

함수와 모듈

함수(Function) 란?

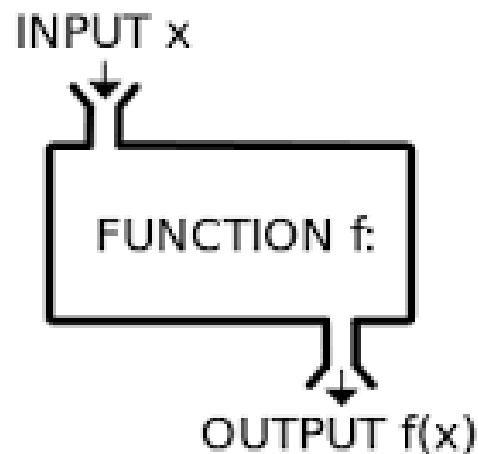
- 사용 이유

- 코드 재사용

- 중복되는 부분을 한 번만 작성. 코드를 간결하게 만들어줌

- 절차적 분해

- 한 번에 큰 처리 절차를 구현하는 것보다 작은 작업을 구현하는 것이 더 쉬움



함수(Function) 란?

- 예)
 - len()이 없었다면?

```
a=[5,5,6,7,8,3]
b='I am a boy.'
# a, b의 길이를 구하고 싶은데 ...

print(len(a))
print(len(b))
```

```
a=[5,5,6,7,8,3]
b='I am a boy.'
# a, b의 길이를 구하고 싶은데 ...

length_a=0
...
print(length_a)
length_b=0
...
print(length_b)
```

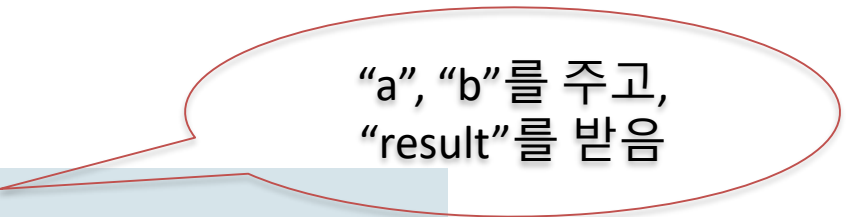
함수(Function) 란?

- 일반적 구조

```
def 함수명(매개변수):  
    수행할 문장  
    ...  
    return 반환값
```

- 예)

```
def add(a, b):  
    result = a + b  
    return result
```



“a”, “b”를 주고,
“result”를 받음

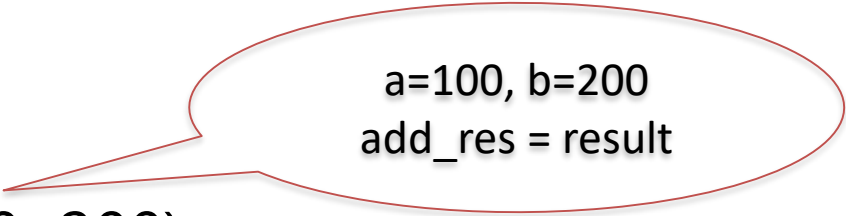
함수(Function) 란?

#함수 정의

```
def add(a, b) :  
    result = a + b  
    return result
```

#함수 호출

```
add_res = add(100, 200)  
print(add_res)
```



a=100, b=200
add_res = result

연습문제 0

- len() 구현

```
def my_len(l):  
    ...  
    return length
```

```
a=[5,5,6,7,8,3]  
b='I am a boy.'  
print(len(a), len(b)) # 내장 함수 len()  
print(my_len(a), my_len(b)) # my_len()
```

함수 – 반환 값 (Return Value)

- 반환 값이 있는 경우

```
def say():  
    return "hello"
```

```
a=say() # a의 값: "hello"
```

- 반환 값이 **없**는 경우

```
def say():  
    print("hello")
```

```
a=say() # a의 값: None
```

None을 반환

함수 – 매개 변수 (Parameter)

- 매개 변수가 있는 경우

```
def add(a, b):  
    result = a + b  
    return result
```

```
a=add(1, 2)  
a=add(2, 3)
```

- 매개 변수가 **없**는 경우

```
def add():  
    return 1+2
```

```
a=add()
```


연습문제 1

- add, sub, mul, div 함수 정의해보기
 - add: 더하기, sub: 빼기, mul: 곱하기, div: 나누기

```
a=1  
b=2
```

```
c=add(a, b)  
d=sub(a, b)  
e=mul(a, b)  
f=div(a, b)
```

```
print(c, d, e, f)
```

```
3 -1 2 0
```

