# Essentail Python 101

Today we're learning Python 101 for beginners.

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

```
# comment
# basic calculation
print(1+1)
print(2*2)
print(5*3)
1+1
2*2
5-3
```

```
2
4
15
3
```

```
print(7//2)
```

```
3
```

```
# ยกกำลัง
pow(5,2)
```

```
25
```

```
# Absolute เปลี่ยนลบเป็นบวก
abs (-666)
```

```
666
```

```
# modulo > reurn เศษการหาร (หารไม่ลงตัว)
5%2
```

```
1
```

```
# assign a variable
my_name = "Boom"
age = 28
gpa = 2.56
movie_lover = True # False
```

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

```
     28 2.56 True Boom
```

```
# Over write a value
age = 25
new_age = age - 5
print(age, new_age)
```

```
     25 20
```

```
s23_price = 29999
discount = 4999
new_s23_price = s23_price - discount
print(new_s23_price)
```

```
     25000
```

```
s23_price = 30000
discount = 0.15
new_s23_price = s23_price * (1-discount)

print(new_s23_price)
```

```
     25500.0
```

```
# remove variable
del new_s23_price
```

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

```
     35.0
```

```
# data types
# int float str bool

age = 34
gpa = 3.55
school = "IU Inter"
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 = True #T=1, F=0
y = int(y)
print(y, type(y))
```

```
    1 <class 'int'>
```

```
y = False
y = float(y)
print(y, type(y))
```

```
    0.0 <class 'float'>
```

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

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

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

```
    68 68 17.0
```

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

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

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

```
hellohellohellohellohellohellohello
```

```
# type hint
age: int = 28
my_name: str = "Boom"
gpa: float = 2.56
fruits: bool = True
```

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

```
28 <class 'int'>
```

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

```
hello world
25 5
```

```
# greeting()
def greeting(name= "Marry"):
    print("hello, geourgous " + name)
```

```
greeting("Paris")
```

```
hello, geourgous Paris
```

```
# greeting() with 2 input
def greeting(name= "Marry", location="London"):
    print("hello, geourgous " + name)
    print("He is in " + location)
```

```
greeting(location ="Japan", name= "John")
```

```
hello, geourgous John
He is in Japan
```

```
def add_two_num(a, b):
    print(a + b)
```

```
add_two_num(5, 15)
```

```
20
```

```
def add_two_num(a, b):
    return a + b
```

return ควรวางที่บรรทัดสุดท้าย เพราะ ถ้าใส่ก่อน มันจะรันแค่ก่อนหน้า return

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

```
hello beautiful
Done!
20
```

```
def add_two_num(a, b):
    print("hello beautiful")
    return a + b
    print("Done!")
```

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

```
hello beautiful
Done!
20
```

```
def add_two_num(a, b):
    print("hello beautiful")
    print("Done!")
    return a + b
```

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

```
hello beautiful
Done!
20
```

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

```
add_two_num(5, 20)
```

```
    25
```

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

long_text = """
Today is a good day
I want to eat Pizza
I like watermelon"""

print(text)
print(long_text)
```

```
    hello world

    Today is a good day
    I want to eat Pizza
    I like watermelon
```

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

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

print(text)
```

```
    Hi! my name is John Rodald and I live in London
```

```
# string template old version: format{} same thing with above
"Hi! my name is {}, location:{}".format(my_name, location)
```

```
    'Hi! my name is John Rodald, location:London'
```

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

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

```
len(text)
```

```
    23
```

```
# slicing, index starts with 0
text[0]
```

```
    'a'
```

```
text[22]
```

```
     '_'
```

```
# -1 is the last index
text[-1]
```

```
     'r'
```

```
text[0], text[-1], text[22]
```

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

```
1 # up to, but not include
text[7:12]
```

```
     'walks'
```

```
text[-3:]
```

```
     'bar'
```

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

```
     Cython
```

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

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

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

```
text
```

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

```
text.lower()
```

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

```
text.title()
```

```
     'A Duck Walks Into A Bar'
```

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

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

     'a lion walks into a bar'


text.split(" ")

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


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

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


words

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


# join the words back to sentences
" ".join(words)

     'a duck walks into a bar'


"+".join(words)

     'a+duck+walks+into+a+bar'


# method = function สร้างขึ้นมาสำหรับ object นั้นๆ
# string is immutable


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


# list
shopping_item = ["apple", "egg", "milk"]
print(shopping_item)

     ['apple', 'egg', 'milk']


shopping_item = ["apple", "egg", "milk"]
print(shopping_item[0])
print(shopping_item[1])
print(shopping_item[2])
print (len(shopping_item))

     apple
     egg
     milk
     3
```

```
# list is mutable > สามารถอัพเดทค่าได้
shopping_item[0] = "pineapple"
print(shopping_item)
```

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

```
# list methods
shopping_item.append("orange")
print(shopping_item)
```

```
     ['pineapple', 'egg', 'milk', 'orange']
```

```
# sort items (ascending order, A>Z)
shopping_item.sort()
print(shopping_item)
```

```
     ['egg', 'milk', 'orange', 'pineapple']
```

```
shopping_item.sort(reverse=True) #descending order
print(shopping_item)
```

```
     ['pineapple', 'orange', 'milk', 'egg']
```

```
# reuseable > We can create our own function
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 on the list
shopping_item.pop()
shopping_item
```

