Function and Module

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

I. def

- 함수(Function): 지정된 입력값을 받아 처리 후 출력값 리턴
  - 입력값 -> 함수(Function) -> 출력값

1) 사용자 정의 함수 with def

- addition( ) 정의 및 사용

```
def addition(m, n):
  k = m + n
  return k
```

```
addition(m = 3, n = 6)
```

```
9
```

2) 사용자 정의 함수 with lambda

- subtraction( ) 정의 및 사용

```
# def subtraction(m, n):
#   k = m - n
#   return k
```

- lambda: 한 줄로 함수 정의

```
subtraction = lambda m, n : m - n
```

3) 매개변수 위치

- m - n

```
subtraction(m = 3, n = 6)
```

```
-3
```

- m - n

```
subtraction(n = 6, m = 3)
```

```
-3
```

- m = 3, n = 6

```
subtraction(3, 6)
```

```
-3
```

- m = 6, n = 3

```
subtraction(6, 3)
```

```
3
```

4) 입력값 개수 미지정

- '*' 기호가 입력값을 모아서 Tuple로 생성

```
def add_all(*inputs):
    result = 0

    for i in inputs:
        result = result + i
    return result
```

```
add_all(1, 3, 5, 7, 9)
```

```
25
```

5) 매개변수 초깃값 설정

- operator = 'addition'

```
def add_mul(*inputs, operator = 'addition'):
    if operator == 'addition':
        result = 0
        for i in inputs:
            result = result + i
    elif operator == 'multiplication':
```

```
        result = 1
        for i in inputs:
            result = result * i
    return result
```

- 마지막 매개변수 생략

```
add_mul(1, 3, 5, 7, 9)
```

```
25
```

- 초깃값이 아닌경우 매개변수 지정

```
add_mul(1, 3, 5, 7, 9, operator = 'multiplication')
```

```
945
```

- 초깃값 설정 - m = 3, n = 6

```
def addition(m = 3, n = 6):
  k = m + n
  return k
```

- 매개변수 없이 실행

```
addition()
```

```
9
```

- 매개변수 지정 실행

```
addition(m = 7, n = 9)
```

```
16
```

II. Module

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

1) 내장 모듈

- 사용구문: import Module_Name

```
import time
```

- 사용구문: Module_Name.Function_Name( )

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

```
스크립트 시작
시작 9초 후 스크립트 종료
```

- Built-in Module Path

```
import sys

sys.path
```

```
['/content',
 '/env/python',
 '/usr/lib/python39.zip',
 '/usr/lib/python3.9',
 '/usr/lib/python3.9/lib-dynload',
 '',
 '/usr/local/lib/python3.9/dist-packages',
 '/usr/lib/python3/dist-packages',
 '/usr/local/lib/python3.9/dist-packages/IPython/extensions',
 '/root/.ipython']
```

2) 내장 모듈 with Alias (import ~ as)

- 사용구문: import Module_Name as Alias

```
import time as t
```

- 사용구문 : Alias.Function_Name( )

```
print('5초 후에 결과 출력')
t.sleep(5)
print('축하합니다. 합격입니다!')
```

```
5초 후에 결과 출력
축하합니다. 합격입니다!
```

3) 사용자 모듈 만들기

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

(1) Module import

- 사용구문: import Module_Name

```
import myModule
```

▼ (2) hi( )

- 사용구문: Module_Name.Function_Name( )

```
myModule.hi()
```

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

▼ (3) addition( )

```
myModule.addition(3, 6)
```

```
9
```

▼ (4) subtraction( )

```
myModule.subtraction(3, 6)
```

```
-3
```

▼ (5) add_all( )

```
myModule.add_all(1, 3, 5, 7, 9)
```

```
25
```

▼ 4) 사용자 모듈 with from ~ import

- Error

```
hi()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-0c415d115f7e> in <module>
----> 1 hi()

NameError: name 'hi' is not defined

SEARCH STACK OVERFLOW
```

- 사용구문: from Module_Name import Function_Name

```
from myModule import hi
```

- 사용구문: Function_Name( )
  - 모듈명 없이 함수명으로 호출

```
hi()
```

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

▼ 5) 사용자 모듈 with Alias (import ~ as)

- 사용구문: import Module_Name as Alias

```
import myModule as m
```

▼ (1) hi( )

- 사용구문 : Alias.Function_Name( )

```
m.hi()
```

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

▼ (2) addition( )

```
m.addition(5, 7)
```

```
12
```

#

#

#

The End

#

#

#