UNIT 29

함수 사용하기

29 함수 사용하기

함수 사용하기

- 파이썬은 함수(function)라는 기능을 제공하는데 특정 용도의 코드를 한 곳에 모아 놓은 것을 뜻함
- 함수는 처음 한 번만 작성해 놓으면 나중에 필요할 때 계속 불러 쓸 수 있음
- 지금까지 사용했던 print, input 등도 모두 파이썬에서 미리 만들어 둔 함수임
- 함수를 사용하면 이런 점이 좋음

- 코드의 용도를 구분할 수 있다.
- 코드를 재사용할 수 있다.
- 실수를 줄일 수 있다.

29.1 Hello, world! 출력 함수 만들기

Hello, world! 출력 함수 만들기

- 함수는 def에 함수 이름을 지정하고 ( )(괄호)와 :(콜론)을 붙인 뒤 다음 줄에 원하는 코드를 작성함(함수의 이름을 짓는 방법은 변수와 같음)
- 이때 코드는 반드시 들여쓰기를 해야 함(들여쓰기 규칙은 if, for, while과 같음)

```
def 함수이름():
    코드
```

- def는 정의하다(define)에서 따온 키워드임

29.1 Hello, world! 출력 함수 만들기

함수 만들기

```
>>> def hello():
...     print('Hello, world!')
...
```

- 함수 이름은 hello로 지정하고, 그다음 줄에서 print로 'Hello, world!' 문자열을 출력하도록 만들었음

29.1 Hello, world! 출력 함수 만들기

함수 호출하기

- 함수를 만든 부분 아래에서 hello()와 같이 함수 이름과 ()를 적어주면 함수를 사용할 수 있음

- 함수()

```
>>> hello()
Hello, world!
```

- 함수를 사용하는 방법을 "함수를 호출(call)한다"라고 부름

29.1 Hello, world! 출력 함수 만들기

소스 파일에서 함수를 만들고 호출하기

function.py

```
def hello():
    print('Hello, world!')

hello()
```

```
Hello, world!
```