```
     ['pineapple', 'orange', 'milk']
```

```
shopping_item.append("egg")
shopping_item
```

```
     ['pineapple', 'orange', 'milk', 'egg']
```

```
# remove specify item
shopping_item.remove("milk")
shopping_item
```

```
['pineapple', 'orange', 'egg']
```

```
# .insert()
shopping_item.insert(1, "Grape")
```

```
shopping_item
```

```
['pineapple', 'Grape', 'orange', 'egg']
```

```
#list + list
item1 = ['egg', 'milk']
item2 = ['banana', 'bread']

print(item1 + item2)
```

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

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

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

```
tup_items.count('egg')
```

```
2
```

```
# username password > Tuple ใช้ในกรณีที่ไม่ต้องการให้มีการแก้ไข username + password ได้
# student1, student2
s1 = ("id001", "123456")
s2 = ("id002", "654321")
user_pw = (s1, s2)

print(user_pw)
```

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

```
# tuple unpacking
username, password = s1

print(username, password)
```

```
id001 123456
```

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

```
John 42 3.98
```

```
# tuple unpacking 3 values but called 2 values only > Using _
name, age, _ = ("John", 42, 3.98)
print(name, age,)
```

```
John 42
```

```
# set {unique} ตัด value ที่ซ้ำ
courses = ["python", "python", "R", "SQL"]
```

```
set(courses)
```

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

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

```
course
```

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

```
course["name"]
```

```
'Data science Bootcamp'
```

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

```
True
```

```
# creat new key
course["start_time"] = "9am"
```

```
course
```

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

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

```
course
```

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

```
# change True to False
course["replay"] = False
```

```
course
```

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

```
course["skill"][0:3]
```

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

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

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

```
course.keys()
```

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

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

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

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

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

```
course.items()
```

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

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

```
     False
```

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

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

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

```
     passed
```

```
score = 105
if score >=120:
    print("passed")
else:
    print("failed")
```

```
     failed
```

```
# create our own function
def grade(score):
      if score >=120:
          return "passed"
      else:
          return "failed"


result = grade(144)
print(result)
```

```
passed
None
```

```
def grade(score):
      if score >=120:
          return "Excellent"
      elif score >= 100:
          return "Good"
      elif score >= 80:
          return "OK"
      else:
          return "Need to catch up more"


result = grade(115)
print(result)
```

```
Good
```

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

```
OK
```

```
# 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 "passed"
    else:
        return "failed"


grade("english", 60)
```

```
'failed'
```

```
grade("data science", 75)
```

```
     'failed'
```

```
not True
```

```
     False
```

```
# for loop
# if score >= 80, passed
scores = [88, 90, 75]

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

```
     88
     90
     75
```

```
#update value

for score in scores:
    print(score-2)
```

```
     86
     88
     73
```

```
# add value in for loop
scores = [88, 90, 75]

new_scores = []
for score in scores:
    new_scores.append(score-2)

print(new_scores)
```

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

```
def grading_all(scores):
  new_scores = []
  for score in scores:
      new_scores.append(score+2)
  return new_scores
```

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

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

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

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

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

```
friends = ["Thang", "Devid", "bee", "Night", "PeePee"]
[f.upper() for f in friends] # This is from

# for f in friends:
 # print(f.upper())
```

```
['THANG', 'DEVID', 'BEE', 'NIGHT', 'PEEPEE']
```

```
# while loop
count = 0

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

```
hello
hello
hello
hello
hello
```

```
# chatbot for fruit order
user_name = input("what is your name?")
```

```
what is your name?Marry Run
```

```
user_name
```

```
'Marry Run'
```

```
def chatbot():
    fruits = []
    while True:
        fruit = input("What kind of fruit would you prefer? ")
        fruits.append(fruit)
        if fruit == "exit":
            return fruits
```

```
chatbot()
```

```
What kind of fruit would you prefer? Mango
What kind of fruit would you prefer? Orange
What kind of fruit would you prefer? Banana
What kind of fruit would you prefer? Tomato
What kind of fruit would you prefer? Apple
What kind of fruit would you prefer? exit
['Mango', 'Orange', 'Banana', 'Tomato', 'Apple', 'exit']
```

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


class Dog:
    pass


dog = Dog()
print(dog)
```

```
<__main__.Dog object at 0x7f33ffa69250>
```

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


dog1 = Dog("Silvy")
dog2 = Dog("Salmon")
dog3 = Dog("Pepsi")


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

```
Silvy Salmon Pepsi
```

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


dog1 = Dog("silvy", 2, "chihuahua")
dog2 = Dog("salmon", 3, "bulldog")
dog3 = Dog("pepsi", 3.5, "german shepherd")


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

```
silvy 2 chihuahua salmon 3 bulldog pepsi 3.5 german shepherd
```

```python
dog4 = Dog("wick", 4, "assasin")
```

```
# Employee class
class Employee:
    pass
```

```
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("hello!")
```

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

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

```
    John Financial Analyst
```

```
emp1.hello()
```

```
    hello!
```

```
class Employee:
    def __init__(self, id, name, dept, pos):
        self.id = id
        self.name = name
        self.dept = dept
        self.pos = pos
    def hello(self):
        print(f"hello! my name is {self.name}")
    def work_hours(self, hours):
        print(f"{self.name} works for {hours} hours.")
```

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

```
    John Financial Analyst
```

```
emp1.hello()
```

```
    hello!
```

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

    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}.")
```

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

```
John Financial Analyst
```

```
emp1.hello()
```

```
hello!
```

```
emp1.dept
```

```
'Finance'
```

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

✓ 0s completed at 8:21 PM