")
```

## class Employee

มี 4 attributes => id, name, dept, pos

มี 3 actions: Employee Method => `hello()`, `work_hours()`, `change_dept()`

```
# Object: attribute => name, id, dept, pos
# Object: mrthod => hello(), change_dept()

# emp1.dept = Attribute
# emp1.change_dept("input arguiment") = Method
```

```
class Employee:
    def __init__(self, id, name, dept, pos):
        self.id = id
        self.name = name
        self.dept = dept
        self.pos = pos # position

    def hello(self):
        print(f"Hello! my name is {self.name}")

    def work_hours(self, hours):
        print(f"{self.name} works for {hours} hours.")

    def change_dept(self, new_dept):
        self.dept = new_dept
        print(f"{self.name} is now in {self.dept}.")
```

```
emp1 = Employee(1, "John", "Finance", "Financial Analyst")
```

```
print(emp1.name, emp1.pos)
```

```
John Financial Analyst
```

```
emp1.hello()
```

```
Hello! my name is John
```

```
emp1.work_hours(10)
```

```
John works for 10 hours.
```

```
emp1.dept
```

```
'Data Science'
```

```
emp1.change_dept("Data Science")
```

```
John is now in Data Science.
```

```
emp1.dept
```

```
'Data Science'
```

```
# HW03 - create new ATM class

class ATM:
    def __init__(self, name, bank, balance):
        self.name = name
        self.bank = bank
```

```
 7         self.balance = balance
 8     def deposit(self, amt):
 9         self.balance += amt
10 
11 scb = ATM("ryugyu", "scb", "500")
12 print(scb.balance)
13 
14 scb.deposit(100)
15 print(scb.balance)
```

```
500
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-99-f42a8163d443> in <cell line: 14>()
     12 print(scb.balance)
     13 
---> 14 scb.deposit(100)
     15 print(scb.balance)

<ipython-input-99-f42a8163d443> in deposit(self, amt)
      7         self.balance = balance
      8     def deposit(self, amt):
----> 9         self.balance += amt
     10 
     11 scb = ATM("ryugyu", "scb", "500")

TypeError: can only concatenate str (not "int") to str
```

SEARCH STACK OVERFLOW

0s completed at 1:09 PM