컴퓨팅 사고력을 키우는 SW 교육

파이썬

Chapter 07
함수 기초

01 함수에 대해 알아봅시다

02 지역변수와 전역변수의 차이는?

03 함수의 반환값과 매개변수를 알아봅시다

04 모듈에 대해 알아봅시다

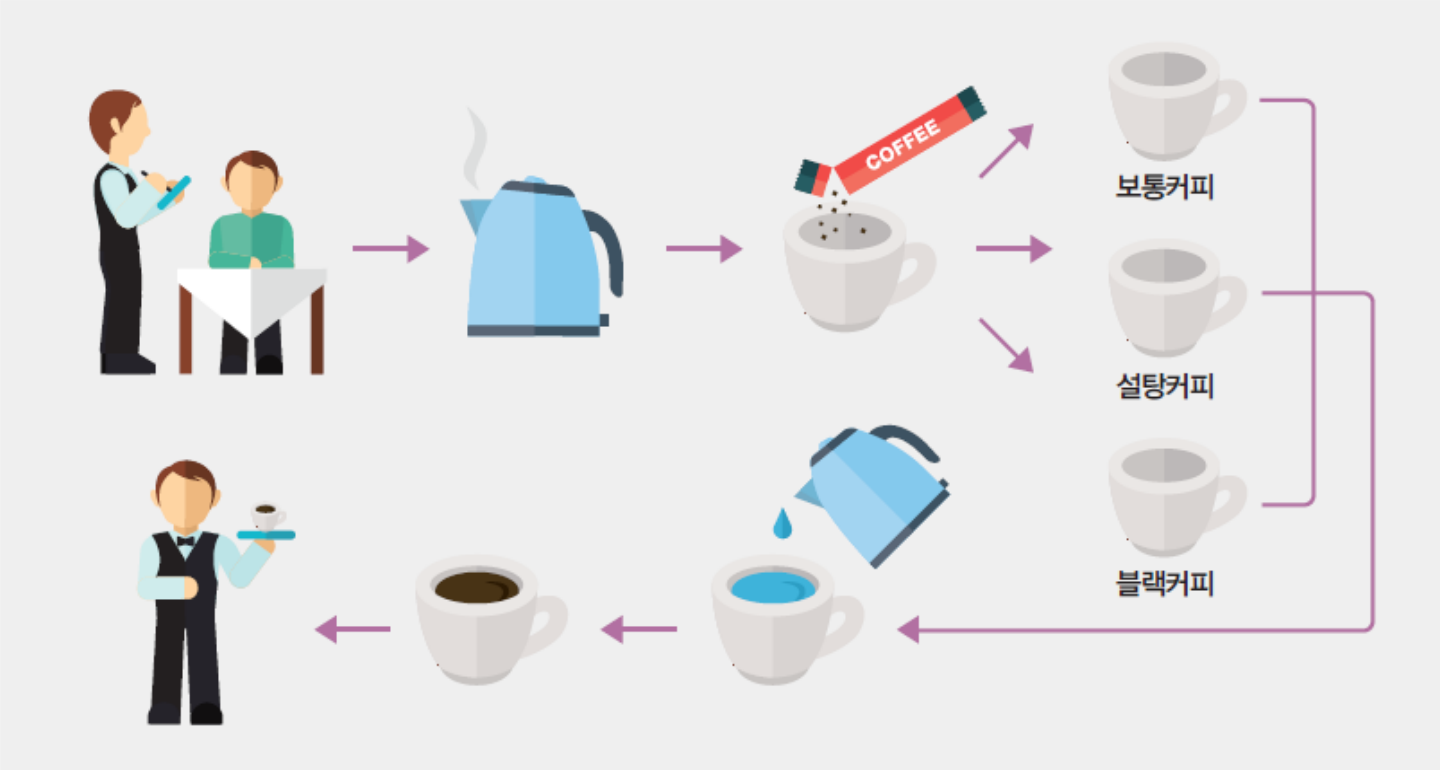
Section 01 함수에 대해 알아봅시다

함수에 대해 알아봅시다(1)

■ 함수의 개념

- 함수는 파이썬 프로그램 자체에서 제공하지만, 사용자가 직접 만들어서 사용하기도 함
- 자판기 없이 커피를 제공하는 프로그램

그림 9-3
직접 커피를 타는 과정



함수에 대해 알아봅시다(2)

소스코드 9-1
(파일명 : 09-01.py)

```
coffee=0

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 저어서 녹인다");
print()
print("손님~ 커피 여기 있습니다.");
```

함수에 대해 알아봅시다(3)

