함수

함수와 인수

❖ 반복되는 코드

- 함수로 정의하여 반복을 없앰

- 함수 정의

```
def 함수명(인수 목록):
    본체
```

- 함수 호출

```
함수명(인수 목록)
```

함수와 인수

❖ 반복되는 코드

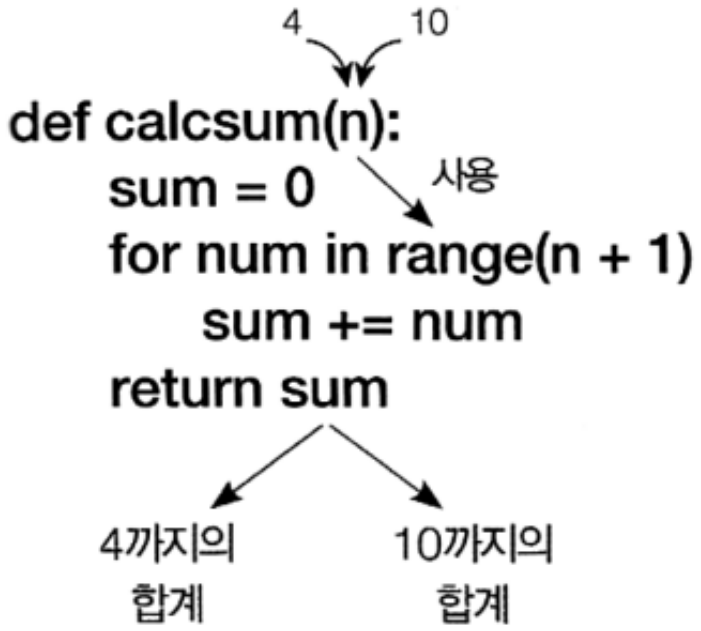
```
def calcsum(n):
    total = 0
    for num in range(n+1):
        total += num

    return total

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

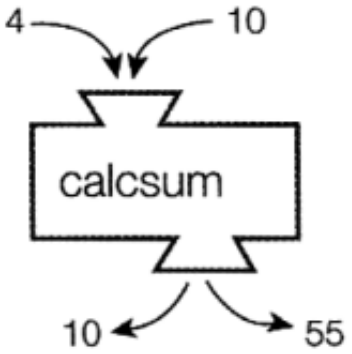
```
 ~ 4 = 10
 ~ 10 = 55
```

함수와 인수

❖ 인수

- 함수로 값을 전달했을 때 이를 저장하는 변수





함수와 인수

❖ 인수

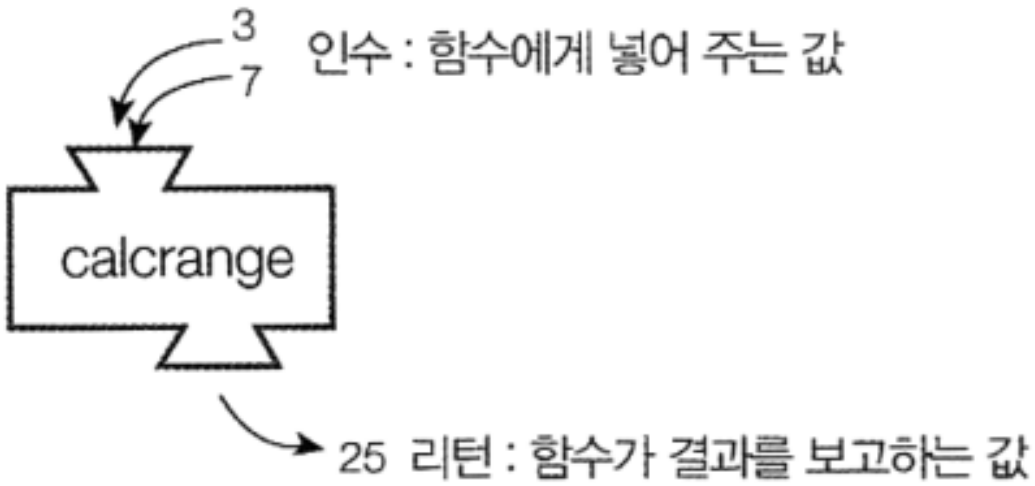
```
def calcrange(begin, end):
    total = 0
    for num in range(begin, end+1):
        total += num

    return total

print("3 ~ 7 =", calcrange(3, 7))
```

```
3 ~ 7 = 25
```

함수와 인수

❖ 리턴값

- 함수의 실행결과를 호출한 곳으로 넘기는 값



함수와 인수

❖ 리턴값

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

printsum(4)
printsum(10)
```