29.1 Hello, world! 출력 함수 만들기

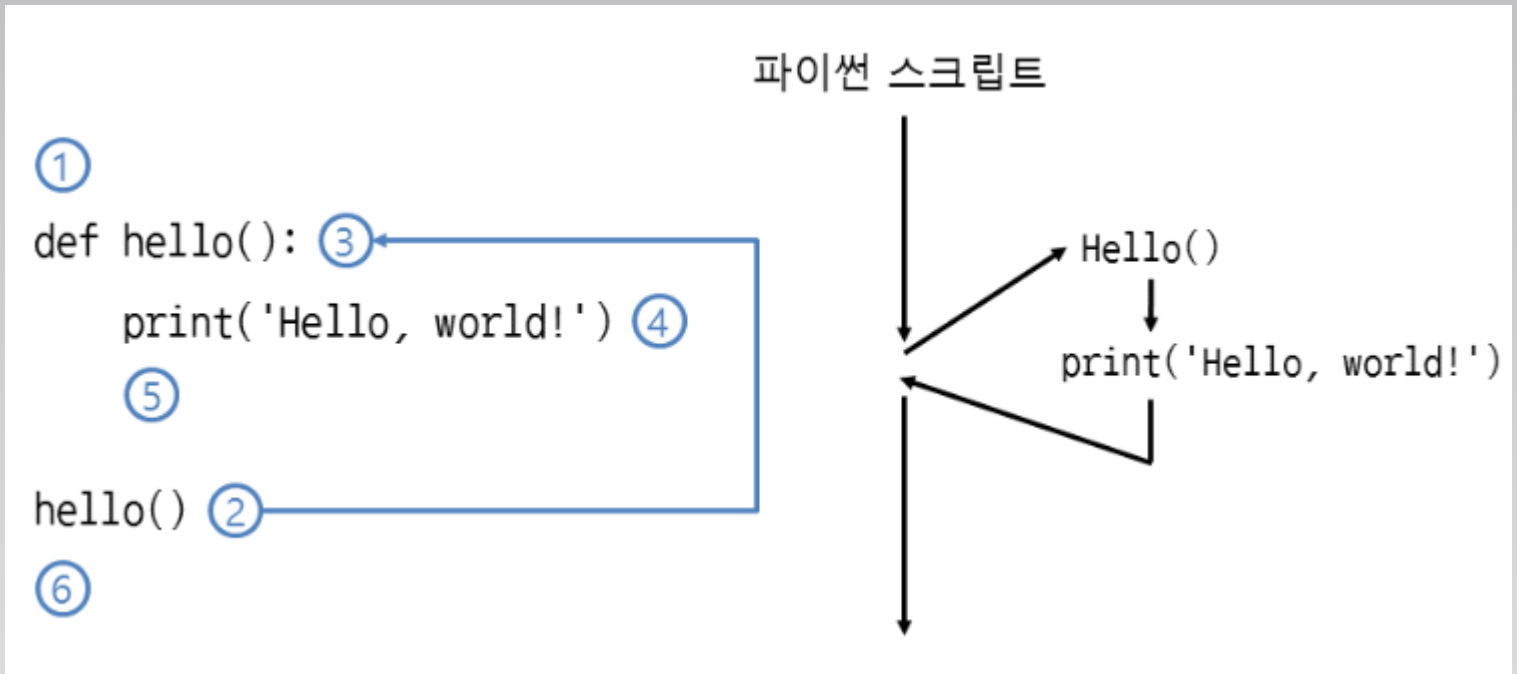
함수의 실행 순서

- hello 함수는 다음과 같은 순서로 실행됨

1. 파이썬 스크립트 최초 실행
2. hello 함수 호출
3. hello 함수 실행
4. print 함수 실행 및 'Hello, world!' 출력
5. hello 함수 종료
6. 파이썬 스크립트 종료

29.1 Hello, world! 출력 함수 만들기

▼ 그림 파이썬 스크립트에서 hello 함수의 실행 순서



29.1 Hello, world! 출력 함수 만들기

함수 작성과 함수 호출 순서

- 다음과 같이 함수를 먼저 호출한 뒤 함수를 만들 수는 없음

```
hello()         # hello 함수를 만들기 전에 함수를 먼저 호출

def hello():    # hello 함수를 만듦
    print('Hello, world!')
```

실행 결과

```
Traceback (most recent call last):
  File "C:\project\function.py", line 1, in <module>
    hello()    # hello 함수를 만들기 전에 함수를 먼저 호출
NameError: name 'hello' is not defined
```

- 함수를 먼저 호출하면 함수가 정의(define)되지 않았다는 에러가 발생함
- 파이썬 코드는 위에서 아래로 순차적으로 실행되기 때문임
- 함수를 먼저 만든 뒤에 함수를 호출해야 함

29.2 덧셈 함수 만들기

덧셈 함수 만들기

- 함수에서 값을 받으려면 ( )(괄호) 안에 변수 이름을 지정해주면 됨
- 이 변수를 매개변수(parameter)라고 부름

```
def 함수이름(매개변수1, 매개변수2):
    코드
```

```
>>> def add(a, b):
...     print(a + b)
...
```

```
>>> add(10, 20)
30
```

