

# Contents

## ❖ 목차

- 1. 함수와 인수
- 2. 인수의 형식
- 3. 변수의 범위

# 1. 함수와 인수

## ❖ 함수

- 일련의 코드 블록에 이름을 붙여 정의한 것
- 자주 반복되는 코드 사용이 용이해짐
- 호출문으로 실행
  - 함수(아규먼트들...)

**함수(function)**: 어떤 일을 수행하는 코드의 코드블록  
또는 코드의 묶음

### 함수의 장점

- ① 필요할 때마다 호출 가능
- ② 논리적인 단위로 분할 가능
- ③ 코드의 캡슐화

calcsun

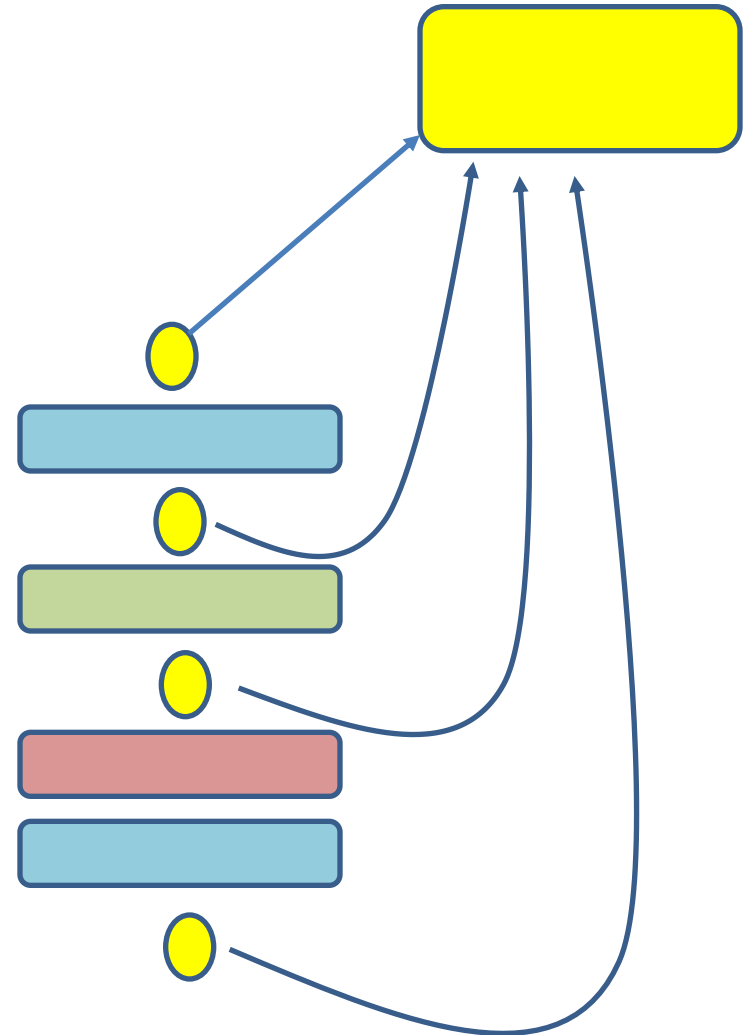
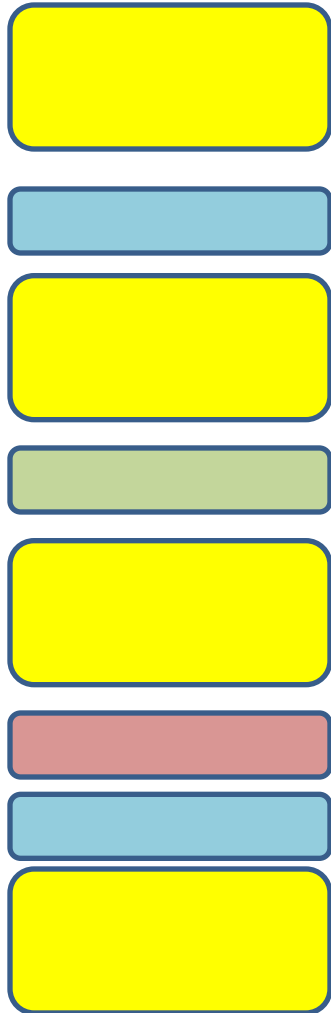
```
def calcsun(n):  
    sum = 0  
    for num in range(n + 1):  
        sum += num  
    return sum
```

```
print("~ 4 =", calcsun(4))  
print("~ 10 =", calcsun(10))
```

