1. 함수와 인수

❖ 함수

- 일련의 코드 블록에 이름을 붙여 정의한 것
- 자주 반복되는 코드의 사용이 용이해짐
- 호출문으로 실행
  - 함수(인수(아규먼트)들...)

함수(function) : 어떤 일을 수행하는 코드의 블록 또는 코드의 묶음

함수의 장점

① 필요할 때마다 호출 가능

② 소스를 논리적인 단위로 분할 가능

③ 효율적인 코드 관리

calcsum

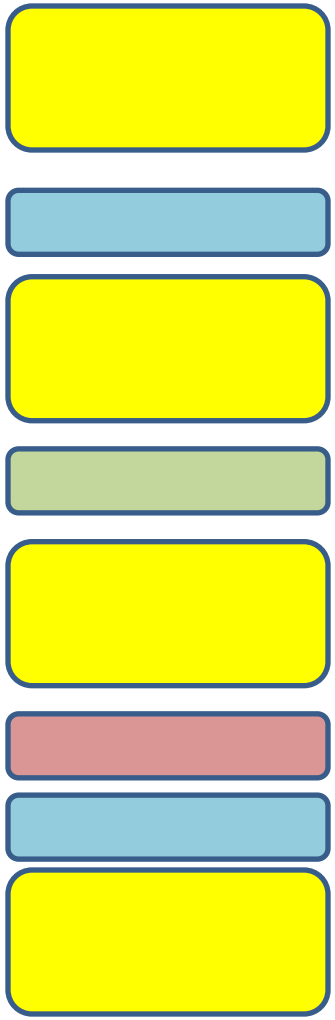
```
def calcSum(n):
    sum = 0
    for num in range(n + 1):
        sum += num
    return sum

print("~ 4 =", calcSum(4))
print("~ 10 =", calcSum(10))
```