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

```
a = printsum(4)
print(a)
```

```
~ 4 = 10
None
```

함수와 인수

❖ 리턴값

```
print(printsum(4)*2)
```

```
~ 4 = 10
----> print(printsum(4)*2)

TypeError: unsupported operand type(s) for *: 'NoneType' and 'int'
```

함수와 인수

❖ **pass**

- 아무것도 안하고 넘어감
- 함수는 반드시 코드 블럭이 있어야 함
  - 실제 구현을 나중으로 미루고자 할 때 pass 지정

```
def calctotal():
    # 나중에 완성할 것
    pass

if score >= 90:
    pass
else:
    pass
```

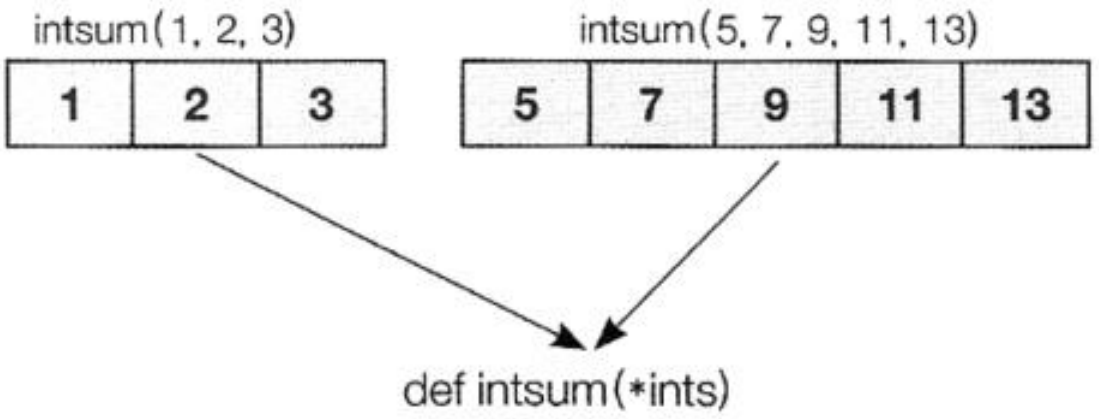
인수의 형식

❖ 가변 인수

- 인수의 수가 고정되지 않음
- 호출시 원하는 만큼 인수를 지정
- 함수에서는 이를 튜플 변수로 받음
- 일반 인수 뒤에만 올 수 있음
- 하나만 사용 가능

```
def 함수명(*인수명):
    명령 블럭
```

```
calcrange(3)
calcrange(3, 7, 10)
```



인수의 형식

❖ 가변인수

```
def intsum(*ints):
    total = 0
    for num in ints:
        total += num

    return total

print(intsum(1, 2, 3))              # ints = (1, 2, 3)
print(intsum(5, 7, 9, 11, 13))      # ints = (5, 7, 9, 11, 13)
print(intsum(8, 9, 6, 2, 9, 7, 5, 8))
```

```
6
45
54
```

인수의 형식

❖ 인수의 기본값

- 함수 호출시 인수가 지정되지 않았을 때 사용할 값
- 함수 정의시 인수에 값을 대입
- 인수 목록의 마지막 부분에 배정
- 중간에 배정시 구분 불가

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

    return total

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

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

인수의 형식

❖ 인수의 기본값

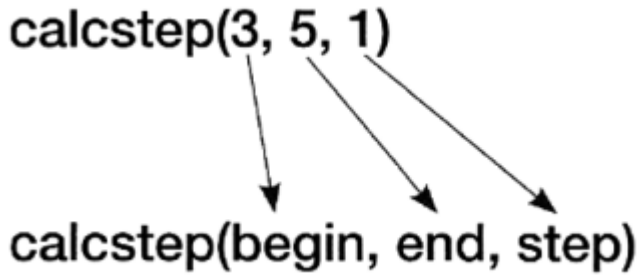
```
def calcstep(begin, end=1, step = 1):
    total = 0
    for num in range(begin, end +1, step):
        total += num

    return total

print("1 ~ 10 =", calcstep(1, 10, 2))
print("1 ~ 100 =", calcstep(1, 100))
```

```
1 ~ 10 = 25
1 ~ 100 = 5050
```

