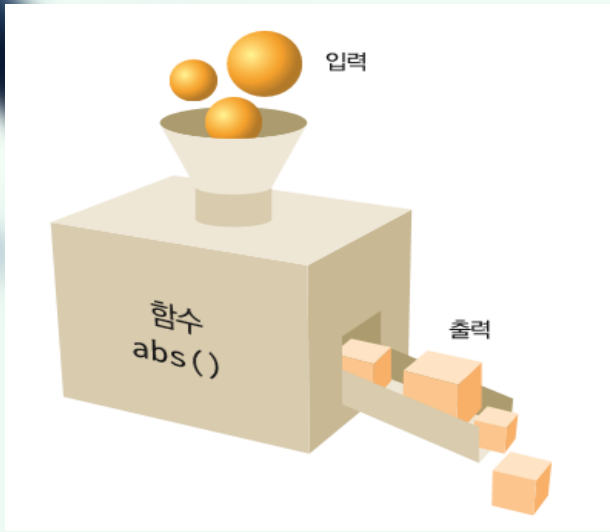
제6장
함수(function)-1

1. 함수란?

1) 함수의 개요

- 함수(function)는 특정 작업을 수행하는 명령어들의 모음을 칭한다. 이미 앞서 많은 함수들을 사용하였다. 화면에 무엇인가를 출력하고자 한다면 print()를 사용하면 될 것이며, 사용자에게 무엇인가 입력을 받고자 한다면 input ()를 사용하면 된다는 것을 잘 알고 지금까지 사용해왔다.
- 함수는 작업에 따라 필요한 데이터를 전달(매개변수, 인자값, parameter, arguments)받을 수 있으며, 작업이 완료된 후에는 작업의 결과를 호출한 곳으로 반환(return값, 반환값)할 수 있다. 함수는 입력을 받아서 처리한 후에 결과를 반환하는 상자로 그릴 수 있을 것이다.
- 아래 그림은 abs() 함수는 외부로부터 정수를 받아서 절대값을 계산하여 반환해주는 것이다.



함수 안의 명령어들을 실행하려면 우리는 함수를 호출(call)하면 되는 것이다. 아래와 같이 말이다.

