Chapter 1

함수

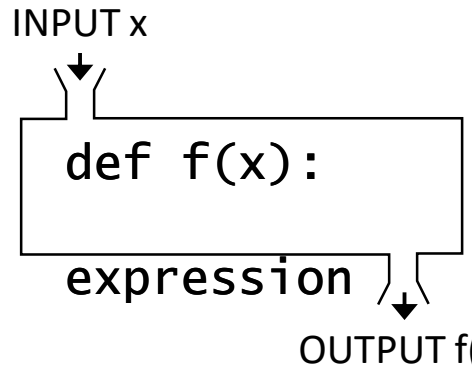
함수

- 함수는 반복되는 코드 블록에 이름을 붙여 놓은 것입니다.

- 함수는 **반복되는 코드를 줄여주고**, 좀 더 구조화시켜 우리가 코드를 알아보기 쉽게 합니다.

- 함수의 이름 뒤에는 반드시 소괄호'()'가 따릅니다.

함수 선언 방법 (반드시 외우자)

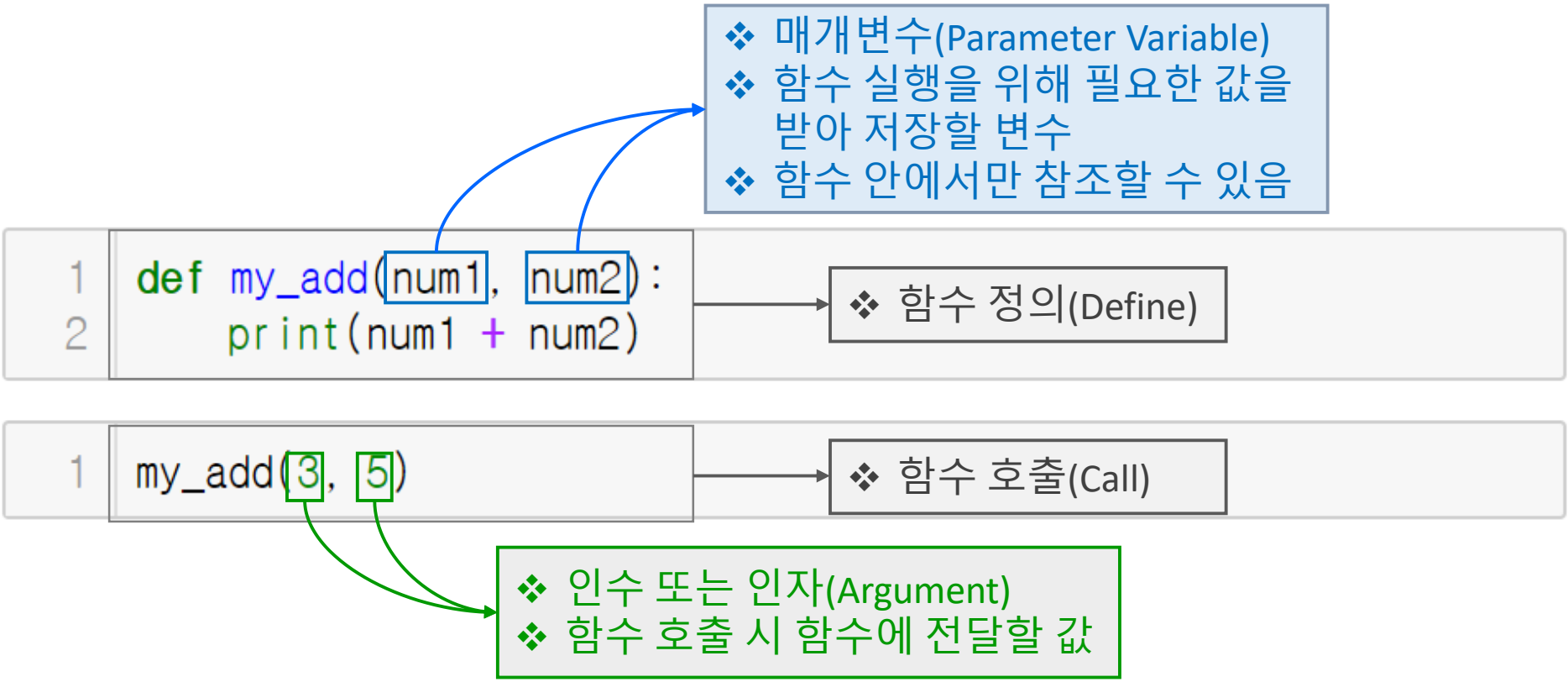


def 이름 (매개변수) :

```
def function_name([param1, param2, ...]):
    expression
```