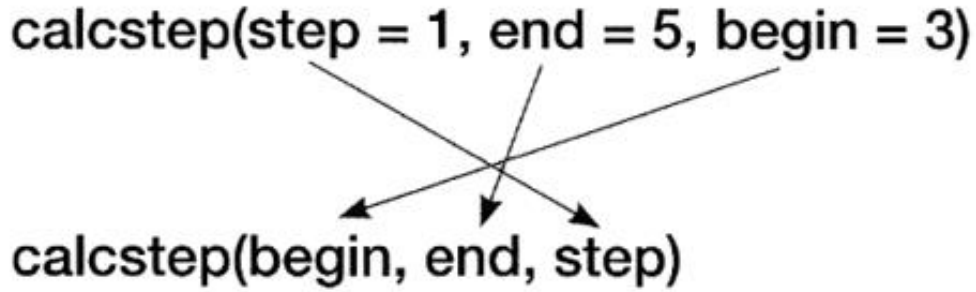
인수의 형식

❖ 키워드 인수

- 일반적으로 함수 호출시 인수의 배치 순서대로 매칭



- 인수 순서가 아닌 인수의 명칭으로 매칭하는 방법



인수의 형식

❖ 키워드 인수

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

    return total

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))
```

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

인수의 형식

❖ 키워드 인수

```
a = 10
b = 20
print(a, b)
print(a, b, sep=',')
print(a, b, sep=',', end='$')
print(a, b, end='$', sep=',')
```

```
10 20
10,20
10,20$10,20$
```

인수의 형식

❖ 키워드 가변 인수

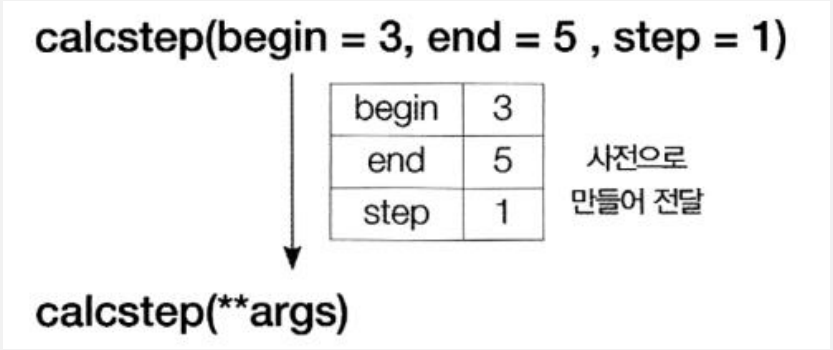
- 키워드 인수를 가변 개수로 전달할 때 사용하는 방법
- **기호로 지정하여 타입은 사전(dictionary)이 됨

```
def 함수명(**인수명):
    명령 블럭
```

인수의 형식

❖ 키워드 가변 인수

```
def calcstep(**args):
    begin = args['begin']
    end = args['end']
    step = args['step']

    total = 0
    for num in range(begin, end +1, step):
        total += num

    return total

print("3 ~ 5 =", calcstep(begin=3, end=5, step=1))
print("3 ~ 5 =", calcstep(step=1, end=5, begin=3))
```

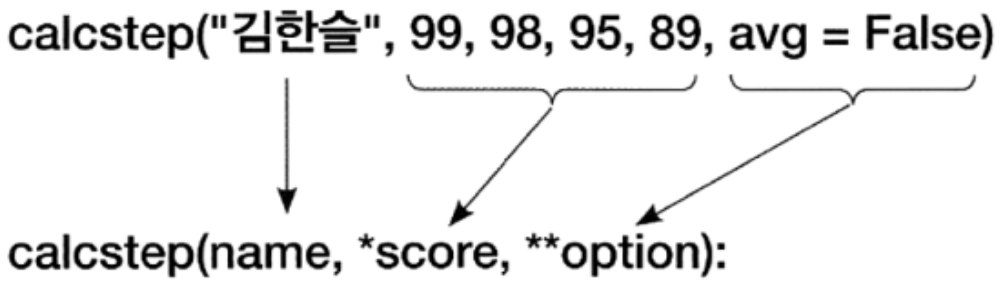



```
3 ~ 5 = 12
3 ~ 5 = 12
```

인수의 형식

❖ **일반 변수, 가변 변수, 키워드 가변 변수 모두 사용**

- 일반 변수, 가변 변수, 키워드 가변 변수 순서로 배치



인수의 형식

❖ 일반 변수, 가변 변수, 키워드 가변 변수 모두 사용

```
def calcscore(name, *score, **option):
    print(name)
    total = 0
    for s in score:
        total += s

    print("총점 :", total)
    if(option['avg'] == True):
        print("평균 :", total/len(score))

calcscore("홍길동", 88, 99, 77, avg=True)
calcscore("고길동", 99, 88, 95, 85, avg=False)
```