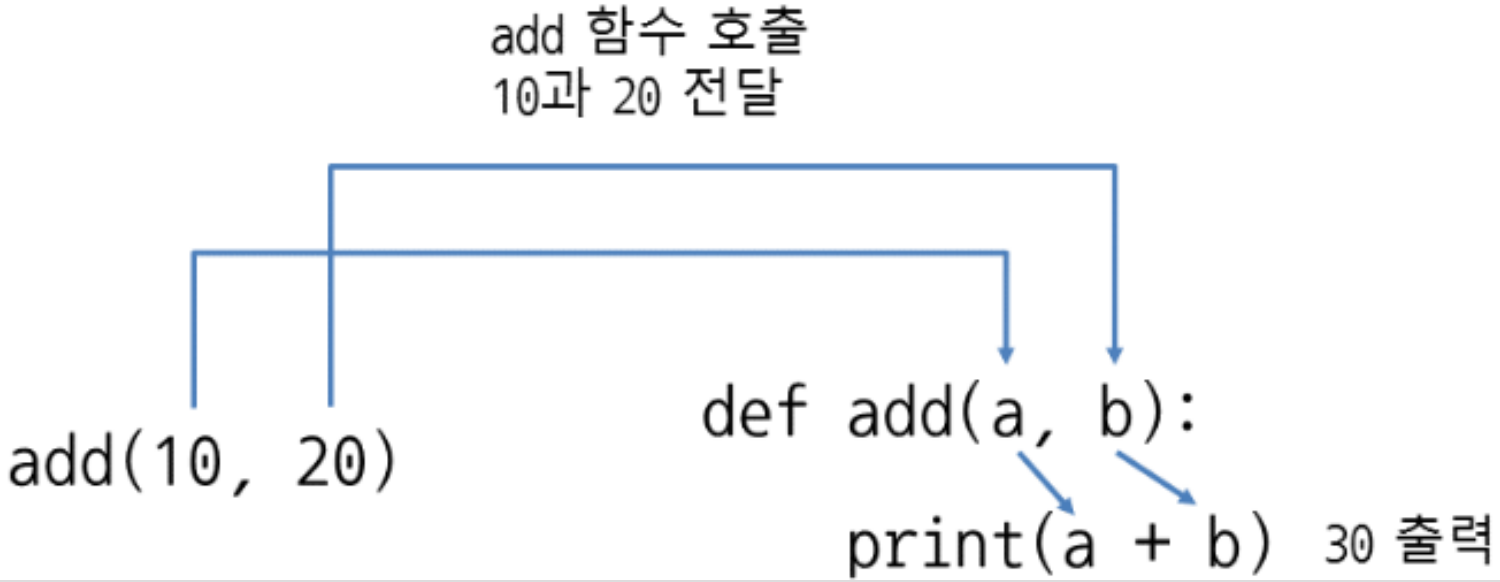
- 함수를 호출할 때 넣는 값을 인수(argument)라고 부름
- add(10, 20)에서 10과 20이 인수임

29.2 덧셈 함수 만들기

덧셈 함수 만들기

- add 함수의 호출 과정을 그림으로 표현하면 다음과 같은 모양이 됨

▼ 그림 함수에 매개변수 사용



29.3 함수의 결과를 반환하기

함수의 결과를 반환하기

- 다음과 같이 함수 안에서 return을 사용하면 값을 함수 바깥으로 반환함(return에 값을 지정하지 않으면 None을 반환)

```
def 함수이름(매개변수):
    return 반환값
```

```
>>> def add(a, b):
...     return a + b
...
```

```
>>> x = add(10, 20)
>>> x
30
```

