

LESSON

함수(Function)

함수가 없는 경우

```
a,b=10,5
add=a+b
sub=a-b
print(f'합:{add}, 차:{sub}')
```

```
a,b=22,77
add=a+b
sub=a-b
print(f'합:{add}, 차:{sub}')
```

반복되는 코드

함수가 있는 경우

```
def calc(a,b):
    add=a+b
    sub=a-b
    print(f'합:{add}, 차:{sub}')
```

calc(10,5) 함수 호출

calc(22,77) 함수 호출

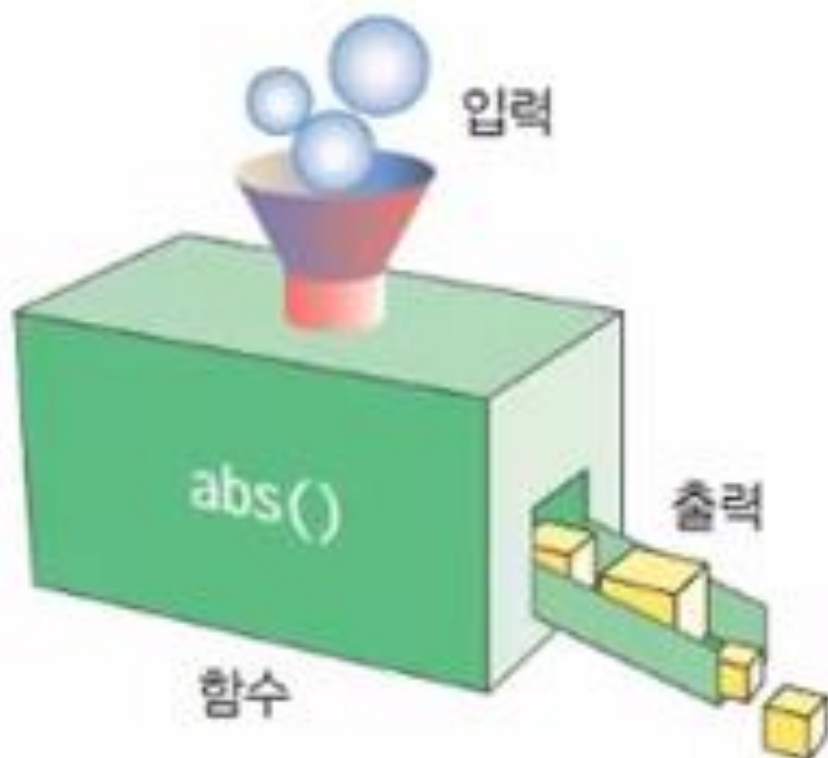


작성된 함수는 몇 번이라도 호출이 가능하다.

- 코드를 묶는 방법
 - 함수(Function) : 반복 사용하는 코드를 묶을 때
 - 클래스(Class) : 서로 관련 있는 변수와 함수를 묶을 때
 - 모듈(Module) : 함수나 클래스 변수 등을 파일로 묶을 때



- 함수(Function)는 특정한 기능을 수행하는 코드의 묶음이다.
- 함수는 입력을 받아서 특정한 작업을 수행하여 결과를 반환하는 블랙박스과 같다.

