

# 함수

---

# 목차

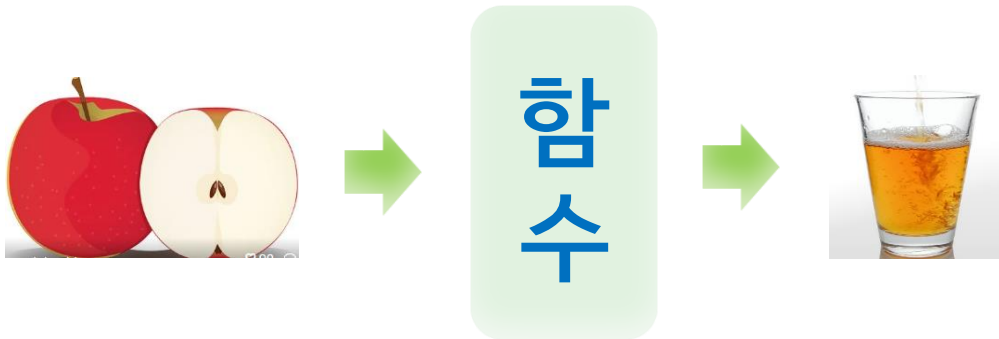
---

1. 함수 정의
2. 함수 구조
3. 매개변수 인수 반환값
4. 함수 예제
5. 디폴트 인수, 키워드 인수
6. Lambda 함수
7. 지역변수/전역변수
8. 내장 함수

# 1. 함수 정의 [1]

- ✦ 코드의 묶음에 이름을 붙인 것
- ✦ 입력을 받아서 출력을 내보내는 요술상자

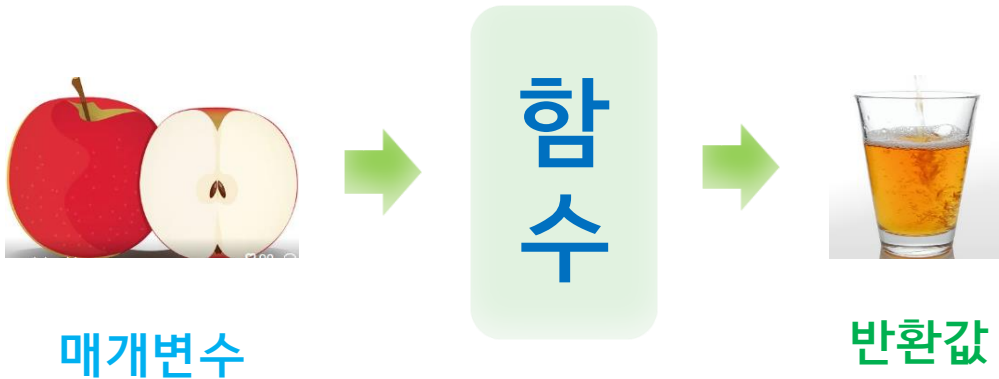
함수는 자료를 전달받아 일을 하고  
그 결과를 돌려 줍니다



# 1. 함수 정의 [2]

✦ 매개변수를 입력받아 처리한 후 ‘반환값’을 돌려 줌.

함수는 자료를 전달받아 일을 하고  
그 결과를 돌려 줍니다



## 2. 함수 구조 [1]

```
def 함수이름(매개변수) :
```

```
    수행할 문장1
```

```
    수행할 문장2
```

```
    ...
```

```
    return 반환값
```

```
def add(a,b):
```

```
    sum = a + b
```

```
    return sum
```

## 2. 함수 구조 [2]

# 함수 정의 (선언)

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

```
c = add(3,5) # 함수 호출  
print( c )
```