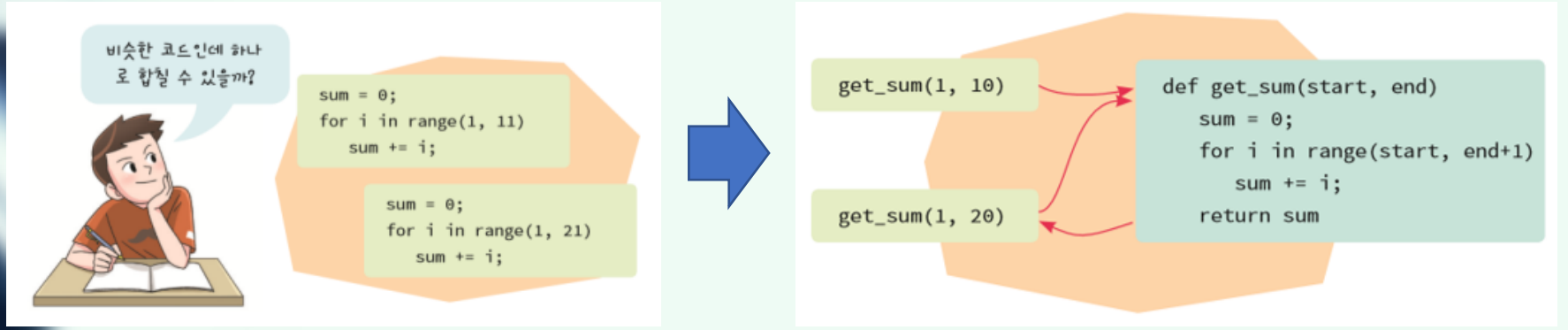
```
>>> value = abs(-100)
```

함수는 초창기부터 많은 프로그래밍 언어에서 지원하고 있으며, 동일한 함수를 여러 차례 호출할 수 있기 때문에 코드를 재사용 하는 기술이기도 하다.즉, 다시 말해 동일한 작업을 하기 위해서 코드를 복사하거나 기술할 필요가 없다는 말이다. 함수를 사용하면 코드를 상당히 간결하게 만들 수 있다. 아울러 하나의 큰 프로그램을 나누어서 작성할 수가 있어서 구조화된 프로그래밍이 가능하다. 또한 특정한 동작을 하는 코드가 독립적으로 작성되어 있기 때문에 코드의 유지보수가 쉬워진다.

1. 함수란?

2) 함수의 필요성

- 컴퓨터의 자원은 한정되어 있다. 프로그램 코드를 저장할 수 있는 공간도 역시 한정되어 있는 것이다. 프로그램을 작성하다 보면 동일한 처리를 반복해야 하는 경우가 많이 발생한다. 하여, 우리가 이미 작성한 코드를 재활용함으로써 자원을 효율적으로 쓰면서, 자주 사용하는 함수를 만들어서 호출만 하면 되는 것이다.



함수는 객체지향 프로그래밍 언어에서는 메서드라 지칭한다.

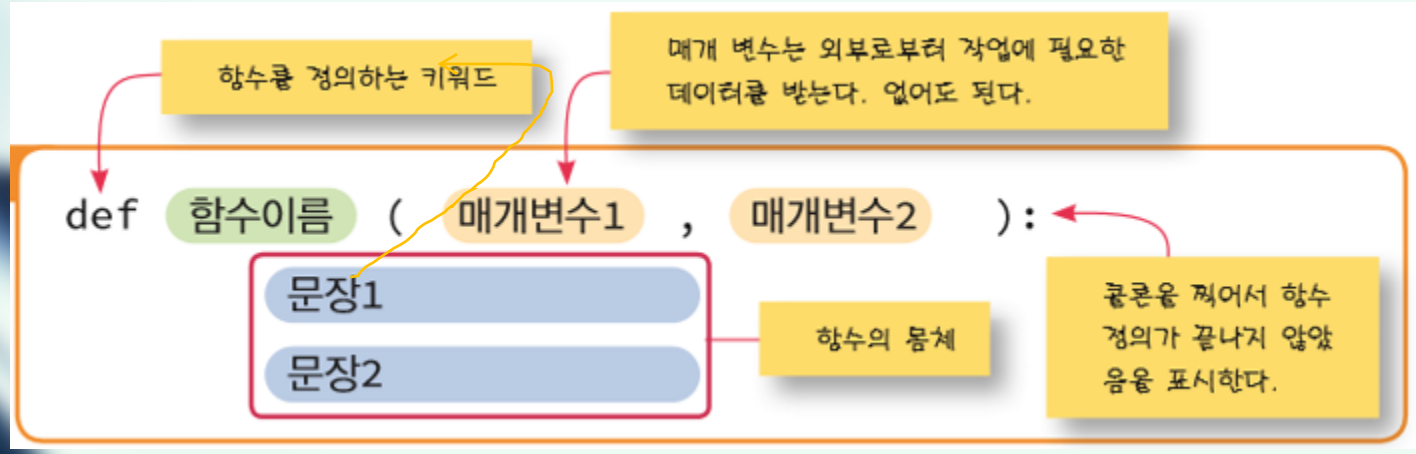
[함수 선언 및 구현]

- 위의 그림과 같이 2개의 코드 조각은 매우 유사하다. 정수의 합을 계산하는 코드인데 매개변수의 값만 다르다는 것을 알 수가 있다. 만약, 함수로 정의하지 않았다면 동일한 코드를 복사해서 매개변수만 바꾸어 붙여넣기를 해야 한다. 이러한 작업은 상당히 귀찮은 작업이며 코드의 증가, 가독성 저하 등의 여러가지 문제를 발생시킨다.
- 하여 이런 경우에 유용하게 사용할 수 있는 도구가 바로 함수(function)인 것이다. 함수를 이용하면 여러 번 반복해야 되는 처리 단계를 하나로 모아 필요할 때 언제든지 호출하면 되는 것이다.

