

학습 내용

1부. 프로그래밍 언어 기본

1장. 파이썬 개요 및 개발환경 구성

2장. 자료형과 연산자

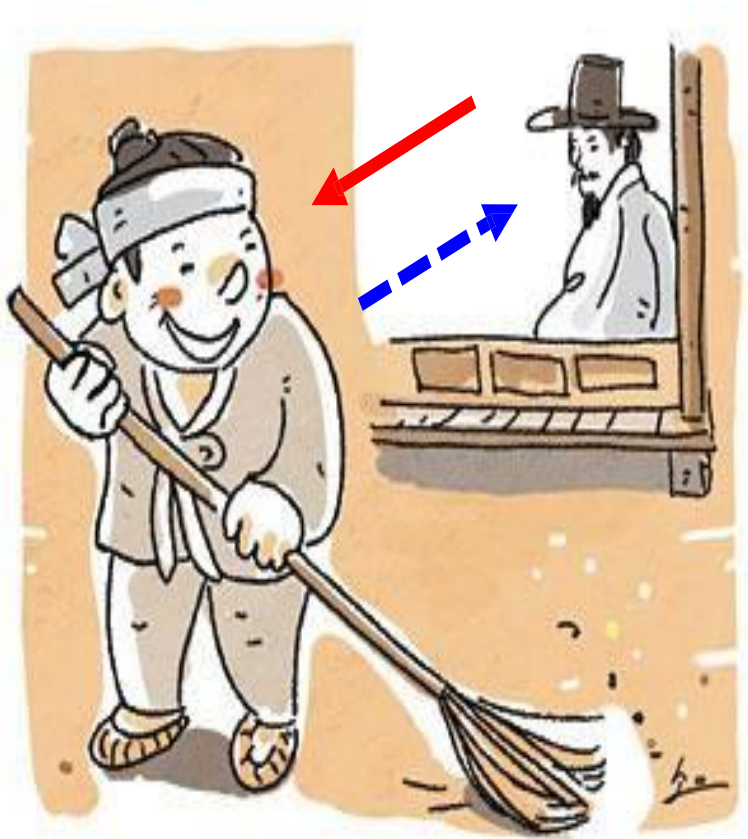
3장. 데이터 구조

4장. 제어문

5장. 함수

- 1. 함수 정의 및 사용
- 2. 함수의 실행결과를 반환하는 return
- 3. 함수 매개변수
- 4. 람다식
- 5. 파이썬 내장함수

함수(function) 개념



마당쇠야...

마당

마당쇠야...

사랑채

마당쇠야...

대문앞

≡ API
(library)

1절. 함수의 정의 및 사용

5장. 함수

- 입력 값을 받아 다른 값을 출력하도록 미리 만들어져 있는 것
- 반복해서 사용한 코드들을 묶어 놓고 그것에 이름을 붙인 것
- 반복해서 사용할 코드는 함수를 이용하면 훨씬 구조적이고 간결한 코드를 작성할 수 있음
- 함수 정의
 - 함수를 사용하려면 먼저 함수가 정의(define)되어 있어야 함
- 함수 호출
 - 정의되어 있는 함수를 사용하려면 '함수명()' 형식으로 사용
 - 한 쌍의 소괄호 안에는 함수가 실행되기 위해 필요한 값을 입력

INPUT x

```
def f(x):
    expression
```

OUTPUT f(x)

