Essential Python

Today we are learning Python gor beginners.

- 1. Variables
- 2. Data Types
- 3. Functions
- 4. Data Structures
- 5. Control Flow
- 6. OOP - Object Oriented Programming

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

```
Hello World
```

```
1 # Comment
2 print("I am learning Python")
3 1 + 1
4 1 - 1
5 2 * 2
6 print(7 / 2)
7 print(7 // 2)
```

```
I am learning Python
3.5
3
```

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

```
25
```

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

```
125
```

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

```
666
```

```
1 # Modulo
2 5 % 3
```

```
2
```

```
## 5 Building Blocks
# 1. Variables
# 2. Data Types
# 3. Functions
# 4. Data Structures
# 5. Control Flow
# 6. OOP - Object Oriented Programming

```

1. Variables

```
## 1. Variables
# Assing a Variable
my_name = "Mon"
age = 21
gpa = 3.30
movie_lover = True # False
```

```
# Case Sensitive
print(age, gpa, movie_lover, my_name)
```

```
21 3.3 True Mon
```

```
# Over write a value
age = 31
new_age = age - 11
print(age, new_age)
```

```
31 20
```

```
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 = 21
age += 1
age += 1
age += 1
age -= 2
age *= 2
age /= 2
print(age)
```

```
22.0
```

2. Data Types

```
## 2. Data Types
# int float str bool
age = 21
gpa = 3.30
school = "KMUTT"
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 Types
x = 100
x = str(x)
print(x, type(x))
```

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

```
y = True # False
y = int(y)
print(y, type(y))
```

```
1 <class 'int'>
```

```
z = 1 # 0
z = bool(z)
```

```
3 print(z, type(z))
```

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

```
1 age = 21
2 print(age+age, age*2, age/2)
```

```
42 42 10.5
```

```
1 text = "Hello"
2 print(text + text)
3 print(text * 4)
```

```
HelloHello
HelloHelloHelloHello
```

```
1 # Type Hint
2 age:  int = 21
3 age:  str = "Mon"
4 gpa: float = 3.30
5 seafood: bool = True
```

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

```
Mon <class 'str'>
```

3. Function

```
1 ## 3. 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)
```

```
1 greeting()
2 greeting("Mon")
3 print("\n")
4 greeting("Naruto", "Washington")
```

```
greeting(location = "Washington",
         name = "Naruto")
```

```
Hello John
He is in London
Hello Mon
He is in London


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

```
def add_two_nums(num1, num2):
    print("Hello World")
    return num1 + num2
    print("Done!")
```

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

```
Hello World
20
```

```
def add_three_nums(a: int, b: int, c: int):
    return a + b + c
```

```
add_three_nums(5, 6, 4)
```

```
15
```

```
# Work with String
text = "Hello World\n"

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
```

```
# tring Template: fstring
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 {} and I live in {}.".format(my_name, location)
```

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

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

len(text)
```

```
a duck walks into a bar
23
```

```
# Slicing, index starts with 0.
print(text[0], text[-1], text[22], text[23-1])
```

```
a r r r
```

```
# Up to, But not include
print(text[13:17])
print(text[7: ])
print(text[-3: ])
```

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

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

```
Cython
```

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

# Function vs. Method
# Sting method
# String is immutable
```

```
# String is immutable
text1 = text.upper()
print(text1)

text2 = text.lower()
print(text2)

text3 = text.title()
print(text3)
```

```
A DUCK WALKS INTO A BAR
a duck walks into a bar
A Duck Walks Into A Bar
```

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

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

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

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

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

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

4. Data Structures

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

```
# 1. list [] is mutable
shopping_items =["banana", "egg", "milk"]
print(shopping_items[0])
print(shopping_items[1])
print(shopping_items[2])
print(shopping_items[1:])
print( len(shopping_items) )
```

```
banana
egg
milk
```

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

```
shopping_items[0] = "pineapple"
shopping_items[1] = "ham"
print(shopping_items)
```

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

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

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

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

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