실행결과

```
~ 4 = 10  
~ 10 = 55
```

# 1. 함수와 인수



# 1. 함수와 인수

## ❖ 함수의 선언

```
def 함수 이름 (매개변수 #1 ...):  
    수행문 1  
    수행문 2  
    return <반환값>
```

① **def**: 'definition'의 줄임말로, 함수를 정의하여 시작한다는 의미이다.

② **함수 이름**:

함수 이름은 개발자가 마음대로 지정할 수 있지만, 파이썬에서는 다음과 같은 규칙을 사용한다.

- 소문자로 입력한다.
- 띄어쓰기를 할 경우에는 \_ 기호를 사용한다. ex) save\_model
- 행위를 기록하므로 동사와 명사를 함께 사용하는 경우가 많다. ex) find\_number

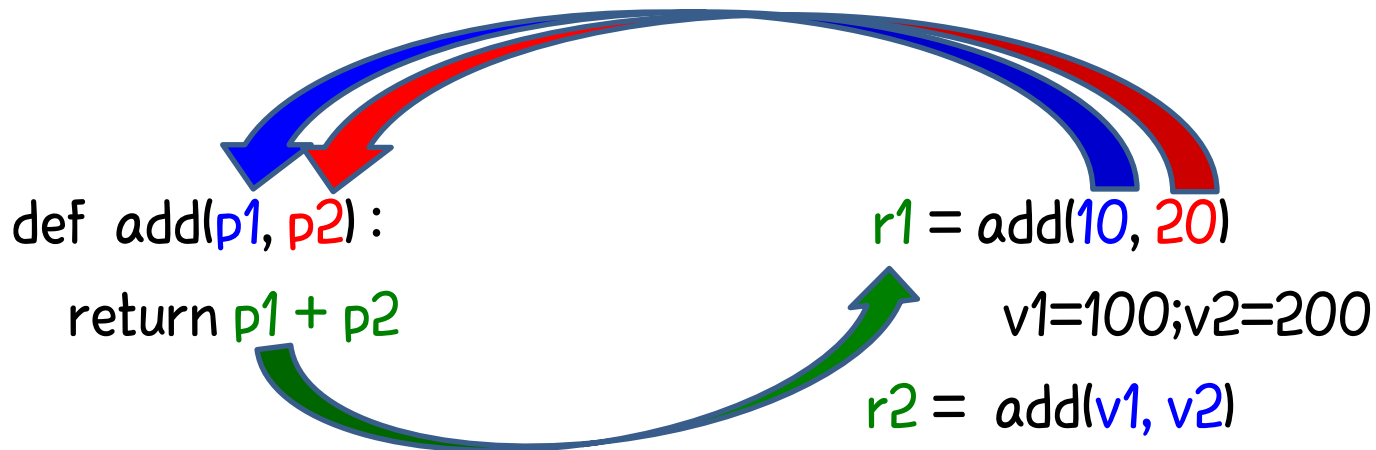
# 1. 함수와 인수

## ❖ 함수의 매개변수의 아규먼트

매개변수 : 함수가 호출될 때 전달받고자 하는 데이터를 저장하는 변수

아규먼트 : 함수를 호출하면서 전달하는 데이터

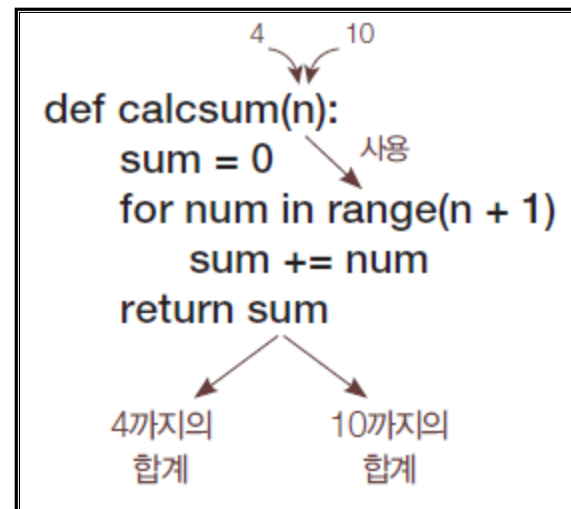
인수



# 1. 함수와 인수

## ❖ 인수(파라미터)

- 호출원에서 함수로 전달되는 작업거리(데이터)
- 함수의 동작에 변화 주어 활용성 높임
- 매개변수
- 형식 인수 = 함수 정의문의 인수
- 실인수 = 함수 호출문에서 전달하는 인수



### calcrange

```
def calcrange(begin, end):  
    sum = 0  
    for num in range(begin, end + 1):  
        sum += num  
    return sum  
  
print("3 ~ 7 =", calcrange(3, 7))
```

실행결과

3 ~ 7 = 25

# 1. 함수와 인수

## ❖ 리턴값

- 함수의 실행 결과를 호출원으로 돌려주는 값
- 리턴값 반환할 때 return 명령 뒤에 반환할 값을 지정
  - 위의 calcsun 함수의 경우

— return sum

- 리턴값이 무조건 있어야 하는 것은 아님

printsum

```
def printsum(n):  
    sum = 0  
    for num in range(n + 1):  
        sum += num  
    print("~", n, "=", sum)
```


```
printsum(4)  
printsum(10)
```