가변 매개변수

- 입력 값(매개변수)이 몇 개인지 모를 경우

```
def 함수이름 (*매개변수명):  
    수행할 문장  
    ...
```

- 예) 짝수 개수 구하기

```
def count_even(*n):  
    cnt = 0  
    for v in n:  
        if v%2 == 0:  
            cnt += 1  
    return cnt
```

```
def f1(a, *b): # 정상  
  
def f2(a, *b, c): # 에러  
  
def f3(*a, *b): # 에러  
  
def f4(a, b, *c):
```

가변 매개변수

- 리스트, 튜플로 대체 가능

```
def count_even_vv(*n):  
    cnt = 0  
    for v in n:  
        if v%2 == 0:  
            cnt += 1  
    return cnt
```

```
def count_even_list(n):  
    cnt = 0  
    for v in n:  
        if v%2 == 0:  
            cnt += 1  
    return cnt
```

```
cnt_even=count_even_vv(1,2,3,4)  
cnt_even2=count_even_list([1,2,3,4])  
  
print(cnt_even)  
print(cnt_even2)
```

지역변수, 전역변수??

- 지역변수 (Local Variable)

- 함수 안에서 만들어져 함수 안에서만 사용하는 변수

```
def add_value(a, b):  
    add_res = a+b  
    return add_res  
print(add_res) # 에러 발생
```

- 전역변수 (Global Variable)

- 함수 밖에서 만들어져 아무데서나 사용할 수 있는 변수

```
x = 1  
def addX(_x):  
    return x+_x  
a = 50  
a1 = addX(a)  
print(a1) # 51
```

지역변수, 전역변수??

- 함수 안에서 전역변수를 선언할 수도 있음

```
def add(a, b) :  
    global add_res # 전역변수로 선언  
    add_res= a+b  
  
add(1,2)  
print(add_res)    # 3
```

연습문제 2

- 양의 정수 n 을 입력받아 1부터 n 까지 곱해주는 factorial 함수를 작성하시오

```
def factorial(n):
```

```
...
```

```
...
```

```
>> factorial(3)
```

```
6
```

```
>> factorial(5)
```

```
120
```

```
>> factorial(8)
```

```
40320
```

```
>> factorial(1)
```

```
1
```

연습문제 3

- n 과 a 를 입력받아 n^a 를 계산해주는 거듭제곱 함수를 작성하시오

```
def pow(n, a):  
    ...  
    ...
```

```
>> pow(5,3)  
125  
>> pow(4,2)  
16  
>> pow(2,4)  
16
```

디폴트 매개 변수

- 매개 변수의 기본값을 지정해줄 수 있다
 - 호출 시, 해당 매개 변수를 주지 않는다면, 정의할 때 지정한 기본값을 가짐

```
def printName(first, second='Lee'): #second 변수 기본값 'Lee'  
    print('My name is ', first, ' ', second + '.')
```

#함수 호출

```
printName('Chulsoo', 'Kim')  
printName('Chulsoo') # second에 대응되는 값을 주지 않았으므로,  
                    # 정의할 때 지정해줬던 'Lee'를 가짐
```


디폴트 매개변수

- 앞 매개변수가 디폴트 값을 가질 때, 뒤에 나타나는 매개변수는 반드시 디폴트 값을 가져야 함

```
def printName(first, second='Kim'): # 정상
```

```
def printName(first='Kim', second): # 에러 발생
```

```
def printName(first, second, third='Kim'): # 정상
```

```
def printName(first, second='Kim', third='M'): # 정상
```

키워드 매개변수

- 호출할 때, 해당 매개변수가 무엇인지 명시적으로 알려줌

```
def calc(x, y, z) :  
    return x+y+z
```

```
result = calc(y=20, x=10, z=30) #매개변수 순서 상관 無  
print(result)
```

```
def calc(x, y=0, z=0) :  
    return x+y+z
```

```
>> calc(y=20, x=10) # 정상  
>> calc(10, y=30, z=20) #정상  
>> calc(10, 30, y=20) # 에러 발생
```

반환 값이 여러 개인 함수

- 튜플이나 리스트를 이용해 여러 개의 값을 리턴할 수 있음

```
a = 3
b = 4
# 함수 정의
def add_and_mul(a, b):
    return (a+b, a*b)

# 함수 호출
add_res, mul_res = add_and_mul(a,b)
print(add_res)
print(mul_res)
```

연습문제 4

- 초를 넘겨주면 시간, 분, 초를 리턴해주는 함수를 작성하시오.
- 예) 57894초를 주면, 16시간 4분 54초를 리턴

```
def hour_min_sec(second):  
    ...  
    ...  
  
hour, min, sec = hour_min_sec(57894)  
print("%d시간 %d분 %d초" % (hour, min, sec))
```

