Live 101 - Python

Today we are learning Python 101 for beginners

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

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

```
Hello World
```

```
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 print(7/2)
3 7//2 #floor division
```

```
3.5
3
```

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

```
125
```

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

```
666
```

```
1 # 5 building blocks
2 # 1. Variables
```

```
# 2. data types
# 3. data structures
# 4. function
# 5. control flow
# 6. OOP
```

```
# Assign a Variables
my_name = "plume"
age = 26
gpa = 2.87
movie_lover = True #or False
```

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

```
plume 26 2.87 True
```

```
# over write a value
age = 22
new_age = age+5
print(age, new_age)
```

```
22 27
```

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

print(new_s23_price)
```

```
25500.0
```

```
# remove variable
del s23_price
```

```
# count variable
age = 26
age += 1
age += 1
age -= 2
print(age)
```

```
26
```

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

```
age = 27
gpa = 2.87
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
type (x)
```

```
int
```

```
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'>
```

```
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
```

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

```
    hello I'm learning Python
```

```
1 #type hint
2 age: int = 27
3 my_name: str = "Plume"
4 gpa: float = 2.87
5 movie_lover : bool = True
```

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

```
    hello world
    25 5
```

```
1 #greeting()
2 def greeting(name = "Plume", location= "London"):
3     print("Hello World! " + name )
4     print("He live in "+ location)
```

```
1 greeting("John Wick", "Thailand")
```

```
    Hello World! John Wick
    He live in Thailand
```

```
1 def add_two_nums(x, y):
2     print(x + y)
3     print("done !")
```

```
1 add_two_nums(10, 30)
```

```
    40
    done !
```

```
1 def add_two_nums(x, y):
2     return x + y
3     print("done")
```

```
result = add_two_nums(15, 30)
print(result)
```

```
45
```

```
# 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 = "Plume"
location = "London"

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

print(text)
```

```
Hi! my name is Plume and I live 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])
```

```
a r
```

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

```
'walks'
```

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

```
Cython
```

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

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

```
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 นั้น
# string methods
# string is immutable
```

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

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

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

print(shopping_items)
```

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

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

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

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

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

```
# reusable
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
```

```
shopping_items
```

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

```
# remove last item in list
shopping_items.pop()
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")
tup_items
```

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

```
#username password
s1= ("id001", "123456")
s2= ("id002", "444444")
user_pw = (s1, s2)

print(user_pw)
```

```
(('id001', '123456'), ('id002', '444444'))
```

```
# tuple unpacking
username, password =s1
print(username, password)
```

```
id001 123456
```

```
# tuple unpacking 3 values
name, age, gpa =("plume", 27, 2.87)
print(name, age, gpa)
```

```
plume 27 2.87
```

```
#set {unique}
courses = ["python", "python", "R", "SQL", "sql"]
set(courses)
```

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

```
# dictionary key : value pairs
course = {
    "name": "Data Science Bootcamp",
```

```
    "duration": "4 months",
    "students": 200,
    "replay" : True,
    "skills": ["google sheets", "SQL", "R", "Python"]

}
```

```
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'],
 'start_time': '9am',
 'language': 'Thai'}
```

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

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': True,
 'skills': ['google sheets', 'SQL', 'R', 'Python'],
 'start_time': '9am'}
```

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

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': False,
 'skills': ['google sheets', 'SQL', 'R', 'Python'],
 'start_time': '9am'}
```

```
course
```

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': False,
 'skills': ['google sheets', 'SQL', 'R', 'Python'],
 'start_time': '9am'}
```

```
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'],
 '9am']
```

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

```
[('name', 'Data Science Bootcamp'),
 ('duration', '4 months'),
 ('students', 200),
 ('replay', False),
 ('skills', ['google sheets', 'SQL', 'R', 'Python']),
 ('start_time', '9am')]
```

```
# จะไม่ error แต่จะไม่ return ค่ากลับมา
course.get("replays")
```

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

```
#control flow
# of 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 "passed"
    else:
        return "failed"
```

```
grade("english", 70)
```

```
'passed'
```

```
#for loop
# 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]
```

```
friends= ["toy","plume","pink","blue"]
[ f.upper() for f in friends]
```

```
['TOY', 'PLUME', 'PINK', 'BLUE']
```

```
# 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?Pasit
```

```
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? yoyo
What fruit do you want to order? ole
What fruit do you want to order? okey
What fruit do you want to order? exit
['yoyo', 'ole', 'okey']
```

```
# HW01 - Chatbot to order pizza
# HW02 - Pao Ying Chub
```

```
# 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("ovaltine", 2, "chihuahua")
dog2 = Dog("milo", 3, "bulldog")
dog3 = Dog("pepsi", 1.5, "golden")
```

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

```
ovaltine 2 chihuahua
milo 3 bulldog
pepsi 1.5 golden
```

```
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(8)
```

```
John works for 8 hours.
```

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

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

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

```
# HW03 - Create new ATM class

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

```
scb = ATM("plume", "scb", 500)
print(scb.balance)
```

```
    500
```

```
scb.deposit(100)
print(scb.balance)
```

```
    600
```

```
1
```