2. 함수의 선언 및 구현

1) 함수의 문법

- 파이썬에서 함수를 선언 및 구현하는 방법은 아래와 같다.



- 함수는 크게 선언부(헤더)와 구현부(정의부, 몸체)로 나누어진다. 선언부에는 def 키워드로 시작하면 이어서 함수 이름, 매개변수(인자값, parameter, arguments)를 적어주며 끝에 콜론(:)을 찍어준다. 아시다시피 콜론(:)은 인터프리터에게 성급하게 해석하지 말라고 하는 의미이다. 함수는 구현부(정의부, 몸체)까지 입력되어야 비로소 해석을 시작할 수가 있다. 매개변수는 외부에서 전달되는 데이터를 함수로 전달하는 변수이다.
  함수의 구현부는 함수가 수행하는 작업을 위한 명령어들이 들어간다. 아래 아주 간단한 함수의 예를 다음 슬라이드에서 보고 이해하도록 하자.

2. 함수의 선언 및 구현

2) 함수의 구현과 호출

- 아래와 같이 함수를 선언 및 구현하고 호출하면 되는 것이다.

```
>>>def say_hello(name):
        print("안녕, ", name);
```

[함수 선언 및 구현]

```
>>>say_hello("철수")
안녕, 철수
```

[함수 호출]

좌측의 함수는 함수 이름은 say_hello()이고 name은 매개변수이다. 함수의 구현부는 print()함수를 통해 매개변수를 출력하는 것이다. 함수는 구현이 되었다고 해서 실행되는 것은 아니고 함수를 호출해야 실행된다.

```
>>>def say_hello(name, msg):
        print("안녕, ", name, "야, msg);
```

[함수 선언 및 구현]

```
>>>name="철수"
>>>msg="어서 집에 오너라"
>>>say_hello(name, msg)
안녕, 철수야, 어서 집에 오너라
```

[함수 호출]

좌측의 함수는 함수 이름은 say_hello()이고 name, msg는 매개변수이다. 함수의 구현부는 print()함수를 통해 매개변수를 출력하는 것이다.

```
>>>def get_sum(start, end) :
    sum=0
    for i in range(start, end+1) :
        sum += i
    return sum
```

[함수 선언 및 구현]

```
>>>value = get_sum(1, 10)
>>>print(value)
55
```

[함수 호출]

좌측의 함수는 함수 이름은 get_sum()이고 start, end는 매개변수이다. 함수의 구현부는 for 문을 활용하여 매개변수로 들어온 2개의 값을 누적합을 구하여 return 키워드로 반환하고 있다.

2. 함수의 선언 및 구현

3) 함수의 장점

- 프로그램 안에서 중복된 코드를 제거한다.
- 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있다. 각 함수들은 레고 블록처럼 다른 함수들과 연결되어서 하나의 프로그램을 구성한다.
- 함수는 한 번 만들어지면 다른 프로그램에서도 얼마든지 재사용될 수 있다.
- 함수를 사용하면 가독성이 증대되고, 유지 관리도 매우 쉬워진다.
- 기존 파이썬에서 제공하는 함수들의 사용법을 알고자 한다면 https://docs.python.org/3.8/ 를 참조하면 된다.

