Essential Python 101

Today we are learning Python 101 for beginners.

- Variables
- Data Types
- Data Structures
- Function
- Control flow
- OOP

```
1 print("hello world")
```

```
hello world
```

```
1 print("I am learning Python 101!")
```

```
I am learning Python 101!
```

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

```
2
4
15
```

```
1 # basic calculation
2 1 + 1
3 2 * 2
4 5 - 3
5 print(7 / 2)
6 print(7 // 2) # floor division
```

```
3.5
3
```

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

```
25
```

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

```
125
```

```
1 abs(-666)
```

```
666
```

```
1 # modulo
2 5 % 3
```

```
2
```

```
1 # 5 building blocks
2 # 1. variables
3 # 2. data type
4 # 3. data structures
5 # 4. function
6 # 5. control flow
7 # 6. OOP = object oriented programming
```

```
1 # assign/create a variables
2 my_name = "nham"
3 age = 28
```

```
gpa = 3.06
movie_lover = True #Boolene (True, False)

```

```
my_name
```

```
'nham'
```

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

```
28 3.06 True nham
```

```
# overwrite a value
age = 28
new_age = age - 3
print(age, new_age)
```

```
28 25
```

```
s23_price = 30000
discount = 0.15 #percent in python
new_s23_price = s23_price * (1 - discount)

print(new_s23_price)
```

```
25500.0
```

```
# remove variable
del s23_price
```

```
# count variable
age = 28
age += 1 # age + 1
age *= 2
age /= 2
print(age)
```

```
29.0
```

```
# data types
# int float str bool
```

```
age = 28
gpa = 3.06
school = "Chulalongkorn"
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
type(x)
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'>
```

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

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

```
1 age = 28
2 print(age+age, age*2, age/2)
```

```
    56 56 14.0
```

```
1 text = "I'm learning Python"
2 text2 = ' "hahaha" '
3 print(text, text2)
```

```
    I'm learning Python  "hahaha"
```

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

```
    'hellohello'
```

```
1 # type hint
2 age: int = 28
3 my_name: str = "Nham"
4 gpa: float = 3.06
5 seafood: bool = True
```

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

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

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

```
    hello world
    25 5
```

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

```
1 greeting(location = "USA", name = "Mangkood")
```

```
    Hello! Mangkood
    He is at USA
```

```
1 def add_two_nums(num1, num2):
2     print("hello world")
3     print("Done!")
4     return num1 + num2 #อะไรที่อยู่หลัง return จะไม่โดนใช้งาน
```

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

```
    hello world
    Done!
    20
```

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

```
1 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 : fstrings
my_name = "Nham Khonlumb"
location = "Bangkok"

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

print(text)
```

```
Hi! my name is Nham Khonlumb and I live in Bangkok.
```

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

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

```
len(text)
```

```
23
```

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

```
'r'
```

```
text[-1]
```

```
'r'
```

```
text
```

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

```
text[2:5]
```

```
'duc'
```

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

```
'duck'
```

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

```
'walks'
```

```
text[13:17]
```

```
'into'
```

```
text[7: ]
```

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

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

```
'bar'
```

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

```
Cython
```

```
name = "Python"
name = "Cython"
```

```
print(name)
```

```
Cython
```

```
text = "a duck walks into a bar"
# function
len(text)
```

```
23
```

```
# function vs. method (function ที่ถูกสร้างมาเพื่อ object นั้นๆ)
# string method
text.upper()
```

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

```
text
```

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

```
text = text.upper()
```

```
print(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
```

```
'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(" ")
print(words, type(words))
```

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

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

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

```
# mothod = function สร้างขึ้นมาสำหรับ object นั้นๆ
# string methods
# string is immutable
```

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

```
# list
shopping_items = ["banana", "egg", "milk"]
print(shopping_items)
```

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

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

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

```
# list is mutable (สามารถ update ค่าที่อยู่ในลิสต์ได้)
shopping_items[0] = "pineapple"
print(shopping_items)
```

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

```
shopping_items[1] = "ham cheese"
print(shopping_items)
```

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

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

```
    ['pineapple', 'milk', 'ham cheese', 'egg', '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
print(shopping_items)
```

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

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

```
    5 430 75 92
```

```
sum(scores)/len(scores)
```

```
    86.0
```

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

```
print(mean(scores))
```

```
print(mean(scores))
```

```
86.0
```

```
shopping_items
```

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

```
shopping_items.pop() # remove the last value in the list
shopping_items
```

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

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

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

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

```
['pineapple', '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")
user_pw = (s1, s2)

print(user_pw)
```

