

# 함수 및 모듈 기본

## 학습 내용 및 목표

- 함수에 대해 이해하고, 함수를 실습을 통해 알아봅니다.
- 함수의 다양한 형태에 대해 알아봅니다.
- 모듈에 대해 알아보고 직접 작성해 봅니다.
- 많이 사용되는 외부 모듈을 불러와서 직접 사용해 봅니다.

## 1-1 함수에 대해 이해해보기

- 어떤 일정한 실행을 반복할 때, 이를 함수로 만들어 효율적 관리유지가 가능하다.

### input().split

- 생각해보기 - 두 값에 입력받고 3번의 더하기, 빼기, 곱하기 연산을 수행해 보기

In [5]:

```
a, b = input().split() # 입력시에 공백을 두어 입력을 수행
print(a, b)
```

3 7  
3 7

In [9]:

```
a, b = input().split(",") # 입력시에 ','을 기준으로 두 개의 값으로 구분
print(a, b)
```

3,7  
3 7

In [10]:

```
a = int(a)
b = int(b)

print(a+b)
print(a-b)
print(a*b)
```

10  
-4  
21

## 두개의 값 입력 후, 연산 실행

In [11]:

```
a, b = input().split(",") # 입력받은 값을 공백을 기준으로 분리

a = int(a)
b = int(b)

print(a+b)
print(a-b)
print(a*b)
```

```
3, 7
10
-4
21
```

## 같은 행동이 반복적으로 수행될때,

- 이를 함수화 시켜보자.

In [6]:

```
def two_plus(a,b):
    print(a + b)
    print(a - b)
    print(a * b)
```

In [7]:

```
a, b = input().split(",") # 입력받은 값을 공백을 기준으로 분리

a = int(a)
b = int(b)
two_plus(a,b)
```

```
4,3
7
1
12
```

## [정리]

함수화를 통해 코드의 반복을 줄일 수 있고, 추후 공동작업시에도 수정 및 기능 추가가 효율적이다.

## 기본 구조1

```
def 함수명():
    실행문장1
    실행문장1
```

```
# 실행을 위해서는 함수명 재호출해야함.
함수명()
```

## 기본 구조2

- 함수를 호출시에 값을 전달할 수 있다.
- 정의하는 곳에서는 이를 매개변수
- 넘겨주는 쪽에서는 이를 인수라한다.

```
def 함수명(매개변수1, 매개변수2):  
    실행문장1  
    실행문장1
```

```
# 실행을 위해서는 함수명 재호출해야함.  
함수명(인자1, 인자2)
```

In [12]:

```
def two_plus(a,b): # a,b 는 매개변수  
    print(a + b)  
    print(a - b)  
    print(a * b)
```

In [13]:

```
a1 = 3  
b1 = 5  
two_plus(a1,b1) # a1, b1는 인수
```

```
8  
-2  
15
```

## 함께 해보기

- 사칙 연산을 계산하는 함수를 작성해 보자.
- +, -, \*, /, //, % 등을 계산해보기

## 생각해 보기

- 좀 더 나의 계산기를 업그레이드 시켜보자.

추가 - eval() 함수를 이용하여 연산 가능

In [17]:

```
# 참고 예
print( eval("3+5") )
print( eval("3-5-10"))
print( eval("3*5*10"))
```

```
8
-12
150
```

## 기본 구조3

- 함수 정의 블록에서 return을 명령을 이용해서 함수 호출한 곳에 실행 후, 값을 전달할 수 있다.

```
def 함수명(매개변수1, 매개변수2):
    실행문장1
    실행문장1
    return [전달내용]
```

```
# 실행을 위해서는 함수명 재호출해야함.
var = 함수명(인자1, 인자2)
```

In [18]:

```
def two_plus(a,b):
    num1 = a+b
    return num1
```

In [19]:

```
a = 3
b = 5
result = two_plus(a,b)
print(result)
```

```
8
```

## 여러개의 값을 전달하기

In [20]:

```
def two_op(a,b):
    num1 = a+b
    num2 = a-b
    num3 = a*b
    num4 = a/b
    return num1, num2, num3, num4
```

In [21]:

```
a = 3
b = 5
r1, r2, r3, r4 = two_op(a,b)
print("더하기 :", r1)
print("빼기   :", r2)
print("곱하기 :", r3)
print("나누기 :", r4)
```

```
더하기 : 8
빼기   : -2
곱하기 : 15
나누기 : 0.6
```