실행결과

```
~ 4 = 10  
~ 10 = 55
```

# 1. 함수와 인수

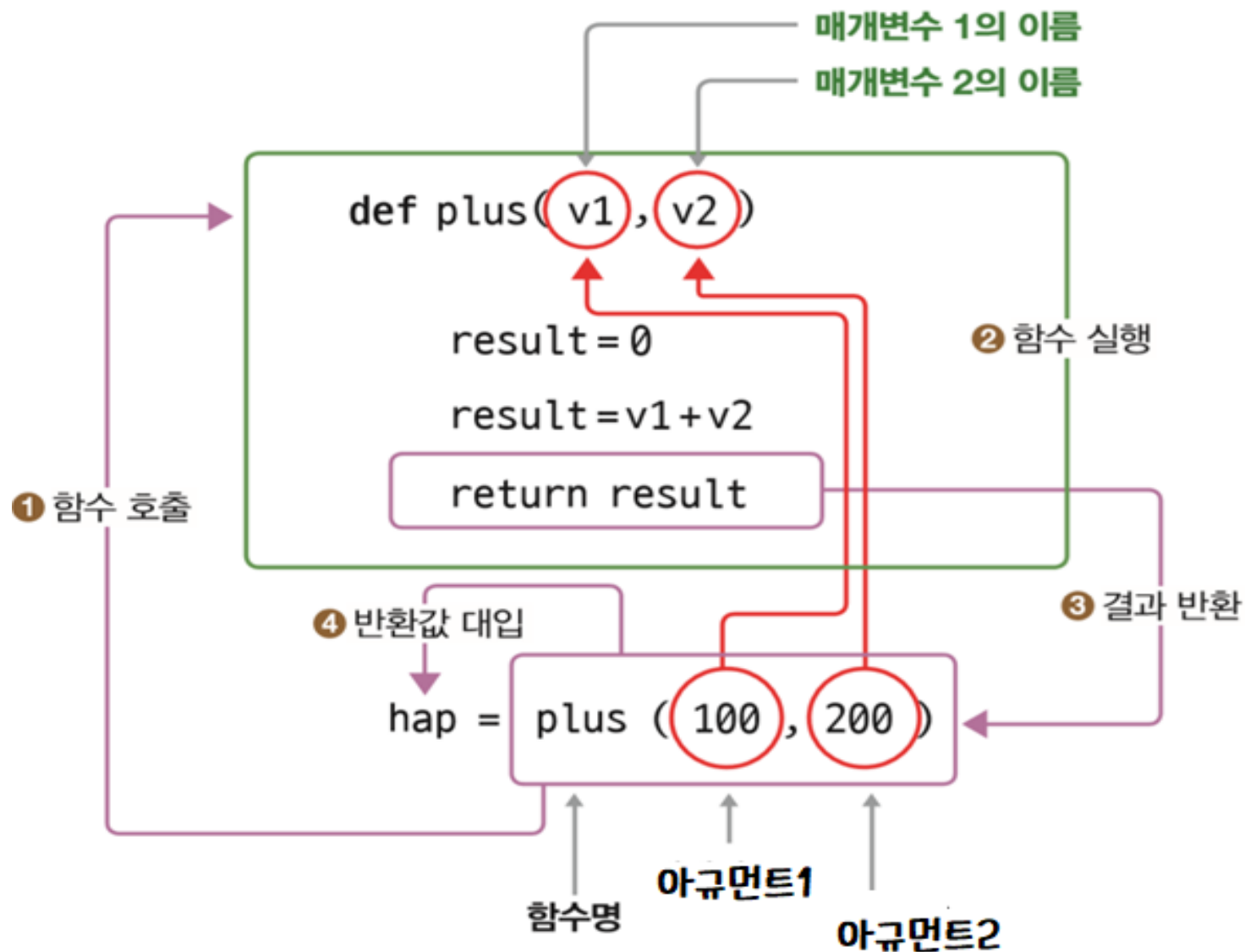
매개변수 유무 반환값 유무	매개변수 없음		매개변수 있음	
	매개변수 없음		매개변수 있음	
반환값 없음	함수 안 수행문만 수행		매개변수를 사용하여 수행문만 수행	
반환값 있음	매개변수 없이 수행문을 수행한 후, 결과값 반환		매개변수를 사용하여 수행문을 수행한 후, 결과값 반환	