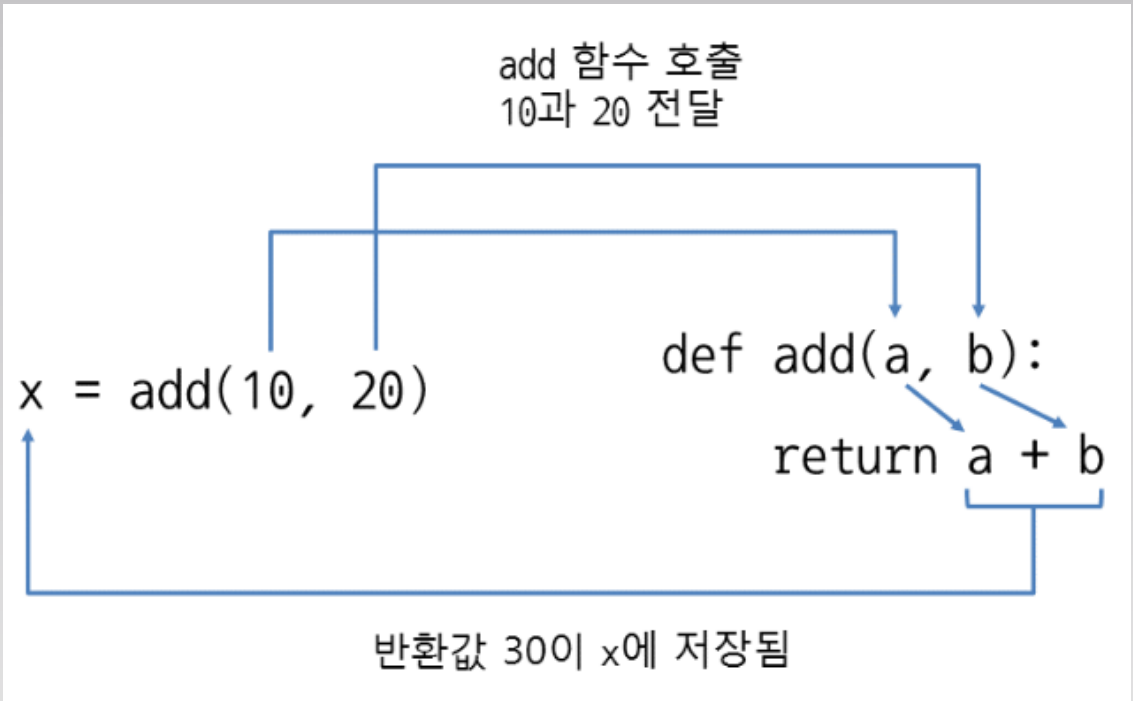
- return을 사용하면 값을 함수 바깥으로 반환할 수 있고, 함수에서 나온 값을 변수에 저장할 수 있음
- return으로 반환하는 값은 반환값이라고 하며 함수를 호출해준 바깥에 결과를 알려주기 위해 사용함

29.3 함수의 결과를 반환하기

함수의 결과를 반환하기

- add 함수의 호출 과정을 그림으로 표현하면 다음과 같은 모양이 됨

▼ 그림 함수의 반환값을 변수에 저장



29.3 함수의 결과를 반환하기

함수의 결과를 반환하기

- 반환값은 변수에 저장하지 않고 바로 다른 함수에 넣을 수도 있음

```
>>> print(add(10, 20))
30
```

29.4 함수에서 값을 여러 개 반환하기

함수에서 값을 여러 개 반환하기

- 함수에서 값을 여러 개 반환할 때는 다음과 같이 return에 값이나 변수를 ,(콤마)로 구분해서 지정하면 됨

```
def 함수이름(매개변수):
    return 반환값1, 반환값2
```

```
>>> def add_sub(a, b):
...     return a + b, a - b
...
```