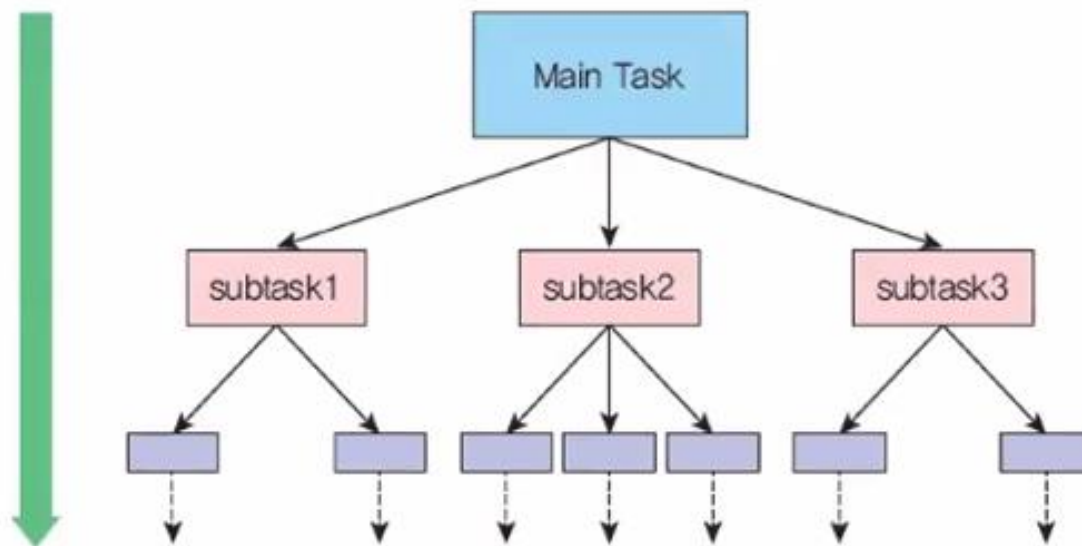


함수는 입력을 받아서 처리한
후에 출력하는 상자과 같습니다.



- 중복된 코드를 제거한다.
- 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다. 각 함수들은 레고의 블록처럼 다른 함수들과 연결되어서 하나의 프로그램을 구성한다.
- 함수는 한번 만들어지면 다른 프로그램에서도 **재사용**될 수 있다.

구조화 프로그래밍



- 내장 함수
 - 파이썬 인터프리터가 기본 제공하는 함수
 - 예를 들면 print, input, len 같은 함수를 내장 함수라고 한다.
- 사용자 정의 함수
 - 개발자가 필요에 의해서 직접 작성한 함수
- 외장 함수
 - 다른 사람이 만들어 제공하는 함수

함수 정의

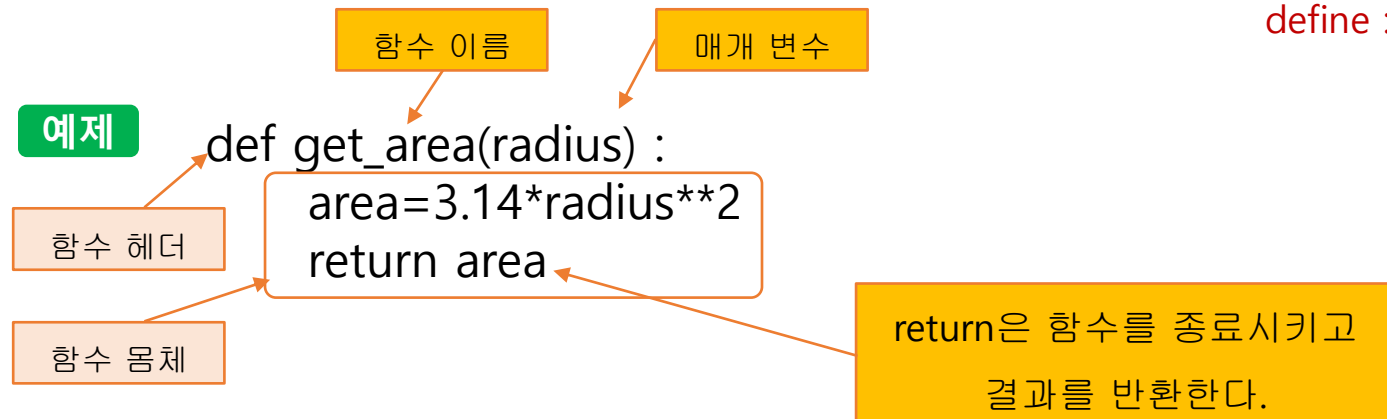
형식

```
def 함수이름(매개변수1, 매개변수2,...) :  
    명령 코드  
    ...  
    return 반환값
```



def는 define
define : 정의하다.