## 2. 함수 구조[3]

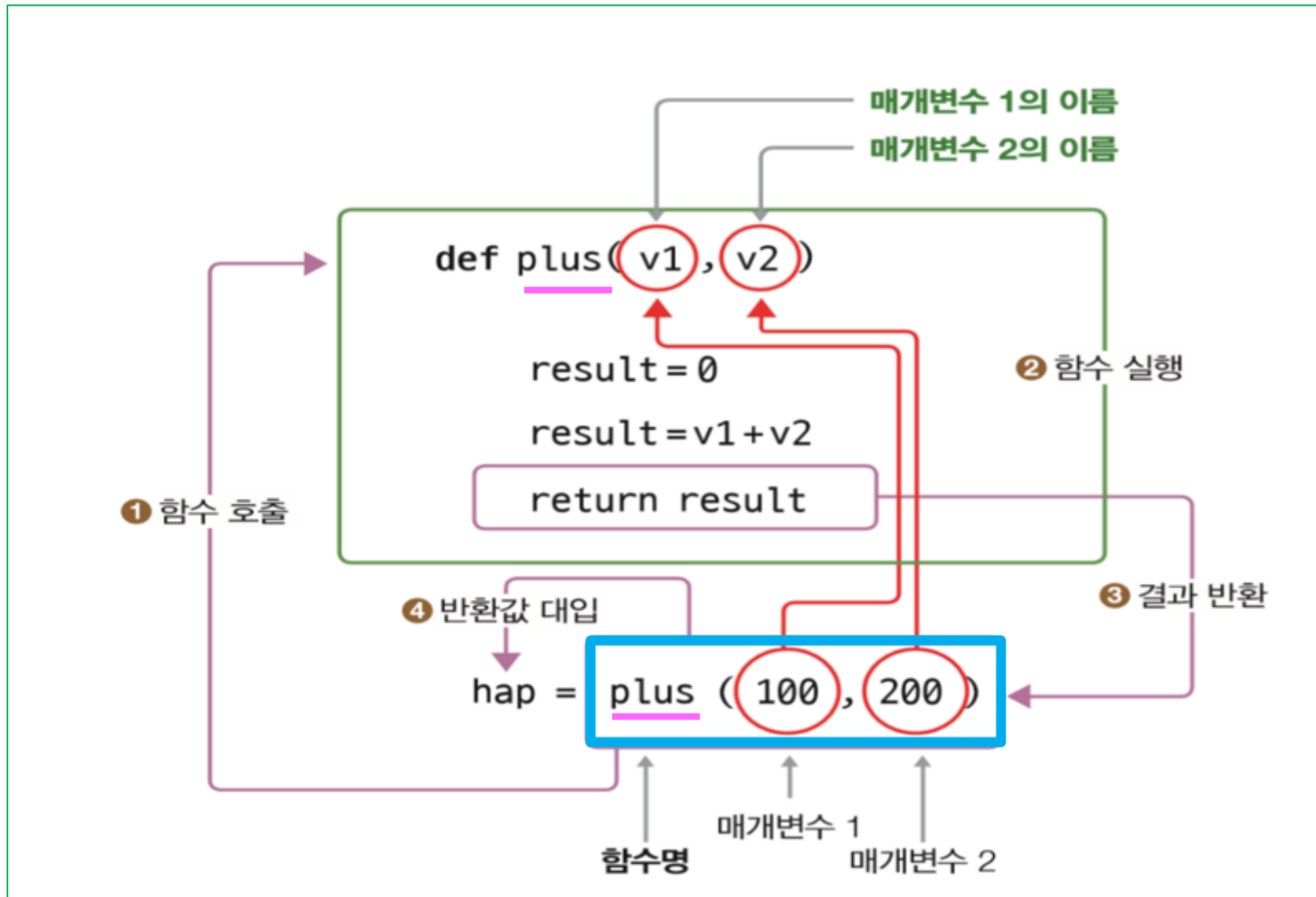
함수 정의(선언)부분과 함수 호출

```
def print_name(): #함수 정의      2  
    print("korea")
```

```
print_name()           #함수 호출      1  
print("happy")         3
```