4) 함수의 이름

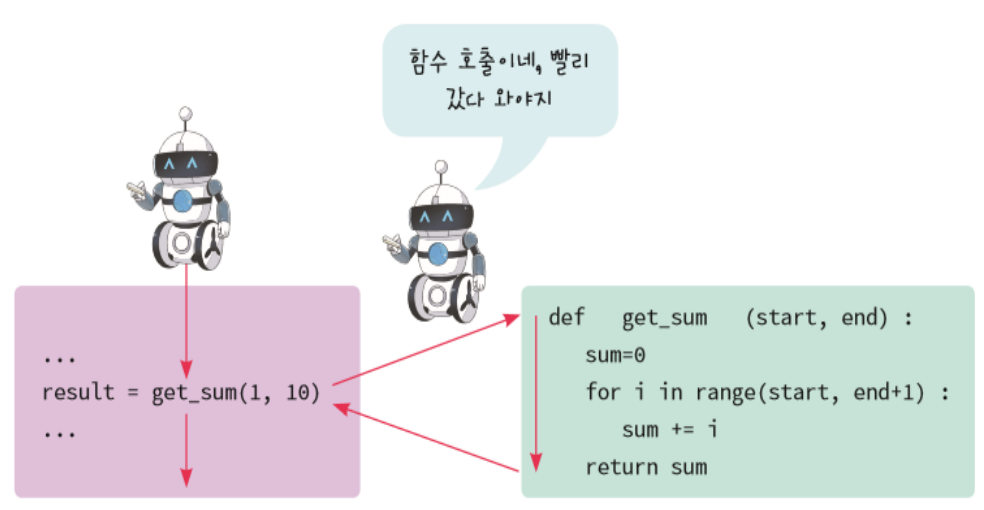
- 함수의 이름은 식별자에 대한 규칙만 따른다면 어떤 이름이라도 상관없다. 다만 가독성이 좋을 수 있도록 함수의 기능을 알려주는 이름을 명명하는 것이 좋다. 일반적으로 함수의 목적을 설명하는 동사 또는 동사+동사를 사용하면 된다. 아래는 함수 이름의 예이다.

```
square(side)              // 정수를 제곱하는 함수
compute_average(list)     // 평균을 구하는 함수
set_cursor_type(c)        // 커서의 타입을 설정하는 함수
```

2. 함수의 선언 및 구현

4) 함수의 호출

- 함수를 선언하고 구현하는 최고의 목적은 바로 함수를 실행시켜 그 기능을 하도록 하는데에 있다. 그렇다면 함수를 사용하려면 당연히 호출(call)을 해야 하는 것이다. 함수 호출(function call)이란 그냥 print()와 같이 함수의 이름을 코드로 적어주면 된다. 함수 안의 문장들은 호출되기 전까지는 전혀 실행되지 아니한다.
- 함수가 호출되면 비로소 함수 안의 있는 문장들이 순차적으로 실행되면 실행이 끝나면 다시 호출한 곳으로 돌아간다. 예를 들어서 get_sum() 을 호출한다고 하면 아래와 같은 순서로 프로그램이 실행된다.




```
def   get_sum(start, end) :
        sum=0
        for i in range(start, end+1) :
                sum += i
        return sum

print( get_sum(1, 10))
print( get_sum(1, 20))
```