예제

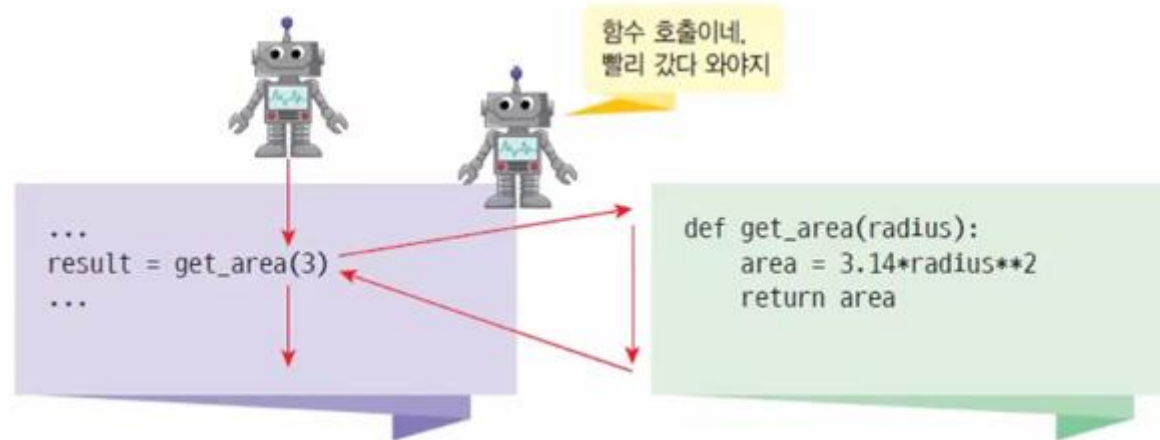


- 함수를 사용하려면 함수를 호출해야 한다.
- 함수가 호출되면 함수 안에 작성한 명령문들이 실행되며 실행이 끝나면 호출한 위치로 되돌아간다.

함수 호출

형식 함수명(데이터1, 데이터2...)

예제



- 정수를 입력받아서 제공한 값을 반환하는 함수를 만들어 보자.

```
def get_square(n):  
    square = n*n  
    return square  
  
result=get_square(4)  
print('4의 제공값=',result)
```

4의 제공값= 16

- 파이썬 인터프리터는 함수가 정의되면 함수 안의 문장들은 실행하지 않는다. 호출되어야 실행이 된다.
- **함수 정의가 먼저 되어야 호출할 수 있다.**
- 단, 함수 안에서 다른 함수를 호출하는 것은 얼마든지 허용된다.

```
result=get_square(4)
print('4의 제곱값=',result)

def get_square(n):
    square = n*n
    return square
```

NameError: name 'get_square' is not defined

```
def main():
    result=get_square(4)
    print('4의 제곱값=',result)

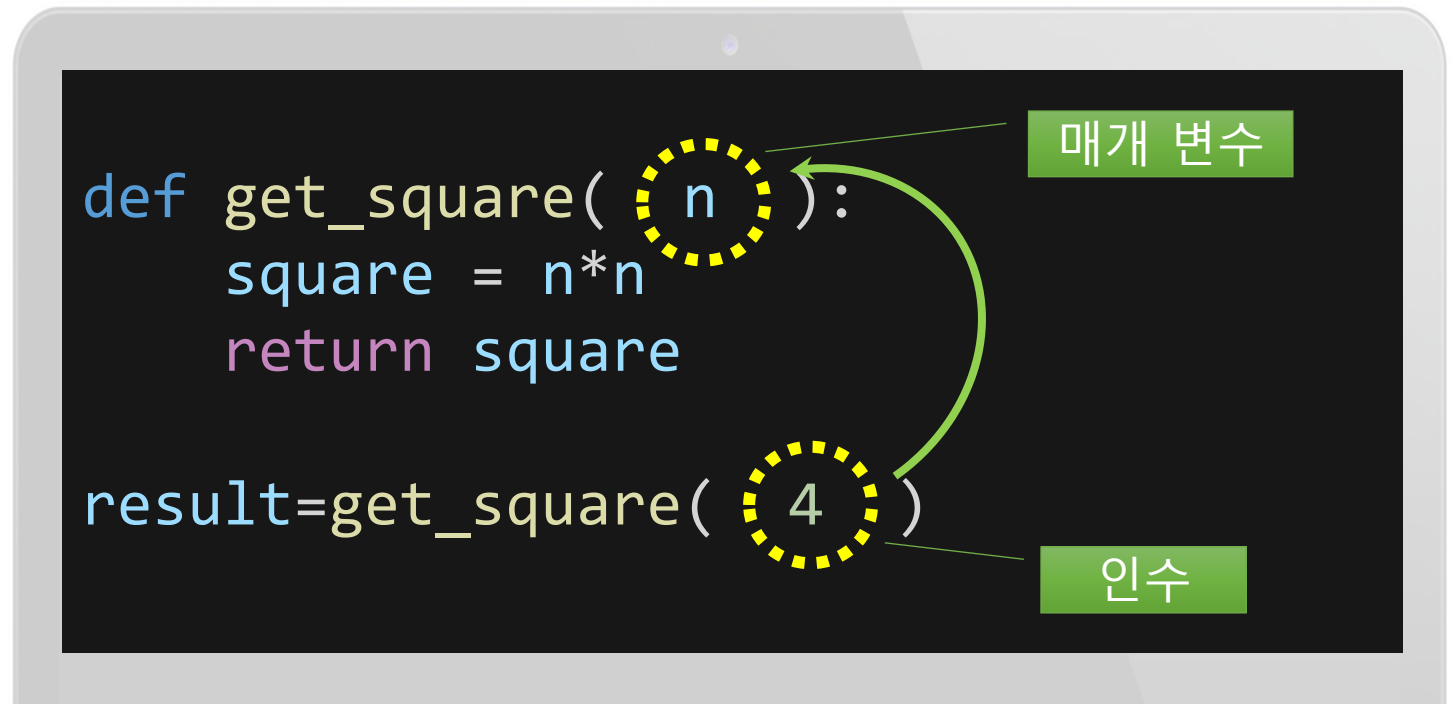
def get_square(n):
    square = n*n
    return square
```