## 2. 함수 구조[3]

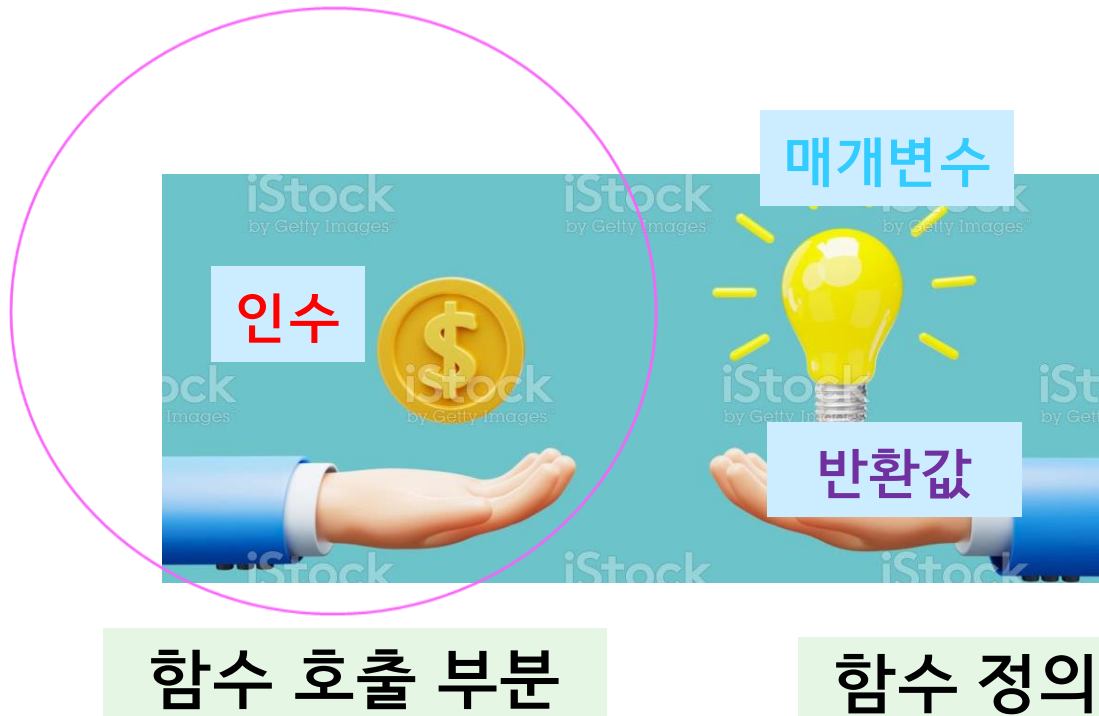
### 함수 정의 / 함수 호출





### 3. 매개변수와 인수

- ✦ **인수** : 함수 호출 시, 전달되는 실제의 값
- ✦ **매개변수** : 함수 내부에서 전달받은 변수
- ✦ **반환값** : 함수는 함수 호출한 곳으로 값을 반환할 수 있음



## 4. 함수 4가지 형태

✦ 매개변수와 반환값의 유무에 따라 분류

✦ 매개변수 ○ 반환값 ○

✦ 매개변수 ○ 반환값 ×

✦ 매개변수 × 반환값 ○

✦ 매개변수 × 반환값 ×



## 4. 함수 4가지 형태

#1

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

#2

```
def say ():  
    print('korea')
```

#3

```
def hello ():  
  
    return 'hi'
```

#4

```
def sub (a,b):  
    print('%d' % (a+b))
```

## 5. 함수 예제 [1]

- ✦ 한 번만 함수를 정의하면 언제든지 필요할 때면 함수를 호출하여 일을 수행할 수 있음.
- ✦ 재사용, 프로그램 간결, 오류 수정 용이

```
def gugudan(dan):  
    for i in range(1, 10):  
        print("%2d * %2d = %2d" % (dan, i, dan * i))  
    print()
```

```
gugudan(5)  
gugudan(20)  
gugudan(7)
```

## 5. 함수 예제 - 함수 정의/ 함수 호출

```
dan = 3
for i in range(1,10): # 1 2 3 4 5 6 7 8 9
    print("%2d *%2d = %2d" % (dan, i, dan * i))
```