실행결과

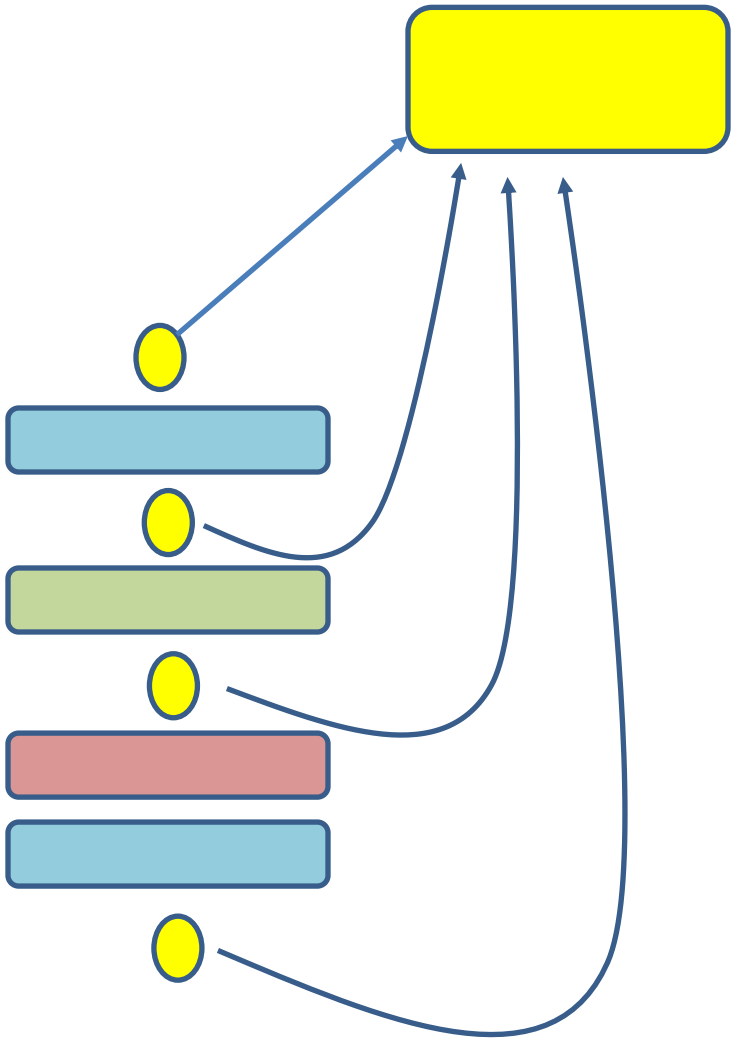
```
~ 4 = 10
~ 10 = 55
```

1. 함수와 인수

함수를 사용하지 않은 소스



함수를 사용하는 소스



1. 함수와 인수

❖ 함수의 선언(정의)

```
def 함수 이름 (매개변수 #1 …):
    수행문 1
    수행문 2
    return <반환값>
```

① **def** : 'definition'의 줄임말로, 함수를 정의하여 시작한다는 의미이다.

② **함수 이름** :

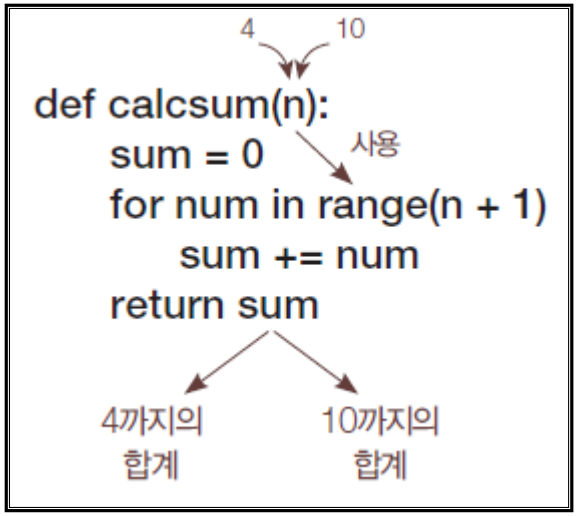
함수 이름은 개발자가 마음대로 지정할 수 있지만, 파이썬에서는 다음과 같은 규칙을 사용한다.

- 소문자로 입력한다.
- 여러 워드를 묶을 경우에는 _ 기호를 사용한다. ex) save_model
- 행위를 기록하므로 동사와 명사를 함께 사용하는 경우가 많다. ex) find_number

1. 함수와 인수

❖ 인수

- 호출할 때 함수로 전달되는 데이터
- 함수의 동작에 변화 주어 활용성을 높임
- 매개변수를 통해서 전달받음



- 형식 인수 = 함수 정의문의 인수 → 매개변수
- 실 인수 = 함수 호출문에서 전달하는 인수 → 아규먼트

calcrange

```
def calcrange(begin, end):
    sum = 0
    for num in range(begin, end + 1):
        sum += num
    return sum

print("3 ~ 7 =", calcrange(3, 7))
```

