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 2*2
```

```
2
4
```

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

```
3.5
3
```

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

```
25
```

```
1 abs(-6)
```

```
6
```

```
1 # modulo will return remainders
2 5%2
```

```
1
```

```
1
```

```
1 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 เหมาะกับงาน software
```

```
1 # assign a variable
2 my_name = "toy"
3 age = 34
4 gpa = 3.38
5 movie_lover = True #False
```

```
1 my_name
```

```
'toy'
```

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

```
34 3.38 True
```

```
1 # overwrite a value
2 age = 22
3 new_age = age-12
4 print(age, new_age)
```

```
22 10
```

```
1 s23_price = 29999
2 discount = 0.15
3 new_s23_price = s23_price * 0.85
4
5 print(new_s23_price)
```

```
25499.149999999998
```

```
1 # remove variable
2 del s23_price
```

```
1 # count variable
2 age = 34
3 age +=1
```

```
4 age -=2
5 age *=2
6 age /=2
7 print(age)
```

```
33.0
```

```
1 # data types
2 # int จำนวนเต็ม float ทศนิยม str = text , bool = True false
3
```

```
1 age = 23
2 gpa = 3.37
3 school = "Chula"
4 movie_lover = True
```

```
1 #check data types
2 type(age)
```

```
int
```

```
1 # convert type
2 x = 100
3 x = str(x)
```

```
1 print (x, type(x))
```

```
100 <class 'str'>
```

```
1 y = True #T=1 F=0
2 y = int(y)
3 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 text = "hello"
2 print (text+text+text, text*4)
```

```
hellohellohello hellohellohellohello
```

```
1 # type hint
2 age: int = 23
3 name: str ="Bright"
4 gpa: float =3.38
5 seafood: bool = True
```

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

```
23 <class 'int'>
Bright <class 'str'>
```

```
1 #function
2 print("Hello", "world")
3 print(pow(5,2), abs(-9))
```

```
Hello world
25 9
```

```
1 #greeting() สร้าง function เอง
2 def greeting(name = "bright", location = "London"):
3     print("Hello " + name)
4     print("She is at " + location)
```

```
1 greeting("Bright","Coventry")
```

```
Hello Bright
She is at Coventry
```

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

```
1 result = add_two_nums(5,25)
2 print(result)
```

```
hello world
30
```

```
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 สร้าง template แล้วดึงค่าตัวแปรเข้าไปใส่
my_name = "Bright"
location = "london"

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

```
print(text)
```

```
Hi! my name is Bright and I live in 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[22]
```

```
'r'
```

```
text[13:17] #ไม่ include number 5
```

```
'into'
```

```
# string is immutable = ถ้าสร้างแล้ว update ค่าไม่ได้
name = "Python"
```

```
name = "C" + name[1: ] # ต้องการจะแก้แค่ ตัว P เป็น C
print(name)
```

```
Cython
```

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

```
23
```

```
len(text)
```

```
23
```

```
#function vs. method(ฟังก์ชันที่ถูกออกแบบมาสำหรับตัว object นั้นๆ)
# string method = function ที่เกิดขึ้นมาสำหรับ string ให้พิม dot after function
text = text.upper()
print(text)
```

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

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

```
print(text)
```

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

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

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

```
words = text.split(" ") #split by comma
```

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

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

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

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

shopping_items[0] = "pineapple"

print(shopping_items)
```

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

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

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

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

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

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

```
5 430 75 92
```

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

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

```
#select specific item to remove
shopping_items.remove("ham")
shopping_items
```

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

```
# .insert() imsert แบบ specific position
shopping_items.insert(3, "ham")
shopping_items
```

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

```
shopping_items.pop()
shopping_items
```

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

```
#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')
```

```
tup_items.count("egg")
```

```
1
```

```
#username password
#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, _ = ("John Wick", 42, 3.98)
print(name, age)
```

```
John Wick 42
```

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

```
set(courses)

```

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

```
#dictionary key: value pairs
course = {
    "name" : "DSBootcamp",
    "duration": "4 months",
    "students": 200,
    "skills": ["Googlesheets","SQL", "R", "Python"]
}
```

```
course
```

```
{'name': 'DSBootcamp',
 'duration': '4 months',
 'students': 200,
 'skills': ['Googlesheets', 'SQL', 'R', 'Python']}
```

```
course["students"]
```

```
200
```

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

```
course
```

```
{'name': 'DSBootcamp',
 'duration': '4 months',
 'students': 200,
 'skills': ['Googlesheets', 'SQL', 'R', 'Python'],
 'start time': '9am'}
```

```
#delete
del course["start time"]
course
```

```
{'name': 'DSBootcamp',
 'duration': '4 months',
 'students': 200,
 'skills': ['Googlesheets', 'SQL', 'R', 'Python']}
```

```
#update
course["students"] = 150
course
```

```
{'name': 'DSBootcamp',
 'duration': '4 months',
 'students': 150,
 'skills': ['Googlesheets', 'SQL', 'R', 'Python']}
```

```
course["skills"][2: ]
```

```
['R', 'Python']
```

```
course.keys()
```

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

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

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

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

```
['DSBootcamp', '4 months', 150, ['Googlesheets', 'SQL', 'R', 'Python']]
```

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

```
[('name', 'DSBootcamp'),
 ('duration', '4 months'),
 ('students', 150),
 ('skills', ['Googlesheets', 'SQL', 'R', 'Python'])]
```

```
course.get("skills")
```

```
['Googlesheets', 'SQL', 'R', 'Python']
```

```
#final exam 150, pass120
def grade(score):
    if score >= 120:
        return "Excellent"
    elif score >= 100:
        return "Good"
    elif score >= 80:
        return "ok"
    else:
        return "failed"
```

```
result = grade(125)
print(result)
```

```
Excellent
```

```
#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: #ใช้ == เพื่อเทียบ 2 ฝั่งสมการ
        return "passed"
    elif course == "data science" and score >= 80:
        return "passed"
```

```
    else:
        return "failed"
```

```
grade("english",60)
```

```
'failed'
```

```
#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 = [75,88,90,92,52]

```

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

```
[150, 176, 180, 184, 104]
```

```
#while loop
count = 0

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

```

```
hello
hello
hello
hello
hello
```

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

```

```
what is your name?Bright
'Bright'
```

```
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?B
What fruit do you want to order?V
What fruit do you want to order?G
What fruit do you want to order?exit
['B', 'V', 'G']
```

```
#OOP object oriented Programming = concept in building object
# Dog class

```

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

```

```

```
#dog 123 refers to self, Dog() refers name
dog1 = Dog("ovaltine",15)
dog2 = Dog("milo",6)
dog3 = Dog("pepsi",4)
```

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

```
ovaltine 15
milo 6
```

```
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): #ประกาศ action ให้เค้าทำอะไรขึ้นมาด้วย
        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}.")
```

```
1 emp1 = Employee(1, "Bright", "Finance", "FA")
```

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

```
Bright FA
```

```
1 emp1.hello()
```

```
hello my name is Bright
```

```
1 emp1.work_hours(6)
```

```
Bright works for 6 hours
```

```
1 emp1.change_dept("Marketing")
```

```
Bright is now in Marketing.
```

```
1
```

✓ 0s completed at 14:31