

■ 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")
```

Section02 함수 기본

■ 커피를 타는 과정

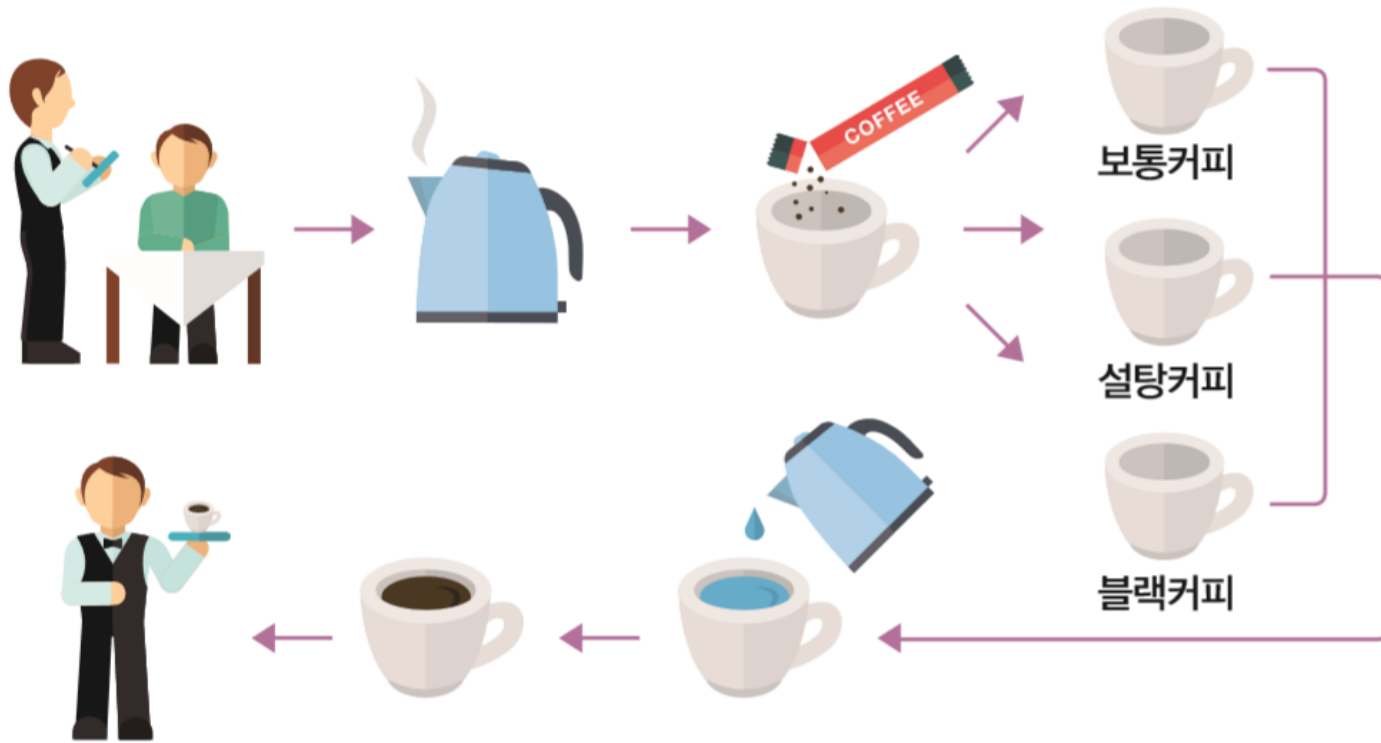


그림 9-1 직접 커피를 타는 과정

Section02 함수 기본

- 커피를 타는 과정의 코드

Code09-01.py

```
1  coffee = 0
2
3  coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)") )
4
5  print()
6  print("#1. 뜨거운 물을 준비한다.");
7  print("#2. 종이컵을 준비한다.");
8
9  if coffee == 1 :
10     print("#3. 보통커피를 탄다.")
11 elif coffee == 2 :
12     print("#3. 설탕커피를 탄다.")
13 elif coffee == 3 :
14     print("#3. 블랙커피를 탄다.")
15 else :
16     print("#3. 아무거나 탄다.\n")
```

1행 : 커피의 종류를 입력받을 변수를 선언

3행 : 손님에게 커피의 종류 입력

6~7행 : 물과 컵을 준비하는 메시지를 출력

9~16행 : 손님이 주문한 커피 종류에 따른 메시지 출력

Section02 함수 기본

```
17
18 print("#4. 물을 붓는다.");
19 print("#5. 스푼으로 젓는다.");
20 print()
21 print("손님~ 커피 여기 있습니다.");
```

18~19행 : 물을 붓고 커피를 스푼으로 잘 저어 녹인 후
21행 : 완성된 커피를 손님에게 제공

출력 결과

어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 2

#1. 뜨거운 물을 준비한다.

#2. 종이컵을 준비한다.

#3. 설탕커피를 탄다.

#4. 물을 붓는다.

#5. 스푼으로 젓는다.

손님~ 커피 여기 있습니다.

Section02 함수 기본

- 손님 3명이 연속해서 들어온다고 했을 때(Code09-01.py의 3~21행을 2번 반복)

```
coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
```

```
print()
```

```
print("#1. 뜨거운 물을 준비한다.");
```

```
print("#2. 종이컵을 준비한다.");
```

```
if coffee == 1 :
```

```
    print("#3. 보통커피를 탄다.")
```

```
elif coffee == 2 :
```

```
    print("#3. 설탕커피를 탄다.")
```

```
elif coffee == 3 :
```

```
    print("#3. 블랙커피를 탄다.")
```

```
else :
```

```
    print("#3. 아무거나 탄다.\n")
```

```
print("#4. 물을 붓는다.");
```

```
print("#5. 스푼으로 젓는다.");
```

```
print()
```

```
print("손님~ 커피 여기 있습니다.");
```

```
    # 두 번째 손님용 코드 다시 반복
```

```
    # 세 번째 손님용 코드 다시 반복
```

Section02 함수 기본

- 손님 3명이 연속해서 들어온다고 했을 때(커피 자판기 만들기)

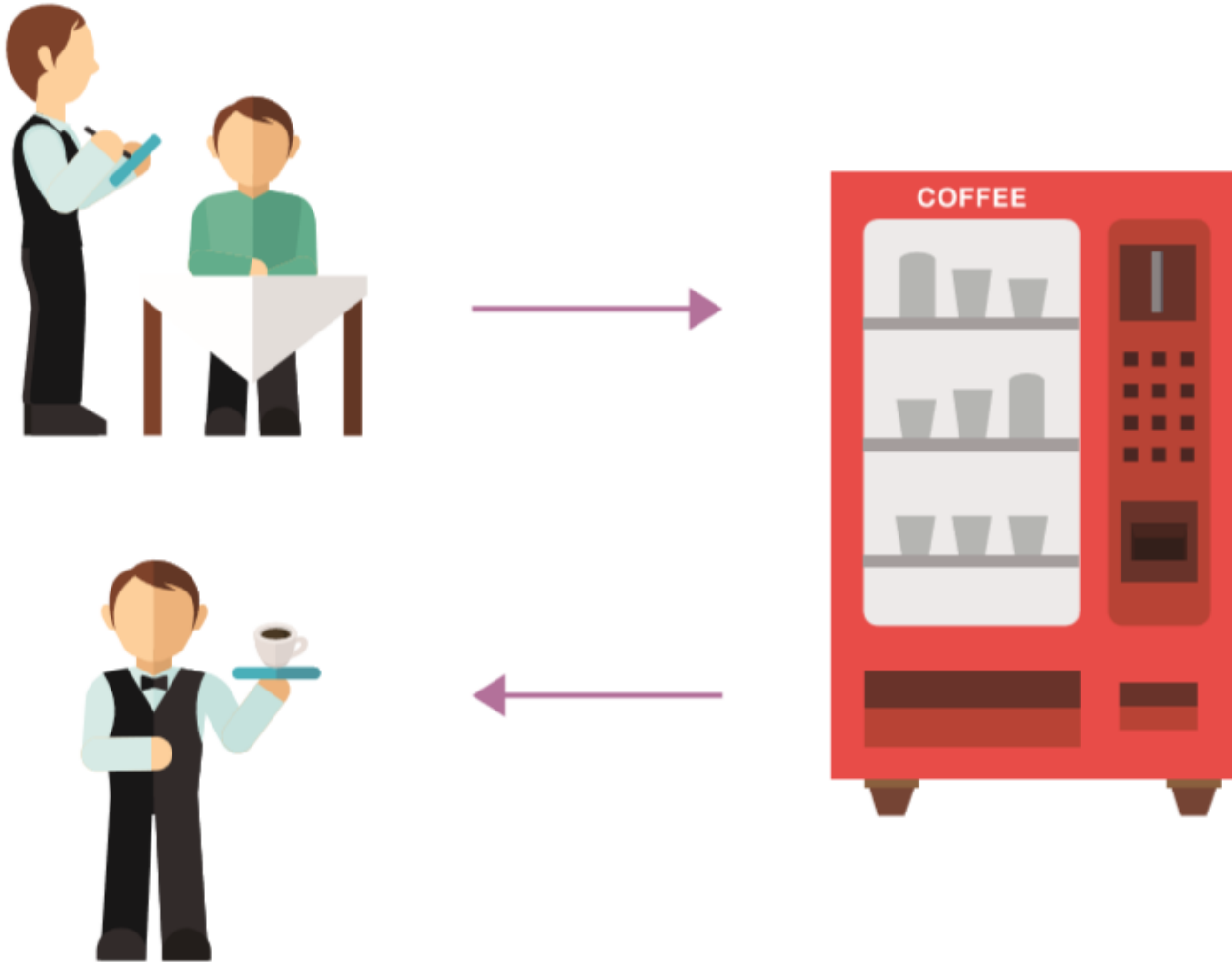


그림 9-2 커피 자판기 사용 예

Section02 함수 기본

- 커피 자판기 실제 프로그램으로 만들기(함수 개념 응용 Code09-01.py 수정)

Code09-02.py