1.1. 함수 정의하기

1절. 함수의 정의 및 사용

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

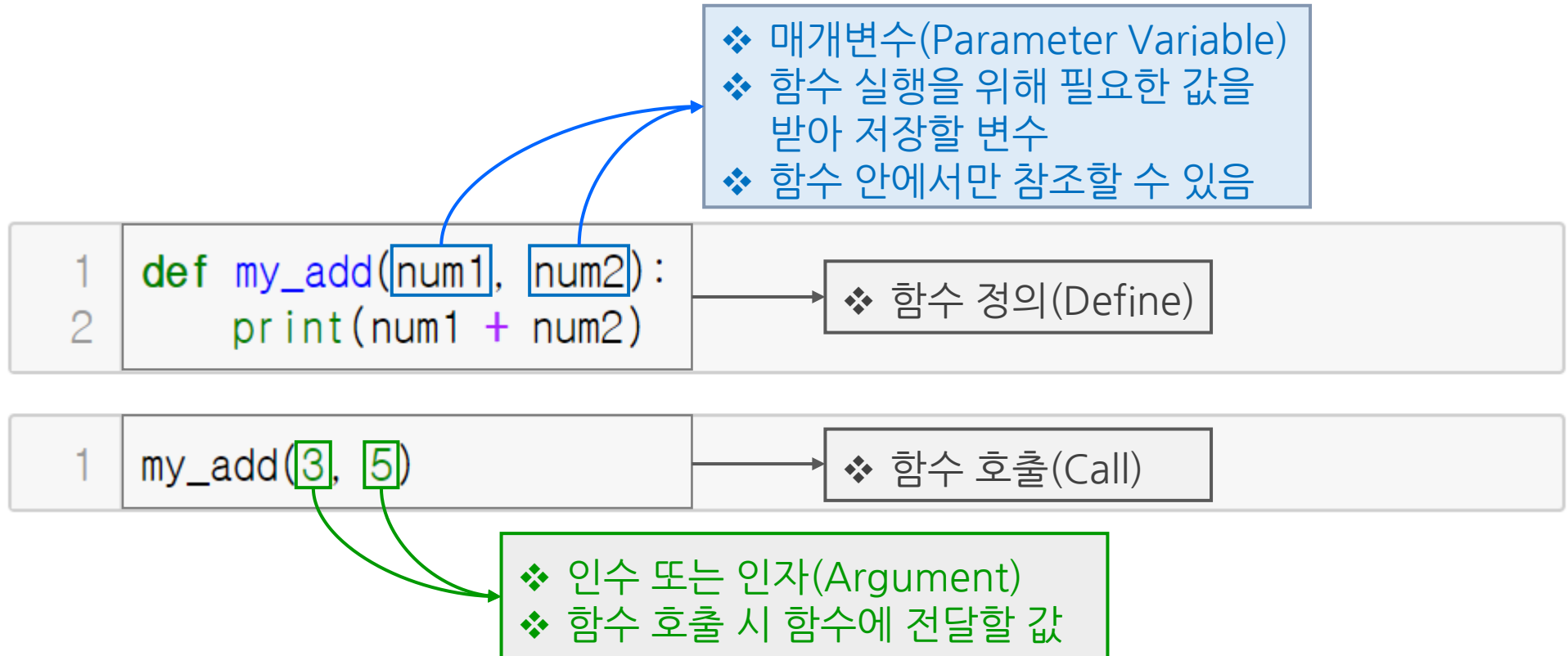
● 구문에서...

- *def* : 함수를 정의하기 위해 사용하는 키워드
- *function_name* : 함수의 이름. 변수 이름 만드는 것처럼 함수를 구분하는 이름. 함수 이름은 문자, 숫자, _를 포함할 수 있으며, 숫자로 시작할 수 없음.
- *param1, param2, ...* : 함수의 매개변수(Parameter variable). 함수가 실행될 때 필요로 하는 값을 받기 위해 사용. 매개변수는 선택사항.
- *expression* : 함수가 실행할 구문. 반드시 들여쓰기가 되어 있어야 함

1) 매개변수와 인수

1절. 함수의 정의 및 사용 > 1.1. 함수 정의하기

- 매개변수(Parameter Variable) : 함수 정의 시 지정하는 함수가 실행을 위해 필요하는 값을 받을 변수들
- 인수(Argument) : 함수 호출 시 함수 실행을 위해 전달하는 값



2) 매개변수가 없는 함수 정의

1절. 함수의 정의 및 사용 > 1.1. 함수 정의하기

- 함수 안에서 실행하는 값을 사용하지 않을 때 매개변수를 선언하지 않는 함수를 정의
- 함수를 만드는 것을 ‘함수를 정의(define)한다’라고 표현

```
def my_hello():  
    print('Hello World')
```