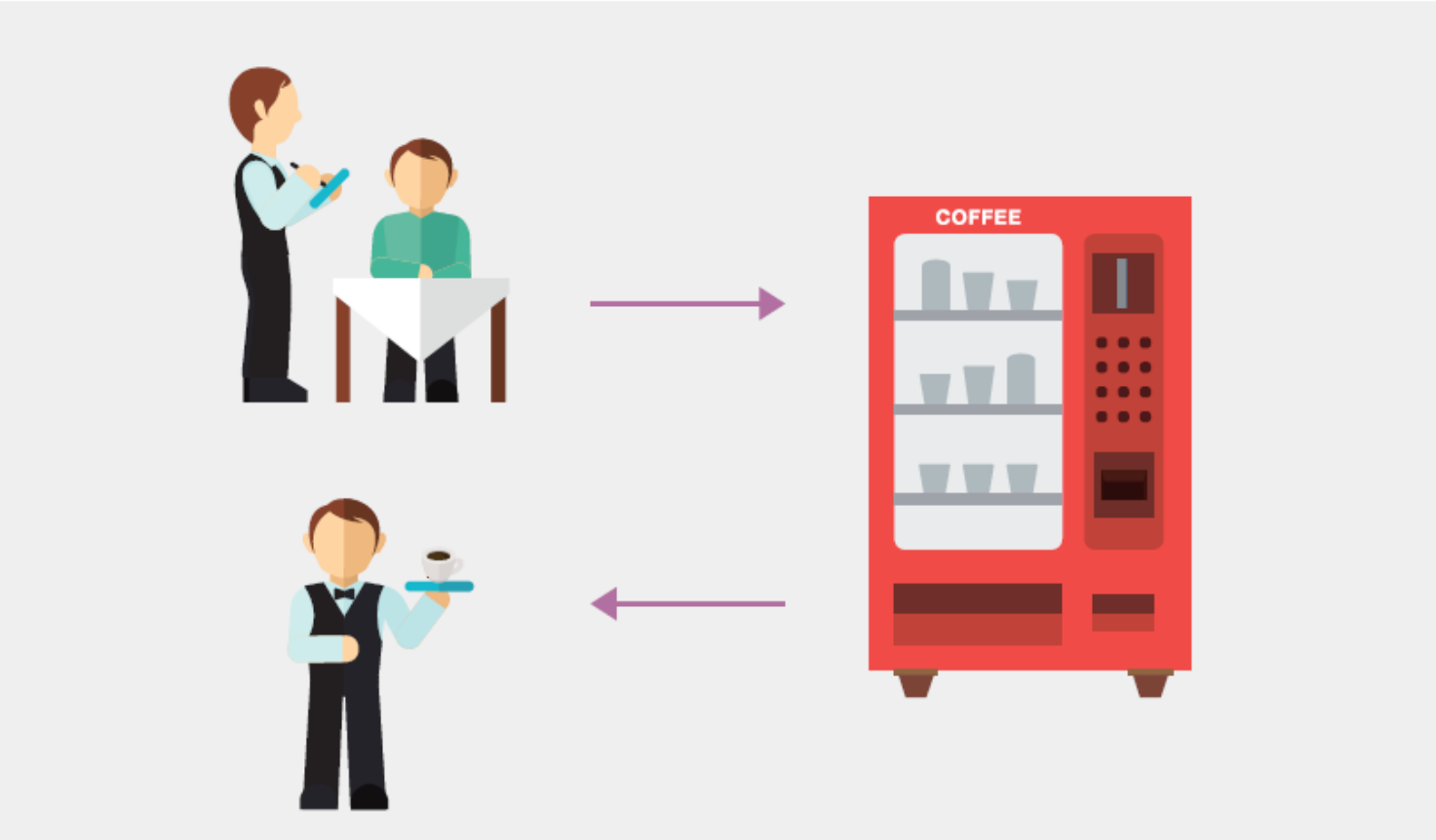
출력 결과

```
어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2  ← 사용자가 입력한 값
# 1. 뜨거운 물을 준비한다.
# 2. 종이컵을 준비한다.
# 3. 설탕커피를 탄다
# 4. 물을 붓는다
# 5. 스푼으로 저어서 녹인다
손님~ 커피 여기 있습니다.
```

- 손님이 세 명 연속 올 경우 [소스코드 9-1]의 3행~21행을 두 번 더 반복 해야 함.
  이럴 경우 소스가 길어지기 때문에 커피자판기 프로그램을 사용

함수에 대해 알아봅시다(4)

그림 9-4
커피 자판기 사용



함수에 대해 알아봅시다(5)

소스코드 9-2
(파일명 : 09-02.py)

```
1  ## 변수 선언 부분
2  coffee=0
3
4  ## 함수 정의 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")
16     else :
17        print("# 3. (자동으로) 아무거나 탄다\n")
```

함수에 대해 알아봅시다(6)

```
18
19      print("# 4. (자동으로) 물을 붓는다");
20      print("# 5. (자동으로) 스푼으로 저어서 녹인다");
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.");
```

출력 결과

```
어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2  ← 사용자가 입력한 값
# 1. (자동으로) 뜨거운 물을 준비한다.
# 2. (자동으로) 종이컵을 준비한다.
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다
손님~ 커피 여기 있습니다.
```

함수에 대해 알아봅시다(7)

- 커피를 타는 [소스코드 9-2] 자판기함수를 이용하여 손님 대접하기

소스코드 9-3
(파일명 : 09-03.py)

```
1  ## 변수 선언 부분
2  coffee=0
3
4  ## 함수 정의 부분
5  def coffee_machine(button) :
…     ~~~ (중간 생략) [소스코드 9-1]의 6~21행과 동일 ~~~
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
```

함수에 대해 알아봅시다(8)

```
32  coffee=int(input("C손님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) "))
33  coffee_machine(coffee)
34  print("C손님~ 커피 여기 있습니다.")
```

