Essential Python 101

Today we are learning python 101 for beginners.

- variables
- datat 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(3*5)
6 print(7 // 2) # floor division ปัดลง
```

```
2
4
15
3
```

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

```
25
```

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

```
125
```

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

```
666
```

```
# modulo # return เศษ
5 % 3
```

```
2
```

```
# 5 building blocks of any language
# 1. variables
# 2. data types
# 3. data structures
# 4. function
# 5. control flow
# 6. OOP (suit for software)
```

```
# assign a variable
# python use snake_case for variable name
# use lower case because python is case sensitive
my_name = "toy"
age = 34
gpa = 3.41
movie_lover = True # False
```

```
my_name
```

```
'toy'
```

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

```
34 3.41 True toy
```

```
# over write a value
age = 34
new_age = age - 12
print(age, new_age)
```

```
34 22
```

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

print(new_s23_price)
```

```
25499.149999999998
```

```
# remove variable
del s23_price
```

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

```
35.0
```

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

```
age = 31
gpa = 2.67
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)
print( 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 = 34
2 print(age+age, age*2, age/2)
```

```
68 68 17.0
```

```
1 text1 = "I'm learning Python"
2 text2 = ' "hahahaha" '
3 print(text1, text2)
```

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

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

```
'hellohello'
```

```
1 # type hint
2 # แค่ใบ้แต่ไม่ได้บังคับ
3 age: int = 34
4 my_name: str = "Toy"
5 gpa: float = 3.41
6 seafood: bool = True
```

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

```
34 <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="John", location="London"):
3     print("Hello! " + name)
4     print("He is in " + location)
```

```
4     print("He is in " + location)
```

```
1 greeting("Wick")
```

```
Hello! Wick
He is in London
```

```
1 greeting("Naruto", "Washington")
```

```
Hello! Naruto
He is in Washington
```

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

```
Hello! Naruto
He is in Japan
```

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

```
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
```

```
1 # work with string
2 text = "hello world"
3 long_text = """
4 this is a
5 very long text
6 this is a new line"""
7
8 print(text)
9 print(long_text)
```

```
hello world

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

```
# string template : fstrings
my_name = "A.T."
location = "Thailand"

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

print(text)
```

```
Hi! my name is A.T. and I live in Thailand.
```

```
# old format
"Hi! my name is {}, location: {}".format(my_name, location)
```

```
'Hi! my name is A.T., location: Thailand'
```

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

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

```
len(text)
```

```
23
```

```
# slicing, index start at 0
print(text[0], text[-1], text[22])
```

```
a r r
```

```
# up to 6 but not include
text[2:6]
text[7:12]
text[-10:-6]
# from 7 to end
text[7: ]
text[-3: ]
```

```
'bar'
```

```
# string is immutable
# can not update value in string variable
name = "Python" # -> Cython
name = "C" + name[1: ]
print(name)
```

```
Cython
```

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

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

```
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'
```

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

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

```
# join list
" ".join(words)
```

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

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

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

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

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

```
['pineapple', 'ham cheese', 'milk']
pineapple
['ham cheese', 'milk']
3
```

```
# list methods
# append -> add new value to the right
# list methods no need to assign value back

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

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

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

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

```
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 in list
shopping_items.pop()
shopping_items
```

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

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

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

```
# search google -> python list method remove
shopping_items.remove("milk")
shopping_items
```

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

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

```
['pineapple', 'milk', 'ham cheese', '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", "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
# ประกาศตัวแปร 2 ตัวพร้อมกัน
username, password = s1
print(username, password)
```

```
id001 123456
```

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

```
John Wick 42
```

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

```
set(courses)
```

```
{'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["students"]
```

```
200
```

```
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'}
```

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

```
1 # delete
2 del course["language"]
3 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'}
```

```
1 del course["start_time"]
2 course
```

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

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

```
{'name': 'Data Science Bootcamp',
 'duration': '4 months',
 'students': 200,
 'replay': False,
```

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

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

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

```
# dictionary methods
list( course.keys() )
```

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

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

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

```
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'])]
```

```
# wrong key but code not error
course.get("replay")
```

```
False
```

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

```
False
```

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

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

```
# final exam 150 question, pass >= 120
score = 105
if score >= 120:
    print("passed")
else:
    print("failed")
```

```
failed
```

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

```
'passed'
```

```
print(not True); print(not False)
```

```
False
True
```

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

```
new_scores = []

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

print(new_scores)
```

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

```
# 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]
```

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

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

```
friends = ["toy", "ink", "bee", "zue", "yos"]
[f.upper() for f in friends]
```

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

```
# 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? John Wick
```

```
user_name
```

```
'John Wick'
```

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

```
how old are you? 34
```

```
type(age)
```

```
str
```

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

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

```
dog1 = Dog("ovaltine", 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)
```

```
ovaltine 2 chihuahua
milo 3 bulldog
```

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

    # action
    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.change_dept("Data Science")
```

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

```
# Object: attribute => name, id, dept, pos
# Object_ method => hello, work_hours, change_dept
```

```
# HW03 - create new ATM class
# ฝากเงิน, ถอนเงิน, ขอ OTP
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("toyeiei", "scb", 500)
print(scb.balance)
```

```
500
```

```
scb.deposit(500)
print(scb.balance)
```

```
1000
```

```
1
```

✓ 0s completed at 12:07 PM