파이썬 함수(Function) 기초

Section02 함수 기본

함수의 개념과 필요성

- 함수(Function) : '무엇'을 넣으면, '어떤 것'을 돌려주는 요술 상자
- 메서드(Method)와 차이점 : 함수는 외부에 별도로 존재, 메서드는 클래스 안에 존재
- 함수의 형식

```
함수명()
```

- print() 함수

```
print("CookBook-파이썬")
```

■ 함수의 형식과 활용

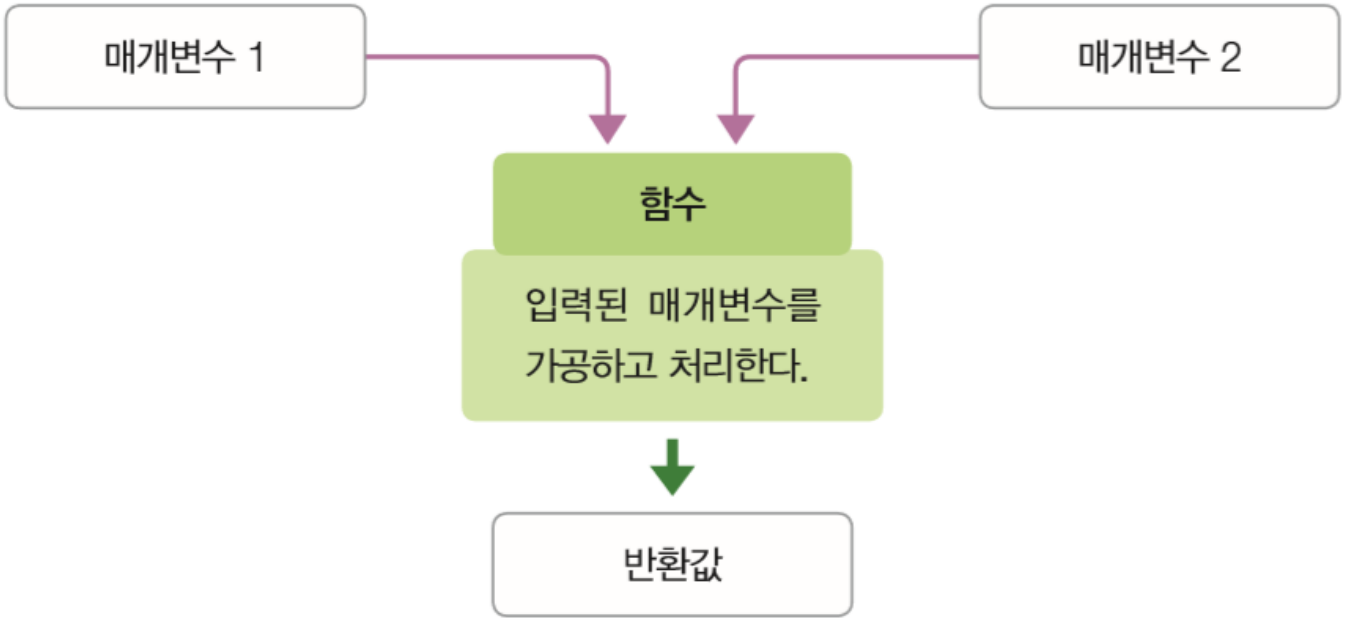

그림 9-3 함수의 기본 형식

- plus() 함수

Code09-04.py

```
1  ## 함수 선언 부분 ##
2  def plus(v1, v2) :
3      result = 0
4      result = v1 + v2
5      return result
6
7  ## 전역 변수 선언 부분 ##
8  hap = 0
9
10 ## 메인 코드 부분 ##
11 hap = plus(100, 200)
12 print("100과 200의 plus() 함수 결과는 %d" % hap)
```