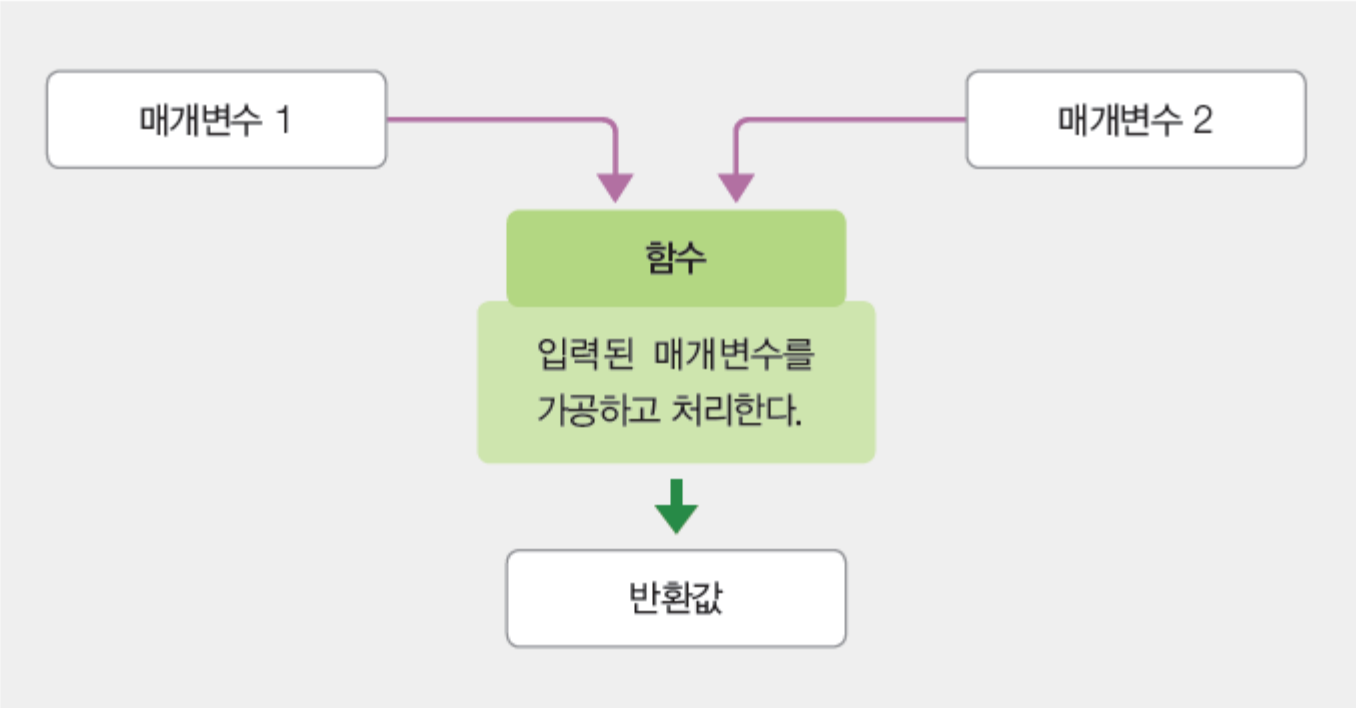
출력 결과

```
A손님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 2  ← 사용자가 입력한 값
# 1. (자동으로) 뜨거운 물을 준비한다.
# 2. (자동으로) 종이컵을 준비한다.
# 3. (자동으로) 설탕커피를 탄다
# 4. (자동으로) 물을 붓는다
# 5. (자동으로) 스푼으로 저어서 녹인다
A 손님~ 커피 여기 있습니다.
B손님, 어떤 커피 드릴까요? (1:보통, 2:설탕, 3:블랙) 3  ← 사용자가 입력한 값
… 이하 생략 …
```

함수에 대해 알아봅시다(9)

- **함수의 모양과 활용**

그림 9-5
함수의 형태



- 함수는 매개변수(Parameter)를 입력 받은 후 그 매개변수를 가공 및 처리한 후에 반환값을 돌려줌

함수에 대해 알아봅시다(10)

- 두 정수를 입력 받아 두 정수의 합계를 반환하는 plus() 함수 만들기

소스코드 9-4
(파일명 : 09-04.py)

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

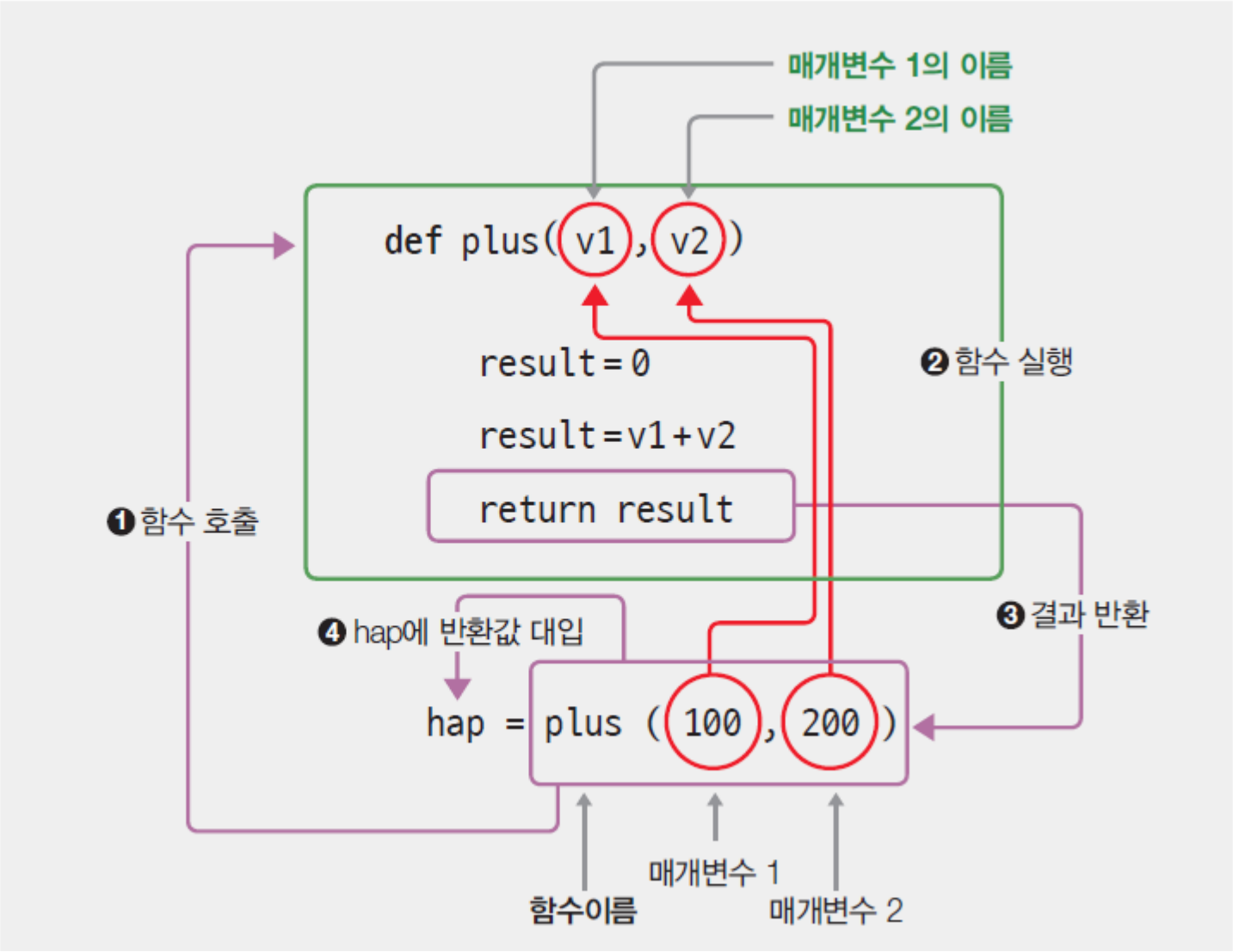
출력 결과

```
100과 200의 plus() 함수 결과는 300
```