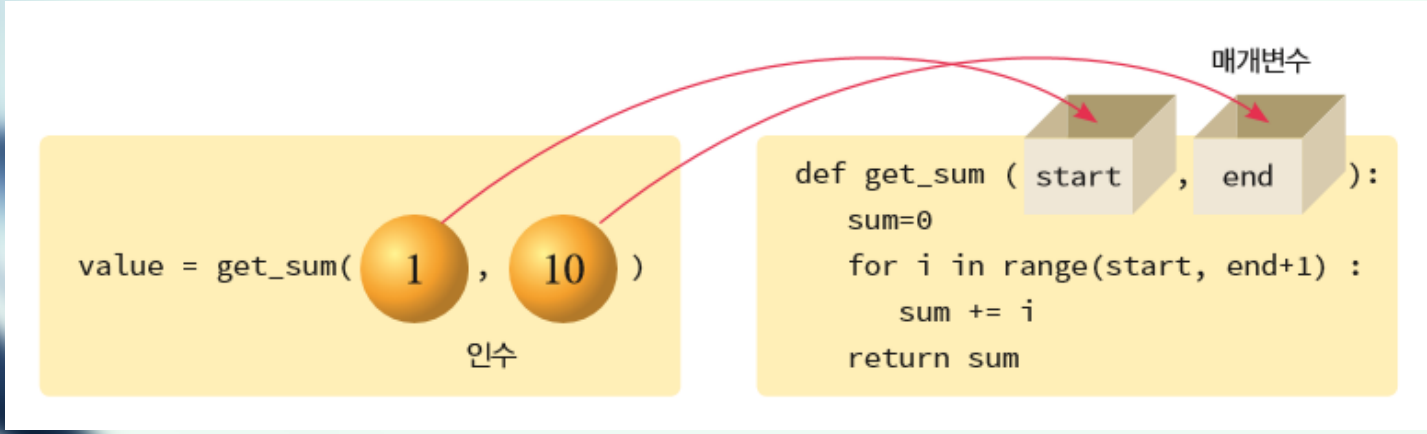
```
55
210
```

- 함수는 일단 작성되면 위와 같이 몇 번에 상관없이 필요하다면 언제든 호출 가능하다. 이것이 바로 함수의 가장 큰 장점인 것이다.

3. 인수와 매개변수 그리고 반환값

1) 인수(arguments)와 매개변수(parameter)의 차이점

- 인수(argument)는 호출 프로그램에 의하여 함수에 실제로 전달되는 값이다.
- 매개 변수(parameter)는 이 값을 전달받는 변수이다.



- 함수가 호출될 때마다 인수는 달라질 수 있다.

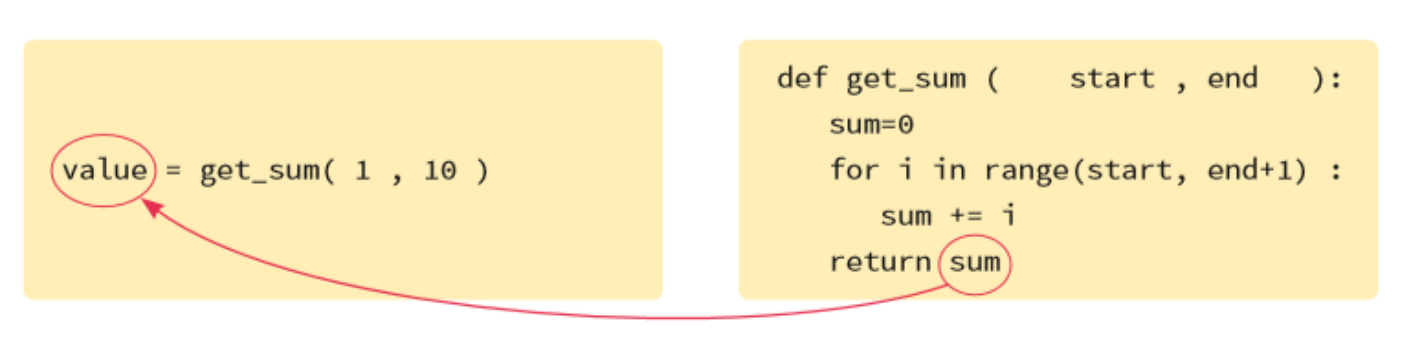
```
# 10이 get_sum()의 인수가 된다
x = get_sum(10)

# 20이 get_sum()의 인수가 된다
y = get_sum(20)
```

- 여기서 반드시 주의해야 할 점은 매개변수의 개수는 정확히 일치하여야 한다는 점이다. 즉, 매개변수가 두 개이면 인수도 두 개를 전달해야 한다. 그렇치 아니하면 에러가 발생한 것을 이미 실습에서 확인한 바 있다.