함수

매개변수(Parameter) : 함수 정의 시 지정하는 함수가 실행을 위해 필요하는 값을 받을 변수들



함수

Return

함수의 반환 값은 함수가 실행한 결과를 함수를 호출한 곳에 전달하기 위해 사용

함수가 실행한 결과를 반환하는 값

변수 또는 표현식의 결과는 모든 자료형 가능

```
def function_name(param) :
    # code
    return return_value
```

함수

함수의 모형

유형 1: 매개변수 있고, 반환 값 있다! 전달인자(○), 반환 값(○)

유형 2: 매개변수 있고, 반환 값 없다! 전달인자(○), 반환 값(×)

유형 3: 매개변수 없고, 반환 값 있다! 전달인자(×), 반환 값(○)

유형 4: 매개변수 없고, 반환 값 없다! 전달인자(×), 반환 값(×)

Chapter 7

1. 지역변수와 전역변수
2. 함수의 다양한 문법+

함수

지역 변수	전역 변수

def func_a():
  num = 10
  print(num) # 10
}
 	
def func_b():
  print(num) # 에러
}
	num = 20
	def func_a():
	  print(num) # 20
	}
	
	def func_b():
	  print(num) # 20
	}

지역 심볼 테이블

num = 10

전역 심볼 테이블

num = 20

함수

Lexical scope

- 함수가 전역변수와 동일한 지역변수를 가지고 있을 때, 함수 안에서 전역변수를 참조하지 못하는 특징을 갖는데 Lexical특성 이라고 합니다.

변수의 값을 찾는 순서

로컬(지역)변수 > 전역 변수 > 내장 된 이름

```
g_var = 100
```

```
def func1():
    print("before", g_var) # 1
    g_var = 200
    print("after", g_var)  # 2
```

```
func1()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-19-d88c41ef3303> in <module>
----> 1 func1()

<ipython-input-18-d2aa724b74da> in func1()
      1 def func1():
----> 2     print("before", g_var) # 1
      3     g_var = 200
      4     print("after", g_var)  # 2

UnboundLocalError: local variable 'g_var' referenced before assignment
```

지역변수로 판별됨

함수

1. Global키워드

전역변수를 참조하려면, 변수 사용 전에 global키워드를 이용해서 선언하면 됩니다.
Global 이후부터 전역변수의 참조가 됩니다.

```
g_var2 = 100
```

```
def func2():
    global g_var2
    print("before", g_var2) # 1
    g_var2 = 200
    print("after", g_var2)  # 2
```

```
func2()
```

```
before 100
after 200
```

```
print(g_var2)
```

```
200
```

함수

2. 함수는 변수에 저장될 수도 있습니다.

그리고, 새로운 변수를 통해 함수를 호출할 수 있습니다.

```
def fibonacci(n):
    "n값 미만까지 피보나치 수열을 출력합니다."
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

```
fibo1 = fibonacci
```

```
type(fibo1)
```

```
function
```

```
print(fibo1)
```

```
<function fibonacci at 0x0000024DBEBBBF28>
```

```
fibo1(2000)
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

함수

3. 기본 값을 갖는 매개변수(가변인수)

함수를 정의할 때에 허용하도록 정의 인수의 수 보다 적은 인수로 호출 할 수 있음

```
def make_url(ip, port=80):
    return "http://{}:{}".format(ip, port)
```

```
make_url("localhost")
```

```
'http://localhost:80'
```

```
make_url("localhost", 8080)
```

```
'http://localhost:8080'
```

```
make_url("coderby.com")
```

```
'http://coderby.com:80'
```

함수

4. 키워드 인수

파라미터 이름을 포함한 인수 사용시 순서를 바꿀 수 있음
기본값을 갖는 파라미터는 생략 가능
필수 인수를 포함하지 않으면 에러

함수

키워드 인수

```
def func4(a, L=None):
    if L is None:
        L = []
    L.append(a)
    return L

```

```
list_ = []
```

```
func4(10, list_)
```

```
[10]
```

```
func4(20, L=list_)
```

```
[10, 20]
```

```
func4(30, L=list_)
```

```
[10, 20, 30]
```

Chapter 7

수고하셨습니다