- 2행~5행에 plus() 함수를 정의하였으나 먼저 실행되지 않음. 11행에서 함수를 호출하면 그때 실행됨

함수에 대해 알아봅시다(11)

그림 9-6
plus( ) 함수의 형태와 호출 순서



함수에 대해 알아봅시다(12)

① 함수 호출

plus() 함수를 호출할 때는 plus(숫자 1, 숫자 2)와 같은 형식으로 호출

② 함수 실행

v1과 v2를 더해 result에 대입한 후, return result 문장에 의해 이 함수를 호출했던 곳(=11행)으로 돌아감

③ 결과 반환

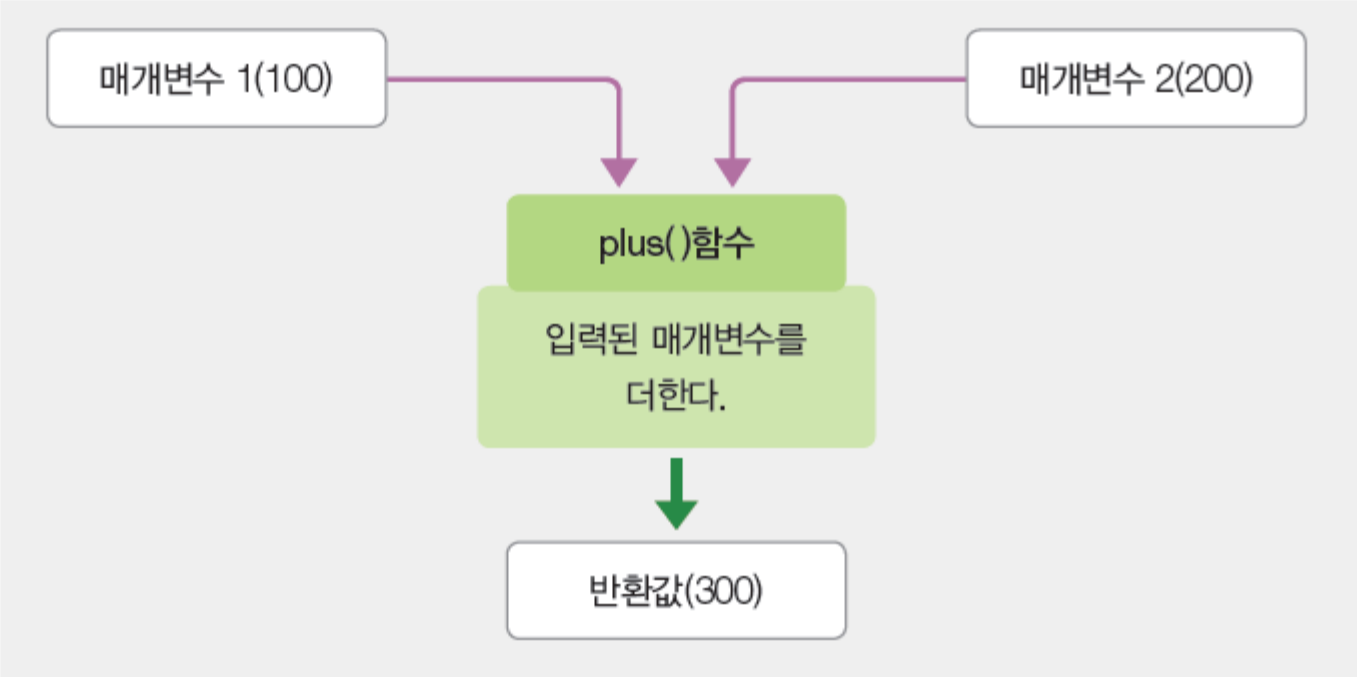
함수를 실행하여 얻은 result값(300)을 plus() 함수를 호출했던 곳으로 돌려줌

④ hap에 반환값 대입

반환된 값 300을 변수 hap에 대입

함수에 대해 알아봅시다(13)

그림 9-7
간단하게 표현한
plus( ) 함수의 호출



함수에 대해 알아봅시다(14)

- 입력한 두 숫자의 덧셈, 뺄셈, 곱셈, 나눗셈을 하는 계산기 함수

소스코드 9-5
(파일명 : 09-05.py)

```
1  ## 함수 정의 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
```

함수에 대해 알아봅시다(15)

```
15  ## 변수 선언 부분
16  res=0
17  var1, var2, oper=0, 0, ""
18
19  ## 메인 코드 부분
20  oper=input("계산 입력 ( +, -, *, / ) : ")
21  var1=int(input("첫 번째 숫자 입력 : "))
22  var2=int(input("두 번째 숫자 입력 : "))
23
24  res=calc(var1, var2, oper)
25
26  print("## 계산기 : %d %s %d=%d" % (var1, oper, var2, res))
```