3. 인수와 매개변수 그리고 반환값

2) 반환값(return value)

- 함수가 호출한 곳으로 반환하는 작업의 결과값이다.
- 함수는 자신을 호출한 곳으로 값을 반환할 수 있다. 값을 반환하려면 return 문장 다음에 수식을 써주면 수식의 값이 반환된다.



- 함수의 반환값은 결국 return 문장 뒤에 있는 수식의 계산값이 된다. return 뒤에 나오는 수식은 파이썬에서 유효한 수식이면 무엇이든 가능하다. 아래 예제 코드를 보자.

```
return 0
return x
return (x*x)+2
```

- 파이썬은 함수가 값을 반환하지 않는 경우에는 None 이라는 특별한 값을 반환한다. None은 어떤 객체도 참조하지 않는다 라는 것을 의미한다.

4. 값을 반환하지 않는 함수

1) 값을 반환하지 않는 함수

- 프로그래밍을 하다 보면 어떤 값을 계산하지 않지만 몇 개의 명령어들을 실행해야 하는 경우도 있다. 만약 이런 경우가 자주 있다면 이 때에서 역시 함수를 만들어서 사용할 수가 있다.

  예를 들어, 화면에 사람들의 이름과 나이를 박스로 만들어서 출력할 때에는 아래와 같이 printInfo() 라는 함수를 작성하여 사용하는 것도 코드 절약과 재활용이 될 것이다.

```
def printInfo(name, age):
        print("====================")
        print("이름: ", name)
        print("나이: ", age)
        print("====================")
        return

printInfo("홍길동", 35)
```

- 함수 안의 return문 뒤에 아무 내용이 없다면 함수의 종료를 알린다. 하여 호출한 곳으로 되돌아가게 된다.

5. 디폴트 인수

1) 디폴트 인수(default argument)

- 파이썬에서는 함수의 매개변수가 기본값을 가질 수 있다. 이것을 디폴트 인수(default argument)라고 한다.

예를 들어보면 아래와 같이 인사를 하는 함수 greet()가 있다고 하자. greet()는 항상 2개의 인수를 받아야 한다.

```
def greet(name, msg="별일 없죠?"):
   print("안녕 ", name + ', ' + msg)

greet("영희")
```

```
안녕  영희, 별일 없죠?
```

- 위의 msg의 값을 미리 “별일 없죠?”라고 정해 놓은 것이다.이런 것을 두고 디폴트 인수라고 한다.

6. 키워드 인수

1) 키워드 인수(default argument)

- 파이썬에서는 대부분의 인수들은 함수 호출 시에 위치에 의하여 구별된다. 예를 들자면 power(1, 10)은 power(10, 1)과 다르다. 이유는 함수 호출 power(1, 10)은 1의 10제곱 값을 구할 것이며, power(10, 1)은 10의 1제곱을 구할 것이다.
- 하지만, 키워드 인수(keyword argument)는 인수들 앞에 키워드를 두어서 인수들을 구분한다. 키워드 인수는 함수를 호출할 때, 인수의 이름을 명시적으로 지정해서 전달하는 방법이다. 예를 들어 아래와 같이 인수가 3개인 함수가 있다고 하자.

```
def calc(x, y, z):
    return x + y + z
```

- 물론 위의 함수는 아래와 같이 호출하는 것이 정상일 것이다. 10은 매개변수 x로 전달되고, 20은 y로, 30은 z로 전달된다. 이와 같은 기본 인수 전달 방식을 위치 인수(positional argument)라고도 한다.

```
calc(10, 20, 30)
60
```

- 하지만, 아래와 같이 매개변수의 이름에 값을 직접 대입하여서도 전달할 수가 있다. 이 방법의 장점은 인수의 위치가 매개변수의 위치와 달라도 된다는 것이다. 키워드 인수를 사용할 때는 인수들이 어떤 순서로 전달 되어도 상관 없다.

```
calc(x=10, y=20, z=30)
60
```

```
calc(y=20, x=10, z=30)
60
```

감사합니다.