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 learning Python 101!")
```

```
I am learning 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
```

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

```
1 # assign a variable
2 my_name = "toy"
3 age = 34
4 gpa = 3.41
5 movie_lover = True # False
```

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

```
    34 3.41 True toy
```

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

```
    34 22
```

```
1 s23_price = 30000
2 discount = 0.15
3 new_s23_price = s23_price * (1-discount)
4
5 print(new_s23_price)
```

```
    25500.0
```

```
# remove variable
del 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 = 3.41
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'>
```

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

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

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

```
    hellohellohellohello hellohellohellohello
```

```
1 5+5
```

```
    10
```

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

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

```
    34 <class 'str'>
```

```
1 # function
2 print("hello", "world")
3 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="Toy")
```

```
Hello! Toy
He is in Washington
```

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

```
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 : fstrings
my_name = "John Wick"
location = "London"

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

print(text)
```

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

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

```
'Hi! my name is John Wick, location: 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 to, but not include
text[-3: ]
```

```
'bar'
```

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

```
Cython
```

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

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

```
# function vs. method
```

```
2 # string methods
3 text = text.upper()
4 print(text)
```

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

```
1 text.title()
```

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

```
1 text = text.lower()
2 text
```

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

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

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

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

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

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

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

```
1 # method = function สร้างขึ้นมาสำหรับ object นั้นๆ
2 # string methods
3 # string is immutable
```

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

```
1 # list is mutable
2 shopping_items = ["banana", "egg", "milk"]
3
4 shopping_items[0] = "pineapple"
5 shopping_items[1] = "ham cheese"
6
7 print(shopping_items)
```

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

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

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

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

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

```
1 # reusable
2 def mean(scores):
3     return sum(scores)/ len(scores)
```

```
1 scores = [90, 88, 85, 92, 75]
2
3 print(len(scores), sum(scores),
4       min(scores), max(scores), mean(scores))
```

```
5 430 75 92 86.0
```

```
1 # remove last item in list
2 shopping_items.pop()
3 shopping_items
```

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

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

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

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

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

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

```
1 shopping_items
```

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

```
1 # list + list
2 items1 = ['egg', 'milk']
3 items2 = ['banana', 'bread']
4
5 print(items1 + items2)
```

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

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

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

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

```
3
```

```
1 # username password
2 # student1, student2
3 s1 = ("id001", "123456")
4 s2 = ("id002", "654321")
5 user_pw = (s1, s2)
6
7 print(user_pw)
```

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

```
1 # tuple unpacking
2 username, password = s1
3
4 print(username, password)
```

```
id001 123456
```

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

```
John Wick 42
```

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

```
1 set(courses)
```

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

```
1 # dictionary key: value pairs
2 course = {
3     "name": "Data Science Bootcamp",
4     "duration": "4 months",
5     "students": 200,
6     "replay": True,
7     "skills": ["Google Sheets", "SQL", "R", "Python",
8                "Stats", "ML", "Dashboard", "Data Transformation"]
9 }
```

```
1 course["start_time"] = "9am"
2
3 course["language"] = "Thai"
4
5 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'}
```

```
1 # delete
2 # del course["start_time"]
3 course["replay"] = False
4 course
```

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

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

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

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

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

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

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

```
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'])]
```

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

```
1 # Recap
2 # list, dictionary = mutable
3 # tuple, string = immutable
```

```
1 # control flow
2 # 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 "passed"
    else:
        return "failed"
```

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

```
'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]
```

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

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

```
1 # list comprehension
2 friends = ["toy", "ink", "bee", "zue", "yos"]
3 [f.upper() for f in friends]
```

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

```
1 # while loop
2 count = 0
3
4 while count < 5:
5     print("hello")
6     count += 1
```

```
    hello
    hello
    hello
    hello
    hello
```

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

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

```
1 def chatbot():
2     fruits = []
3     while True:
4         fruit = input("What fruit do you want to order? ")
5         if fruit == "exit":
6             return fruits
7         fruits.append(fruit)
```

```
1 chatbot()
```

```
    What fruit do you want to order? milo
    What fruit do you want to order? ovaltine
    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', 'ovaltine', 'pepsi', 'coke']
```

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

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

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

```
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("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
```

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

```
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
```

```
'Finance'
```

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

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

```
emp1.dept
```

```
'Data Science'
```

```
# Object: attribute => name, id, dept, pos
# Object: 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, amt):
        self.balance += amt

scb = ATM("toyeiei", "scb", 500)
print(scb.balance)

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

```
    500
    600


1
```