- 함수를 사용하는 것을 ‘함수를 호출(call)한다’라고 표현

```
my_hello()
```

Hello World

3) 매개변수가 있는 함수 정의

1절. 함수의 정의 및 사용 > 1.1. 함수 정의하기

- 함수가 실행하기 위해 어떤 값을 받아야 한다면 매개변수를 선언
- 매개변수는 일반 변수와 동일한 사용법과 특징을 가지고 있음
- 매개변수는 함수 안에서 사용할 값을 받는 변수
- 매개변수의 이름은 변수 이름 만드는 규칙에만 맞으면 자유롭게 선언할 수 있음
- 함수 정의 시 괄호('('와 ')') 안에 선언한 이름으로 함수 안에서 사용할 수 있음
- 매개변수는 여러 개 사용할 수 있음, 그러나 함수 호출 시 인수의 개수는 함수 정의 시 매개변수의 개수와 일치해야 함

```
def my_add(num1, num2):  
    print(num1 + num2)
```

```
my_add(3, 5)
```

8

1.2. docstring

1절. 함수의 정의 및 사용

- 함수 본문의 첫 번째 문장에 문자열을 포함
- 함수의 설명서 문자열 또는 docstring
- 겹따옴표 3개(''')와 """) 또는 홑따옴표 3개(''')를 이용
- 문자열 스트링이 여러 개 일 경우 맨 처음의 문자열만 독스트링이 됨

```
def my_function():  
    """함수의 첫 라인에 독스트링을  
    포함시킬 수 있습니다.  
    """  
  
    pass
```

```
print(my_function.__doc__)
```

함수의 첫 라인에 독스트링을
포함시킬 수 있습니다.

1.3. 함수 정의하고 호출하기 예

1절. 함수의 정의 및 사용

- 숫자 하나를 입력 받고 입력 받은 숫자까지 피보나치수열을 출력

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

```
fibonacci(200)
```

0 1 1 2 3 5 8 13 21 34 55 89 144

```
fibonacci(2000)
```

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

1.4 지역변수와 전역변수

1절. 함수의 정의 및 사용

| 지역 변수 | 전역 변수 |
|---|---|
| <pre>def func_a(): num = 10 print(num) # 10</pre> <p>↑↓ 생명주기</p> <pre>def func_b(): print(num) # 에러</pre> | <pre>num = 20 def func_a(): print(num) # 20</pre> <p>↑↓ 생명주기</p> <pre>def func_b(): print(num) # 20</pre> |

지역 심볼 테이블

num = 10

전역 심볼 테이블

num = 20

1.5. 변수의 참조

1절. 함수의 정의 및 사용

- 함수 안에 정의된 변수들을 지역변수(Local Variable)
- 로컬 심볼 테이블(Local Symbol Table)
 - 함수가 실행될 때에 지역변수들은 함수 실행을 위한 특별한 영역에 저장
 - 함수가 실행될 때 함수내의 모든 지역변수들은 해당 함수의 로컬 심볼 테이블에 값을 저장
- 전역 테이블(Global Symbol Table)
 - 함수 밖에 정의된 전역변수들을 저장하는 공간
- **변수의 값을 조회 순서**
 - 먼저 로컬 심볼 테이블 > 전역 심볼 테이블 > 내장 된 이름 테이블

1.5. 변수의 참조

1절. 함수의 정의 및 사용

- 다음 코드는 함수 내에서 전역변수를 참조하고 있음
- func1() 함수 내에 global_var 변수가 선언되어 있지 않으므로 로컬 심볼 테이블에서 값을 찾을 수 없음
- 전역 심볼 테이블에서 global_var 변수를 찾아 출력함

```
global_var = 100
```

```
def func1():  
    print(global_var)
```

```
func1()
```

100

1.6. 변수의 참조

1절. 함수의 정의 및 사용

- 반대로 함수 안에 선언한 변수를 함수 밖에서 참조할 수 없음

```
def func2():
    local_var = 200
    print(local_var)
```

```
func2()
```

200

```
print(local_var)
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-20-3bfff76e6cd3> in <module>
----> 1 print(local_var)
```