### 함수 정의

```
def gugudan(dan):
    for i in range(1, 10):
        print("%2d *%2d = %2d" % (dan, i, dan * i))
```

```
gugudan(5)
gugudan(20)
gugudan(7)
gugudan(13)
```

### 함수 호출

## 5. 함수 예제

```
hap = 0
for i in range(1,11): # 1 2 3 4 ...10
    hap = hap + i

print(hap)
```



```
def sum (number):
    hap = 0

    for i in range(1,number+1): #1 2 3.. number
        hap = hap + i

    return hap #반환값 유무 print(hap)
```

## 5. 함수 예제 [2]

✦ 매개변수○ 반환값○

```
def sum (number):  
    hap = 0  
  
    for i in range(1, number+1): #1 2 3  
        hap = hap + i  
  
    return hap
```

```
sum_result = sum (3)  
print(sum_result)  
sum_result = sum (100)  
print(sum_result)
```

## 5. 함수 예제 [2]

✦ 매개변수 ○ 반환값 × ← 조건

Add (3)

Add (20)

Add (100)

1에서 3까지의 합계: 6

1에서 20까지의 합계: 210

1에서 100까지의 합계: 5050



## 5. 함수 예제 [2]

★ 매개변수 ○ 반환값 ×

```
def Add (number):  
    hap = 0  
  
    for i in range(1, number+1): #1 2 3 ... number  
        hap = hap + i  
  
    print("1에서 %d까지의 합계: %d" % (number, hap))
```

Add (3)

Add (20)

Add (100)

1에서 3 까지의 합계: 6

1에서 20 까지의 합계: 210

1에서 100 까지의 합계: 5050

## 5. 함수 예제 [3]

✦ 함수에 여러 개의 매개변수 전달하기

```
def get_sum(start, end):  
    sum = 0  
    for i in range(start, end+1): # 2 3 4 5  
        sum = sum + i  
  
    return sum  
  
print(get_sum(2, 5))
```

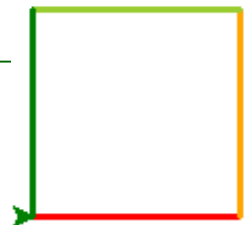
## 5. 함수 예제 [4]

```
import turtle  
t = turtle.Turtle()  
c = ['red', 'orange', 'yellowgreen', 'green', 'blue', 'indigo']  
t.width(3)
```

```
length = 100
```

```
for count in range(4):  
    t.color(c[count])  
    t.fd(length)  
    t.left(90)
```

```
turtle.done()
```

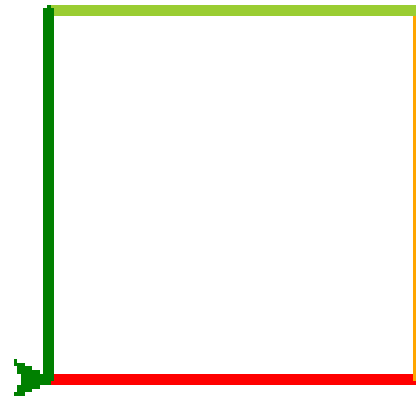


## 5. 도전

- ✦ 사각형 그리는 함수를 작성해 보세요
- ✦ 조건) 함수 이름 - square
- ✦ 매개 변수 1개 있음 ( 매개변수는 한 변의 길이를 저장한다)
- ✦ 반환값 없음

예) **square(50)** 이면

한 변의 길이가 50픽셀인 사각형을 그린다.



## 5. 도전

```
import turtle  
t = turtle.Turtle()  
c=['red','orange','yellowgreen','green']  
t.width(3)
```

```
def square(length):  
    for count in range(4):  
        t.color(c[count])  
        t.fd(length)  
        t.left(90)
```

```
square(50)  
square(70)  
square(30)  
turtle.done()
```