## 기본 구조4

- 전달되는 입력값이 몇 개인지 모를때는 다음과 같이 여러개의 값을 전달 가능하다.
- 입력되는 값의 개수가 때에 따라 다를 경우

```
def 함수명(*매개변수):
    실행문장1
```

```
# 실행을 위해서는 함수명 재호출해야함.
var = 함수명(인자1, 인자2)
```

In [22]:

```
def mul_op(*args):
    sum = 0
    for i in args:
        sum += i
    return sum
```

In [23]:

```
mul_op(1,2,3,4,5)
```

Out[23]:

```
15
```

In [24]:

```
mul_op(3,4,5)
```

Out[24]:

```
12
```

## 기본 구조5

- 매개변수의 초기값 설정이 가능하다.
- 함수를 호출시에 해당 값이 전달이 되지 않으면 초기값으로 값을 가지게 된다.

```
def 함수명(매개변수1, 매개변수2=0):
```

```
    실행문장1
```

```
# 실행을 위해서는 함수명 재호출해야함.
```

```
var = 함수명(인자1, 인자2)
```

In [25]:

```
def two_op2(a,b=0):
    num1 = a+b
    num2 = a-b
    return num1, num2
```

In [26]:

```
a = 3
b = 5
r1, r2 = two_op2(a,b)
print("더하기 :", r1)
print("빼기   :", r2)
```

```
더하기 : 8
```

```
빼기   : -2
```

## 함수의 매개변수 값, b가 없을 경우,

In [27]:

```
a = 3
r1, r2 = two_op2(a) # 원래 인수가 없을 때는 함수에서 정의된 초기값을 쓴다.
print("더하기 :", r1) # 3+0
print("빼기   :", r2) # 3-0
```

```
더하기 : 3
```

```
빼기   : 3
```

## 함께 해보기

- 함수 실습해보기

In [ ]:

## 1-2 모듈(module)과 import에 대해 알아보자.

- 모듈이란 함수나 변수 또는 클래스를 모아놓은 파일입니다.
- 다른 파이썬 프로그램에서 불러와 사용할 수 있도록 만든 파이썬 파일로 보면 됩니다.

## 내가 직접 모듈을 만들어 사용해 보기

파일 만들기 mymod.py

```
def two_op2(a,b=0):  
    num1 = a+b  
    num2 = a-b  
    return num1, num2
```

## 모듈을 불러오기 (import 명령을 이용)

[syntax] import 모듈이름

In [31]:

```
import mymod
```

In [32]:

```
a = 5  
b = 10  
two_op2(a, b)
```

Out[32]:

(15, -5)

## 외부 모듈 사용해 보기

- os 모듈 : 시스템의 디렉터리, 파일, os 자원을 제어할 수 있게 해주는 모듈
- shutil 파일 복사 모듈
- time : 시간관련 모듈
- random : 난수(규칙이 없는 임의의 수)를 발생시키는 모듈

### 2-1 외부 모듈(라이브러리) 사용해 보기 - os

In [33]:

```
import os

# 현재 위치 확인
print( os.getcwd() )
```

D:\Github\WCLASS\_PYTHON\_START\_Code

In [34]:

```
# 현재 위치의 디렉터리 및 파일 확인
print( os.listdir() )
```

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_
python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_star
t_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ip
ynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mymod.py', 'pdf_h
tml', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.pptx', 'REA
DME.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_programmin
g', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데이터분석
을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

In [35]:

```
# 현재 위치에 새로운 디렉터리 만들기
print( os.mkdir("mydir") )
print( os.listdir() )
```

None

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_
python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_star
t_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ip
ynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mydir', 'mymod.p
y', 'pdf_html', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.p
ptx', 'README.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_pr
ogramming', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데
이터분석을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

In [36]:

```
# 현재 위치에 디렉터리 삭제
print( os.rmdir("mydir") )
print( os.listdir() )
```

None

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_
python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_star
t_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ip
ynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mymod.py', 'pdf_h
tml', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.pptx', 'REA
DME.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_programmin
g', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데이터분석
을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

## ### 2-2 외부 모듈(라이브러리) 사용해 보기 - shutil

\* 파일을 복사해 주는 파이썬 모듈



In [38]:

```
print( os.listdir() )
```

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_start_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ipynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mymod.py', 'pdf_html', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.pptx', 'README.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_programming', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