출력 결과

```
100과 200의 plus() 함수 결과는 300
```

- plus() 함수

Code09-04.py

```
1  ## 함수 선언 부분 ##
2  def plus(v1, v2) :
3      result = 0
4      result = v1 + v2
5      return result
6
7  ## 전역 변수 선언 부분 ##
8  hap = 0
9
10 ## 메인 코드 부분 ##
11 hap = plus(100, 200)
12 print("100과 200의 plus() 함수 결과는 %d" % hap)
```

2~5행 : plus() 함수를 정의
4행 : 매개변수로 받은 두 값의 합계를 구함
5행 : 반환
11행 : 100, 200 두 값 을 전달하면서 plus() 함수를 호출해 hap에 대입
12행 : plus() 함수에서 반환된 값 출력

출력 결과

```
100과 200의 plus() 함수 결과는 300
```

- plus() 함수의 형식과 호출 순서

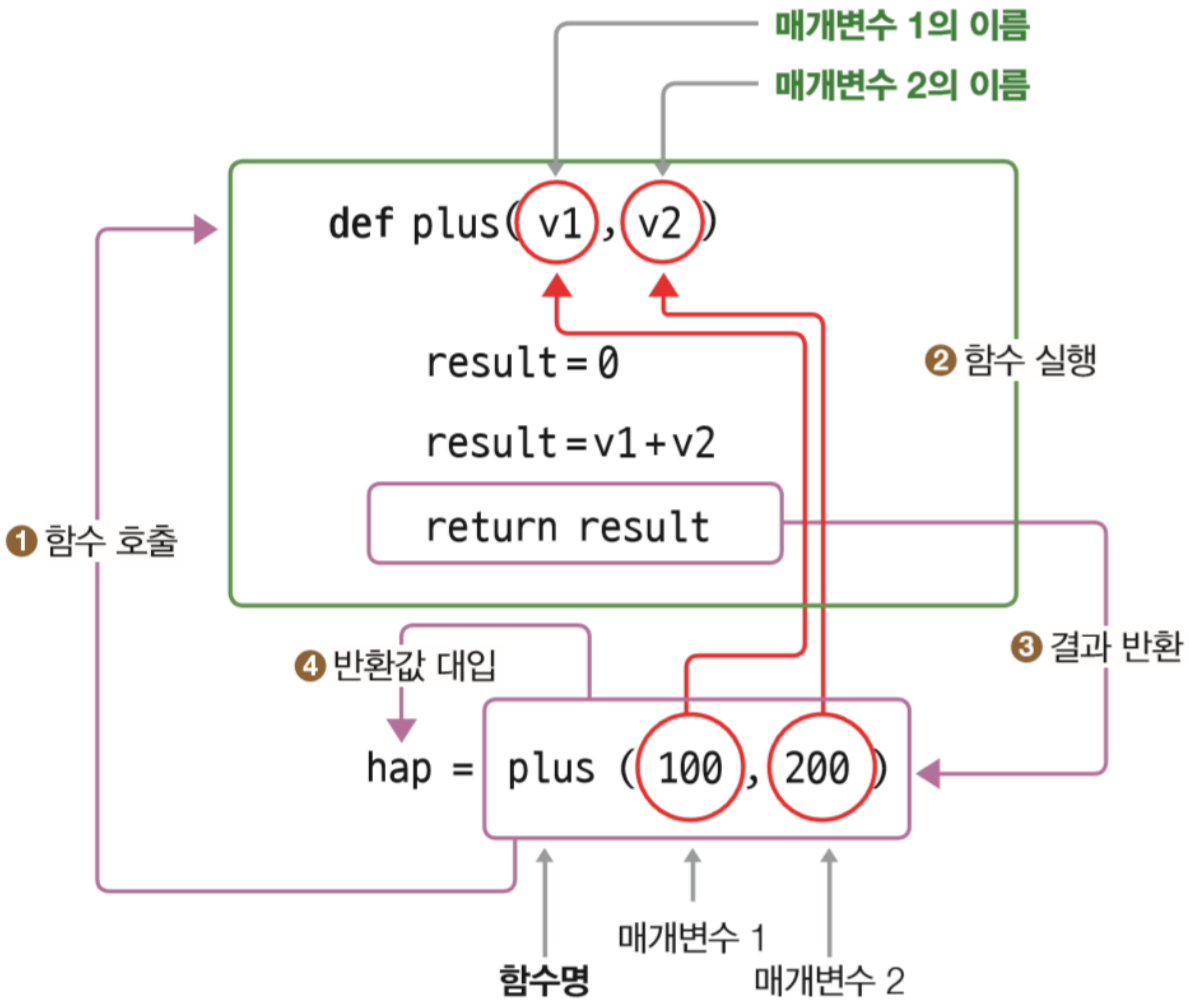

그림 9-4 plus() 함수의 형식과 호출 순서

- plus() 함수의 형식과 호출 순서

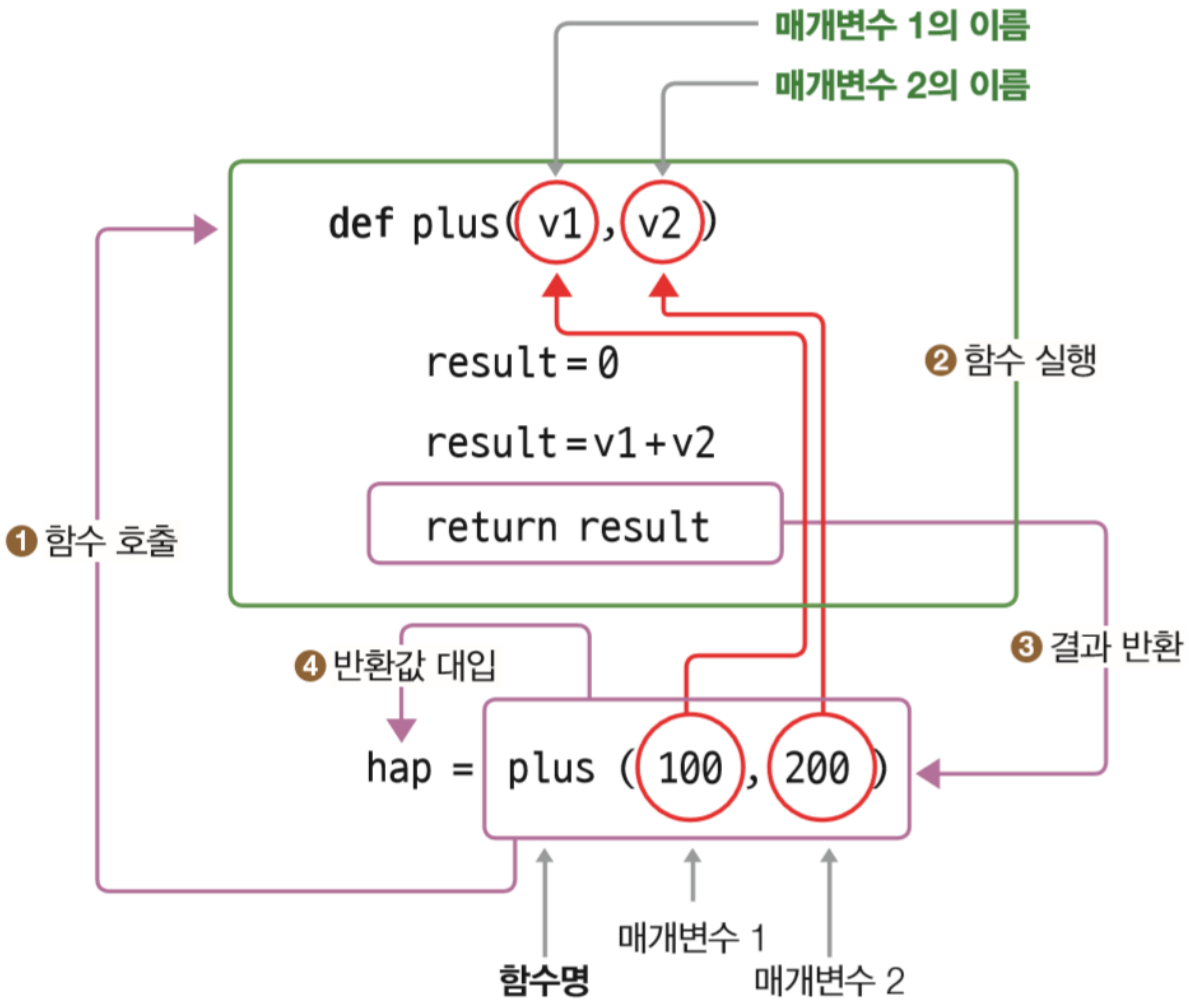

그림 9-4 plus() 함수의 형식과 호출 순서

- plus() 함수의 호출 간략 표현

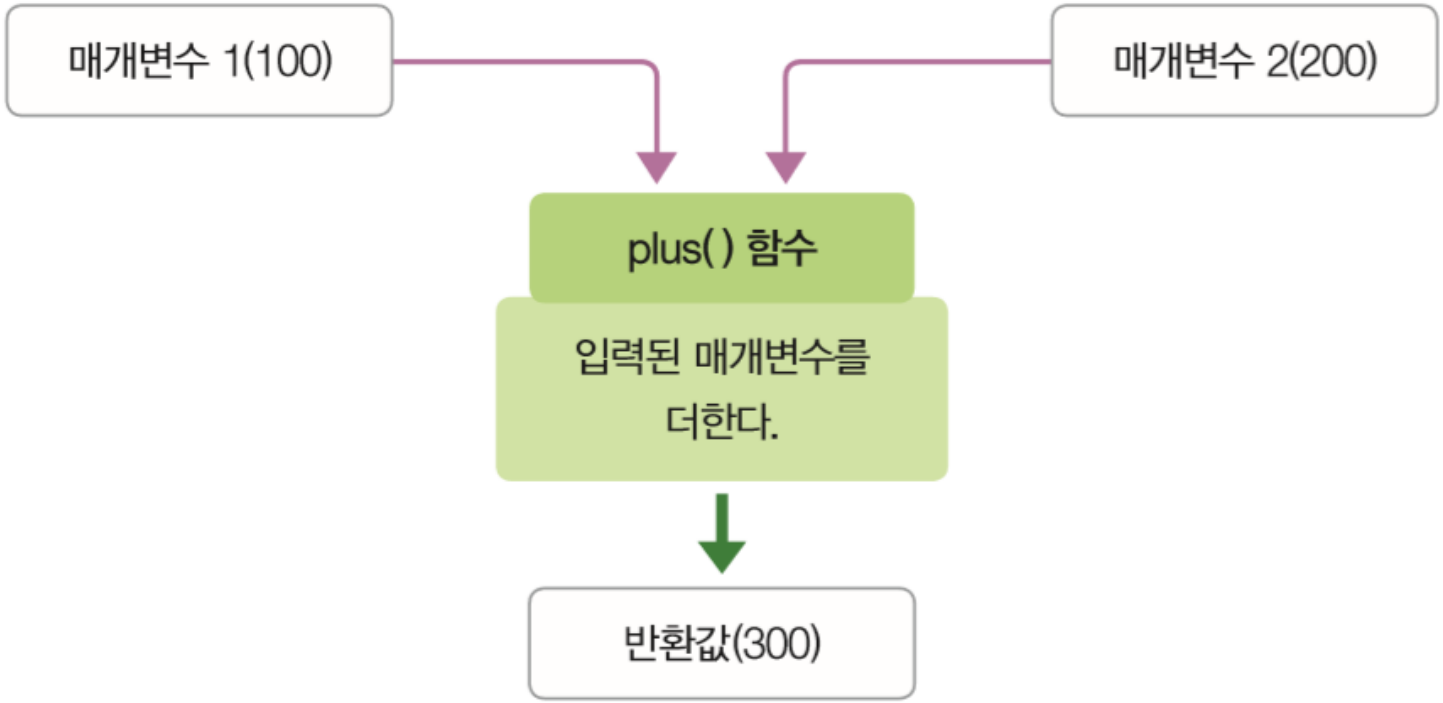

그림 9-5 plus( ) 함수의 호출 간략 표현

Section02 함수 기본

- 덧셈, 뺄셈, 곱셈, 나눗셈을 하는 계산기 함수를 작성

Code09-05.py

```
1  ## 함수 선언 부분 ##
2  def calc(v1, v2, op) :
3      result = 0
4      if op == '+' :
5          result = v1 + v2