main()

4의 제곱값= 16

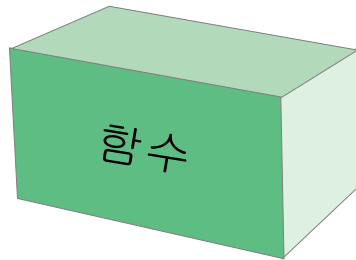
인수(argument) 와 매개 변수(parameter)

- 인수와 매개 변수는 함수 호출시 데이터를 주고받기 위해 필요하다.
- 인수(argument)는 호출 프로그램에 의해 함수에 **전달되는 값**이다.
- 매개 변수(parameter)는 이 값을 **전달받는 변수**이다.



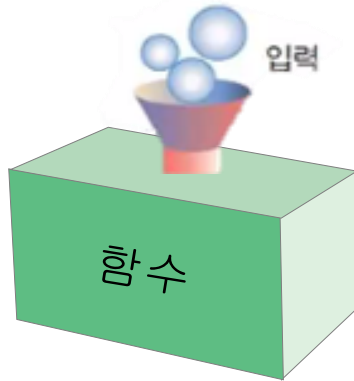
- 반환값은 함수가 호출한 곳으로 반환되는 작업의 결과값이다.
- 값을 반환하려면 **return 문** 다음에 써주면 값이 반환된다.
- 반환값이 없으면 return 문을 생략하면 된다.