NameError: name 'local_var' is not defined

1.6. Lexical 특성

1절. 함수의 정의 및 사용

```
g_var = 100
```

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

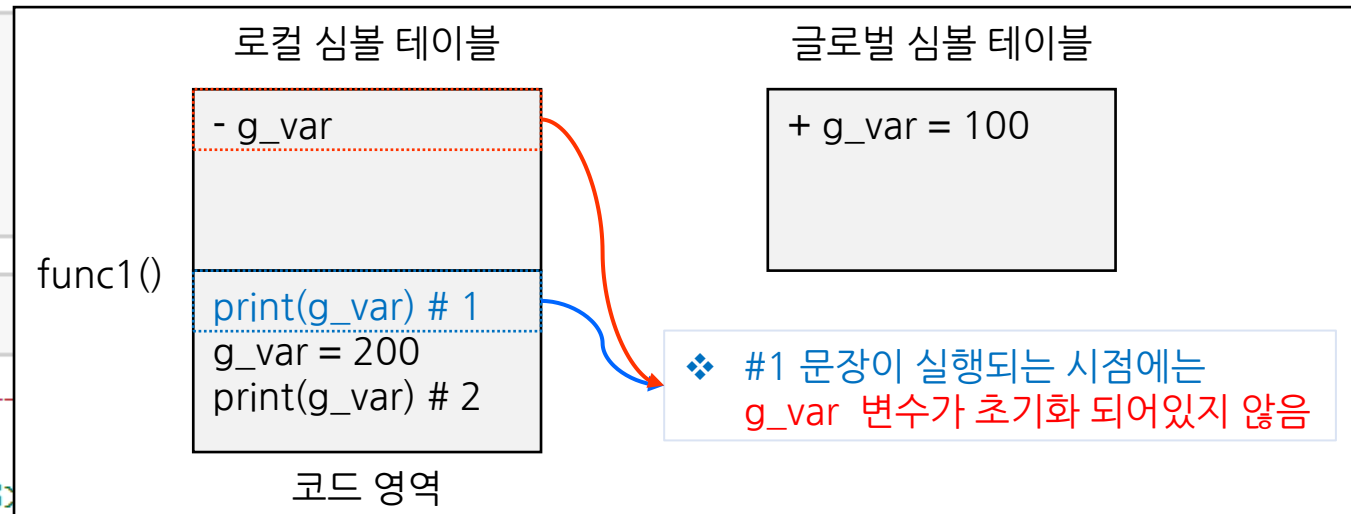
```
func1()
```

UnboundLocalError

```
<ipython-input-19-d88c41ef3303>
--> 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



1.7. 전역변수 수정

1절. 함수의 정의 및 사용

```
g_var2 = 100
```

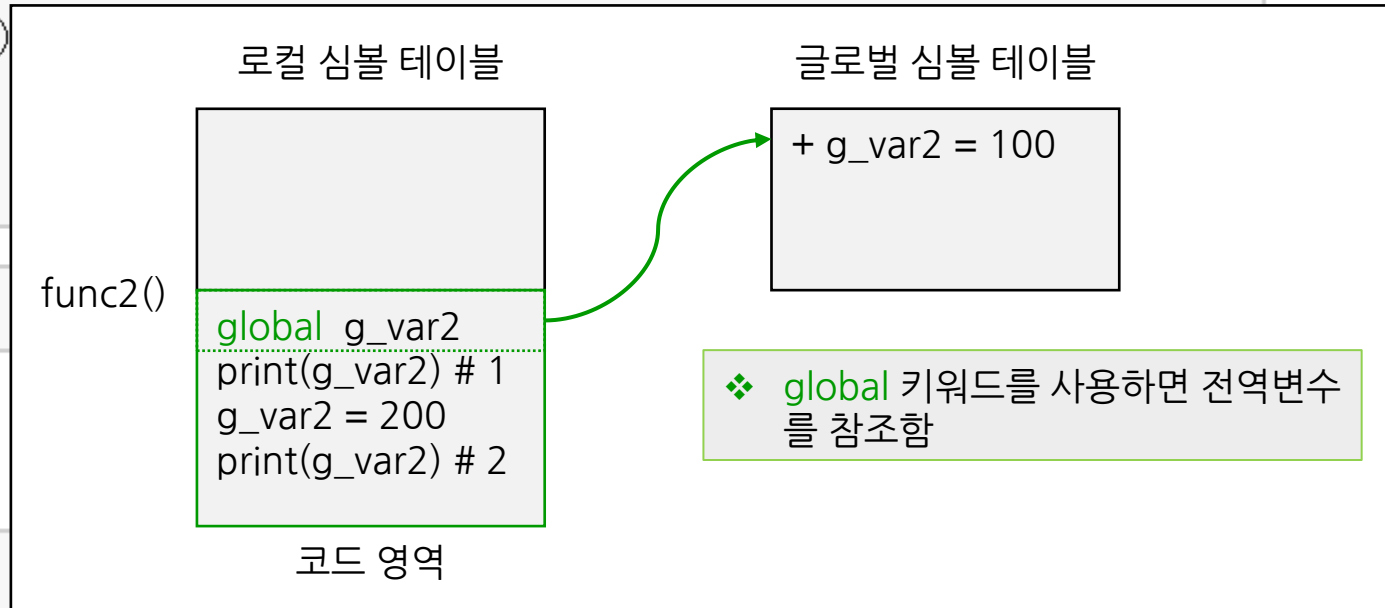
```
def func2():
    global g_var2
    print("before", g_var2)
    g_var2 = 200
    print("after", g_var2)
```