출력 결과

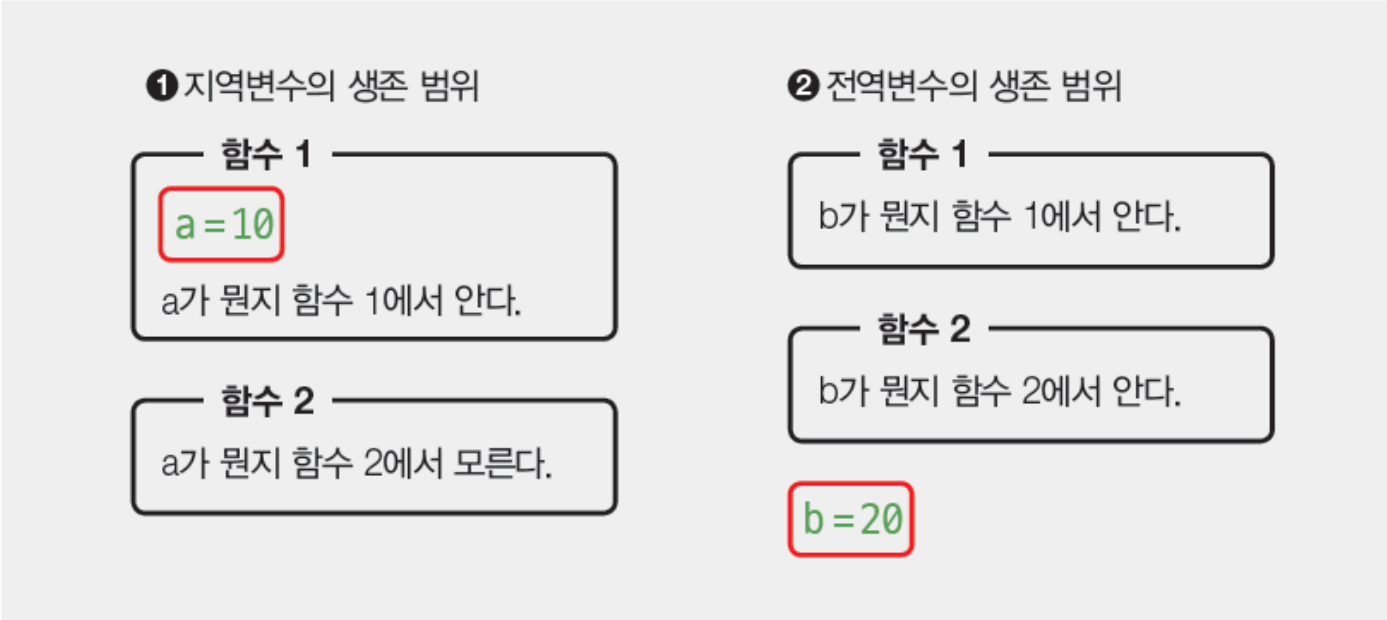
```
계산 입력 ( +, -, *, / ) : *  ← 사용자가 입력한 값
첫 번째 숫자 입력 : 7  ← 사용자가 입력한 값
두 번째 숫자 입력 : 8  ← 사용자가 입력한 값
## 계산기 : 7*8=56
```

Section 02 지역변수와 전역변수의 차이는?

지역변수와 전역변수의 차이는?(1)

- **지역변수와 전역변수의 이해**
  - 지역변수는 한정된 지역(Local)에서만 사용되는 변수, 전역변수는 프로그램 전체(Global)에서 사용되는 변수

그림 9-8
지역변수와 전역변수의 생존 범위



① 에서 a는 현재 함수 1 안에 선언. 즉 a는 함수 1 안에서만 사용될 수 있음

② b는 함수(함수 1, 함수 2) 안이 아니라 바깥에 선언, 모든 함수에서 b의 존재를 안다.

지역변수와 전역변수의 차이는?(2)

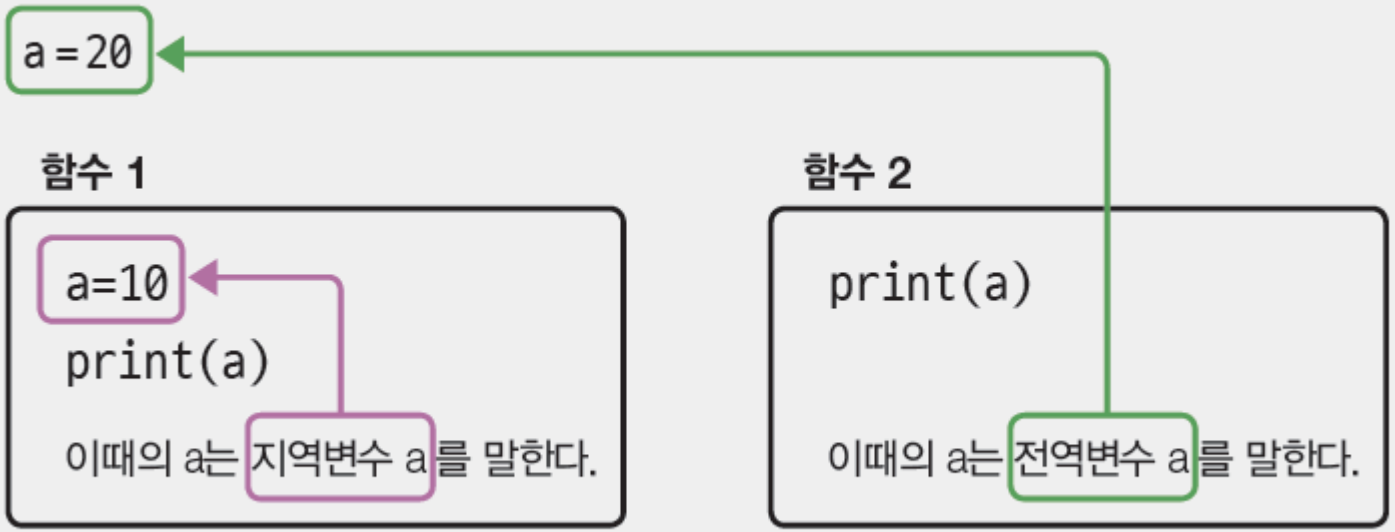


그림 9-9
지역변수와 전역변수의 공존

지역변수와 전역변수의 차이는?(3)

소스코드 9-6
(파일명 : 09-06.py)

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

출력 결과

```
func1()에서 a의 값 10
func2()에서 a의 값 20
```

- 3행의 a는 func1( ) 함수 안에서 선언했으므로 지역변수, 10행의 a는 함수 밖에서 선언했으므로 전역변수

지역변수와 전역변수의 차이는?(4)

- [소스코드 9-6]에서 10행의 전역변수가 없다면 7행은 어떻게 될까?