16시간 4분 54초

매개변수 종류

- Mutable 타입의 매개변수:
 - Call by reference 방식과 동일

```
t = [1, 2, 3]
```

```
def test(a):  
    a += [4, 5, 6]
```

```
test(t)  
print(t)
```

```
t = {1:'a'}
```

```
def test(a):  
    a[2] = 'b'
```

```
test(t)  
print(t)
```

- Mutable
 - 리스트, 딕셔너리 등

매개변수 종류

- Immutable 타입의 매개변수:
 - Call by value 방식과 동일

```
t = (1, 2, 3)
```

```
def test(a):  
    a += (4, 5, 6)
```

```
test(t)  
print(t)
```

```
t = 1
```

```
def test(a):  
    a += 1
```

```
test(t)  
print(t)
```

- Immutable
 - 숫자, 문자열, 튜플 등

연습문제 5

- 숫자로 구성된 리스트를 입력받아 최대값, 최소값을 리턴하고, 두 수를 리스트에서 제거하는 함수를 작성하시오.

```
def get_min_max(l):  
    ...  
    ...  
  
l = [3, 5, 9, 1, 2]  
(min_val, max_val) = get_min_max(l)  
  
print(min_val)  
print(max_val)  
print(l)
```

```
1  
9  
[3, 5, 2]
```

연습문제 6

- Dot product (스칼라곱, 내적)을 계산하는 `sparseVectorDotProduct(v1, v2)` 함수 작성

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

- 테스트 케이스

- `v1={'a':5}, v2={'a':3, 'b':2}`
 - 15
- `v1={'c':5}, v2={'a':2, 'b':1}`
 - 0
- `v1={'a':5, 'b':4}, v2={'a':-1, 'b':2}`
 - 3

난수(Random Number) 생성

- `import random`

- 리스트에서 무작위로 선택

```
>>> random.choice('abcdefg')
```

- a 이상 b 이하의 임의의 정수 뽑기 (a, b: int 타입)

- `>>> random.randint(a, b)` `#a~b 정수 값 중 하나;`
- `>>> random.randrange(a, b+1)` `#randint(a, b)와 완전 동일한 기능`

난수(Random Number) 생성

- 리스트 내용 무작위 섞기

```
>>> random.shuffle([1, 2, 3, 4, 5])
```

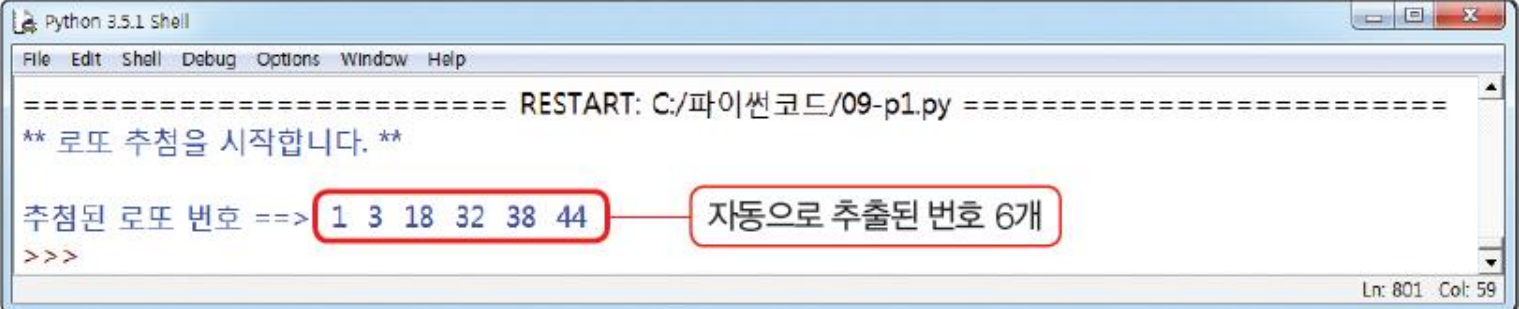
- 리스트에서 정해진 개수만큼 무작위 뽑기

```
>>> random.sample([1, 2, 3, 4, 5, 6], 3) # 3개 샘플링
```

연습문제 7

- 로또 추첨 프로그램
 - 1~45 번호 중 6개 임의의 선택 후 출력

그림 9-1
로또 복권 번호
추첨 프로그램



A screenshot of a Python 3.5.1 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following content: a separator line of equals signs, the text 'RESTART: C:/파이썬코드/09-p1.py', another separator line, the text '** 로또 추첨을 시작합니다. **', and the text '추첨된 로또 번호 ==>'. Below this, the numbers '1 3 18 32 38 44' are displayed and enclosed in a red rounded rectangle. To the right of this rectangle, a red line connects to another red rounded rectangle containing the text '자동으로 추출된 번호 6개'. Below the numbers is the text '>>>'. The status bar at the bottom right shows 'Ln: 801 Col: 59'.

```
Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/파이썬코드/09-p1.py =====
** 로또 추첨을 시작합니다. **
추첨된 로또 번호 ==> 1 3 18 32 38 44 — 자동으로 추출된 번호 6개
>>>
Ln: 801 Col: 59
```