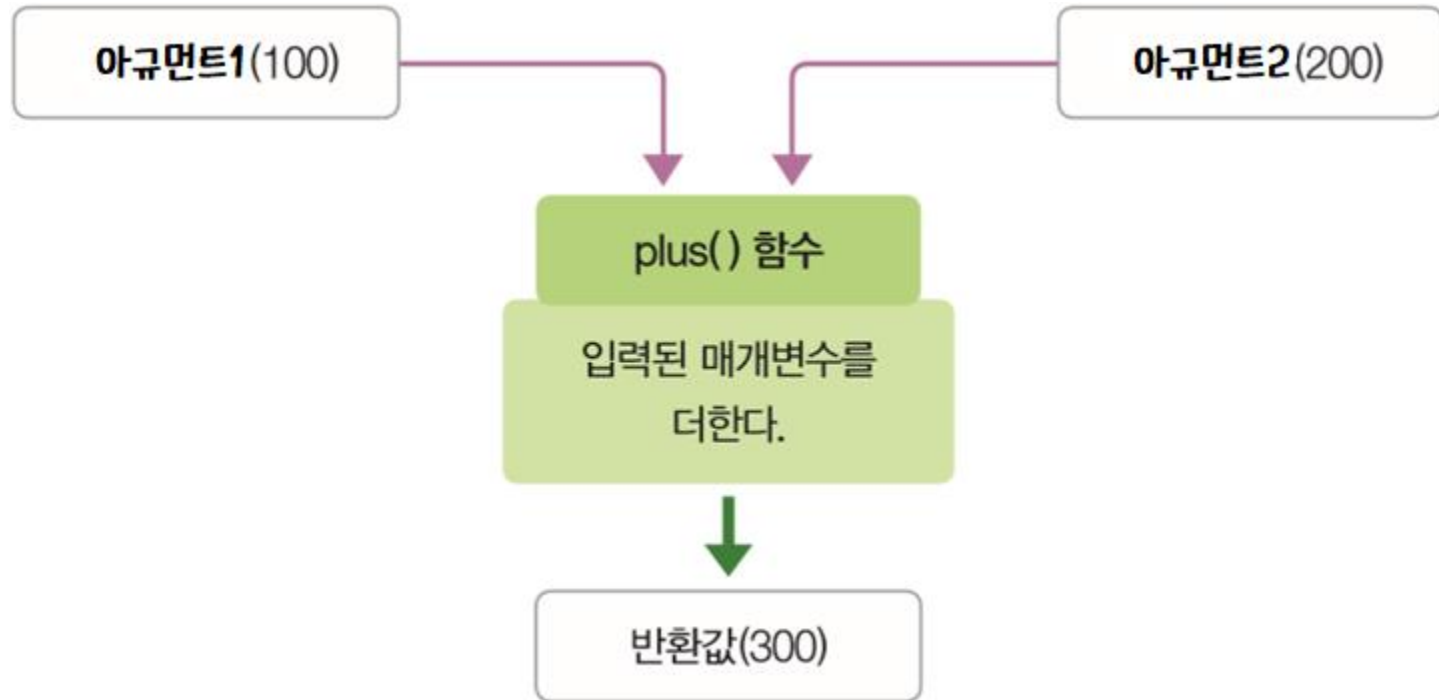
리턴 값이 없는 함수는 리턴 값으로 None이 대신 리턴  
None : 아무 값도 없음을 나타냄



# 1. 함수와 인수



# 1. 함수와 인수



## 2. 인수의 형식

### ❖ 가변 인수

- 고정되지 않은 임의의 개수의 인수를 받음
- \* 기호를 인수 이름 앞에 붙임

vararg

```
def intsum(*ints):  
    sum = 0  
    for num in ints:  
        sum += num  
    return sum  
  
print(intsum(1, 2, 3))  
print(intsum(5, 7, 9, 11, 13))  
print(intsum(8, 9, 6, 2, 9, 7, 5, 8))
```

실행결과

6  
45  
54

## 2. 인수의 형식

- 인수 목록의 마지막에 와야 함

```
intsum(s, *ints)      # 가능
intsum(*ints, s)      # 에러
intsum(*ints, *nums)  # 에러
```

### ❖ 인수의 기본값

- 위 calcrange 함수에 중간값을 인수로 전달할 경우
  - 범위 내 수를 건너뛰어 합계 구함

defaultarg

```
def calcstep(begin, end, step):
    sum = 0
    for num in range(begin, end + 1, step):
        sum += num
    return sum

print("1 ~ 10 =", calcstep(1, 10, 2))
print("2 ~ 10 =", calcstep(2, 10, 2))
```

실행결과

```
1 ~ 10 = 25
2 ~ 10 = 30
```

## 2. 인수의 형식

- 잘 바뀌지 않는 인수는 인수 목록에 기본값 지정
  - 실인수 생략하면 기본값 전달한 것으로 가정

calcstep

```
def calcstep(begin, end, step = 1):  
    sum = 0  
    for num in range(begin, end + 1, step):  
        sum += num  
    return sum  
  
print("1 ~ 10 =", calcstep(1, 10, 2))  
print("1 ~ 100 =", calcstep(1, 100))
```