```
func2()
```

```
before 100
after 200
```

```
print(g_var2)
```

```
200
```



1.8. 값에 의한 호출

1절. 함수의 정의 및 사용

```
foo = 100
```

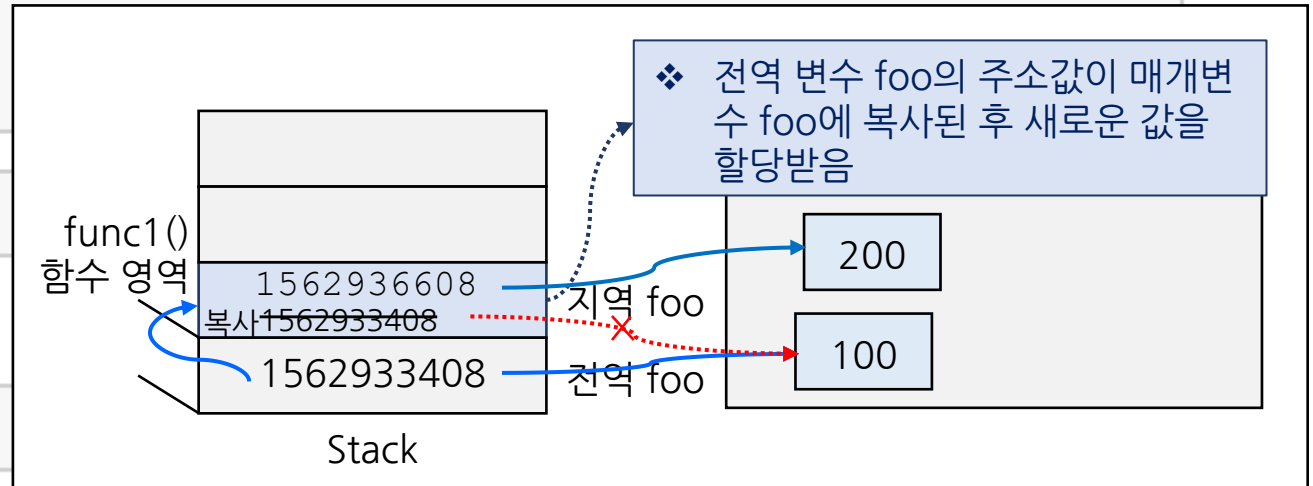
```
def func1(foo):
    foo = 200
    print(foo)
```