실행결과

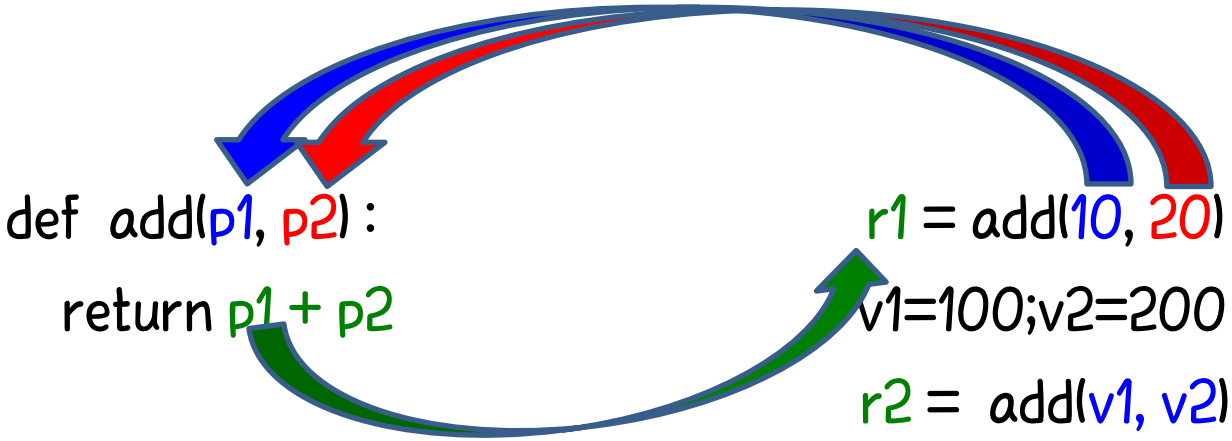
```
3 ~ 7 = 25
```

1. 함수와 인수

❖ 함수의 매개변수의 아규먼트

매개변수 : 함수가 호출될 때 전달받고자 하는 데이터를 저장하는 변수

아규먼트 : 함수를 호출하면서 전달하는 데이터



1. 함수와 인수

❖ 리턴 값

- 함수의 실행 결과를 호출한 곳으로 돌려주는 값
- 리턴 값을 반환할 때는 return 명령 뒤에 반환할 값을 지정

return sum return 100 return v1 + v2 return len('abc')

- 리턴 값이 무조건 있어야 하는 것은 아님(리턴값이 없는 함수는 자동으로 None 이 리턴 됨)

printsum

```
def printsum(n):
    sum = 0
    for num in range(n + 1):
        sum += num
    print("~", n, "=", sum)

printsum(4)
printsum(10)
```