인수와 반환값이 없는 함수



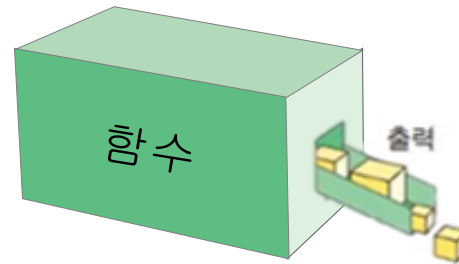
정해진 일을 시키고 싶을 때

인수만 있는 함수



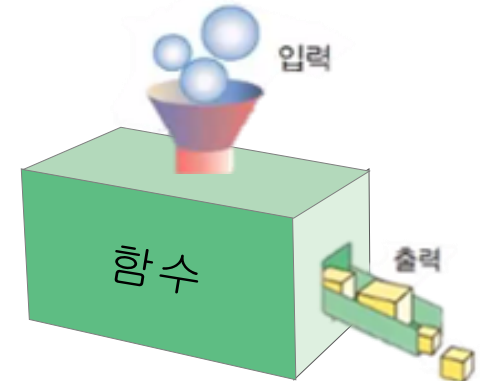
다른 데이터를 전달해
처리를 조정하고 싶을 때

반환값만 있는 함수



결과에 변화가 있으므로
알고 싶을 때

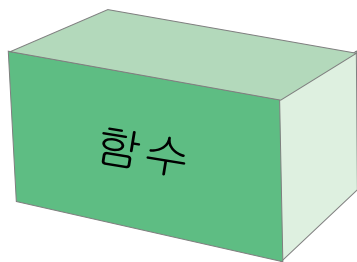
인수와 반환값이 있는 함수



데이터를 전달해 계산하거나
실행 결과를 알고 싶을 때

인수와 반환값이 없는 경우

인수와 반환값이 없는 함수

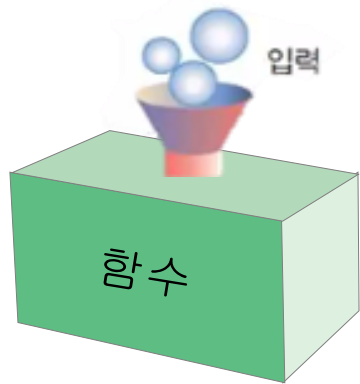


정해진 일을 시키고 싶을 때

```
def say_hello():  
    print('Hello!')  
  
say_hello()
```

Hello!

인수만 있는 함수

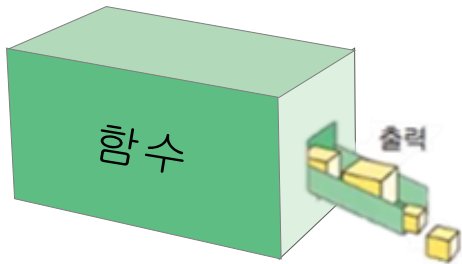


다른 데이터를 전달해
처리를 조정하고 싶을 때

```
def input_check(msg):  
    input(msg)  
  
input_check('첫 번째 입력:')  
input_check('두 번째 입력:')
```

첫 번째 입력:5
두 번째 입력:6

반환값만 있는 함수



결과에 변화가 있으므로
알고 싶을 때

```
import random

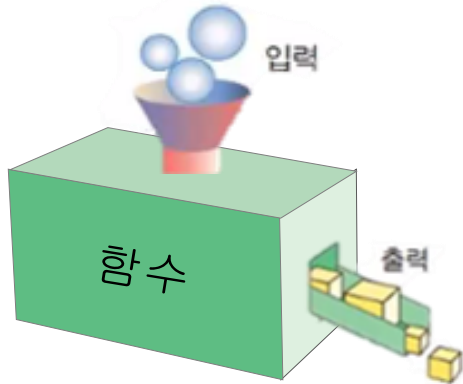
def get_color():
    color=['red','green','blue']
    return random.choice(color)

result=get_color()
print('컬러:',result)
```

컬러: blue

인수와 반환값이 있는 경우

인수와 반환값이 있는 함수



데이터를 전달해 계산하거나
실행 결과를 알고 싶을 때



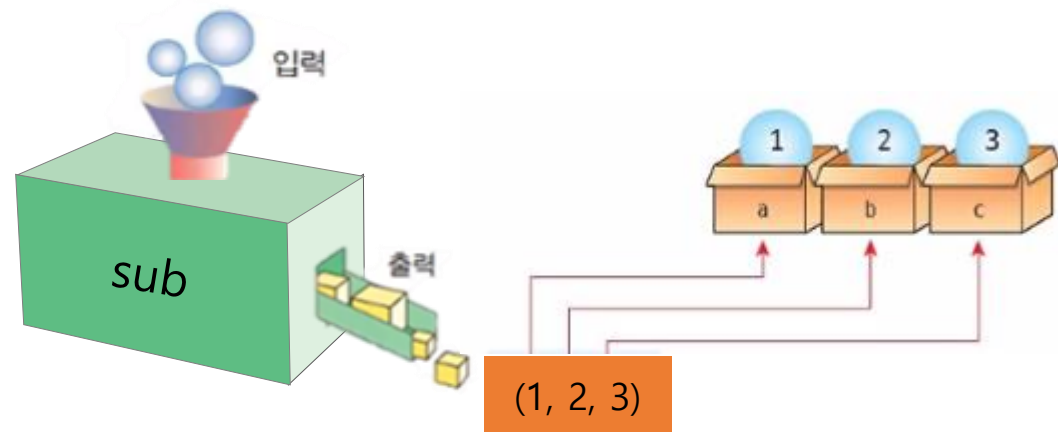
```
def get_sum(num):  
    sum=0  
    for i in range(1,num+1):  
        sum+=i  
  
    return sum  
  
result=get_sum(10)  
if result%2==0:  
    print('짝수')  
else:  
    print('홀수')
```