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 a의 값 %d" % a)
NameError: name 'a' is not defined
```

func1( )에는 a가 있으므로 잘 출력. func2( )에는 a가 없으므로 오류가 발생

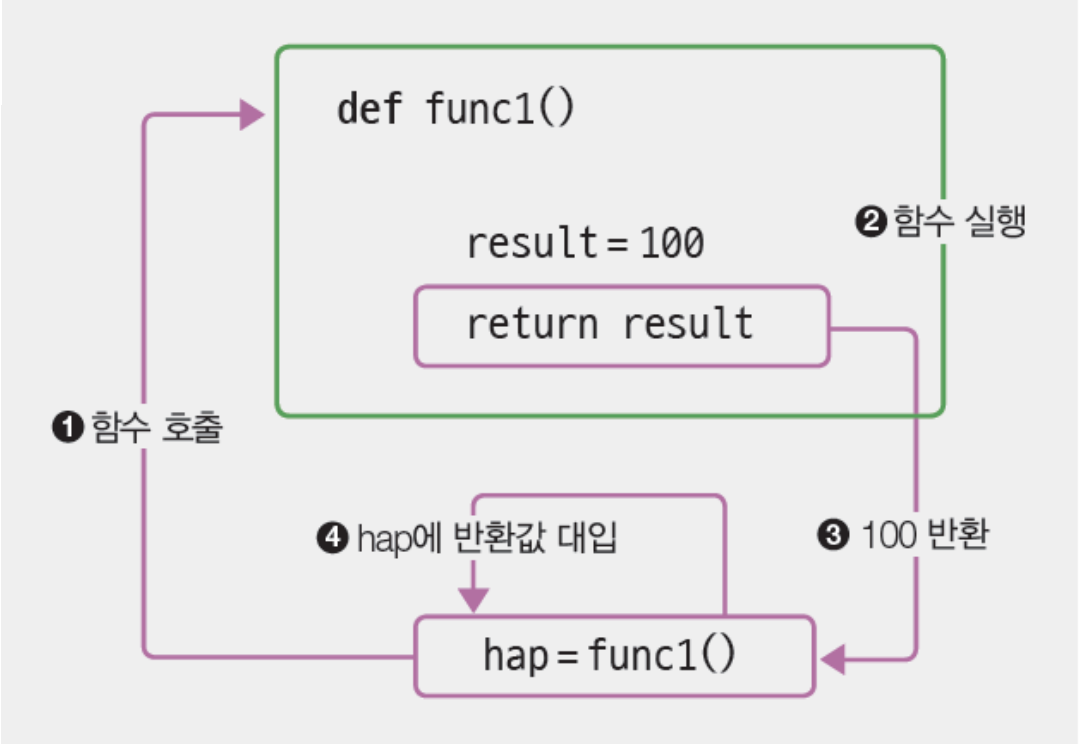
결국 func2( )에서는 func1( )의 a(지역변수)가 있는지 전혀 인식하지 못함.

Section 03 함수의 반환값과 매개변수를 알아봅시다

함수의 반환값과 매개변수를 알아봅시다(1)

- **반환값 유무에 따른 함수 구분**
  - 반환값은 '리턴값'이라 함. 매개변수(Parameter)는 '파라미터'라고 부르기도 함
  - 반환값이 있는 함수

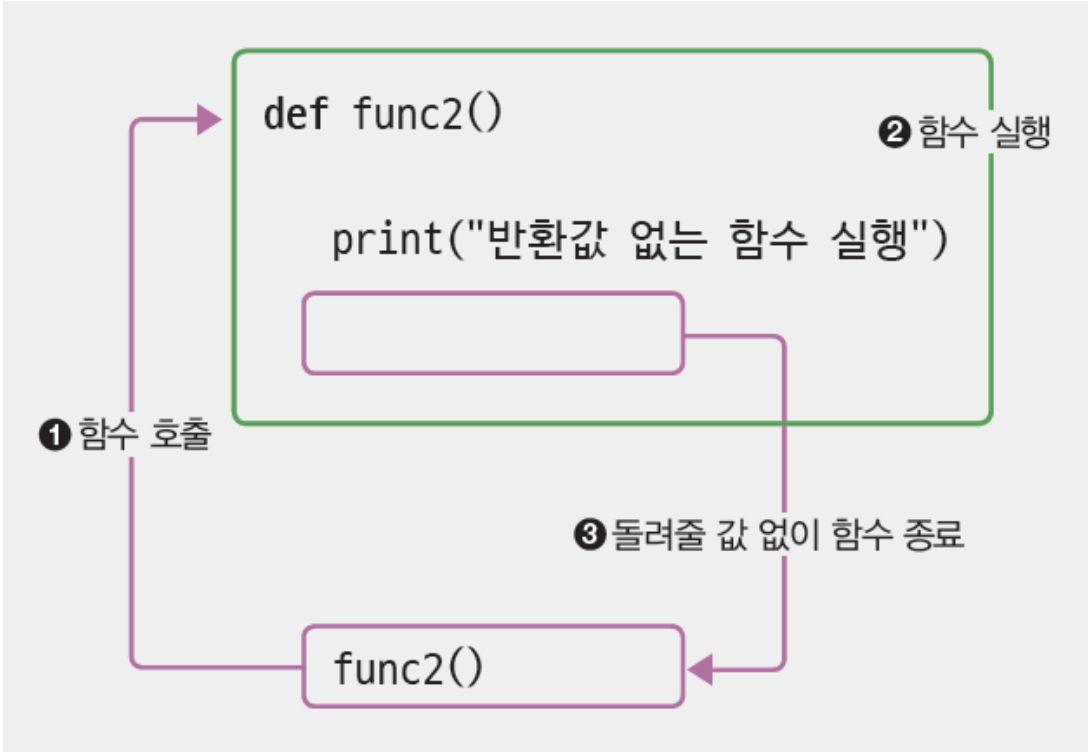
그림 9-10
반환값이 있는 함수의 작동



함수의 반환값과 매개변수를 알아봅시다(2)

- 반환값이 없는 함수
  - return문 생략. 또는 반환값 없이 return만 씀.

그림 9-11
반환값이 없는 함수의 작동



함수의 반환값과 매개변수를 알아봅시다(3)

소스코드 9-7
(파일명 : 09-07.py)