```
홍길동
총점 : 264
평균 : 88.0
고길동
총점 : 367
```

변수의 범위

❖ **지역 변수**

- 함수 내에서 사용된 변수
- 함수 내에서 만 사용 가능
  - 함수 밖에서 사용 불가

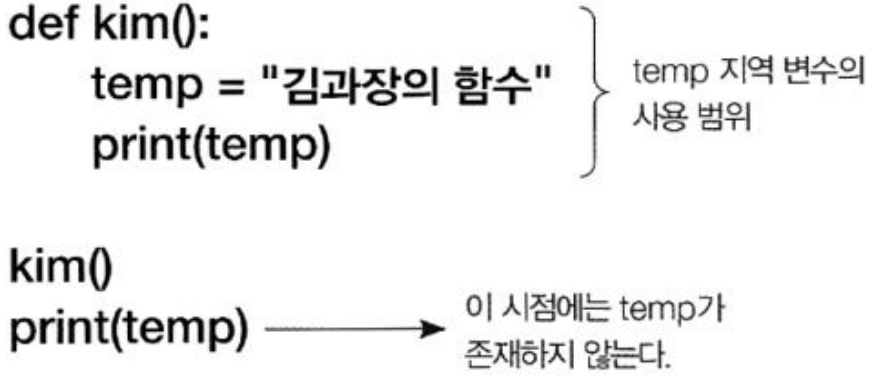
```
def calcsum(n):
    total = 0
    for num in range(n + 1):
        total += num

    return total
```

변수의 범위

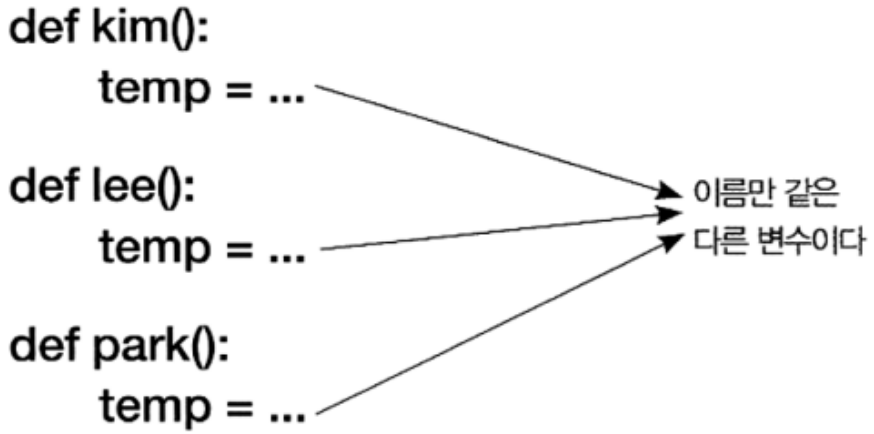
❖ 지역 변수

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

kim()
print(temp)
```

```
김과장의 함수
Traceback (most recent call last):
  File, line 6, in <module>
    print(temp)
NameError: name 'temp' is not defined
```



변수의 범위



❖ 지역 변수

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

def lee():
    temp = 2**10
    return temp

def park(a):
    temp = a*2
    print(temp)

kim()
print(lee())
park(6)
```

```
김과장의 함수
1024
12
```

변수의 범위

❖ 전역 변수

- 어디서든 접근 가능한 변수
- 탑 레벨에서 사용된 변수

```
salerate = 0.9

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

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

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

```
오늘의 할인율: 0.9
가격 : 1100.0
```

변수의 범위

❖ 전역 변수

```
price = 1000

def sale():
    price = 500

sale()
print(price)
```

```
1000
```

```
price = 1000

def sale():
    price = 500
    print("sale", id(price))

sale()
print("global", id(price))
```

```
sale 140412151296944
global 140412151297040
```

변수의 범위

❖ 전역 변수

```
price = 1000

def sale():

    global price
    price = 500

sale()
print(price)
```

```
500
```

변수의 범위

❖ docstring

- 함수의 도움말
- 함수의 코드 블록 앞에 문자열로 지정
- help(함수명) 호출 시 출력될 문자열

```
def calcsum(n):
    """1 ~ n까지의 합계를 구해 리턴한다"""
    total = 0
    for i in range(n+1):
        total += i

    return total

help(calcsum)
```

```
Help on function calcsum in module __main__:

calcsum(n)
    1 ~ n까지의 합계를 구해 리턴한다
```