Function and Module

```
import warnings
warnings.filterwarnings('ignore')
```

I. def

1) 사용자 정의 함수 with def

- hap( ) 정의 및 사용

```
def hap(x, y):
  z = x + y
  return z
```

```
hap(3, 6)
```

```
9
```

2) 사용자 정의 함수 with lambda

- cha( ) 정의 및 사용

```
cha = lambda x, y : x - y
```

```
cha(3, 6)
```

```
-3
```

II. Module

- Module : 사용자 정의 함수를 포함하고 있는 파이썬 스크립트(*.py)
- import : 파이썬 스크립트에 정의된 함수를 메모리로 호출

1) 파이썬 내장 모듈

- 사용구문 : import Module_Name

```
import time
```

- 사용구문 : Module_Name.Function_Name( )

```
print('스크립트 시작')
time.sleep(9)
print('9초 후 스크립트 종료')
```

```
스크립트 시작
9초 후 스크립트 종료
```

- Built-in Module Path

```
import sys

sys.path
```

```
['',
 '/env/python',
 '/usr/lib/python36.zip',
 '/usr/lib/python3.6',
 '/usr/lib/python3.6/lib-dynload',
 '/usr/local/lib/python3.6/dist-packages',
 '/usr/lib/python3/dist-packages',
 '/usr/local/lib/python3.6/dist-packages/IPython/extensions',
 '/root/.ipython']
```

2) Alias로 import

- 사용구문 : import Module_Name as Alias

```
import time as t
```

- 사용구문 : Alias.Function_Name( )

```
print('3초 후에 결과 출력')
t.sleep(3)
print('축하합니다. 합격입니다!')
```

```
3초 후에 결과 출력
축하합니다. 합격입니다!
```

3) 사용자 모듈 만들기

- hello.py 스크립트 생성(UTF-8 Encoding)
- 스크립트에 hi( ) 함수 정의
- Colab에 hello.py 업로드 후 실습 진행

```
import hello
```

```
hello.hi()
```

```
사용자 정의 함수 실행
Hello World
사용자 정의 함수 종료
```

III. Package

- 관련된 모듈을 디렉토리 단위로 관리
- import 또는 from import 구문으로 함수를 호출하여 사용

1) 사용자 패키지 만들기

- Colab에 myPKG 디렉토리 생성
- myPKG 디렉토리 내에 **init**.py 생성
- **init**.py에 version=1.0 작성 후 UTF-8로 저장
- myLibrary.py Module 생성(UTF-8)
- myLibrary.py 내에 hi( ) 및 hap( ) 함수 정의
- myLibrary.py 파일을 myPKG 디렉토리에 복사

2) 사용자 패키지 사용 with import

- 사용구문 : import Directory_Path.Module_Name

```
import myPKG.myLibrary
```

- 사용구문 : Directory_Name.Module_Name.Function_Name

```
myPKG.myLibrary.hi()
```

```
첫번째 함수 실행
```

```
myPKG.myLibrary.hap(3, 5)
```

```
8
```

3) 사용자 패키지 사용 with from ~ import

- 사용구문 : from Directory_Name.Module_Name import Function_Name

```
from myPKG.myLibrary import hi
from myPKG.myLibrary import hap
```

- 사용구문 : Function_Name

```
hi()
```

```
첫번째 함수 실행
```

```
hap(2, 6)
```

```
8
```

IV. Class

- Class/Name Space : 개발자에 의해 지정된 독립 공간
- Class Member : Class 내에 선언된 변수
- Class Method : Class 내에 선언된 함수

1) Class 선언

```
class myClass:
    var_1 = 'Hello Class'

    def func_1(self):
        print(self.var_1, '클래스 함수 실행')
```

2) Instance 생성

```
obj = myClass()
```

3) Member 호출

```
obj.var_1
```

```
'Hello Class'
```

4) Method 호출

```
obj.func_1()
```

```
Hello Class 클래스 함수 실행
```

V. Class Member vs. Instance Member

1) Class 선언

- Class Member
- Instance Member

```
class myClass_1:
    var_1 = 'Class Member'

    def func_1(self):
        var_2 = 'Instance Member'
        print('Method 내에 선언', var_2)
        print(' Class 내에 선언', self.var_1)

    def func_2(self, x, y):
        z = x + y
        print(x, '더하기', y, '는', z)
```

2) Instance 생성

```
obj_1 = myClass_1()
```

3) Class Member 호출

```
obj_1.var_1
```

```
'Class Member'
```

4) Instance Member 호출 에러

```
obj_1.var_2
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-27-062f88125ce0> in <module>()
----> 1 obj_1.var_2

AttributeError: 'myClass_1' object has no attribute 'var_2'

SEARCH STACK OVERFLOW
```

5) Class Method 내에서 Class Member 및 Instance Member 호출 성공

```
obj_1.func_1()
```

```
Method 내에 선언 Instance Member
 Class 내에 선언 Class Member
```

6) 인자가 있는 Class Method - func_2(x, y)

```
obj_1.func_2(3, 5)
```

```
3 더하기 5 는 8
```

VI. Class 생성자 vs. Class 소멸자

1) Class 선언

```
class myClass_2:
    def __init__(self):
        print('myClass_2 인스턴스가 메모리에 생성되었습니다')

    var_1 = 'Class Member'

    def func_1(self):
        print('Class Method')

    def __del__(self):
```

```
        print('myClass_2 인스턴스가 메모리에서 소멸되었습니다')
```

2) 인스턴스 생성 메시지 출력

```
obj_2 = myClass_2()
```

```
myClass_2 인스턴스가 메모리에 생성되었습니다
```

3) Class Member & Class Method

```
obj_2.var_1
```

```
'Class Member'
```

```
obj_2.func_1()
```

```
Class Method
```

4) 인스턴스 소멸 메시지 출력

```
del obj_2
```

```
myClass_2 인스턴스가 메모리에서 소멸되었습니다
```

VII. Class 상속

1) Class 선언

- Class Sum, Cop 선언 후 Computer Class에 전달
- Class Computer는 상속 받은 add( ), mul( ) Method 사용 가능

```
class Sum:
    def add(self, x, y):
        z = x + y
        return z

class Cop:
    def mul(self, x, y):
        z = x * y
        return z
```

```
class Computer(Sum, Cop):
    def sub(self, x, y):
        z = x - y
        return z
```

2) Instance 생성

```
obj_3 = Computer()
```

3) 상속 받은 Method 사용

- Computer Class의 Method

```
obj_3.sub(3, 5)
```

```
-2
```

- Sum Class로 부터 상속받은 add( ) Method

```
obj_3.add(3, 5)
```

```
8
```

- Cop Class로 부터 상속받은 Mul( ) Method

```
obj_3.mul(3, 5)
```

```
15
```

#

#

#

The End

#

#

#