```
1  ## 전역 변수 선언 부분 ##
2  coffee = 0
3
4  ## 함수 선언 부분 ##
5  def coffee_machine(button) :
6      print()
7      print("#1. (자동으로) 뜨거운 물을 준비한다.");
8      print("#2. (자동으로) 종이컵을 준비한다.");
9
10     if button == 1 :
11         print("#3. (자동으로) 보통커피를 탄다.")
12     elif button == 2 :
13         print("#3. (자동으로) 설탕커피를 탄다.")
14     elif button == 3 :
15         print("#3. (자동으로) 블랙커피를 탄다.")
```

Section02 함수 기본

```
16     else :
17         print("#3. (자동으로) 아무거나 탄다.\n")
18
19     print("#4. (자동으로) 물을 붓는다.");
20     print("#5. (자동으로) 스푼으로 젓는다.");
21     print()
22
23     ## 메인 코드 부분 ##
24     coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
25     coffee_machine(coffee)
26     print("손님~ 커피 여기 있습니다.");
```

24행 : 손님에게 커피 종류를 물어봄
25행 : 커피 자판기만 있음
26행 : 손님에게 커피를 대접

출력 결과

어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 2
#1. (자동으로) 뜨거운 물을 준비한다.
#2. (자동으로) 종이컵을 준비한다.
#3. (자동으로) 설탕커피를 탄다.
#4. (자동으로) 물을 붓는다.
#5. (자동으로) 스푼으로 젓는다.
손님~ 커피 여기 있습니다.

Section02 함수 기본

- Code09-02.py를 이용 연속해서 방문한 손님(A, B, C) 3명에게 커피 대접

Code09-03.py