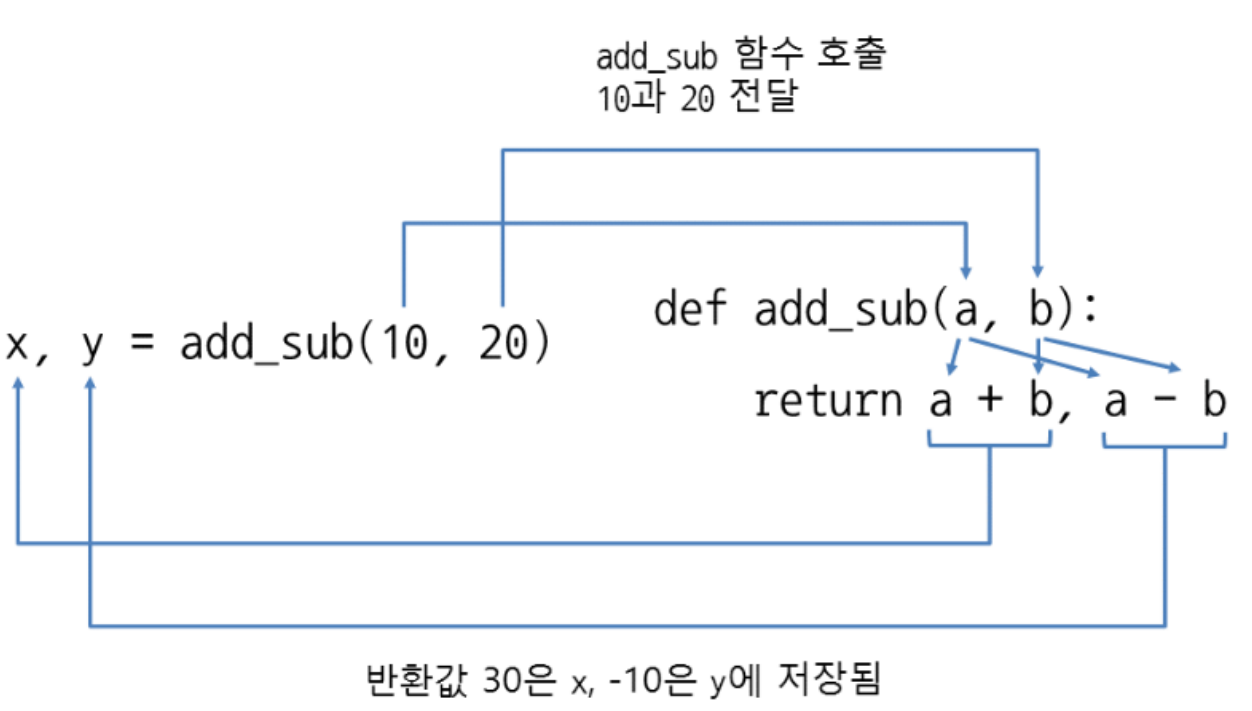
```
>>> x, y = add_sub(10, 20)
>>> x
30
>>> y
-10
```

29.4 함수에서 값을 여러 개 반환하기

함수에서 값을 여러 개 반환하기

- add_sub 함수의 호출 과정을 그림으로 표현하면 다음과 같은 모양이 됨

▼ 그림 반환값 여러 개를 변수 여러 개에 저장



29.4 함수에서 값을 여러 개 반환하기

함수에서 값을 여러 개 반환하기

- 다음과 같이 add_sub의 결과를 변수 한 개에 저장해서 출력해보면 튜플이 반환되는 것을 볼 수 있음

```
>>> x = add_sub(10, 20)
>>> x
(30, -10)
```

- 즉, 튜플이 변수 여러 개에 할당되는 특성을 이용한 것임(언패킹)

```
>>> x, y = (30, -10)
>>> x
30
>>> y
-10
```

29.5 함수의 호출 과정 알아보기

함수의 호출 과정 알아보기

- 함수 여러 개를 만든 뒤에 각 함수의 호출 과정을 스택 다이어그램(stack diagram)이라고함
- 스택은 접시 쌓기와 같은데 접시를 차곡차곡 쌓고 꺼낼 때는 위쪽부터 차례대로 꺼내는 방식임(단, 중간에 있는 접시는 뺄 수 없음)
- 파이썬에서는 접시 쌓기와 방향이 반대인데, 함수가 아래쪽 방향으로 추가되고 함수가 끝나면 위쪽 방향으로 사라짐

29.5 함수의 호출 과정 알아보기

함수의 호출 과정 알아보기

- 다음은 덧셈 함수 add와 곱셈 함수 mul이 있고, add 함수 안에서 mul 함수를 호출하는 방식으로 만들어져 있음

function_call.py

```
def mul(a, b):
    c = a * b
    return c

def add(a, b):
    c = a + b
    print(c)
    d = mul(a, b)
    print(d)

x = 10
y = 20
add(x, y)
```