홀수

- 파이썬에서는 함수가 여러 개 값을 반환할 수 있다.

```
def sub():  
    return 1,2,3  
  
a,b,c=sub()  
print(a,b,c)
```

1,2,3



- 함수로 전달해야 하는 인수의 개수가 정해지지 않은 경우 가변 인수를 사용한다.
- 매개 변수명 앞에 *(에스터리스크)를 붙이면 된다.
- "*"를 붙이면 매개 변수가 튜플 형식으로 넘어온다.

```
def average(*args):  
    result=sum(args)/len(args)  
    print('평균:',result)  
  
average(1)  
average(1,2,3)  
average(1,3,5,7)
```



관행적으로 매개변수명을 **args**로 한다.
변수명을 다른 걸로 해도 된다.

```
평균: 1.0  
평균: 2.0  
평균: 4.0
```

키워드 가변 인수(keyword variable arguments)

- 함수로 전달해야 하는 인수의 개수가 정해지지 않은 경우 가변 인수를 사용한다.
- `**`를 붙이면 매개 변수에 인자값이 딕셔너리 형식으로 넘어온다.

```
def student(**kwargs):  
    print(kwargs)  
  
student(name="James")  
student(name="James", age=22)
```



관행적으로 매개변수명을 **kwargs**로 한다.
변수명을 다른 걸로 해도 된다.

```
{'name': 'James'}  
{'name': 'James', 'age': 22}
```

```
def order_pizza(size, *toppings, **details):  
    print(f"Size: {size}")  
    print(f"Toppings:")  
    for topping in toppings:  
        print(f"- {topping}")  
  
    print('\nDetails: ')  
    for key, value in details.items():  
        print(f"- {key}:{value}")  
  
order_pizza("large", "peperoni", "olives", delivery="Yes", beverage="Coke")  
order_pizza("large", "spinach", delivery="No", time="12:00~12:30")
```

Size: large

Toppings:

- peperoni
- olives

Details:

- delivery:Yes
- beverage:Coke

- 파이썬에서는 함수의 **매개 변수가 기본값을 가질 수 있다**. 이것을 디폴트 매개 변수(default parameters)라고 한다.

```
def greet(name, msg):  
    print(name + ', ' + msg)
```

```
greet('미미', '안녕!')
```



인수와 매개 변수의 개수가 같아야 실행된다.

미미, 안녕

```
def greet(name, msg='반가워!'):  
    print(name + ', ' + msg)
```

```
greet('미미')
```

미미, 반가워!

기본값이 있는 매개 변수의 위치

- 디폴트 값을 가지는 매개 변수는 **뒤에 배치**해야 한다.

```
def greet(msg='반가워!', name):  
    print(name+', '+msg)  
  
greet('미미')
```



SyntaxError: non-default argument
follows default argument

```
def greet(name, msg='반가워!'):  
    print(name+', '+msg)  
  
greet('미미')
```



미미, 반가워



- 함수에 사용하는 매개변수가 많을 때 인수 값의 순서가 틀리면 안된다.
- 키워드를 사용해서 인수를 전달하면 함수 선언시 사용한 매개변수의 순서와 관계없이 인수를 전달할 수 있다.

```
def student(id, name, phone, addr, birth):  
    print(f'학번:{id}, 이름:{name}, 생년월일:{birth}')  
    print(f'연락처:{phone}, 주소:{addr}')  
  
student('1234', 'kim', '02-1234-5678', 'seoul', '2000-10-10')  
  
# 키워드를 이용한 인수의 전달  
student(id='1234', name='kim', birth='2000-10-10', phone='02-1234-5678', addr='seoul')
```