```
1  ## 전역 변수 선언 부분 ##
2  coffee = 0
3
4  ## 함수 선언 부분 ##
5  def coffee_machine(button) :
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23  ## 메인 코드 부분 ##
24  coffee = int(input("A손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
25  coffee_machine(coffee)
26  print("A손님~ 커피 여기 있습니다.")
27
28  coffee = int(input("B손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
29  coffee_machine(coffee)
30  print("B손님~ 커피 여기 있습니다.")
31
32  coffee = int(input("C손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
33  coffee_machine(coffee)
34  print("C손님~ 커피 여기 있습니다.")
```

Section02 함수 기본

출력 결과

A손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 2

#1. (자동으로) 뜨거운 물을 준비한다.

#2. (자동으로) 종이컵을 준비한다.

#3. (자동으로) 설탕커피를 탄다.

#4. (자동으로) 물을 붓는다.

#5. (자동으로) 스푼으로 젓는다.

A손님~ 커피 여기 있습니다.

B손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 3

... 생략 ...

Section02 함수 기본

SELF STUDY 9-1

Code09-03.py를 수정해서 커피 종류를 아메리카노, 카페라떼, 카푸치노, 에스프레소 중 하나를 선택할 수 있도록 하자.
그리고 로제, 리사, 지수, 제니라는 손님 4명의 주문을 받아 보자.

출력 결과

로제씨, 어떤 커피 드릴까요?(1:아메리카노, 2:카페라떼, 3:카푸치노, 4:에스프레소) 4

#1. (자동으로) 뜨거운 물을 준비한다.

#2. (자동으로) 종이컵을 준비한다.

#3. (자동으로) 에스프레소를 탄다.

#4. (자동으로) 물을 붓는다.

#5. (자동으로) 스푼으로 젓는다.

로제씨~ 커피 여기 있습니다.

... 생략 ...

Section02 함수 기본

■ 함수의 형식과 활용

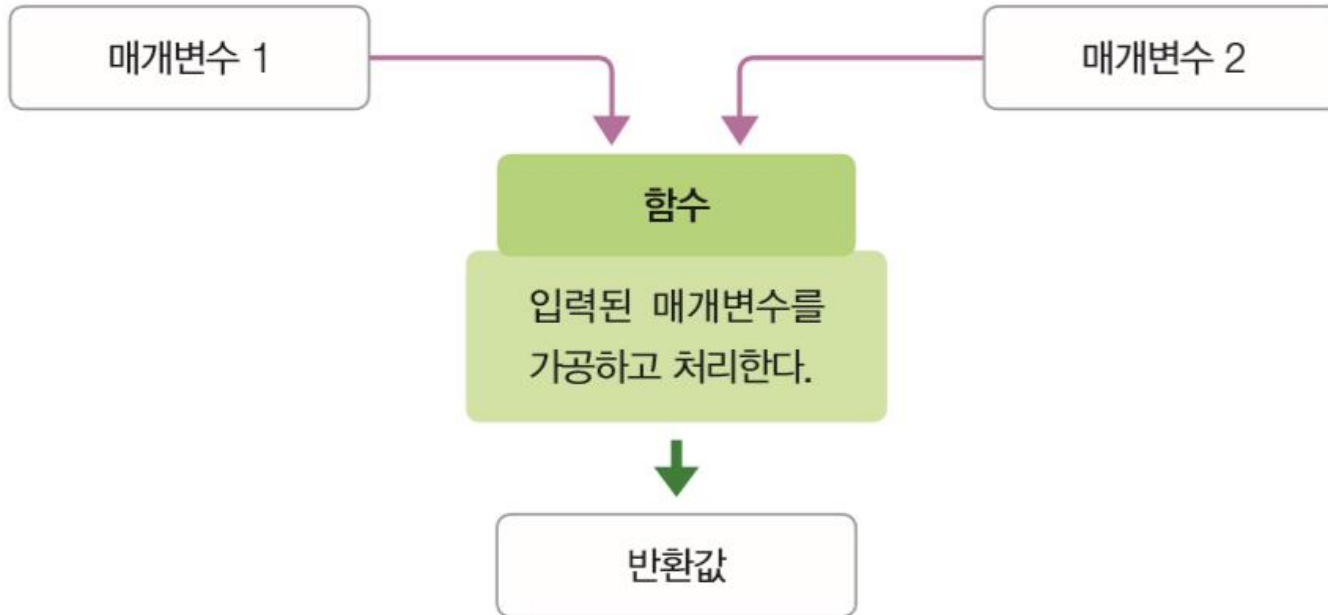


그림 9-3 함수의 기본 형식

Section02 함수 기본

- 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

Section02 함수 기본

▪ 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

Section02 함수 기본

- plus() 함수의 형식과 호출 순서

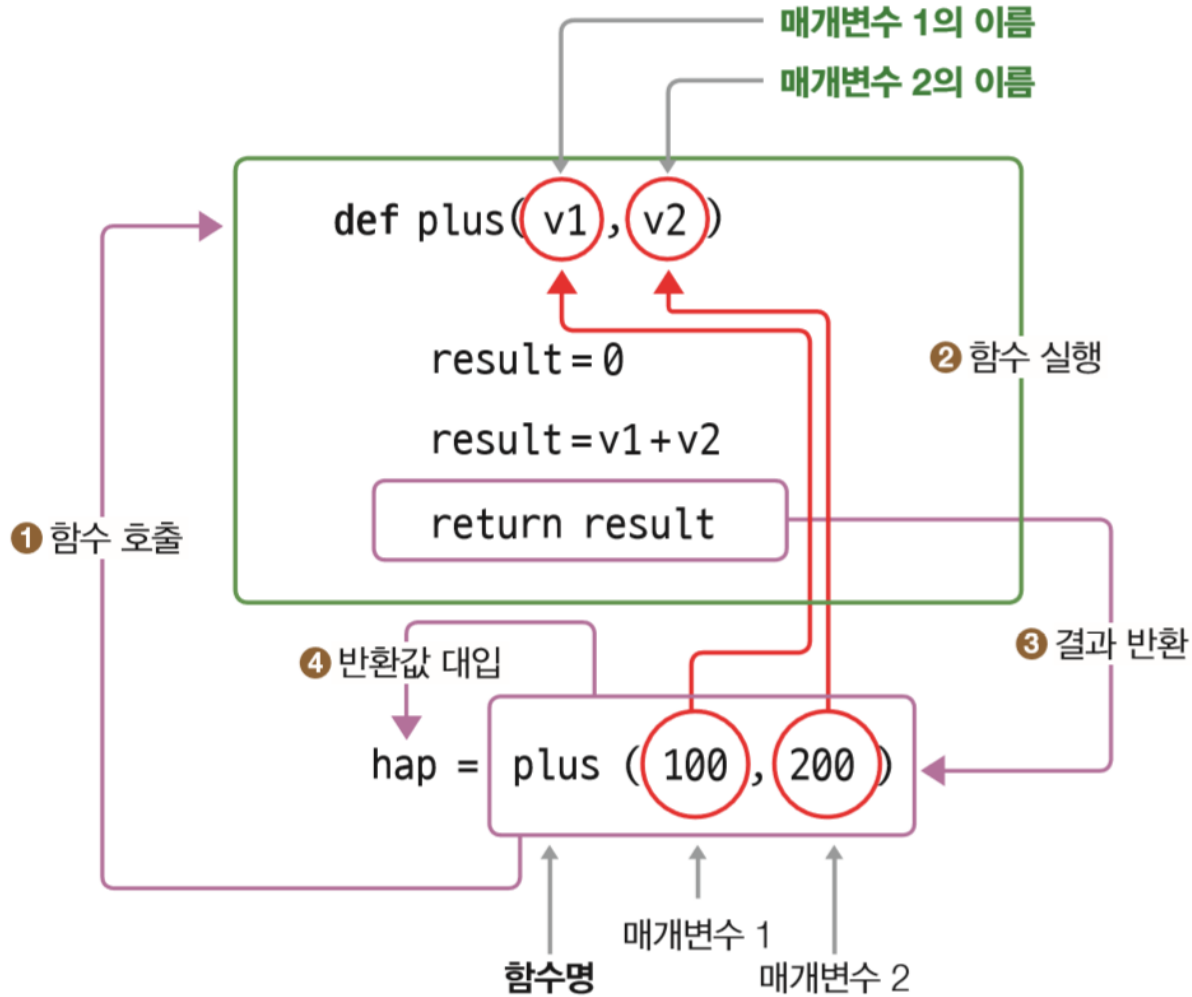


그림 9-4 plus() 함수의 형식과 호출 순서

Section02 함수 기본

- plus() 함수의 형식과 호출 순서

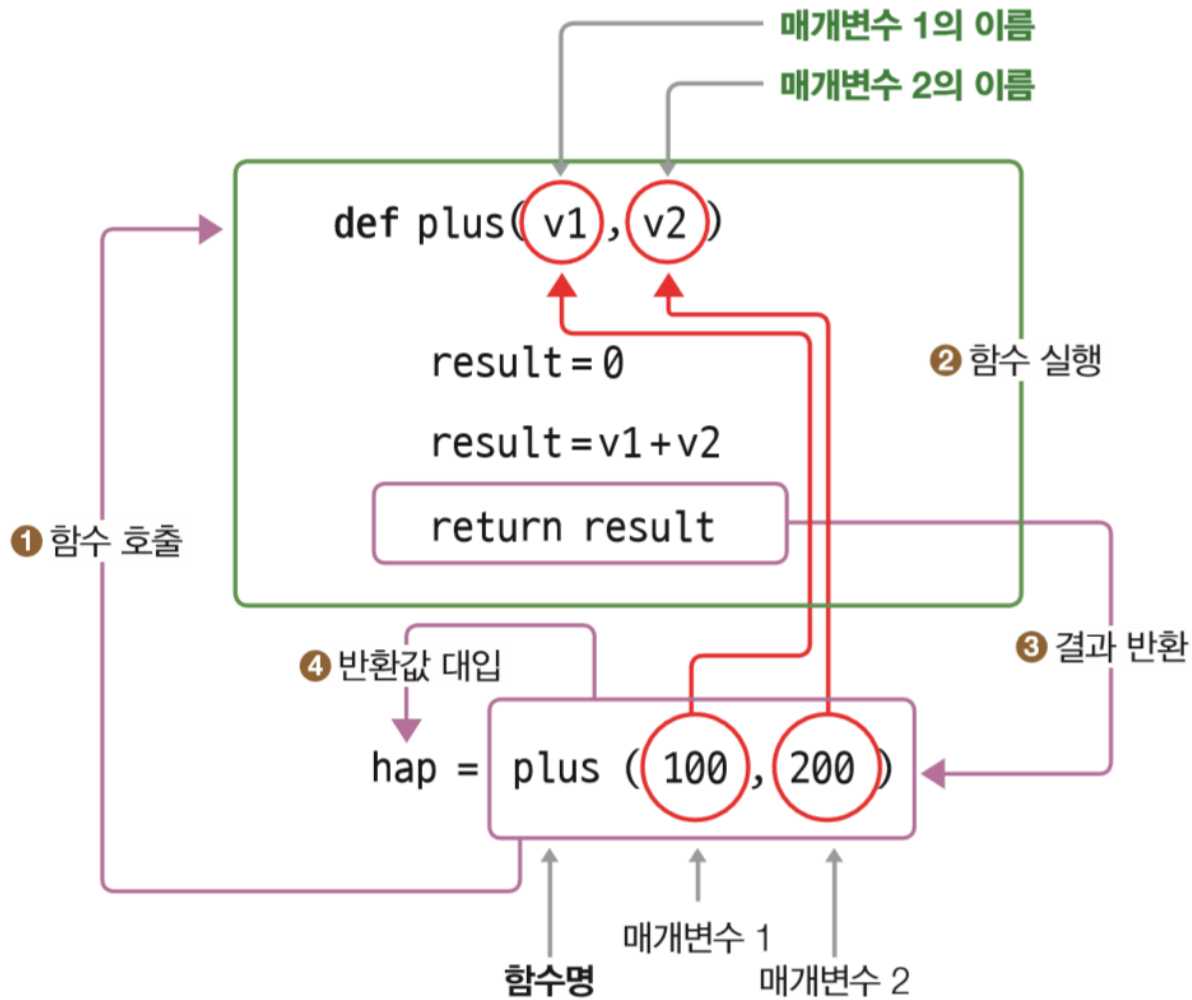


그림 9-4 plus() 함수의 형식과 호출 순서

Section02 함수 기본

- plus() 함수의 호출 간략 표현

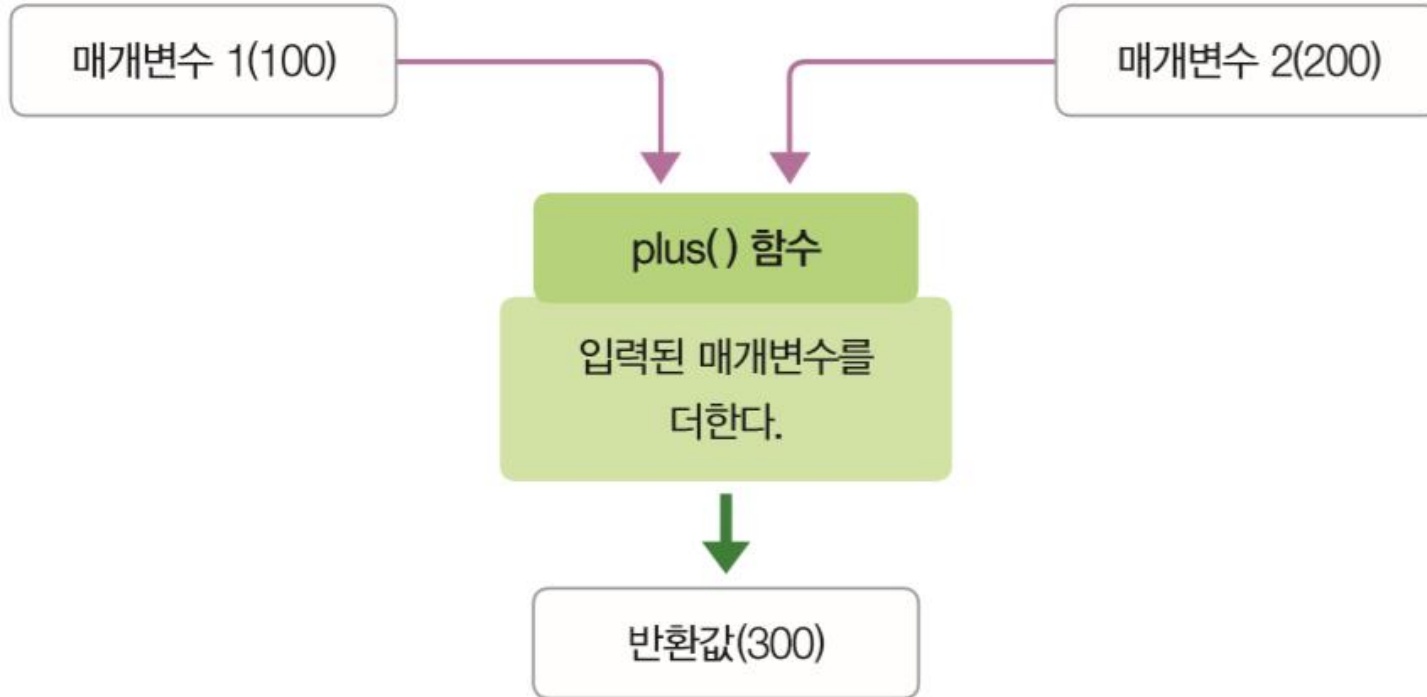


그림 9-5 plus() 함수의 호출 간략 표현

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

Section02 함수 기본