## 5. 도전 - 도형 그리기

```
for i in range(n):  
    t.forward(w)  
    t.left(360/n)
```

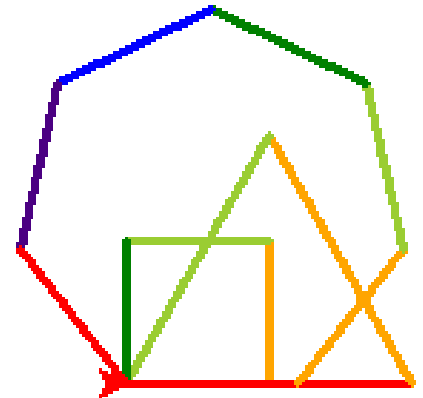


★ 함수 - **n\_polygon** (n, length)

★ 매개 변수 2개 있음 / 반환값 없음

예) **n\_polygon(3,100)** : 한 변의 길이가 100픽셀인 삼각형

```
n_polygon(3,100)  
n_polygon(4,50)  
n_polygon(7,60)
```



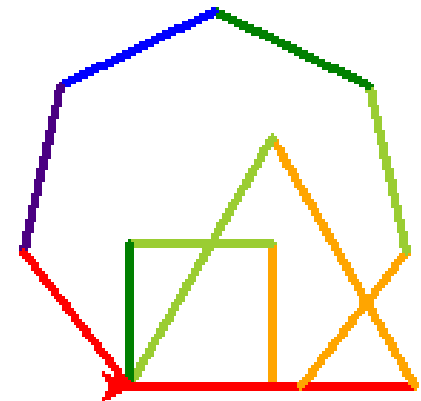
```
import turtle
t = turtle.Turtle()
c=['red','orange','yellowgreen','green','blue','indigo']
t.width(3)
```

```
def n_polygon(n,length):
```

이 부분을 완성하세요

```
n_polygon(3,100)
n_polygon(4,50)
n_polygon(7,60)

turtle.done()
```

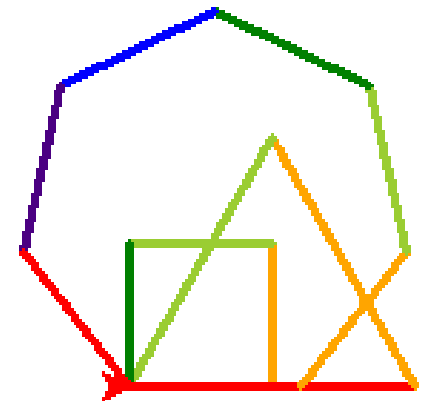


```
import turtle
t = turtle.Turtle()
c=['red','orange','yellowgreen','green','blue','indigo']
t.width(3)
```

```
def n_polygon(n,length):
    for i in range(n):
        t.color(c[i % 6])
        t.fd(length)
        t.left(360 / n)
```

```
n_polygon(3,100)
n_polygon(4,50)
n_polygon(7,60)
```

```
turtle.done()
```





## 6. 디폴트 인수

- ✦ 함수의 매개변수가 기본값을 가질 수 있음.
- ✦ 이것을 디폴트 인수(default argument) 라고 함.

```
def hap(n1,n2=10):  
    sum = n1 + n2  
    return sum
```

```
print(hap(5,20))
```

1

```
print(hap(5))
```

2

25  
15

## 6. 키워드 인수

✦ 매개변수 이름을 명시적으로 지정해서 전달하는 방법

```
def power(a,b):  
    print(a**b)
```

2<sup>5</sup>

power(2,5)

power(b=5,a=2)

1

#위치 인수

2

#키워드 인수

32  
32

## 7. Lambda 함수

★ lambda (매개변수 : 매개변수를 사용한 표현식)

```
print( (lambda a: a+10 ) (5) )
```

1

```
add = lambda a: a+10
```

```
print(add(5) )
```

```
print(add(100))
```

2

3

## 6. Lambda 함수 2

```
s = lambda x: x**2
```

```
print(s(5))
```

```
print(s(7))
```

25

49

# Filter 함수

- ✦ `Filter(함수 이름, 데이터)`
- ✦ 두번째 인수 데이터를 첫번째 인수인 함수에 입력했을때,
- ✦ 반환값이 참인 것만 묶어서(`filter`)하여 돌려준다.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
```