Time 모듈

- `import time`
- 1970년 1월 1일 0시 이후 누적된 초를 리턴
`>>> time.time()`
- 현재 한국 시각 리턴
`>>> time.localtime()`
- 시각을 원하는 문자열 표현으로 리턴
`>>> time.strftime('date: %dth', t)`

연습문제 8

1. 문장을 입력받으면 해당 문장에서 각 알파벳이 몇 개씩 나오는지 저장하는 딕셔너리를 반환하는 함수를 작성하시오
2. 1.의 함수를 통해 생성된 딕셔너리를 입력받으면, 가장 많이 등장한 알파벳을 리턴하는 함수를 작성하시오. (여러 개일 경우 하나만 리턴해도 무관)

```
def letter_dict(str):  
    ....  
    ....  
  
def max_letter(dict):  
    ....  
    ....  
  
a = letter_dict('red apple')  
print(a)  
print(max_letter(a))
```

```
{'r':1, 'e':2, 'd':1, ' ':1, 'a':1,  
'p':2, 'l':1} # 순서 다를 수 있음  
  
e
```

연습문제 9

1. 'yellow banana'를 letter_dict()에 입력하여 새로운 딕셔너리 b를 리턴 받는다
2. 두 딕셔너리 a, b를 입력하면 각 딕셔너리에 등장한 값을 모두 더한 새로운 딕셔너리 c를 리턴하는 함수를 작성하시오.

```
def comb_dict (dict1, dict2):  
    ....  
    ....  
  
b = letter_dict('yellow banana')  
c = comb_dict(a, b)  
  
print(c)  
print(max_letter(c))
```

```
{'r':1, 'e':3, 'd':1, ' ':2, 'a':4,  
'p':2, 'l':3, 'y':1, 'o':1, 'w':1  
, 'b':1, 'n':2} # 순서 다를 수 있음  
  
a
```

과제 1 – prime.py

- 소수(Prime Number) 판단하는 함수 정의하기
 - 소수이면, True 반환
 - 소수가 아니면, False 반환

```
def check_prime(num):  
    # TO BE IMPLEMENTED  
    ...  
  
def main():  
    a = 13  
    b = 15  
    if check_prime(a):  
        print(str(a)+'는 소수입니다.')  
    else:  
        print(str(a)+'는 소수가 아닙니다.')  
  
    if check_prime(b):  
        print(str(b)+'는 소수입니다.')  
    else:  
        print(str(b)+'는 소수가 아닙니다.')
```

과제 2 – add_comma.py

- 천 단위마다 쉼표(,) 추가하는 함수 정의하기
 - 1000000(int)이면, '1,000,000'(str) 반환

```
def add_comma(val):  
    # TO BE IMPLEMENTED  
    ...  
  
def main():  
    comma_added_1234 = add_comma(1234)  
    comma_added_12345678 = add_comma(12345678)  
    comma_added_12 = add_comma(12)  
  
    print(comma_added_1234)           # '1,234'  
    print(comma_added_12345678)      # '12,345,678'  
    print(comma_added_12)            # '12'
```


과제 3 – tok.py

- 입력받은 단어열의 n-gram을 구해주는 tokenize() 함수 정의하기

```
def tokenize(trg, N=1):  
    # TO BE IMPLEMENTED
```

```
def main():  
    a="There was a farmer who had a dog ."  
    print(tokenize(a))  
    print(tokenize(a, 2))
```

```
['There', 'was', 'a', 'farmer', 'who', 'had', 'a', 'dog', '.']
```

```
['There was', 'was a', 'a farmer', 'farmer who', 'who had', 'had a', 'a dog', 'dog .']
```

과제 4 – mean_and_var.py

- 여러 벡터를 입력받아, 평균(Mean)과 분산(Variance)을 구하는 함수 정의하기

```
def mean_and_var(*val):  
    # TO BE IMPLEMENTED  
    ...  
  
def main():  
    v1=(0, 1)  
    v2=(0.5, 0.5)  
    v3=(1, 0)  
  
    m, var = mean_and_var(v1, v2, v3)  
    print('평균: ', m, '분산: ', var)
```

부록

- python 000.py 를 실행할 때 main() 호출 방법
 - 내장변수 `__name__`이 `'__main__'`이라는 문자열을 갖는 것을 이용

```
...
```

```
if __name__ == '__main__': # 파일 제일 아래에 작성  
    main()
```