- 파이썬에서 인수들은 함수 호출 시에 위치에 의해 구별된다. 위치 인수(positional argument)
- 키워드 인수(keyword argument)는 인수들 앞에 키워드를 두어서 인수들을 구분한다.

```
def calc(x,y,z):  
    return x+y+z  
  
print(calc(x=1,y=2,z=3))  
print(calc(y=2,z=3,x=1))
```

6
6

```
def calc(x,y,z):  
    return x+y+z  
  
print(calc(1,y=2,z=3))
```

6

```
def calc(x,y,z):  
    return x+y+z  
  
print(calc(x=1,2,3))
```

SyntaxError: positional argument
follows keyword argument



위치 인수와 키워드 인수가 같이 있는 경우
위치 인수가 앞에 나와야 한다.

- Docstring : 프로그램 문서화(설명서), documentation strings.
 - 3중 따옴표(""" """) 안에 적으면 해당 내용은 특수 속성 doc로 변환된다.
 - 모듈, 함수, 클래스, 메소드 정의할 때 첫 부분에 작성하는 문자열이다.
 - Docstring 확인하는 방법-> `__doc__`

```
"""
```

```
파일명: 계산기 만들기
```

```
작성자: 아우라
```

```
작성일: 2022-1-3
```

```
"""
```

```
a=10
```

```
b=20
```

```
c=a+b
```

```
print(__doc__)
```

```
파일명: 계산기 만들기
```

```
작성자: 아우라
```

```
작성일: 2022-1-3
```

```
def test(name,age):  
    """  
        이름과 나이를 입력받아 출력하는 함수이다.  
  
        name  
            이름을 문자로 입력  
        age  
            나이는 숫자로 입력  
    """  
    print('{} , {}'.format(name,age))  
  
help(test)
```

Help on function test in module __main__:

```
test(name, age)  
    이름과 나이를 입력받아 출력하는 함수이다.  
  
    name  
        이름을 문자로 입력  
    age  
        나이는 숫자로 입력
```

- 파이썬은 동적 프로그래밍 언어이므로 변수에 자료형을 지정하지 않아도 자동으로 지정된다.
- 개발을 용이하게 위해 어노테이션을 사용하여 코드의 가독성을 높일 수 있다.

변수의 타입 어노테이션

```
변수명:자료형 = 값  
a=10      ---> a:int=10  
b='hi'     ---> b:str='hi'
```

함수의 타입 어노테이션

```
함수명(매개변수명:자료형,...) -> 자료형:  
def test(name:str,age:int) -> None:  
    """  
        이름과 나이를 입력받아 출력하는 함수이다.  
  
        name  
            이름을 문자로 입력  
        age  
            나이는 숫자로 입력  
    """  
    print('{}, {}'.format(name,age))  
    test()
```

```
def hint_test2(num1:int, num2:int)->tuple:
    '''
        이 함수는 두 수의 사칙연산 결과를 돌려줍니다.

        num1
            숫자 입력
        num2
            숫자 입력
        ...
    '''
    return (num1+num2, num1-num2, num1*num2, num1/num2)

print(hint_test2(5,5))
```

- 함수 내부에서 자기 자신을 호출하는 함수.
- 주의 사항
 - 반드시 종료 조건을 추가해야 한다.
- 재귀 함수 사용하는 이유:
 - 반복문을 사용하지 않아도 되므로 코드가 간단해짐.
 - 변수 사용을 줄일 수 있다.
- 재귀 함수의 단점:
 - 계속된 함수 호출로 메모리를 많이 사용하며 속도가 느림.
 - 재귀 함수는 되도록 사용을 제한하도록 권장함.
- 재귀 함수가 사용되는 곳
 - 이진 탐색, 퀵 정렬, 팩토리얼 연산, 피보나치 수열, ...

재귀 함수(Recursion Function)

```
def recall_func():  
    print("hi~")  
    recall_func()  
  
recall_func()
```



종료 조건이 없는 경우 최대 재귀
깊이(1000)만큼만 수행

hi~

hi~

...