```
print( " 짝수 in the list:", even_numbers)
```

```
짝수 in the list: [2, 4, 6, 8, 10]
```

# Filter 함수

```
# 함수
def is_even(num):
    return num % 2 == 0

# 데이터
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# filter() 함수를 사용하여 짝수만 걸러내기
even_numbers = list(filter(is_even, numbers))
print("짝수 in the list:", even_numbers)
```

짝수 in the list: [2, 4, 6, 8, 10]

# map 함수

- ✦ map( 함수 이름, 반복가능한 데이터 )
- ✦ 두번째 인수 데이터를 첫번째 인수인 함수에 입력했을때,
- ✦ 수행결과를 묶어서 돌려준다.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
two_times = list(map(lambda x: x * 2, numbers))

print(two_times)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

## enumerate 함수

- ✦ 데이터(리스트, 튜플, 문자열)을 입력으로 받아,
- ✦ 각 항목의 **인덱스**를 추적할 수 있다.

```
data = ['a', 1, 10, 999, 3, 7, 2000, 8, '가']  
for i, v in enumerate(data):  
    print(i,v)
```

```
0 a  
1 1  
2 10  
3 999  
4 3  
5 7  
6 2000  
7 8  
8 가
```



# enumerate 함수

- ✦ 데이터(리스트, 튜플, 문자열)을 입력으로 받아,
- ✦ 각 항목의 **인덱스**를 추적할 수 있다.

```
my_list = ['apple', 'banana', 'orange']  
for i, v in enumerate(my_list):  
    print(f'Index: {i}, Value: {v}')
```

```
Index: 0, Value: apple  
Index: 1, Value: banana  
Index: 2, Value: orange
```

## max, min, chr , ord(ordinal)