6      elif op == '-' :
7          result = v1 - v2
8      elif op == '*' :
9          result = v1 * v2
10     elif op == '/' :
11         result = v1 / v2
12
13     return result
14
15 ## 전역 변수 선언 부분 ##
16 res = 0
17 var1, var2, oper = 0, 0, ""
18
19 ## 메인 코드 부분 ##
20 oper = input("계산을 입력하세요(+, -, *, /) : ")
21 var1 = int(input("첫 번째 수를 입력하세요 : "))
```

2~13행 : 매개변수를 3개 받는 calc() 함수를 정의
20행 : 어떤 연산을 할지 연산자를 입력
21~22행 : 계산할 숫자 2개를 입력
24행 : calc() 함수에 매개 변수 3개를 넘겨주며 호출
4~11행 : 사용자가 입력한 연산자에 따라 필요한 연산을 수행
13행 : 계산된 값을 return으로 반환
24행 : calc() 함수에서 반환한 값을 res에 넣음
26행 : 계산식 출력

```
22  var2 = int(input("두 번째 수를 입력하세요 : "))
23
24  res = calc(var1, var2, oper)
25
26  print("## 계산기 : %d %s %d = %d" % (var1, oper, var2, res))
```

출력 결과

```
계산을 입력하세요(+, -, * , /) : *
첫 번째 수를 입력하세요 : 7
두 번째 수를 입력하세요 : 8
## 계산기 : 7 * 8 = 56
```

Tip • 매개변수(파라미터)는 지역 변수로 취급. Code09-05.py의 calc( ) 함수가 받는 매개변수 v1, v2, op는 모두 calc() 함수 안에서만 사용되는 지역 변수. 지역 변수와 전역 변수는 바로 이어서 설명