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

```
# tuple unpacking
s1
```

```
('id001', '123456')
```

```
username, password = s1
print(username, password)
```

```
id001 123456
```

```
# tuple unpacking 3 values
name, age, gpa = ("Nham", 28, 3.06)
print(name, age, gpa)
```

```
Nham 28 3.06
```

```
name, age, _ = ("Nham", 28, 3.06)
print(name, age)
```

```
    Nham 28
```

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

```
set(courses)
```

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

```
# dictionary key: value pairs (Key is immutable) dictionary is mutable object
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']}
```

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

```
    'Data Science Bootcamp'
```

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

```
    True
```

```
course["start time"] = "9am"
```

```
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'}
```

```
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 in dict
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'}
```

```
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"]
```

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

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

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

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

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

```
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.keys()
```

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

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

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

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

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

```
1 course.items()
```

```
dict_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')])
```

```
1 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')]
```

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

```
False
```

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

```
False
```

```
1 # Recap
2 # list, dictionary = mutable
3 # tuple, string = immutable
4 # set = unique value
```

```
1 # control flow
2 # if for while
```

```
1 # final exam 150 questions, pass >= 120
2 score = 125
3 if score >= 120:
4     print("passed")
5 else:
6     print("failed")
```

```
passed
```

```
1 def grade(score):
2     if score >= 120:
3         print("passed")
```

```
    else:
        print("failed")
```

```
grade(125)
```

```
passed
```

```
def grade(score):
    if score >= 120:
        return("passed")
    else:
        return("failed")
```

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

```
passed
```

```
def grade(score):
    if score >= 120:
        return "Excellent"
    elif score >= 100:
        return "Good"
    elif score >= 80:
        return "Okay"
    else:
        return "Need to study 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 "passed"
    else:
        return "failed"
```

```
grade("english", 72)
```

```
'passed'
```

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

```
'passed'
```

```
not True; not False
```

```
True
```

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

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

```
86
88
73
```

```
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, 90, 95, 52])
```

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

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

```
for s in scores:
    print(s*2)
```

```
150
176
180
190
104
```

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

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

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

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

```
friends = ["nham", "junior", "mangkood", "pudding", "miso"]
for f in friends:
    print(f.upper())
```

```
NHAM
JUNIOR
MANGKOOD
PUDDING
MISO
```

```
[f.upper() for f in friends]
```

```
['NHAM', 'JUNIOR', 'MANGKOOD', 'PUDDING', 'MISO']
```

```
# 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? Nham
```

```
user_name
```

```
'Nham'
```

```
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']
```

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

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

```
how old are you? 28
```

```
type(age)
```

```
str
```

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

```
how old are you? 28
```

```
type(age)
```

```
int
```

```
# not include 'exit' in the 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? strawberry
What fruit do you want to order? apple
What fruit do you want to order? kiwi
What fruit do you want to order? grape
What fruit do you want to order? exit
['strawberry', 'apple', 'kiwi', 'grape']
```

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

```
class Dog:
    pass
```

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

```
<__main__.Dog object at 0x79901c086590>
```

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

```
dog1 = Dog("gigi")
dog2 = Dog("bowie")
dog3 = Dog("kimchi")
```

```
dog1.name
```

```
'gigi'
```

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

```
gigi bowie kimchi
```

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

```
dog1 = Dog("gigi",2 , "chihuahua")
dog2 = Dog("bowie", 3, "bulldog")
dog3 = Dog("kimchi", 3.5, "german shepherd")
```

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

```
gigi 2 chihuahua
bowie 3 bulldog
kimchi 3.5 german shepherd
```

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

```
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): # method of employee class
        print("Hello!")
```

```
emp1 = Employee(1, "Nham", "Management", "CEO")
```

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

```
Nham CEO
```

```
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.")
```

```
emp1 = Employee(1, "Nham", "Management", "CEO")
```

```
emp1.hello()
```

```
Hello! my name is Nham
```

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

```
Nham works for 10 hours.
```

```
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}.")
```

```
emp1 = Employee(1, "Nham", "Management", "CEO")
```

```
emp1.change_dept("Strategies") # mothod
```

```
Nham is now in Strategies.
```

```
emp1.dept # attribute
```

```
'Strategies'
```

```
# object: attribute => name, id, dept, pos
# object: mothod => hello(), change_dept()
```

```
# HW03 - create new ATM class

class ATM:
    pass
```

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

scb = ATM("nham", "scb", 1000000)
print(scb.balance)

scb.deposit(500000)
print(scb.balance)
```

```
1000000
1500000
```

```

```

✓ 0 วินาที เสร็จสมบูรณ์เมื่อ 01:14