```
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

Section02 함수 기본

SELF STUDY 9-2

Code09-05.py에 다음 기능을 추가해 보자.

- ① 숫자1, 연산자, 숫자2 순서로 입력받는다.
- ② 제곱(**) 연산자를 추가한다.
- ③ 0으로 나누려고 하면 메시지를 출력하고 계산되지 않도록 한다.

힌트 메인 코드 부분에 if~else 문을 활용한다.

출력 결과

```
첫 번째 수를 입력하세요 : 2
계산을 입력하세요(+, -, *, /, **) : **
두 번째 수를 입력하세요 : 4
## 계산기 : 2 ** 4 = 16
```

출력 결과

```
첫 번째 수를 입력하세요 : 8
계산을 입력하세요(+, -, *, /, **) : /
두 번째 수를 입력하세요 : 0
0으로는 나누면 안 됩니다.ππ
```

Section03 지역 변수, 전역 변수

■ 지역 변수와 전역 변수의 이해

- 지역 변수 : 한정된 지역에서만 사용
- 전역 변수 : 프로그램 전체에서 사용

① 지역 변수의 생존 범위

함수 1

$a = 10$

a가 뭔지 함수 1에서 안다.

함수 2

a가 뭔지 함수 2에서 모른다.

② 전역 변수의 생존 범위

함수 1

b가 뭔지 함수 1에서 안다.

함수 2

b가 뭔지 함수 2에서 안다.

$b = 20$

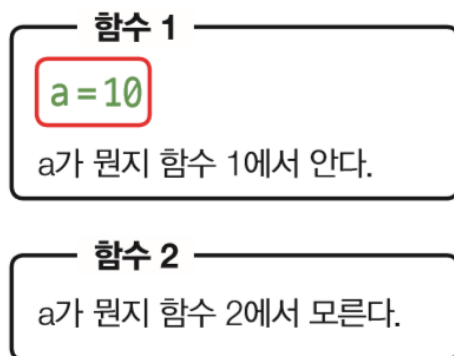
그림 9-6 지역 변수와 전역 변수의 생존 범위

Section03 지역 변수, 전역 변수

■ 지역 변수와 전역 변수의 이해

- 지역 변수 : 한정된 지역에서만 사용
- 전역 변수 : 프로그램 전체에서 사용

① 지역 변수의 생존 범위



② 전역 변수의 생존 범위

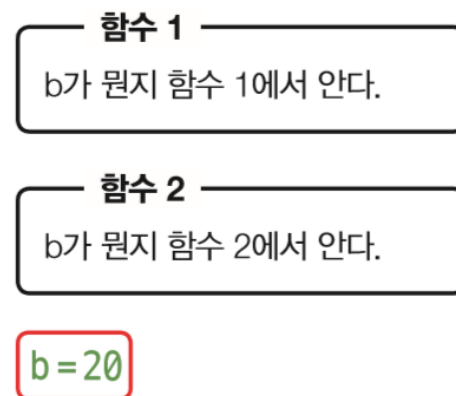


그림 9-6 지역 변수와 전역 변수의 생존 범위

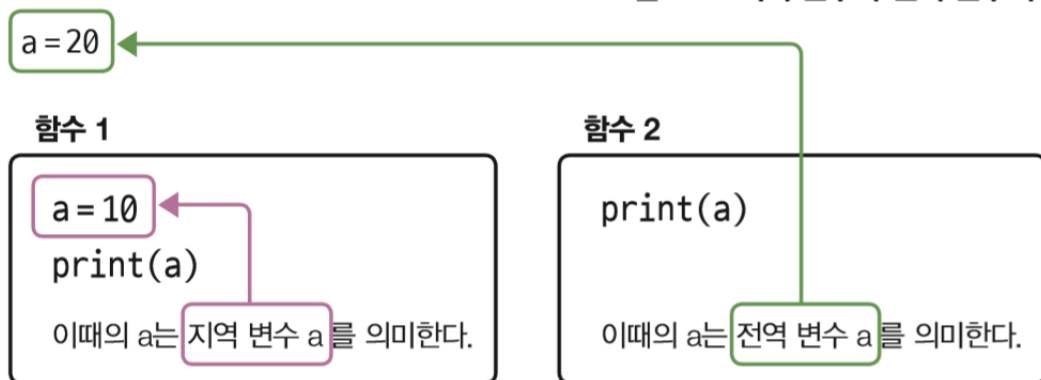


그림 9-7 지역 변수와 전역 변수의 공존

Section03 지역 변수, 전역 변수

Code09-06.py

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      a = 10    # 지역 변수
4      print("func1()에서 a값 %d" % a)
5
6  def func2() :
7      print("func2()에서 a값 %d" % a)
8
9  ## 전역 변수 선언 부분 ##
10 a = 20        # 전역 변수
11
12 ## 메인 코드 부분 ##
13 func1()
14 func2()
```

2~4행 : 한 func1() 함수 정의
3행 : a를 선언(지역 변수)
10행 : a는 선언(전역 변수)
13행 : func1() 함수 호출
14행 : func2() 함수 호출

출력 결과

```
func1()에서 a값 10
func2()에서 a값 20
```

Section03 지역 변수, 전역 변수

- 10행의 전역 변수가 없다면 7행은?

출력 결과

func1()에서 a의 값 10

Traceback (most recent call last):