```
numbers = [1, 10, 999, 3, 7, 2000, 8]
```

```
strings="pYthon"
```

```
print(max(numbers))
```

```
print(max(strings))
```

```
print(min(numbers))
```

```
print(min(strings))
```

```
print(chr(97))
```

```
print(chr(65))
```

```
print(chr(44032))
```

```
print(ord('a')) #문자를 정수(유니코드)로 변환
```

```
print(ord('A'))
```

```
print(ord('가'))
```

2000

t

1

Y

a

A

가

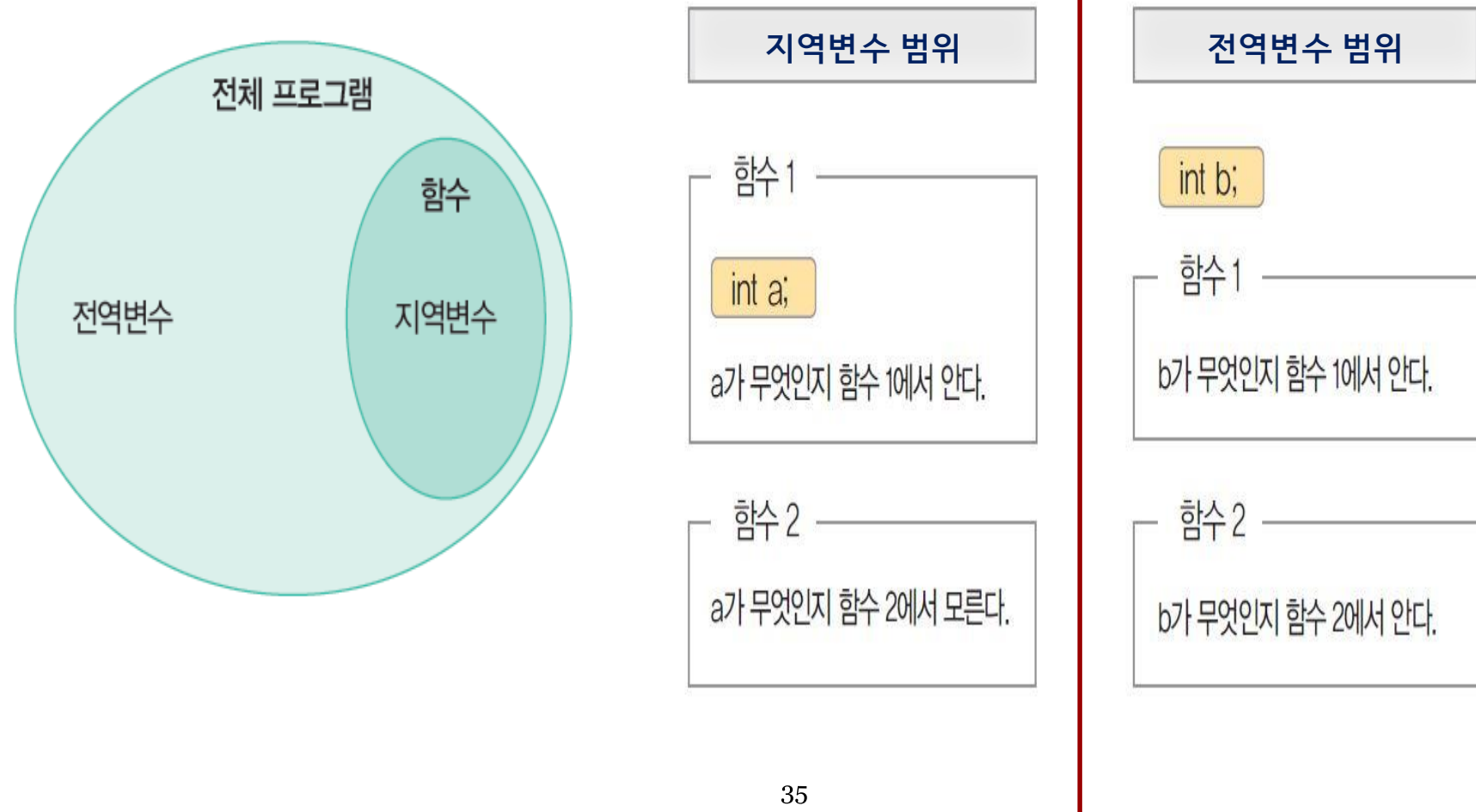
97

65

44032

## 7. 변수 범위 [1]

- ✦ **지역 변수** : 함수 안에서 선언되는 변수 (**한정된** 지역에서만 사용)
- ✦ **전역 변수** : 함수 외부에서 선언되는 변수 (**프로그램 전체**에서 사용)



## 7. 변수의 범위 예제

```
def count():  
    hap = 200    # 지역변수  
    print(hap)
```

```
def count1():  
    print(hap)
```

```
hap = 100 # 전역변수  
count()  
count1()
```

```
print(hap)
```

200  
100  
100

## 8. 매개변수의 개수를 지정하지 않고 전달하는 방법

✦ 입력값이 여러 개, 입력하려는 데이터의 개수를 모를 때

```
def 함수이름 (*매개변수):
```

수행할 문장

```
def func(*para) :  
    result = 0  
    for num in para :  
        result = result + num  
    return result
```

## 8. 매개변수의 개수를 지정하지 않고 전달하는 방법

```
def func(*para) :  
    result = 0  
    for num in para :  
        result = result + num  
    return result
```

```
hap = 0
```

```
hap = func(10, 20)
```

```
print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
```

```
hap = func(10, 20, 30)
```

```
print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
```

## 9. 함수의 결과값은 하나

✦ 반환값이 여러 개이면, 반환값을 튜플형태로 만들어 하나 반환

```
def add_and_mul(a,b):  
    return a+b, a*b    # (a+b, a*b) (8,15) 튜플
```

```
result = add_and_mul(3,5)  
print(result)
```

```
result1, result2 = add_and_mul(3,5)  
print(result1)  
print(result2)
```

(8, 15)

8

15

# 마무리

---

1. 함수 정의
2. 함수 구조
3. 매개변수 인수 반환값
4. 함수 예제
5. 디폴트 인수, 키워드 인수
6. Lambda 함수
7. 지역변수/전역변수