실행결과

```
~ 4 = 10
~ 10 = 55
```

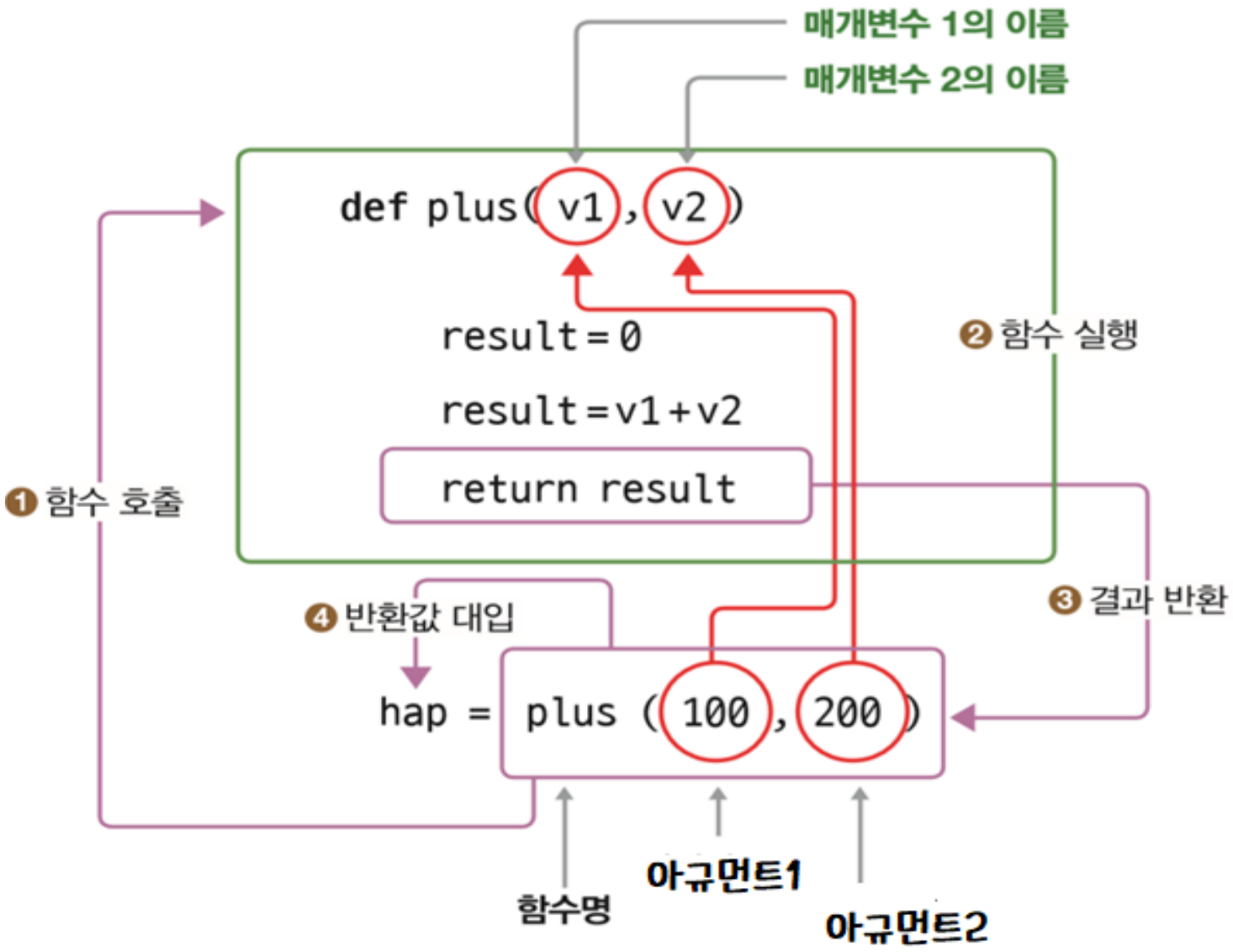
1. 함수와 인수

매개변수 유무 / 반환값 유무	매개변수 없음	매개변수 있음
반환값 없음	함수 안 수행문만 수행	매개변수를 사용하여 수행문만 수행
반환값 있음	매개변수 없이 수행문을 수행한 후, 결과값 반환	매개변수를 사용하여 수행문을 수행한 후, 결과값 반환