```
## 함수 정의 부분
def func1() :
    result = 100
    return result

def func2() :
    print("반환값 없는 함수 실행")

## 변수 선언 부분
hap = 0

## 메인 코드 부분
hap = func1()
print("func1()에서 돌려준 값 ==> %d" % hap)
func2()
```

출력 결과

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

함수의 반환값과 매개변수를 알아봅시다(4)

매개변수 전달 방법

- 매개변수의 개수를 정해놓는 방법

소스코드 9-8
(파일명 : 09-08.py)

```
## 함수 정의 부분
def para2_func(v1, v2) :
    result = 0
    result = v1 + v2
    return result

def para3_func(v1, v2, v3) :
    result = 0
    result = v1 + v2 + v3
    return result

## 변수 선언 부분
hap = 0

## 메인 코드 부분
hap = para2_func(10, 20)
print("매개변수 2개 함수 호출 결과==>%d" % hap)
```

함수의 반환값과 매개변수를 알아봅시다(5)

```
18  hap=para3_func(10, 20, 30)
19  print("매개변수 3개 함수 호출 결과==>%d" % hap)
```

출력 결과

```
매개변수 2개 함수 호출 결과==>30
매개변수 3개 함수 호출 결과==>60
```

함수의 반환값과 매개변수를 알아봅시다(6)

- 매개변수에 기본값을 설정해놓는 방법

소스코드 9-9
(파일명 : 09-09.py)

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

출력 결과

[소스코드 9-8]과 동일

함수의 반환값과 매개변수를 알아봅시다(7)

- 매개변수의 숫자를 정해놓지 않는 방법 - 가변 매개변수(Arbitrary Argument List) 방식

소스코드 9-10
(파일명 : 09-10.py)

```
## 함수 정의 부분
def para_func(*para) :
    result = 0
    for num in para :
        result = result + num

    return result

## 변수 선언 부분
hap = 0

## 메인 코드 부분
hap = para_func(10, 20)
print("매개변수 2개 함수 호출 결과 ==> %d" % hap)
hap = para_func(10, 20, 30)
print("매개변수 3개 함수 호출 결과 ==> %d" % hap)
```

출력 결과

```
[소스코드 9-8]과 동일
```

함수의 반환값과 매개변수를 알아봅시다(8)

- 2행 : *para로 매개변수를 설정.
- 13행 : 호출한 매개변수는 (10, 20)
- 15행 : 호출한 매개변수는 (10, 20, 30) 형식의 튜플로 전달됨
- 4행에서 넘겨 받은 매개변수를 for문으로 하나씩 추출해서 5행에서 result에 누적. 예로 (10, 20, 30)을 매개변수로 받았다면 4행~5행은 다음과 같이 3회 반복.
  - 1회 : num에 10을 저장한 후, result =result +10 → result에 10 저장됨
  - 2회 : num에 20을 저장한 후, result =result +20 → result에 30 저장됨
  - 3회 : num에 30을 저장한 후, result =result +30 → result에 60 저장됨

- 열 개 의 매개변수를 넘겨도 아래와 같이 잘 처리됨

```
hap=para_func(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
print(hap)
```

출력 결과

```
550
```

함수의 반환값과 매개변수를 알아봅시다(9)

- 함수의 매개변수 앞에 **를 붙이면 튜플이 아닌 딕셔너리 형식으로 전달. 함수를 호출할 때도 딕셔너리 형식의 매개변수를 '이야기 키=값' 형식으로 사용함

```
def  dic_func(**para) :
    for k  in  para.keys() :
        print ("%s  --> %d 명입니다." % (k, para[k]))

dic_func(아이오아이=11, 소녀시대=8, 걸스데이=4, AOA=7)
```

출력 결과

```
소녀시대  --> 8 명입니다.
걸스데이  --> 4 명입니다.
AOA  --> 7 명입니다.
아이오아이  --> 11 명입니다.
```

함수의 반환값과 매개변수를 알아봅시다(10)

- **로또 복권 번호 추첨 프로그램 완성**

소스코드 9-11
(파일명 : 09-11.py)

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

함수의 반환값과 매개변수를 알아봅시다(11)