```
File "C:/파이썬코드/09-06.py", line 14, in <module>
```

```
    func2()
```

```
File "C:/파이썬코드/09-06.py", line 7, in func2
```

```
    print("func2()에서 a값 %d" % a)
```

```
NameError: name 'a' is not defined
```


Section03 지역 변수, 전역 변수

■ 글로벌 예약어

Code09-07.py

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      global a    # 이 함수 안에서 a는 전역 변수
4      a = 10
5      print("func1()에서 a값 %d" % a)
6
7  def func2() :
8      print("func2()에서 a값 %d" % a)
9
10 ## 함수 변수 선언 부분 ##
11 a = 20          # 전역 변수
12
13 ## 메인 코드 부분 ##
14 func1()
15 func2()
```

출력 결과

```
func1()에서 a값 10
func2()에서 a값 10
```

Section03 지역 변수, 전역 변수

■ 글로벌 예약어

Code09-07.py

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      global a    # 이 함수 안에서 a는 전역 변수
4      a = 10
5      print("func1()에서 a값 %d" % a)
6
7  def func2() :
8      print("func2()에서 a값 %d" % a)
9
10 ## 함수 변수 선언 부분 ##
11 a = 20          # 전역 변수
12
13 ## 메인 코드 부분 ##
14 func1()
15 func2()
```

3행 : global 예약어로 a 변수를 전역 변수로 지정
4행 : 전역 변수 a값을 10으로 변경하므로 func1()과
func2() 함수에서 모두 전역 변수 a값을 10으로 출력

출력 결과

```
func1()에서 a값 10
func2()에서 a값 10
```

Section04 함수의 반환값과 매개변수

■ 함수의 반환값

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

- 반환값이 있는 함수

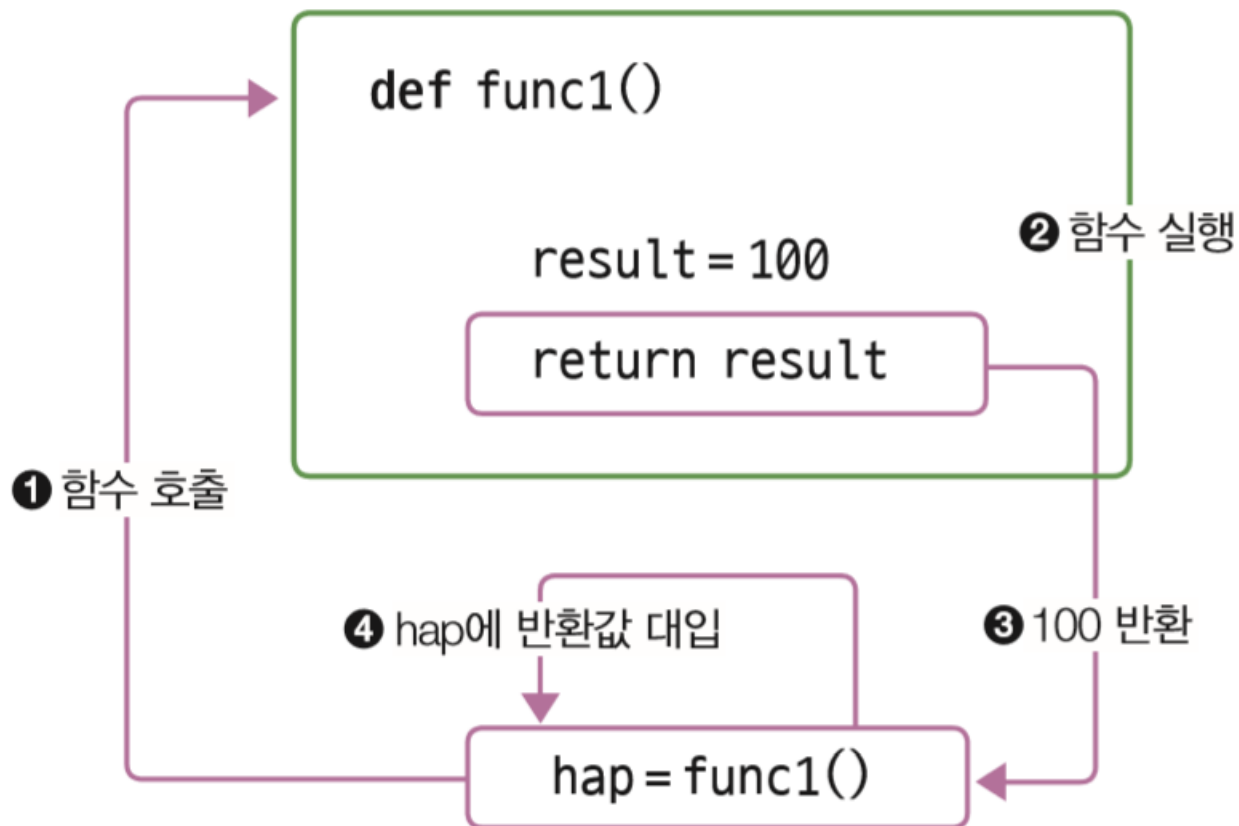


그림 9-8 값의 반환

Section04 함수의 반환값과 매개변수

- 반환값이 없는 함수

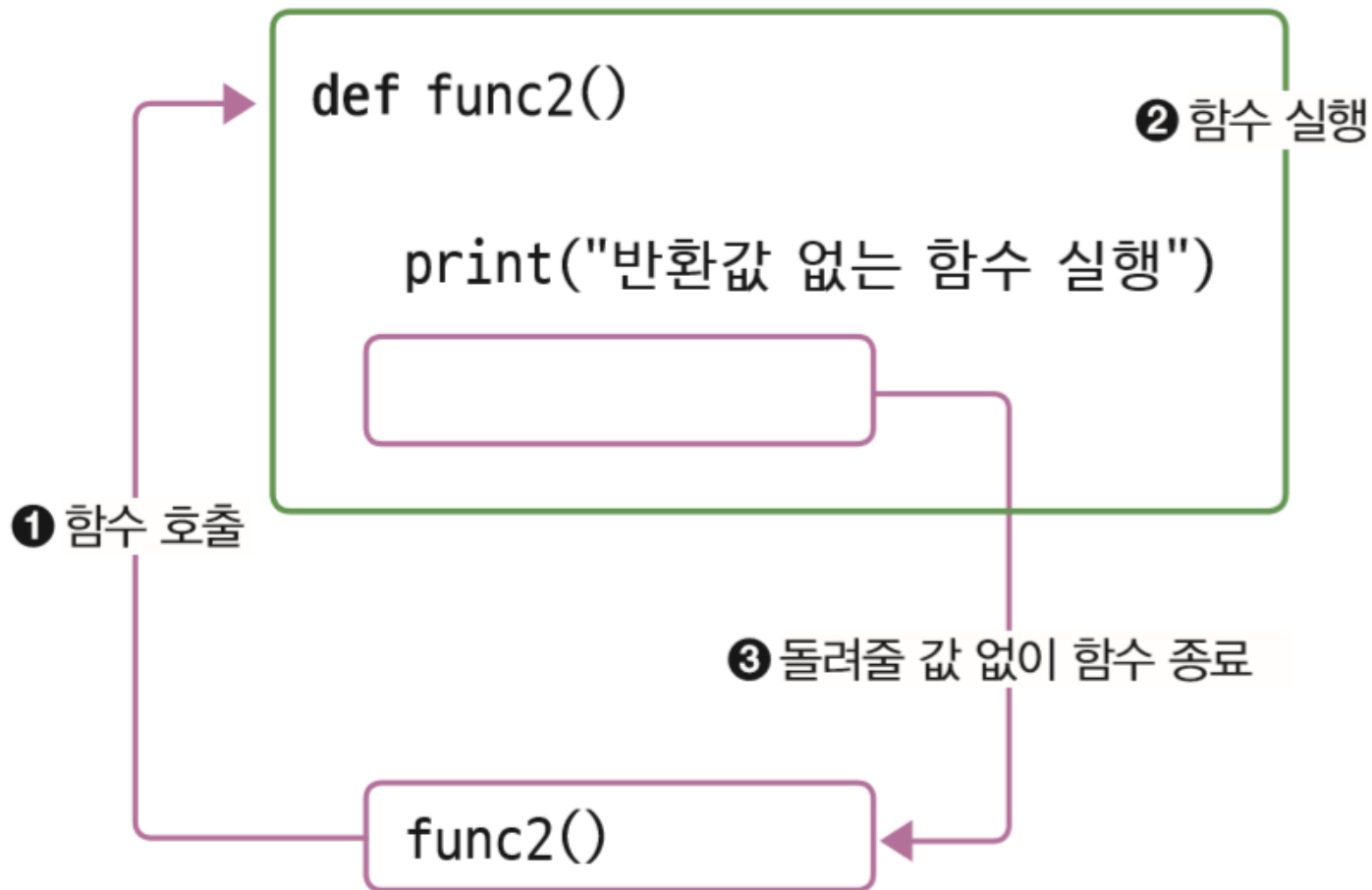


그림 9-9 반환값이 없는 함수의 작동

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 없음

출력 결과

func1()에서 돌려준 값 ==> 100
반환값이 없는 함수 실행

Section04 함수의 반환값과 매개변수

- 반환값이 여러 개인 함수

Code09-09.py