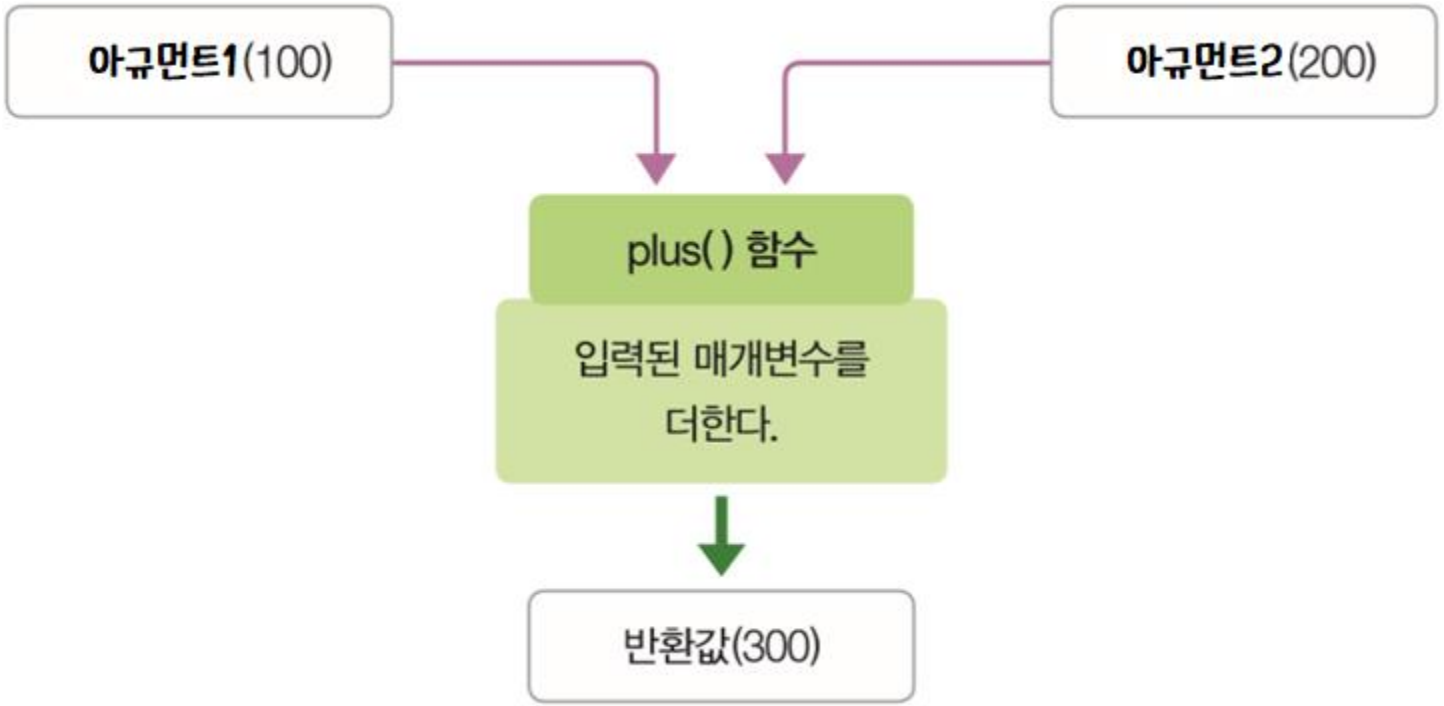
리턴 값이 없는 함수는 리턴 값으로 None이 대신 리턴

None : 아무 값도 없음을 나타냄

1. 함수와 인수


매개변수 1의 이름
매개변수 2의 이름
def plus(v1 , v2)
result = 0
result = v1 + v2
return result
❷ 함수 실행
❶ 함수 호출
❸ 결과 반환
❹ 반환값 대입
hap = plus (100 , 200)
함수명
아규먼트1
아규먼트2

1. 함수와 인수



4.8 입력함수와 출력함수

❖ input() 함수

- 사용자로부터 입력을 받기 위한 함수

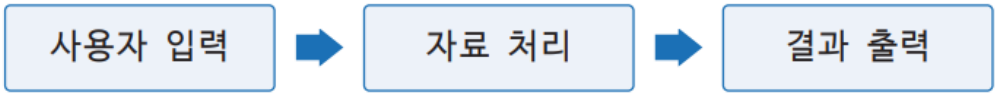
코드 4-31 : input() 입력함수를 사용한 name의 입력 방법

input_name1.py

```
print('Enter your name : ')
name = input()    # 사용자의 키보드 입력 값을 name이 참조함
print('Hello', name, '!')
```

실행결과

```
Enter your name :
Hong GilDong
Hello Hong GilDong !
```



[그림 4-14] 사용자 입력과 처리, 결과 출력을 가지는 프로그램의 흐름

코드 4-32 : 문자열을 이용한 input 입력함수의 사용법

input_name2.py

```
name = input('Enter your name : ')
print('Hello', name, '!')
```

실행결과

```
Enter your name : Hong GilDong
Hello Hong GilDong !
```

4.8.1 input() 함수와 int() 함수

대화창 실습 : 정수형과 문자열형의 덧셈, 서로 다른 자료형의 덧셈

```
>>> 100 + 1     # 정수의 덧셈으로 수+수
101
>>> '100' + '1' # 문자열의 덧셈으로 문자열+문자열 연산을 통해 문자를 연결
'1001'
>>> '100' + 1   # 문자열과 정수의 덧셈(오류 발생)
Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
TypeError: must be str, not int
```

대화창 실습 : int() 함수와 float(), str() 함수를 이용한 문자열의 변환

```
>>> int('100') + 1    # 문자열 '100 ' 의 덧셈을 위해 정수(int)로 변환
101
>>> float('100') + 1  # 문자열 '100'을 덧셈을 위해 실수(float)로 변환
101.0
>>> '100' + str(1)    # 문자열 덧셈을 위해서 1을 문자열 '1'로 변환
'1001'
```

대화창 실습 : 대화형 모드를 통한

int() 함수로 감싸야 정수값이 됨

```
>>> num1 = int(input("숫자를 입력하세요: "))
숫자를 입력하세요: 100
>>> num2 = int(input("숫자를 입력하세요: "))
숫자를 입력하세요: 200
>>> num3 = num1 + num2
>>> print("두 수의 합은", num3, "입니다.")
두 수의 합은 300 입니다.
```

int() 함수로 감싸야 정수값이 됨

- '100'이라는 문자형을 정수 100으로 변환할 수 있는 방법은?
- int() 함수를 이용하여 괄호 안의 문자열을 정수형으로 변환하기

- int() 함수 대신에 float() 함수를 사용하면 실수 값으로 변환할 수 있다

대화창 실습 : int() 함수와 float() 함수를 이용한 문자열의 변환

```
>>> int('100') + 1
101
>> float('100') + 1
101.0
```