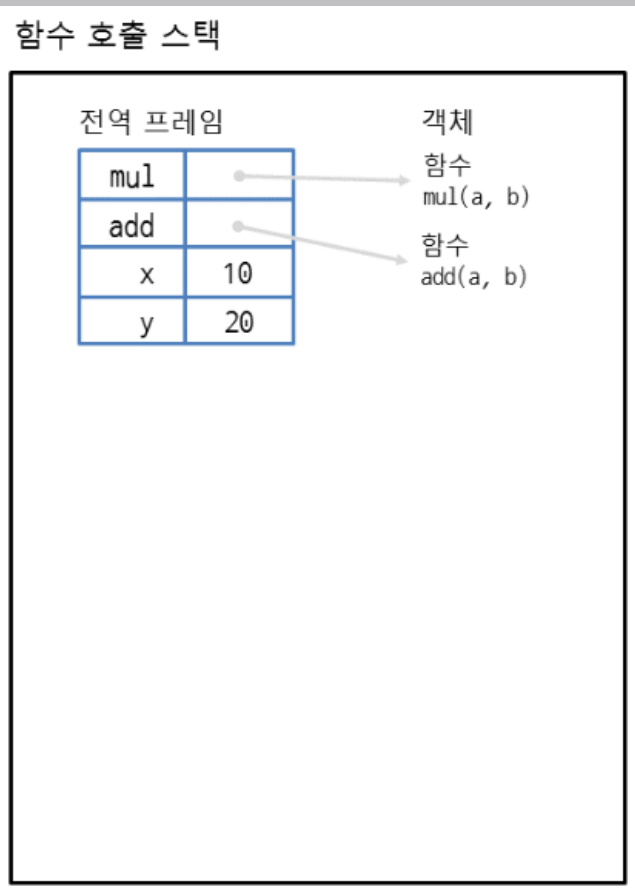
실행 결과

```
30
200
```

29.5 함수의 호출 과정 알아보기

▼ 그림 전역 프레임

```
1   def mul(a, b):
2       c = a * b
3       return c
4
5   def add(a, b):
6       c = a + b
7       print(c)
8       d = mul(a, b)
9       print(d)
10
11  x = 10
12  y = 20
13  add(x, y)
```



29.5 함수의 호출 과정 알아보기

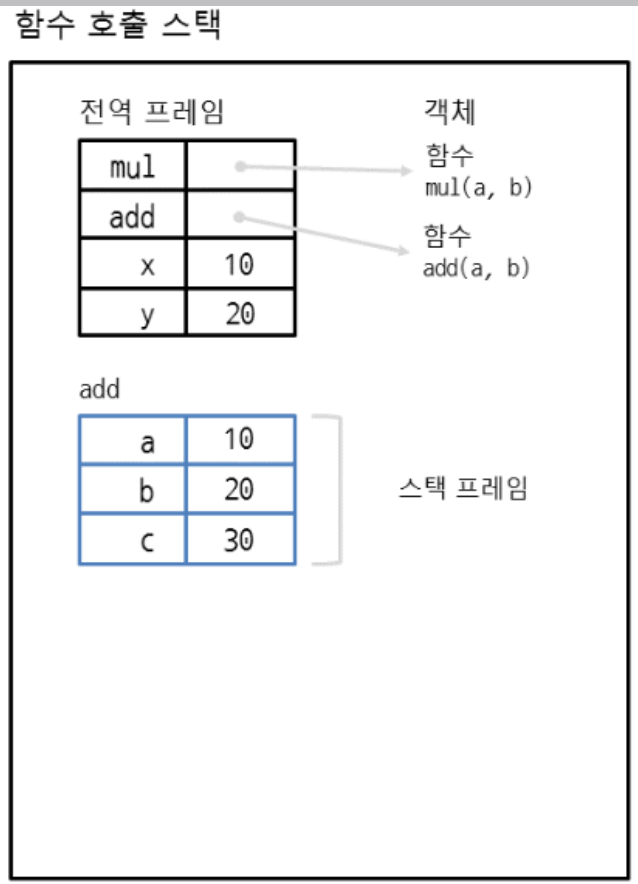
함수의 호출 과정 알아보기

- 프레임이란 메모리에서 함수와 함수에 속한 변수가 저장되는 독립적인 공간임
- 전역 프레임은 파이썬 스크립트 전체에서 접근할 수 있어서 전역 프레임이라 부름
- 함수 add를 호출한 뒤 안으로 들어가서 줄 7 print(c)까지 실행하면 다음과 같은 모양이 됨
- 함수 add의 스택 프레임이 만들어지고 매개변수 a와 b 그리고 변수 c가 들어감