```
func1(foo)
```

```
200
```

```
foo
```

```
100
```



1.9. 참조에 의한 호출

1절. 함수의 정의 및 사용

```
L = [1, 2, 3, 4, 5]
id(L)
```

2532935735816

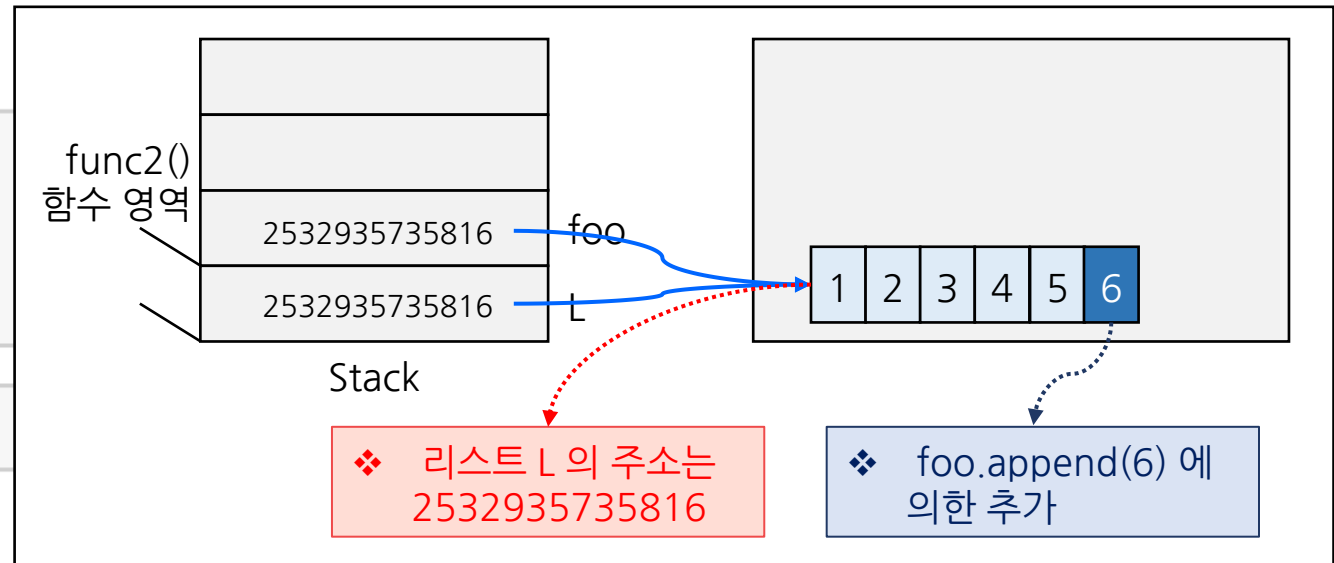
```
def func1(foo):
    foo.append(6)
    print(id(foo))
    print(foo)
```

```
func1(L)
```

2532935735816
[1, 2, 3, 4, 5, 6]

L

[1, 2, 3, 4, 5, 6]



1.10. 함수 이름 변경

1절. 함수의 정의 및 사용

- 함수 정의는 현재 심볼 테이블에 함수 이름을 지정
- 함수 이름의 값은 인터프리터가 사용자 정의 함수로 인식하는 유형

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

```
fibo = fibonacci
```

```
fibonacci(2000)
```

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

```
id(fibonacci), id(fibo)
```

```
(2532935712552, 2532935712552)
```

1.11. 함수 이름 변경과 실행 결과 저장

1절. 함수의 정의 및 사용

- 함수를 다른 이름의 변수에 할당하는 것

```
fibol = fibonacci
```

```
type(fibol)
```

```
function
```

```
print(fibol)
```

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

```
fibol(2000)
```

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

1.11. 함수 이름 변경과 실행 결과 저장

1절. 함수의 정의 및 사용

● 함수 실행 결과를 저장

```
fibonacci(2000)
```

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

```
type(fibo2)
```

NoneType

```
print(fibo2)
```

None

```
fibo2(2000)
```

TypeError

Traceback (most recent call last)

```
<ipython-input-63-96dd13014279> in <module>
```

```
----> 1 fibo2(2000)
```

TypeError: 'NoneType' object is not callable

2절. 함수의 실행 결과를 반환하는 return

5장. 함수

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

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

- return
함수의 결과 값을 반환하기 위한 키워드
- *return_value*
함수가 실행한 결과를 반환하는 값
변수이름 또는 표현식으로 사용
변수 또는 표현식의 결과는 모든 자료형 가능

2.1. 반환 값이 없는 함수

2절. 함수의 실행 결과를 반환하는 return

- fibonacci() 함수는 리턴 값이 없기 때문에 f200 변수에는 아무것도 저장되지 않음

```
f200 = fibonacci(200)
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144
```

```
f200
```

```
f200(10)
```

TypeError

Traceback (most recent call last)

```
<ipython-input-67-a380534fcffd> in <module>
```

```
----> 1 f200(10)
```

TypeError: 'NoneType' object is not callable

2.2. 반환 값이 있는 함수

2절. 함수의 실행 결과를 반환하는 return