대화창 실습 : format() 메소드와 플레이스홀더의 인덱스

```
>>> '{} Python!'.format('Hello')
'Hello Python!'
>>> '{} Python!'.format('Hi')
'Hi Python!'
```

플레이스 홀더에 들어갈 내용

플레이스 홀더

```
>>> '{0} Python!'.format('Hello')
'Hello Python!'
```

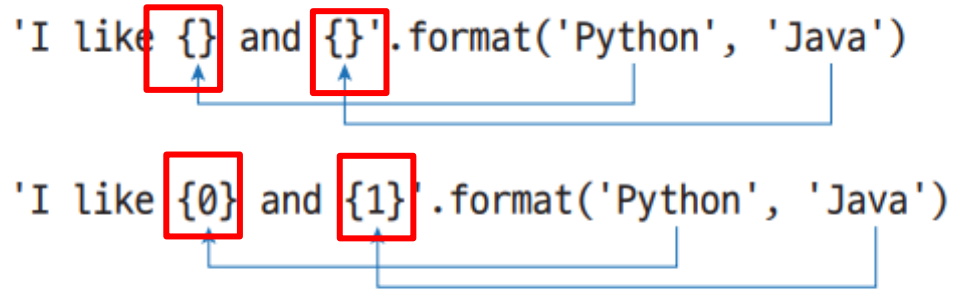
베이스 문자열, 템플릿 문자열

대화창 실습 : format() 메소드와 플레이스홀더

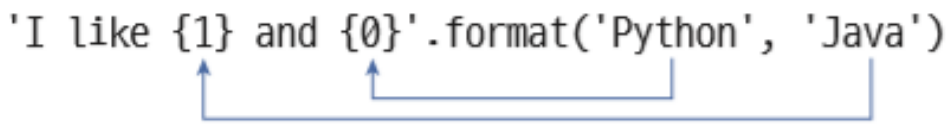
```
>>> 'I like {} and {}'.format('Python', 'Java')
'I like Python and Java'
>>> 'I like {0} and {1}'.format('Python', 'Java')
'I like Python and Java'
```

인덱스에 따라 Python, Java가 각각 들어감

❖ 플레이스홀더 내에 필요한 정수 값(인덱스)을 할당하여 출력 순서를 제어할 수 있다(디폴트로 0, 1, .. 이 할당됨)



[그림 4-15] 플레이스홀더와 인덱스를 이용한 출력제어



[그림 4-16] 플레이스홀더와 전달인자

- {0}, {1}, {2}와 같이 플레이스홀더의 번호를 이용하여 다양한 출력을 할 수 있다
- 플레이스홀더에는 문자열뿐만 아니라 100, 200과 같은 정수형이나 실수형 등 임의의 자료형도 올 수 있음

대화창 실습 : format() 메소드와 플레이스홀더의 인덱스

```
>>> 'I like {1} and {0}'.format('Python', 'Java')
'I like Java and Python'
```

인덱스를 중복할 수 있음

대화창 실습 : format() 메소드와 플레이스홀더의 인덱스 사용법

```
>>> '{0}, {0}, {0}! Python'.format('Hello')
'Hello, Hello, Hello! Python'
>>> '{0}, {0}, {0}! Python'.format('Hello', 'Hi')
'Hello, Hello, Hello! Python'
>>> '{0} {1}, {0} {1}, {0} {1}!'.format('Hello', 'Python')
'Hello Python, Hello Python, Hello Python!'
>>> '{0} {1}, {0} {1}, {0} {1}!'.format(100, 200)
'100 200, 100 200, 100 200!'
```

❖ format() 내부에는 문자열 리터럴 뿐만 아니라 다음과 같이 변수나 객체를 넣을 수 있음

대화창 실습 : 플레이스홀더 내의 객체 출력

```
>>> greet = 'Hello'
>>> '{} World!'.format(greet)
'Hello World!'
```

코드 4-33 : 플레이스홀더와 format() 메소드의 사용

print_format1.py

```
name = 'Hong GilDong‘
print('My Name is {}!'.format(name))
```