29.5 함수의 호출 과정 알아보기

▼ 그림 함수 add의 스택 프레임

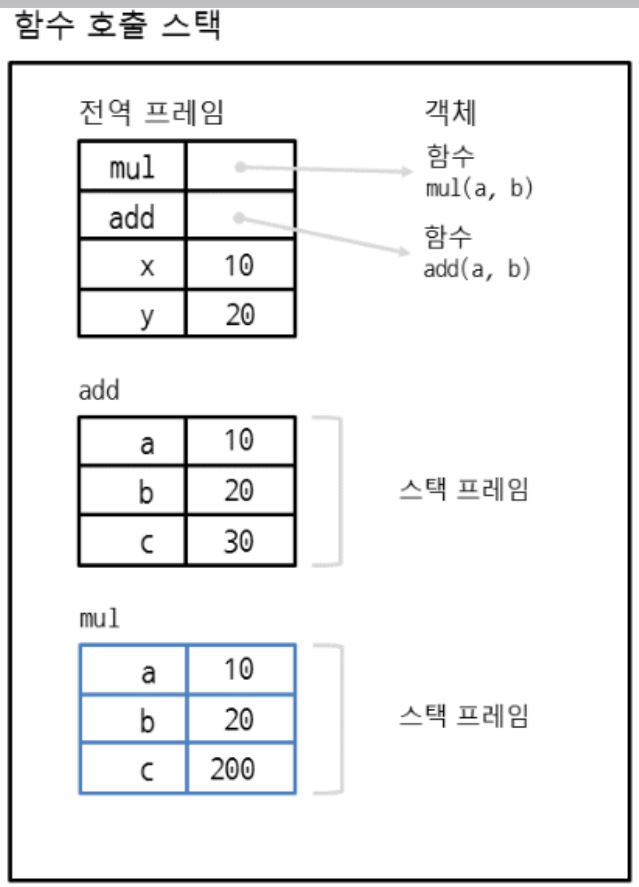
```
1   def mul(a, b):
2       c = a * b
3       return c
4
5   def add(a, b):
6       c = a + b
7       print(c)
8       d = mul(a, b)
9       print(d)
10
11  x = 10
12  y = 20
13  add(x, y)
```



29.5 함수의 호출 과정 알아보기

▼ 그림 mul 함수의 스택 프레임

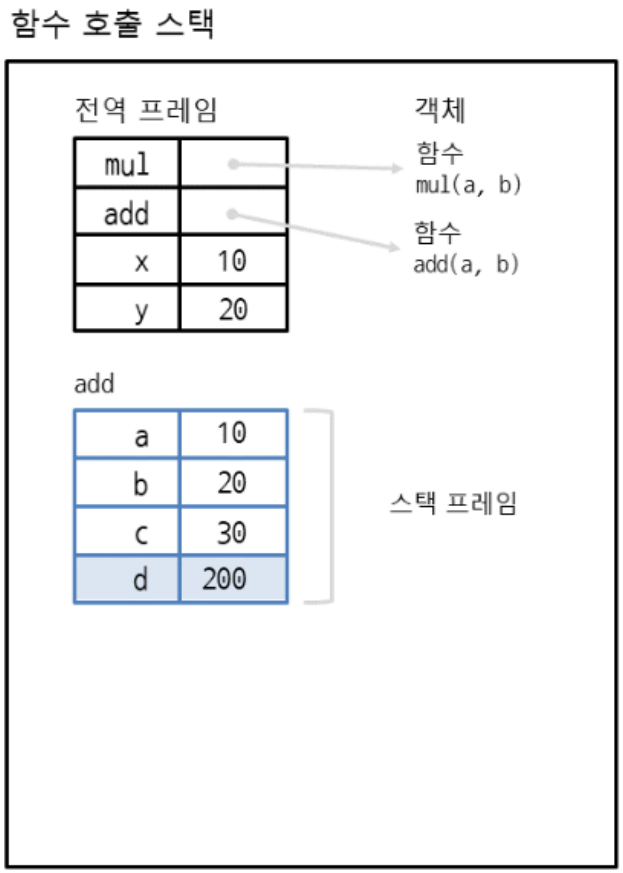
```
1   def mul(a, b):
2       c = a * b
3       return c
4
5   def add(a, b):
6       c = a + b
7       print(c)
8       d = mul(a, b)
9       print(d)
10
11  x = 10
12  y = 20
13  add(x, y)
```