```
def fibonacci2(n):  
    """n값 미만까지 피보나치 수열을 출력합니다."""  
    result = []  
    a, b = 0, 1  
    while a < n:  
        result.append(a)  
        a, b = b, a+b  
    return result
```

```
f100 = fibonacci2(100)
```

```
f100
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

```
print(fibonacci2(100))
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

```
print(f100)
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

2.3. 여러 개 값 반환

2절. 함수의 실행 결과를 반환하는 return

- 여러 개 값을 반환하면 그 값들은 튜플에 저장되어 반환
- 반환 값을 하나의 튜플 변수에 저장하거나 함수가 반환하는 값의 개수 만큼 변수를 선언하여 반환 값이 저장되도록 할 수 있음

```
def swap(a,b):  
    return b,a
```

```
a = swap(1,2)  
type(a)
```

tuple

```
print(a)
```

(2, 1)

```
x, y = swap(1,2)
```

```
print(x, y)
```

2 1

꼭 알아야 하는 함수의 매개변수의 주요 주제들

3절. 함수 매개변수

- 매개변수가 기본값을 가질 때 : 기본값이 있는 매개변수는 선택적으로 사용 가능
- 매개변수의 기본값이 스칼라 변수일 때
- 매개변수의 기본값이 변경가능한 객체일 경우
- 키워드 인수 : 함수 호출 시 인수가 매개변수 이름을 갖는 것
- 튜플 매개변수 : 가변인수를 사용할 수 있음 *args
- 딕셔너리 매개변수 **kwargs
- 매개변수와 인수의 순서
 - 인수의 순서 : 순서인수, 튜플인수, 키워드 인수, 딕셔너리 인수
- 튜플 인수 언패킹
- 딕셔너리 인수 언패킹

3.1. 기본 값을 갖는 매개변수

3절. 함수 매개변수

- 하나 이상의 인수에 대한 기본값을 지정
- 가변인수 : 함수를 정의할 때에 허용하도록 정의 인수의 수 보다 적은 인수로 호출 할 수 있음

```
def make_url(ip, port=80):  
    return "http://{0}:{1}".format(ip, port)
```

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

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

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

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

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

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

3.2. 기본 변수를 갖는 매개변수

3절. 함수 매개변수

- 함수의 매개변수가 기본 값으로 변수 이름을 가질 수 있음
- 이때 기본 값은 함수가 정의되는 지점에 평가됨

```
1 i = 5
2 def func2(arg=i):
3     print(arg)
```

```
1 i = 6
2 func2()
```

5

- 함수가 정의될 때 func2() 함수의 arg 변수에 전달되는 값은 5
- i 값이 바뀌더라도 함수를 호출할 때에 arg 인수를 지정하지 않는다면 arg의 기본 값은 5가 됨

변경 가능한 객체를 기본변수로 갖는 매개변수

3절. 함수 매개변수

- 기본 변수가 리스트, 딕셔너리 또는 대부분의 클래스의 인스턴스와 같은 변경 가능한 객체 일 때 호출시마다 전달된 인수를 사용

```
1 def func3(a, L=[]):  
2     L.append(a)  
3     return L  
4
```

```
1 print(func3(1))
```

[1]

```
1 print(func3(2))
```

[1, 2]

```
1 print(func3(3))
```

[1, 2, 3]

변경 가능한 객체를 기본변수로 갖는 매개변수

3절. 함수 매개변수

- 기본 변수를 함수들 호출 사이에 공유하지 않으려면 다음과 같은 형식으로 함수를 작성

```
1 def func4(a, L=None):  
2     if L is None:  
3         L = []  
4     L.append(a)  
5     return L  
6
```

```
1 print(func4(1))
```

[1]

```
1 print(func4(2))
```

[2]

3.3. 키워드 인수

3절. 함수 매개변수

- 함수를 호출 할 때에 kwarg=value 형식의 인수를 사용하여 호출 할 수 있음



3.3. 키워드 인수

3절. 함수 매개변수

```
1 def func4(a, L=None):  
2     if L is None:  
3         L = []  
4     L.append(a)  
5     return L  
6
```

```
1 list_ = []
```

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

[10]

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

[10, 20]

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

[10, 20, 30]

3.3. 키워드 인수

3절. 함수 매개변수

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

```
1 func4(L=list_, a=40)
```