RecursionError: maximum recursion depth exceeded while calling a Python object

```
def fact(n):  
    result=1  
    for i in range(1,n+1):  
        result *=i  
  
    return result  
  
print(fact(3))
```

```
def fact(n):  
    if n==1:return 1    #종료 조건을 반드시 설정  
    return n*fact(n-1)  
  
print(f' {4} != {fact(4)} '))
```

 1!=1
2!=2*1

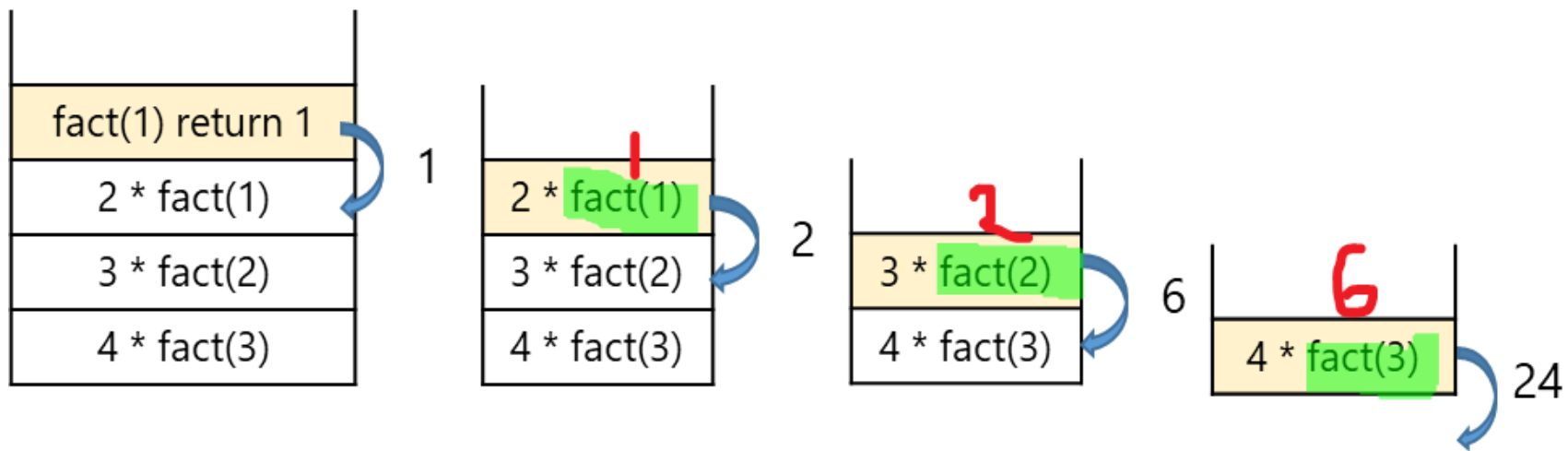
...

$n! = n * (n-1) * (n-2) \dots * 2 * 1$

재귀함수에서 프로그램이 종료되는 조건을 지정하지 않으면 무한루프에 빠지게 된다. n이 1이면 1을 반환하도록 해야 한다.

n!값 계산

네 번째	fact(1) return 1
세 번째	2 * fact(1)
두 번째	3 * fact(2)
첫 번째	4 * fact(3)



0~100 사이의 정수 2개를 입력받아서 더한 값을 출력한다.

첫 번째 숫자 입력:5.5
정수가 아닙니다.
첫 번째 숫자 입력:abc
정수가 아닙니다.
첫 번째 숫자 입력:5
두 번째 숫자 입력:400
0~100사이의 값을
입력하세요.
두 번째 숫자 입력:4
합계: 9

```
def input_check(msg):  
    while True:  
        try:  
            num=int(input(msg))  
            if num<0 or num>100:  
                print('0~100사이의 값을 입력하세요.')  
                continue  
            return num  
        except ValueError:  
            print('정수가 아닙니다.')  
  
num1=input_check('첫 번째 숫자 입력:')  
num2=input_check('두 번째 숫자 입력:')  
print('합계:',num1+num2)
```