29.5 함수의 호출 과정 알아보기

▼ 그림 mul 함수가 끝남

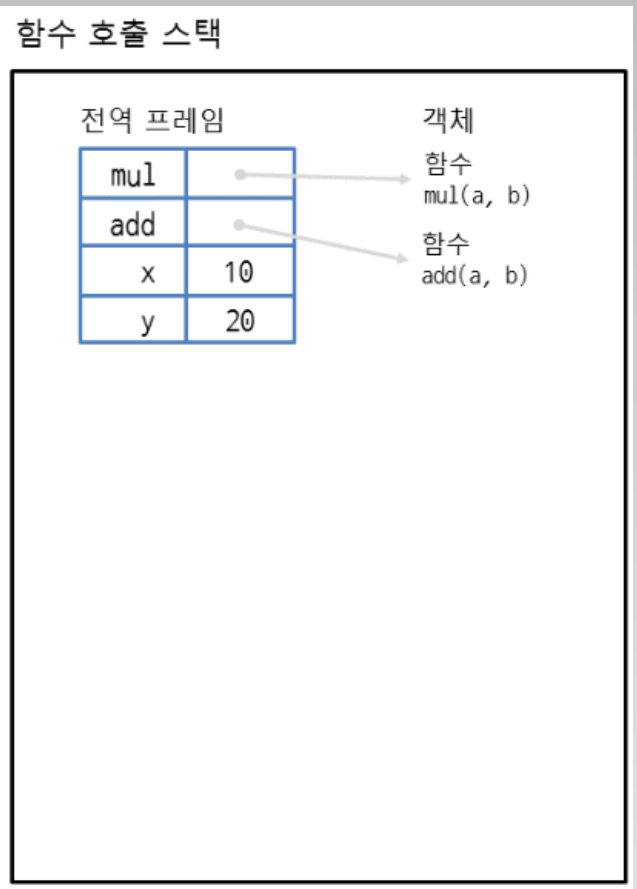
```
1  def mul(a, b):
2      c = a * b
3      return c
4
5  def add(a, b):
6      c = a + b
7      print(c)
8      d = mul(a, b)
9      print(d)
10
11 x = 10
12 y = 20
13 add(x, y)
```



29.5 함수의 호출 과정 알아보기

▼ 그림 add 함수가 끝남

```
1   def mul(a, b):
2       c = a * b
3       return c
4
5   def add(a, b):
6       c = a + b
7       print(c)
8       d = mul(a, b)
9       print(d)
10
11  x = 10
12  y = 20
13  add(x, y)
```



29.5 함수의 호출 과정 알아보기

함수의 호출 과정 알아보기

- 함수는 스택(stack) 방식으로 호출됨
- 함수를 호출하면 스택의 아래쪽 방향으로 함수가 추가되고 함수가 끝나면 위쪽 방향으로 사라짐
- 프레임은 스택 안에 있어서 각 프레임을 스택 프레임이라고 부름
- 전역 프레임은 스크립트 파일의 실행이 끝나면 사라짐