■ 함수의 반환값

Tip • 반환값은 return 문으로 반환되므로 리턴값이라고도 함. 매개변수는 파라미터라고도 함

- 반환값이 있는 함수

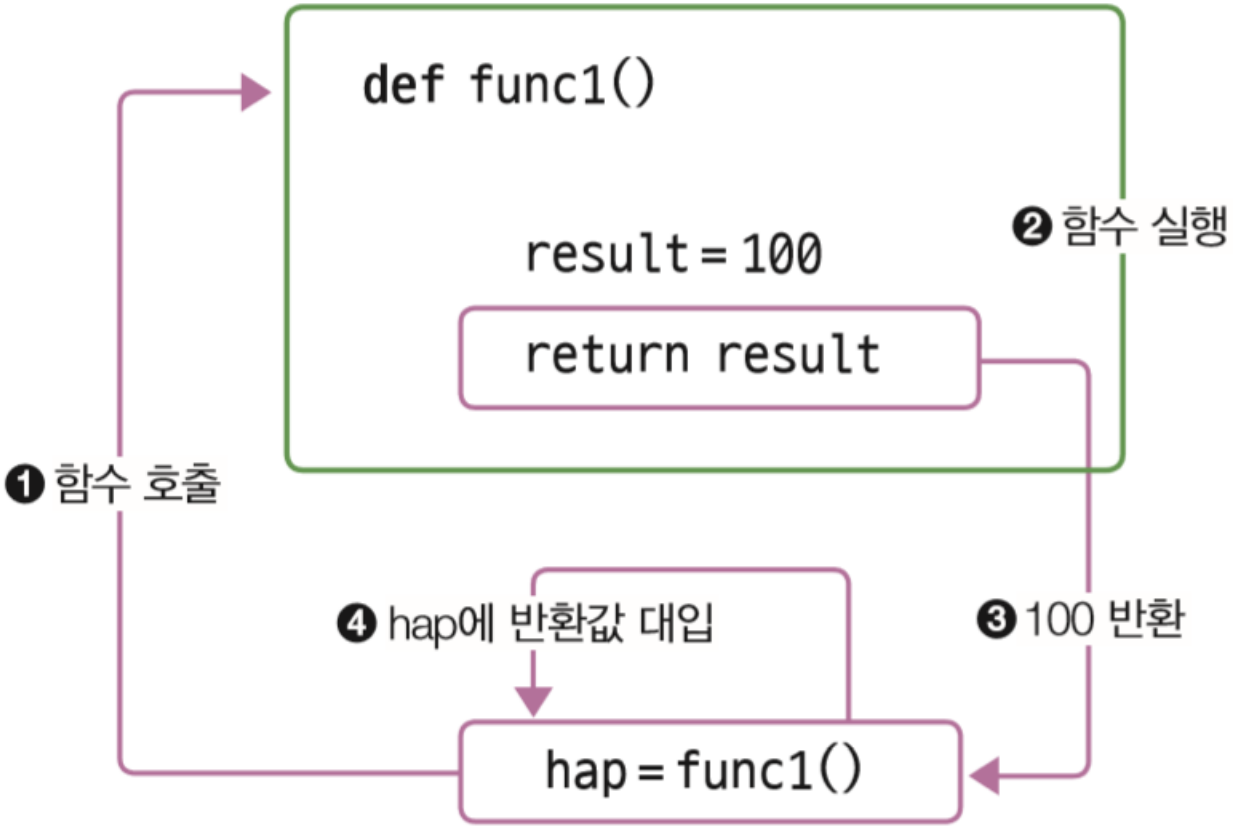

그림 9-8 값의 반환

Section04 함수의 반환값과 매개변수

- 반환값이 없는 함수

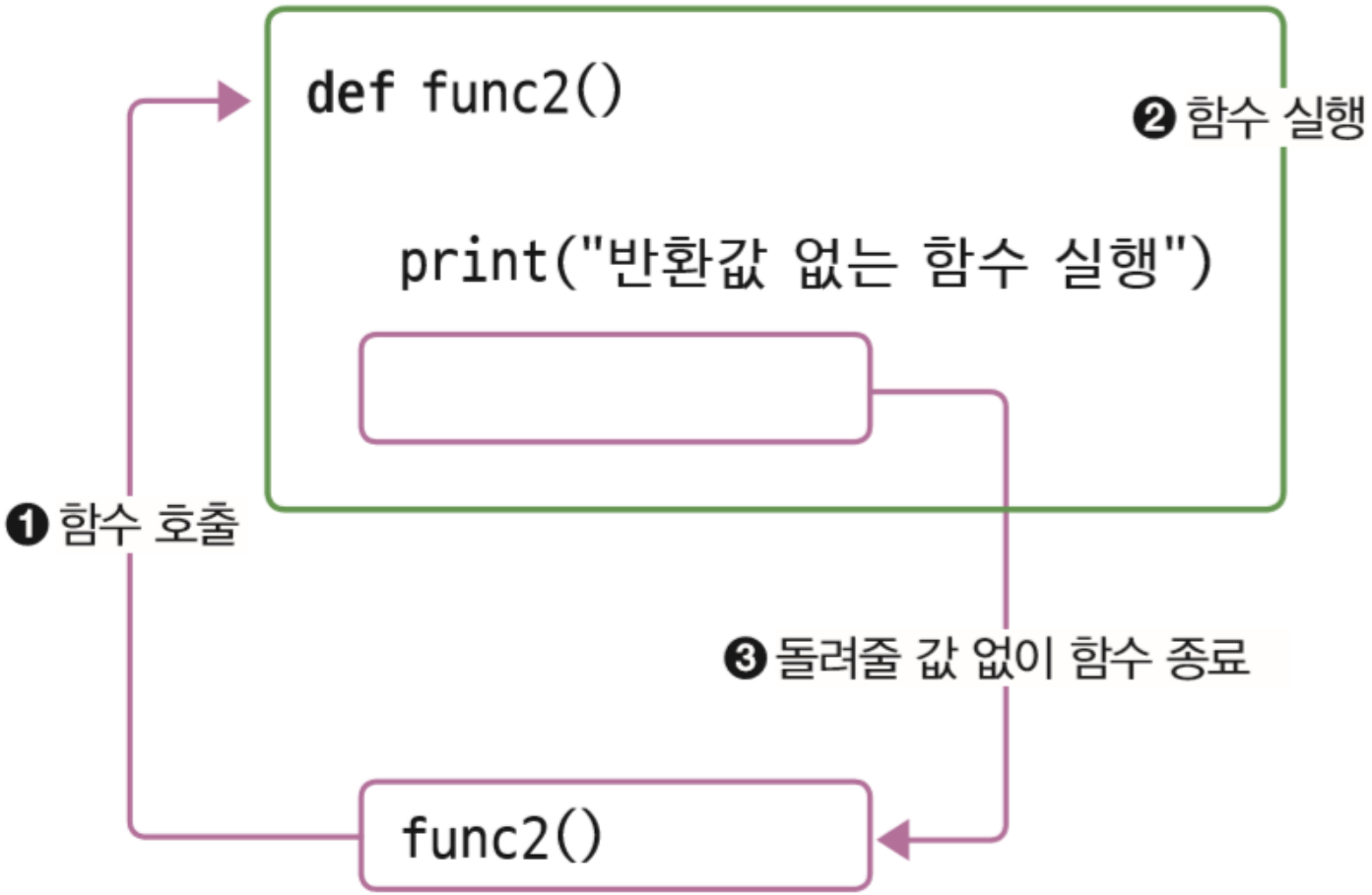

그림 9-9 반환값이 없는 함수의 작동

Section04 함수의 반환값과 매개변수

- 반환값이 없는 함수

Code09-08.py

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      result = 100
4      return result
5
6  def func2() :
7      print("반환값이 없는 함수 실행")
8
9  ## 전역 변수 선언 부분 ##
10 hap = 0
11
12 ## 메인 코드 부분 ##
13 hap = func1()
14 print("func1()에서 돌려준 값 ==> %d" % hap)
15 func2()
```

13행 : 반환값이 있는 함수인 func1()을 호출하면 func1() 실행 후 func1()의 반환값을 hap에 넣고
14행 : 출력
15행 : 반환값이 없는 함수인 func2()를 호출하면 반환 않음

출력 결과

```
func1()에서 돌려준 값 ==> 100
반환값이 없는 함수 실행
```

List, Tuple

- List
  - 기본 자료형의 연속적인 자료구조
    - ex) 배열
  - mutable
    - 내용의 변경이 가능함

```
myFruits = [‘Orange’, ‘Apple’, “Mango’]
append(), pop()
```

- Tuple
  - immutable
    - List와 비슷하나 내용의 변경이 불가능

```
myFruits = (‘orange’, ‘apple’, ‘mango’) 혹은
myFruits = ‘orange, ‘apple’, ‘mango’
```

Set

- 유일한 요소로 구성된 연속형 자료구조

```
mySet = {1,2,3,5}

mySet.add(4)

print(mySet)

{1,2,3,4,5}

mySet.add(2)

print(mySet)

{1,2,3,4,,5}
```

딕셔너리(Dictionary)

- key, value 로 구성된 연속형 자료형

```
myFruits = {'orange':100, 'apple':150, 'mango':60}

for k in myFruits:
    print(k)

for k, v in myFruits.items():
    print(k, v)

for k in myFruits.keys():
    print(k)

for v in myFruits.values():
    print(v)
```