```
1  ## 함수 선언 부분 ##
2  def multi(v1, v2) :
3      retList = []      # 반환할 리스트
4      res1 = v1 + v2
5      res2 = v1 - v2
6      retList.append(res1)
7      retList.append(res2)
8      return retList
9
10 ## 전역 변수 선언 부분 ##
11 myList = []
12 hap, sub = 0, 0
13
14 ## 메인 코드 부분 ##
15 myList = multi(100, 200)
16 hap = myList[0]
17 sub = myList[1]
18 print("multi()에서 돌려준 값 ==> %d, %d" % (hap, sub))
```

3행 : 빈 리스트를 준비

6~7행 : 리스트에 값을 추가

8행 : 리스트 반환

15~17행 : 반환한 리스트의 값을 각 변수에 대입

출력 결과

multi()에서 돌려준 값 ==> 300, -100

Section04 함수의 반환값과 매개변수

- pass 예약어

```
def myFunc() :  
    pass
```

- True일 때 아무런 할 일이 없다고 빈 줄로 둘 때 오류 발생

```
if True :  
  
else :  
    print('거짓이네요')
```

- 오류 해결

```
if True :  
    pass  
else :  
    print('거짓이네요')
```

Section04 함수의 반환값과 매개변수

■ 함수의 매개변수 전달

- 매개변수의 개수를 지정해 전달하는 방법
 - 숫자 2개의 합과 숫자 3개의 합을 구 하는 코드

Code09-10.py

```
1  ## 함수 선언 부분 ##
2  def para2_func( v1, v2 ) :
3      result = 0
4      result = v1 + v2      2~5행은매개변수를 2개, 7~10행은 매개변수를 3개 받아 합계를
5      return result        반환하는 함수 정의
6
7  def para3_func( v1, v2, v3 ) :
8      result = 0
9      result = v1 + v2 + v3
10     return result
11
12  ## 전역 변수 선언 부분 ##
13  hap = 0
```


Section04 함수의 반환값과 매개변수

```
14
15  ## 메인 코드 부분 ##
16  hap = para2_func(10, 20)
17  print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
18  hap = para3_func(10, 20, 30)
19  print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
```

16~19행 : 각 함수를 호출하고 결과 출력

출력 결과

매개변수가 2개인 함수를 호출한 결과 ==> 30

매개변수가 3개인 함수를 호출한 결과 ==> 60

Section04 함수의 반환값과 매개변수

- 매개변수에 기본값을 설정해 놓고 전달하는 방법

Code09-11.py

```
1  ## 함수 선언 부분 ##
2  def para_func( v1, v2, v3 = 0 ) :
3      result = 0
4      result = v1 + v2 + v3
5      return result
6
7  ## 전역 변수 선언 부분 ##
8  hap = 0
9
10 ## 메인 코드 부분 ##
11 hap = para_func(10, 20)
12 print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
13 hap = para_func(10, 20, 30)
14 print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
```

Section04 함수의 반환값과 매개변수

SELF STUDY 9-3

Code09-11.py에서 2에서 10개까지 몇 개를 매개변수로 사용하든지 합계를 구하도록 para_func() 함수를 수정해 보자.

출력 결과

매개변수가 2개인 함수를 호출한 결과 ==> 30

매개변수가 10개인 함수를 호출한 결과 ==> 550

Section04 함수의 반환값과 매개변수

■ [프로그램 1]의 완성

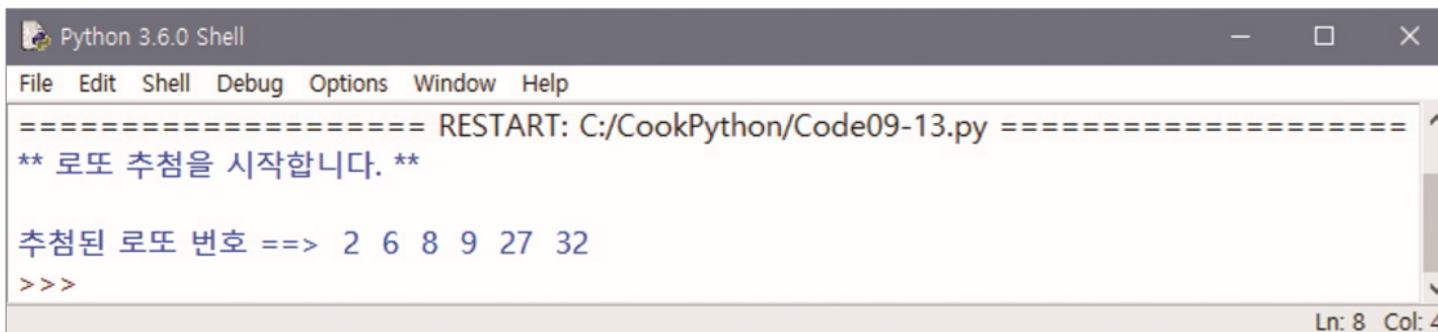
Code09-13.py

```
1 import random
2
3 ## 함수 선언 부분 ##
4 def getNumber() :
5     return random.randrange(1, 46)
6
7 ## 전역 변수 선언 부분 ##
8 lotto = []
9 num = 0
10
11 ## 메인 코드 부분 ##
12 print("** 로또 추첨을 시작합니다. ** \n");
13
14 while True :
15     num = getNumber()
```

4~5행 : getNumber() 함수 정의
5행 : random.randrange(시작값, 끝값) 함수는 시작값~끝값-1 중에 임의의 숫자 하나를 추출. 이 함수를 사용하려고 1행에 import random을 입력
8~9행 : 추첨된 로또 숫자를 저장할 lotto 리스트와 추첨된 숫자를 임시로 저장할 num 변수 준비

Section04 함수의 반환값과 매개변수

```
16
17     if lotto.count(num) == 0 :
18         lotto.append(num)    14~21행 : 무한 반복
19                               17~18행 : 이미 뽑힌 숫자가 lotto[] 리스트에 들어 있지 않아야
20     if len(lotto) >= 6 :    lotto.append(num) 함수로 추가
21         break              24행 : 뽑힌 숫자 6개를 lotto.sort()로 정렬
                               25~26행 : 출력
22
23 print("추첨된 로또 번호 ==> ", end = '')
24 lotto.sort()
25 for i in range(0, 6) :
26     print("%d " % lotto[i], end = '')
```



The screenshot shows a Python 3.6.0 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The command prompt displays the following text:

```
===== RESTART: C:/CookPython/Code09-13.py =====
** 로또 추첨을 시작합니다. **

추첨된 로또 번호 ==> 2 6 8 9 27 32
>>>
```

The status bar at the bottom right indicates "Ln: 8 Col: 4".

■ 모듈 : 함수의 집합

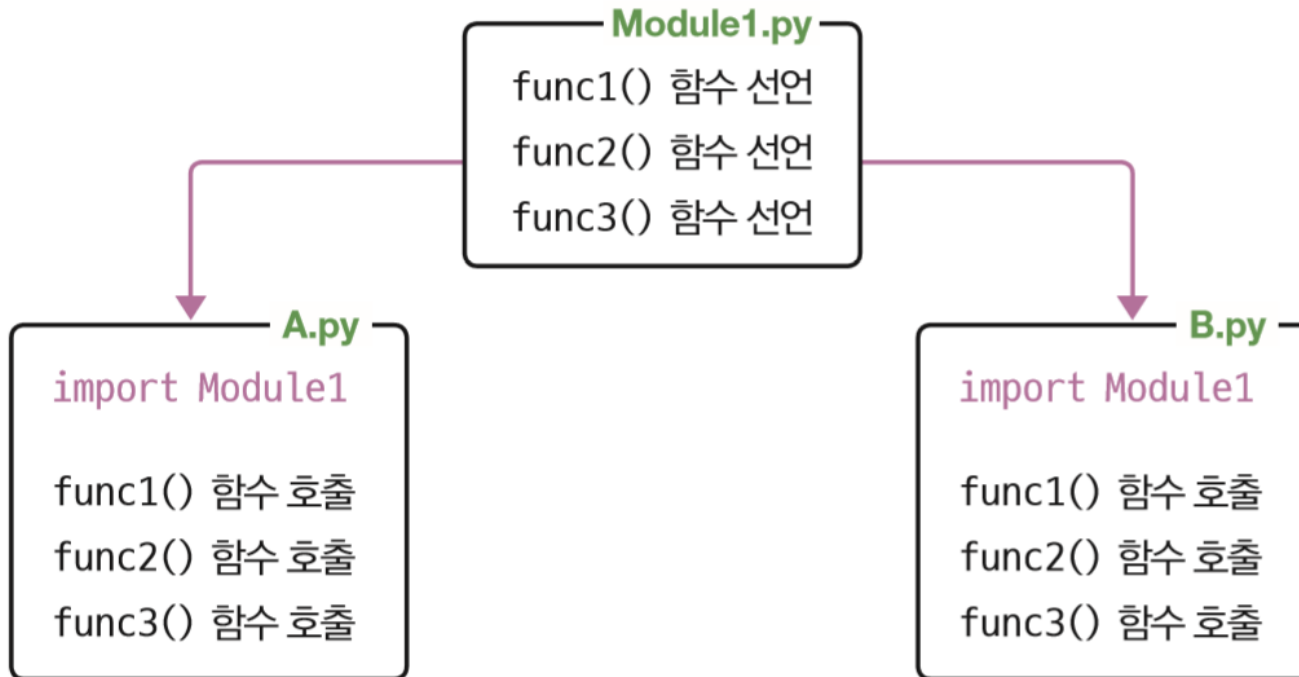


그림 9-10 모듈 사용 예

■ 모듈의 생성과 사용

Module1.py

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      print("Module1.py의 func1()이 호출됨.")
4
5  def func2() :
6      print("Module1.py의 func2()가 호출됨.")
7
8  def func3() :
9      print("Module1.py의 func3()이 호출됨.")
```

Section05 모듈

A.py

```
1 import Module1
2
3 ## 메인 코드 부분 ##
4 Module1.func1()
5 Module1.func2()
6 Module1.func3()
```

출력 결과

Module1.py의 func1()이 호출됨.

Module1.py의 func2()가 호출됨.

Module1.py의 func3()이 호출됨.

- 모듈명을 생략하고 함수명만 쓸 때 1행 형식

```
from 모듈명 import 함수1, 함수2, 함수3
또는
from 모듈명 import *
```


Section05 모듈

- 4~6행 함수명으로만 호출

B.py

```
1 from Module1 import func1, func2, func3    # 또는 from Module1 import *
2
3 ## 메인 코드 부분 ##
4 func1()
5 func2()
6 func3()
```

■ 모듈의 종류

- 표준 모듈, 사용자 정의 모듈, 서드 파티 모듈로 구분
- 표준 모듈 : 파이썬에서 제공하는 모듈
- 사용자 정의 모듈 : 직접 만들어서 사용하는 모듈
- 서드 파티(3rd Party) 모듈 : 파이썬이 아닌 외부 회사나 단체에서 제공하는 모듈
 - 파이썬 표준 모듈이 모든 기능을 제공 않음
 - 서드 파티 모듈 덕분에 파이썬에서도 고급 프로그래밍 가능
 - 게임 개발 기능이 있는 pyGame, 윈도우창을 제공 하는 PyGTK, 데이터베이스 기능의 SQLAlchemy 등

Section05 모듈

- 파이썬에서 제공하는 표준 모듈의 목록을 일부 확인

```
import sys
print(sys.builtin_module_names)
```

출력 결과

```
('_ast', '_bisect', '_codecs', '_codecs_cn', '_codecs_hk', '_codecs_iso2022', '_codecs_
jp', '_codecs_kr', '_codecs_tw', '_collections', '_csv', '_datetime', '_functools',
'_heapq', '_imp', '_io', '_json', '_locale', '_lsprof', '_md5', '_multibytecodec',
'_opcode', '_operator', '_pickle', '_random', '_sha1', '_sha256', '_sha512', '_signal',
'_sre', '_stat', '_string', '_struct', '_symtable', '_thread', '_tracemalloc',
'_warnings', '_weakref', '_winapi', 'array', 'atexit', 'audioop', 'binascii', 'builtins',
'cmath', 'errno', 'faulthandler', 'gc', 'itertools', 'marshal', 'math', 'mmap', 'msvcrt',
'nt', 'parser', 'sys', 'time', 'winreg', 'xxsubtype', 'zipimport', 'zlib')
```

Section05 모듈

- 파이썬에서 제공하는 표준 모듈의 목록을 일부 확인

```
import sys
print(sys.builtin_module_names)
```

출력 결과

```
('_ast', '_bisect', '_codecs', '_codecs_cn', '_codecs_hk', '_codecs_iso2022', '_codecs_
jp', '_codecs_kr', '_codecs_tw', '_collections', '_csv', '_datetime', '_functools',
'_heapq', '_imp', '_io', '_json', '_locale', '_lsprof', '_md5', '_multibytecodec',
'_opcode', '_operator', '_pickle', '_random', '_sha1', '_sha256', '_sha512', '_signal',
'_sre', '_stat', '_string', '_struct', '_symtable', '_thread', '_tracemalloc',
'_warnings', '_weakref', '_winapi', 'array', 'atexit', 'audioop', 'binascii', 'builtins',
'cmath', 'errno', 'faulthandler', 'gc', 'itertools', 'marshal', 'math', 'mmap', 'msvcrt',
'nt', 'parser', 'sys', 'time', 'winreg', 'xxsubtype', 'zipimport', 'zlib')
```

Tip • `dir(__builtins__)` 명령어로도 제공하는 모듈과 예약어 확인

Section05 모듈

- 수학 계산 모듈인 math 모듈이 제공하는 함수의 목록 보기

```
import math  
dir(math)
```

출력 결과

```
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',  
'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e',  
'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',  
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp',  
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin',  
'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

■ [프로그램 2]의 완성

myTurtle.py

```
1 import random
2 from tkinter.simpledialog import *
3
4 def getString() :
5     retStr = ''
6     retStr = askstring('문자열 입력', '거북이 쓸 문자열을 입력')
7     return retStr
8
9 def getRGB() :
10     r, g, b = 0, 0, 0
11     r = random.random()
12     g = random.random()
13     b = random.random()
14     return(r, g, b)
15
16 def getXYAS(sw, sh) :
17     x, y, angle, size = 0, 0, 0, 0
18     x = random.randrange(-sw / 2, sw / 2)
19     y = random.randrange(-sh / 2, sh / 2)
```

4~7행 : 문자열을 입력받아 반환하는 함수 생성
9~14행 : 무작위로 RGB 색상 추출해서 튜플 반환하는 함수 생성
16~22행 : x, y, 각도, 크기를 무작위로 추출해서 리스트로 묶어 반환하는 함수 생성

Section05 모듈