```
14 while True :
15     num=getNumber()
16 
17     if lotto.count(num)==0 :
18         lotto.append(num)
19 
20     if len(lotto)>=6 :
21         break
22 
23 print("추첨된 로또 번호==>  ", end='')
24 lotto.sort()
25 for i in range(0, 6) :
26     print("%d  " % lotto[i], end='')
```

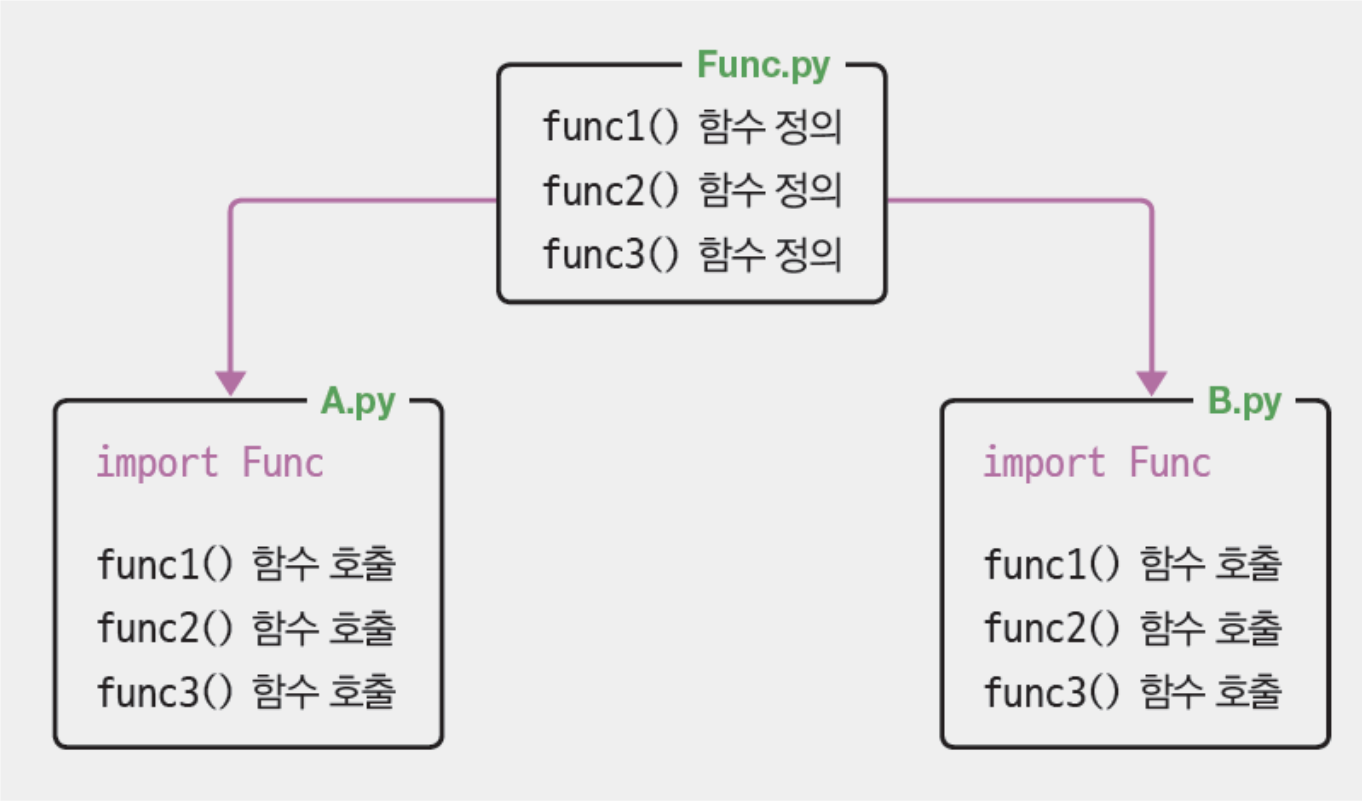
- 5행 : random.randrange (시작, 끝 + 1) 함수는 '시작 숫자~끝 숫자' 중 임의의 숫자 하나를 추출
- 14행~21행 : 무한 반복. 17행~18행에서 이미 뽑힌 숫자가 lotto[ ] 리스트에 들어있지 않아야만 lotto.append(num)함수로 lotto[ ] 리스트에 숫자를 추가함
- lotto.count(num) 함수는 lotto[ ] 리스트에서 num 숫자의 개수를 세어 6이 될 경우 빠져 나옴.

Section 04 모듈에 대해 알아봅시다

모듈에 대해 알아봅시다(1)

- **모듈의 이해**
  - 많이 사용하는 함수를 만들어 놓고, 프로그램에서 해당 함수를 사용할 때 함수를 Import 하여 사용하면 편리함

그림 9-12
모듈의 사용



모듈에 대해 알아봅시다(2)

■ 모듈의 생성과 사용

- 모듈로 사용할 파일과 호출하는 파일은 모두 같은 폴더에 저장

소스코드 9-12
(파일명 : Func.py)

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

소스코드 9-13
(파일명 : A.py)

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

모듈에 대해 알아봅시다(3)

출력 결과

```
Func.py의 func1()이 호출됨.
Func.py의 func2()가 호출됨.
Func.py의 func3()이 호출됨.
```

- 4행~6행 : 호출 할 때 'Func.함수이름( )' 형식으로 모듈이름을 앞에 붙였지만 모듈이름을 생략하고 함수이름만으로 사용하고 싶다면 다음과 같이 수정

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

소스코드 9-14
(파일명 : B.py)

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

출력 결과

```
[소스코드 9-13] 출력결과와 동일
```

모듈에 대해 알아봅시다(4)

■ 모듈의 종류

- 표준 모듈은 파이썬에서 제공하는 모듈, 사용자 정의 모듈은 직접 만들어서 사용하는 모듈, 서드 파티(3rd Party) 모듈은 파이썬이 아닌 다른 회사나 기관에서 제공하는 모듈

- 파이썬에서 제공하는 표준 모듈의 목록

```
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')
```

모듈에 대해 알아봅시다(5)

- 모듈 별로 제공되는 함수는 dir() 함수로 알 수 있음. 즉 math 모듈이 제공하는 함수의 목록을 보려면 다음 명령을 사용함

```
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']
```

모듈에 대해 알아봅시다(6)

- **모듈을 활용한 계산기 프로그램 완성**

소스코드 9-15
(파일명 : myCalc.py)

```
1  def plus(num1, num2) :
2      return num1+num2
3
4  def minus(num1, num2) :
5      return num1-num2
6
7  def multiply(num1, num2) :
8      return num1*num2
9
10 def divide(num1, num2) :
11     if num2!=0 :
12         return num1/num2
13     else :
14         print("0으로 나누면 안되요 ㅠㅠ")
15         return -1
```

모듈에 대해 알아봅시다(7)

- myCalc.py 모듈을 임포트

소스코드 9-16
(파일명 : 09-16.py)

```
1  from myCalc import  *
2
3  ## 메인 코드 부분
4  in1=float(input("첫 번째 숫자를 입력하세요 : "))
5  op=input("연산자(+, -, *, /)를 입력하세요 : ")
6  in2=float(input("두 번째 숫자를 입력하세요 : "))
7
8  print()
9  print("*** 모듈로 작성한 계산기 호출 결과 ***")
10 if op=='+' :
11     print("%5.1f+%5.1f=%5.1f" % (in1, in2, plus(in1, in2)))
12 elif op=='-' :
13     print("%5.1f-%5.1f=%5.1f" % (in1, in2, minus(in1, in2)))
14 elif op=='*' :
15     print("%5.1f*%5.1f=%5.1f" % (in1, in2, multiply(in1, in2)))
16 elif op=='/' :
17     print("%5.1f/%5.1f=%5.1f" % (in1, in2, divide(in1, in2)))
18 else :
19     print("연산자를 모르겠네요. ㅠㅠ")
```

Thank You