In [39]:

```
import shutil
```

```
shutil.copy("mydata.txt", "mydata_bak.txt")  
print( os.listdir() )
```

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_start_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ipynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mydata_bak.txt', 'mymod.py', 'pdf_html', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.pptx', 'README.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_programming', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

In [40]:

```
shutil.copy("mymod.py", "mymod1.py")  
print( os.listdir() )
```

```
['.git', '.gitattributes', '.ipynb_checkpoints', '01_01_python_start.ipynb', '01_02_python_start_if_for_list.ipynb', '01_03_python_start_file.ipynb', '01_04_python_start_fnc_module.ipynb', '02_01_programming.ipynb', '03_02_ml_start.ipynb', '03_03_DB.ipynb', '03_file실습', 'file01.txt', 'hello_utf.txt', 'mydata.txt', 'mydata_bak.txt', 'mymod.py', 'mymod1.py', 'pdf_html', 'price01.txt', 'Python_Basic_Summary.pptx', 'Python_Basic_v10_1911.pptx', 'README.md', 'unit08_package', 'unit09_addressprogramming', 'unit09_tkinter_programming', '__pycache__', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pdf', '데이터분석을위한파이썬_16h_v01_202202_커리큘럼.pptx']
```

## 2-3 외부 모듈(라이브러리) 사용해 보기 - time

- 시간과 관련된 다양한 기능

In [41]:

```
import time
```

In [42]:

```
# 현재 시간을 실수형태로 표현
time.time()
```

Out[42]:

1649310589.6926239

In [43]:

```
# 현재 시간을 실수형태로 표현된 것을 년월일시분초로 표현
a = time.time()
time.localtime(a)
```

Out[43]:

```
time.struct_time(tm_year=2022, tm_mon=4, tm_mday=7, tm_hour=14, tm_min=49, tm_sec=5
0, tm_wday=3, tm_yday=97, tm_isdst=0)
```

In [44]:

```
# 원하는 포맷으로 출력
# 자세한 포맷은 링크 참조 : https://wikidocs.net/33
loc_time = time.localtime(time.time())
time.strftime("%Y/%m/%d %H/%M/%S", loc_time)
```

Out[44]:

'2022/04/07 14/49/52'

In [45]:

```
### 몇초동안 실행을 지연시킨다.
import time

print("지금부터 5초")
time.sleep(5)
print("5초후 출력")
```

지금부터 5초  
5초후 출력

## 2-4 외부 모듈(라이브러리) 사용해 보기 - random

In [46]:

```
import random
```

In [47]:

```
# 0.0에서 1.0사이의 실수 중에서 난수값을 발생  
random.random()
```

Out[47]:

0.36188454742029896

In [48]:

```
# 1,10 사이의 정수 중에서 난수 발생  
random.randint(1,10)
```

Out[48]:

8

In [49]:

```
# 1,30 사이의 정수 중에서 난수 값 발생  
random.randint(1,30)
```

Out[49]:

26

In [50]:

```
# 가위바위보 중에 하나 난수 발생  
a = ['가위', '바위', '보']  
  
num1 = random.randint(0,2)  
print(a[num1])
```

가위

## 응용 프로그램

- 가위바위보 게임을 만들어보자.

In [51]:

```
# 가위바위보 중에 하나 난수 발생
a = ['가위', '바위', '보']

me = int( input("가위(0)/바위(1)/보(2) 중의 하나를 선택해 주세요(0,1,2) : "))

num1 = random.randint(0,2)
print("컴퓨터 : ",a[num1])
print("선택 : ",a[me])

if a[me]==a[num1]:
    print("비겼습니다.")
elif (a[me]=="가위" and (a[num1]=="바위")):
    print("졌습니다.")
elif (a[me]=="바위" and (a[num1]=="가위")):
    print("이겼어요")
else:
    print("잘못된 값을 입력했어요.")
```

가위(0)/바위(1)/보(2) 중의 하나를 선택해 주세요(0,1,2) : 0  
컴퓨터 : 가위  
선택 : 가위  
비겼습니다.

## 실습

- 위의 가위바위보 프로그램은 일부만 동작하고 일부는 동작하지 않습니다. 보완해 주세요.