실행결과

```
1~10 = 25  
2~10 = 5050
```

## 2. 인수의 형식

### ❖ 키워드 인수

- 인수 이름 지정하여 대입 형태로 전달하는 방식

#### keywordarg

```
def calcstep(begin, end, step):  
    sum = 0  
    for num in range(begin, end + 1, step):  
        sum += num  
    return sum  
  
print("3 ~ 5 =", calcstep(3, 5, 1))  
print("3 ~ 5 =", calcstep(begin = 3, end = 5, step = 1))  
print("3 ~ 5 =", calcstep(step = 1, end = 5, begin = 3))  
print("3 ~ 5 =", calcstep(3, 5, step = 1))  
print("3 ~ 5 =", calcstep(3, step = 1, end = 5))
```

#### 실행결과

```
3 ~ 5 = 12  
3 ~ 5 = 12  
3 ~ 5 = 12  
3 ~ 5 = 12  
3 ~ 5 = 12
```

- 앞쪽에 키워드 인수 있으면 뒤쪽에 위치 인수 올 수 없음

## 2. 인수의 형식

### ❖ 키워드 가변 인수

- **\*\* 기호**를 인수 목록에 붙여 키워드 인수를 가변 개수 전달함

keywordvararg

```
def calcstep(**args):  
    begin = args['begin']  
    end = args['end']  
    step = args['step']  
  
    sum = 0  
    for num in range(begin, end + 1, step):  
        sum += num  
    return sum  
  
print("3 ~ 5 =", calcstep(begin = 3, end = 5, step = 1))  
print("3 ~ 5 =", calcstep(step = 1, end = 5, begin = 3))
```

실행결과

```
3 ~ 5 = 12  
3 ~ 5 = 12
```

- 위치 인수와 키워드 인수를 동시에 가변으로 취할 수도 있음

### 3. 변수의 범위

#### ❖ 지역 변수

- 함수 내부에서 선언하는 변수

```
def calcsun(n):  
    sum = 0                # 지역 변수 초기화  
    for num in range(n + 1):  
        sum += num         # 누적  
    return sum             # 리턴
```

- 함수 내부에서만 사용되고 밖으로는 알려지지 않음

local

```
def kim():  
    temp = "김과장의 함수"  
    print(temp)
```

```
kim()  
print(temp)
```

실행결과

```
김과장의 함수  
Traceback (most recent call last):  
  File "C:/Python/test.py", line 6, in <module>  
    print(temp)  
NameError: name 'temp' is not defined
```



### 3. 변수의 범위

#### ❖ 전역 변수

- 함수 바깥에서 선언하는 변수
- 어디에서나 참조할 수 있음

global

```
salerate = 0.9

def kim():
    print("오늘의 할인율 :", salerate)

def lee():
    price = 1000
    print("가격 :", price * salerate)

kim()
salerate = 1.1
lee()
```

실행결과

```
오늘의 할인율 : 0.9
가격 : 1100.0
```

### 3. 변수의 범위

#### ■ 쓰기에 주의

- 초기화하는 장소에 따라 변수의 범위가 결정

global2

```
price = 1000

def sale():
    price = 500

sale()
print(price)
```

실행결과     1000

- 함수 내부에서 전역 변수 대입하여 초기화하면 새로운 지역 변수가 생성
- global 함수
  - 함수 내부에서 지역 변수 새로 만들지 않고 전역 변수 참조하게 함

### 3. 변수의 범위

#### ❖ docstring

- 함수 선언문과 본체 사이에 작성하는 문자열
- 함수의 사용법, 인수의 의미, 주의사항 등 설명 작성
- 실행에는 영향 없음

##### docstring

```
def calcsun(n):  
    """1 ~ n까지의 합계를 구해 리턴한다."""  
    sum = 0  
    for i in range(n+1):  
        sum += i  
    return sum
```

```
help(calcsun)
```

##### 실행결과

```
Help on function calcsun in module __main__:
```

```
calcsun(n)  
    1 ~ n까지의 합계를 구해 리턴한다.
```