실행결과

```
My Name is Hong GilDong!
```

코드 4-34 : 플레이스홀더와 format() 메소드의 사용

print_format2.py

```
name = input('당신의 이름을 입력해주세요 : ')
age = input('나이를 입력해주세요 : ')
job = input('직업을 입력해주세요 : ')

print('당신의 이름은 {}, 나이는 {}살, 직업은 {}입니다.'.format(name, age, job))
```

실행결과

```
당신의 이름을 입력해주세요 : 김철수
나이를 입력해주세요 : 21
직업을 입력해주세요 : 학생
당신의 이름은 김철수, 나이는 21살, 직업은 학생입니다.
```

- print() 함수 내부에 있는 {}의 의미는 {}가 있는 곳에 format() 함수의 인자 값 name을 출력하라는 뜻
- 사용자의 이름과 나이를 입력으로 받아서 name, age라는 이름의 변수에 저장한 후 format() 메소드를 이용하여 출력이 가능
- 문자열의 format() 메소드를 사용해 {} 필드에 각각 이름(name)과 나이(age), 직업(job)이 위치하도록 문자열 포매팅을 함

```
print('당신의 이름은', name, '나이는', age, '살, 직업은', job, '입니다.')
```

- format() 메소드를 사용하지 않는다면 출력문은 복잡한 형태로 나타남. 이 경우 쉼표가 너무 많고 따옴표도 많아서 코드를 읽기도 힘들고 오류가 날 가능성도 매우 높다

4.11 내장함수

- 프로그래밍에서 함수는 흔히 **블랙박스**black box라는 비유를 함
- 함수를 호출하는 사용자는 제대로 된 입력 값을 주고 그 결과인 출력(결과 값)을 사용만 하면 되기 때문

[그림 4-18] 입력에 대해 정해진 출력을 내보내는 블랙박스 역할을 하는 함수

[그림 4-19] 내장함수 len()의 동작

- len() 함수가 어떤 값을 반환하는지 알고 있기 때문에 자세한 동작 과정을 몰라도 사용함
- print() 함수, input() 함수도 이런 함수에 속함

- 파이썬에서 기본으로 구현되어 있어 제공하는 함수를 파이썬의 **내장함수**built-in function라고 한다

[표 4-2] 파이썬의 내장함수 목록

파이썬의 내장 함수				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	_import_()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

대화창 실습 : 대화형 모드를 통한 여러 가지 내장 함수 실습

```
>>> abs(-100)                    # 절대값을 반환하는 함수
100
>>> min(200, 100, 300, 400)     # 여러 원소들 중 최솟값을 반환하는 함수
100
>>> max(200, 100, 300, 400)     # 여러 원소들 중 최댓값을 반환하는 함수
400
>>> str1 = "FOO"                # "Foo" 혹은 'FOO' 형식으로 문자열 객체를 생성함
>>> len(str1)                   # 문자열의 길이를 반환
3
>>> eval("100+200+300")        # 문자열을 수치값과 연산자로 변환하여 평가
600
>>> sorted("EABFD")            # 문자열을 정렬
['A', 'B', 'D', 'E', 'F']
>>> list = [200, 100, 300, 400]
>>> sorted(list)
[100, 200, 300, 400]
>>> sorted(list, reverse=True)
[400, 300, 200, 100]
```

- abs() 함수는 -100이라는 정수를 입력 받아서 그 절댓값인 100을 반환
- min() 함수는 200, 100, 300, 400 의 값을 가지는 정수들 중에서 가장 작은 값을 반환
- max() 함수는 200, 100, 300, 400 의 값을 가지는 정수들 중에서 가장 큰 값을 반환
- type() 함수는 해당 변수의 자료형을 반환
- len() 함수는 변수 str1에 저장된 문자열의 길이를 반환
- eval() 함수는 문자열을 받아와 해당 문자열의 내용을 수식화해서 평가(evaluate)한 다음 평가한 값을 반환
- sorted() 함수는 문자열을 받아와 해당 문자열을 구성하는 문자들을 알파벳 순으로 정렬해 반환