```
20     angle = random.randrange(0, 360)
21     size = random.randrange(10, 50)
22     return [x, y, angle, size]
```

Section05 모듈

Code09-14.py

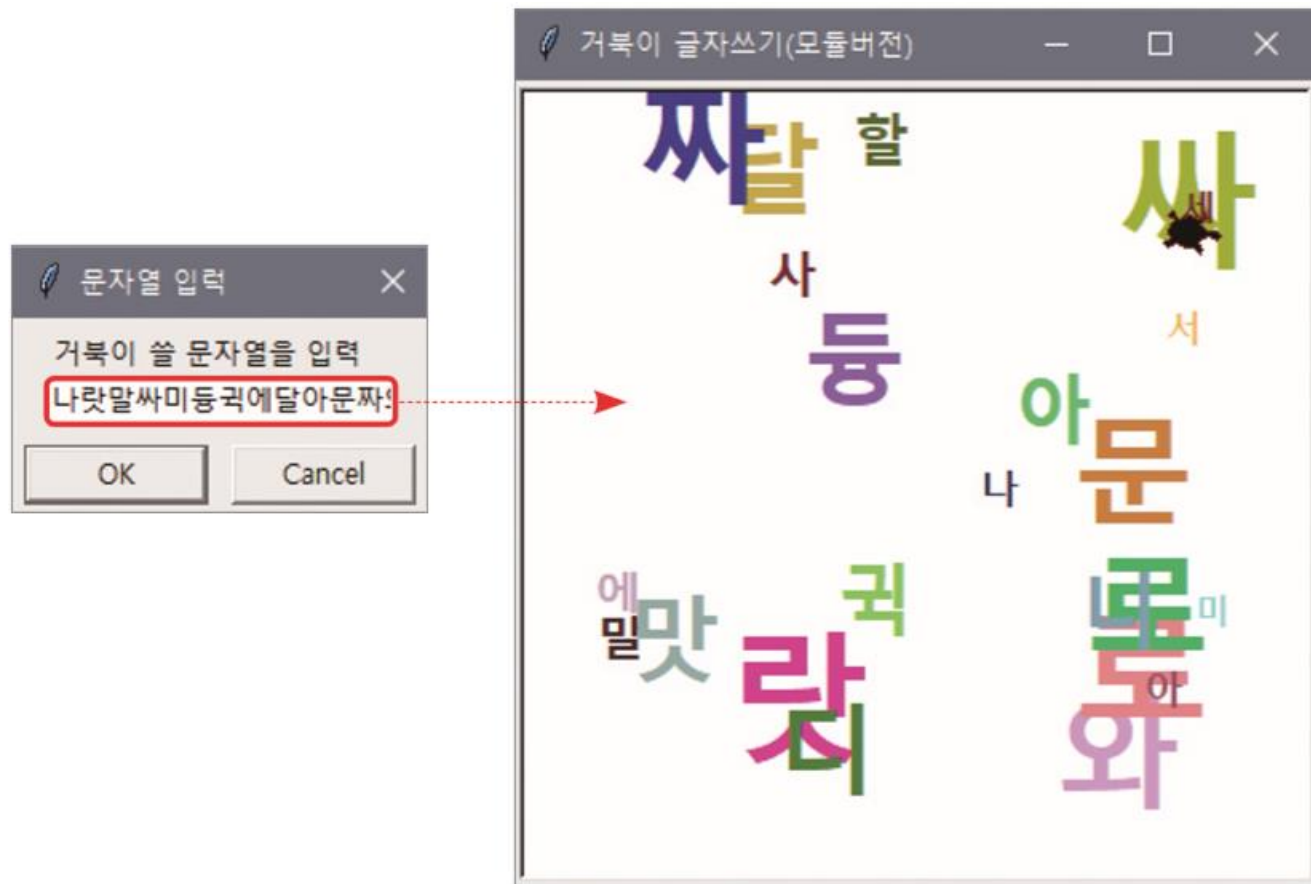
```
1  from myTurtle import *
2  import turtle
3                                     1행 : myTurtle 모듈 импорт
4  ## 전역 변수 선언 부분 ##
5  inStr = ''
6  swidth, sheight = 300, 300
7  tX, tY, tAngle, tSize = [0] * 4
8
9  ## 메인 코드 부분 ##
10 turtle.title('거북이 글자쓰기(모듈버전)')
11 turtle.shape('turtle')
12 turtle.setup(width = swidth + 50, height = sheight + 50)
13 turtle.screensize(swidth, sheight)
14 turtle.penup()
15 turtle.speed(5)
16
```


Section05 모듈

```
17 inStr = getString()
18
19 for ch in inStr :
20
21     tX, tY, tAngle, txtSize = getXYAS(swidth, sheight)
22     r, g, b = getRGB()
23
24     turtle.goto(tX, tY)
25     turtle.left(tAngle)
26
27     turtle.pencolor((r, g, b))
28     turtle.write(ch, font = ('맑은고딕', txtSize, 'bold'))
29
30 turtle.done()
```

17행과 21~22행 : 모듈의 함수를 사용

Section05 모듈



Section06 함수의 심화 내용

■ 패키지

- 모듈이 하나의 *.py 파일 안에 함수가 여러 개 들어 있는 것이라면, 패키지(Package)는 여러 모듈을 모아 놓은 것으로 폴더의 형태로 나타냄
- 모듈을 주제별로 분리할 때 주로 사용

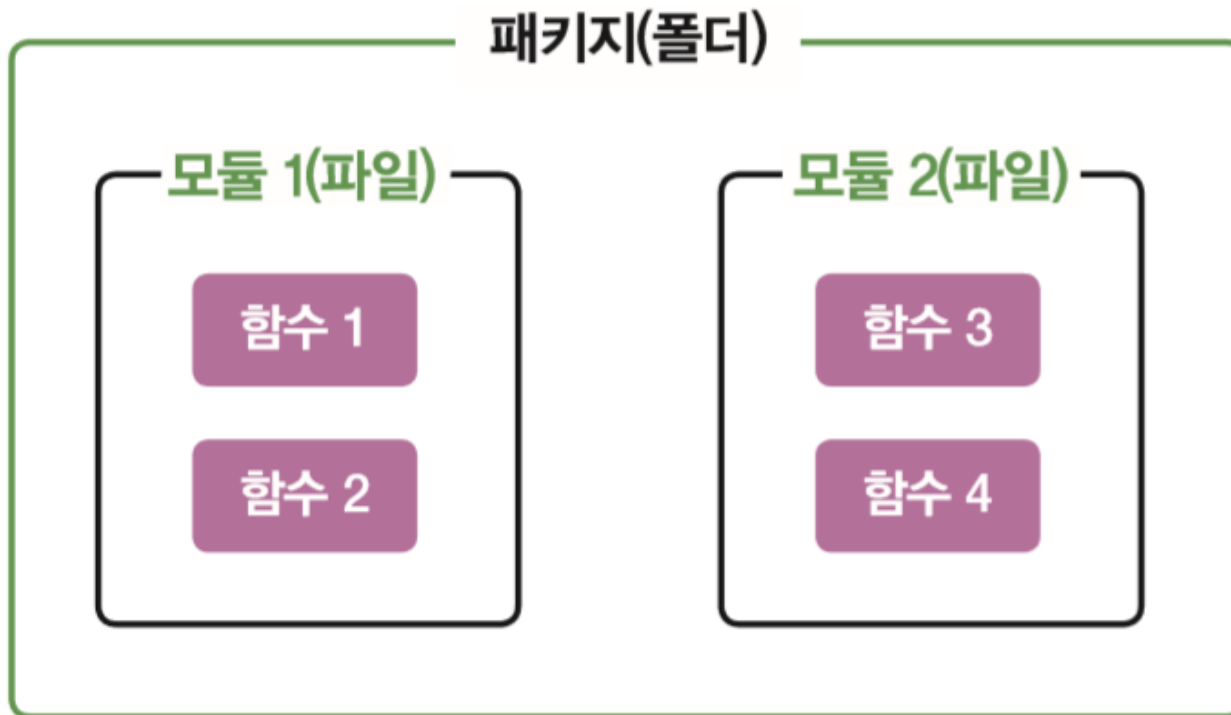


그림 9-11 패키지의 개념

Section06 함수의 심화 내용

- 임포트 형식

```
from 패키지명.모듈명 import 함수명
```

- [그림 9-11]과 같은 형태로 구현 임포트

```
from package.Module1 import *
```



Thank You