```
# Remove last item in list
shopping_items.pop()
shopping_items
```

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

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

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

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

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

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

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

```
# list + list
item1 = ['egg', 'milk']
item2 = ['banana', 'bread']
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
username, password = s1

print(username, password)
```

```
id001 123456
```

```
# tuple unpacking 3 values
name, age, _ = ("Mon", 21, 3.30)
```

```
print(name, age)
```

```
Mon 21
```

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

```
set(courses)
```

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

```
# dictionary key: value pairs
courses = {
    "name": "Data Science",
    "duration": "4 months",
    "replay": True,
    "skills": ["Google Sheets", "SQL", "R", "Data Tranformation",
               "Data Visualization", "Stats", "ML", "Python", "Dashboard"]
}

print(courses)
```

```
{'name': 'Data Science', 'duration': '4 months', 'replay': True, 'skills': ['Google Sheets', 'SQL', 'R', 'Data Tranfor
```

```
courses["replay"] = False
courses["start_time"] = "9am"
courses["language"] = "Thai"

courses
```

```
{'name': 'Data Science',
 'duration': '4 months',
 'replay': False,
 'skills': ['Google Sheets',
  'SQL',
  'R',
  'Data Tranformation',
  'Data Visualization',
  'Stats',
  'ML',
  'Python',
  'Dashboard'],
 'start_time': '9am',
 'language': 'Thai'}
```

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

```
3 del courses["start_time"]
4 courses
```

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

```
1 print(courses["skills"][0:3])
2 courses["skills"][-3: ]
```

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

```
1 list( courses.keys() )
```

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

```
1 list( courses.values() )
```

```
['Data Science',
 '4 months',
 False,
 ['Google Sheets',
  'SQL',
  'R',
  'Data Tranformation',
  'Data Visualization',
  'Stats',
  'ML',
  'Python',
  'Dashboard']]
```

```
1 list( courses.items() )
```

```
[('name', 'Data Science'),
 ('duration', '4 months'),
 ('replay', False),
 ('skills',
  ['Google Sheets',
   'SQL',
   'R',
   'Data Tranformation',
   'Data Visualization',
```

```
            'Stats',
            'ML',
            'Python',
            'Dashboard'])]
```

```
courses.get("replays")
```

```
courses.get("replay")
```

```
False
```

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

5. Control Flow

```
## 5. Control Flow
# if for while
```

```
 # final exam 150 questions, pass >= 120
def grade(score):
    if score >= 120:
        return "Passed"
    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

#scores = [88, 90, 75]
#new_scores = []
#for score in scores:
#    new_scores.append(score-2)

#print(new_scores)
```

```
grading_all([88, 90, 75])
```

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

```
# list comprehension
scores = [88, 90, 75]
```

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

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

```
[176, 180, 150]
```

```
friends = ["mon", "drew", "ya"]
new_friends = [f.upper() for f in friends]
new_friends
```

```
['MON', 'DREW', 'YA']
```

```
# while loop
count = 0

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 ?: Mon
```

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

```
1 chatbot()
```

```
What fruits do you want to order ?: Mon
What fruits do you want to order ?: exit
[]
```

```
1 age = int( input("How old are you ?:") )
```

```
How old are you ?:21
```

```
1 type(age)
```

```
int
```

6. OOP - Object Oriented Programming

```
1 ## 6. OOP - Object Oriented Programming
2 # Dog Class
```

```
1 #class Dog:
2 #     pass
```

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

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

```
Ovaltine 2 Chihuahua
Milo 3 Bulldog
Pepsi 3.5 German Shepherd
```

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

```
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
```

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

```
Hello! my name is John
```

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

```
John works for 10 hours.
```

```
1 emp1.dept
```

```
'Finance'
```

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

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

```
1 emp1.dept
```

```
'Data Science'
```

✓ 0 วินาที เสร็จสมบูรณ์เมื่อ 12:22