[10, 20, 30, 40]

```
1 func4(a=60)
```

[60]

```
1 func4()
```

```

TypeError                                Traceback (most recent call last)
<ipython-input-27-0e6ad11a93c1> in <module>()
----> 1 func4()
```

TypeError: func4() missing 1 required positional argument: 'a'

3.3. 키워드 인수

3절. 함수 매개변수

```
1 func4(50, a=60)
```

TypeError Traceback (most recent call last)

```
<ipython-input-29-4a36985e9362> in <module>()
--> 1 func4(50, a=60)
```

```
TypeError: func4() got multiple values for argument 'a'
```

필수 매개변수에 매개변수의 이름을 지정하지 않은 인수와 매개변수 이름을 지정한 인수를 동시에 사용할 수 없음

```
1 func4(b=70)
```

TypeError Traceback (most recent call last)

```
<ipython-input-30-e445497b0d28> in <module>()
--> 1 func4(b=70)
```

```
TypeError: func4() got an unexpected keyword argument 'b'
```

함수 정의 시 존재하지 않는 매개변수 이름을 사용할 수

```
1 func4(L=list_, 70)
```

```
File "<ipython-input-31-79b4f6d43508>", line 1
func4(L=list_, 70)
      ^
```

키워드 인수는 순서 인수 뒤에 와야 함

SyntaxError: positional argument follows keyword argument

3.4. 튜플 매개변수를 이용한 가변인수 설정

3절. 함수 매개변수

- 매개변수 앞에 *를 붙여 정의
- 인수들이 튜플에 저장되어 전달

```
1 def add(*args):  
2     sum = 0  
3     for num in args:  
4         sum = sum + num  
5     return sum  
6
```

```
1 add(10, 20)
```

30

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

60

```
1 add(10, 20, 30, 40)
```

100

튜플 인수의 순서

3절. 함수 매개변수 > 3.4. 튜플 매개변수를 이용한 가변인수 설정

- 가변 인수 앞에 0 개 이상의 일반 인수가 올 수 있음
- 가변 인수는 함수에 전달되는 나머지 모든 입력 인수를 스쿠핑하기 때문에 형식 인수 목록의 마지막에 음
- *args 매개변수 다음에 나오는 형식적 매개 변수는 '키워드 전용' 인수

```
def concat(*args, sep):
    return sep.join(args)
```

```
concat("earth", "mars", "venus", "/")
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-92-513211c6e4fd> in <module>
----> 1 concat("earth", "mars", "venus", "/")
```

```
TypeError: concat() missing 1 required keyword-only argument: 'sep'
```

튜플 인수의 순서

3절. 함수 매개변수 > 3.4. 튜플 매개변수를 이용한 가변인수 설정

```
def concat(*args, sep):
    return sep.join(args)
```

파라미터의
이름을 지정

```
concat("earth", "mars", "venus", sep="/")
```

```
'earth/mars/venus'
```

순서인수가 앞에 올수 있
도록 파라미터 선언

```
def concat(sep, *args):
    return sep.join(args)
```

```
concat("/", "earth", "mars", "venus")
```

```
'earth/mars/venus'
```

디폴트 파라미터로 선언

```
def concat(*args, sep="/"):
```

```
    return sep.join(args)
```

```
concat("earth", "mars", "venus")
```

```
'earth/mars/venus'
```

3.5. 딕셔너리 인수

3절. 함수 매개변수

- `**name` 형식의 최종 형식 매개 변수가 있으면 형식 매개 변수에 해당하는 것을 제외하고 모든 키워드 인수가 들어있는 딕셔너리를 받음

```
1 def func5(**data):  
2     for item in data.items():  
3         print(item)  
4
```

```
1 func5(name="Ki IDong", age=30, address="서울시 강남구")
```

('name', 'Ki IDong')

('age', 30)

('address', '서울시 강남구')

3.5. 딕셔너리 매개변수

3절. 함수 매개변수

- 딕셔너리 인수는 위치 인수, 키워드 인수, 그리고 튜플을 받는 *name 형식의 가변 인수와 같이 사용될 수 있음
- *name은 **name 앞에 나와야 함

```
1 def func6(a, *b, **c):  
2     print(a)  
3     print(b)  
4     print(c)  
5
```

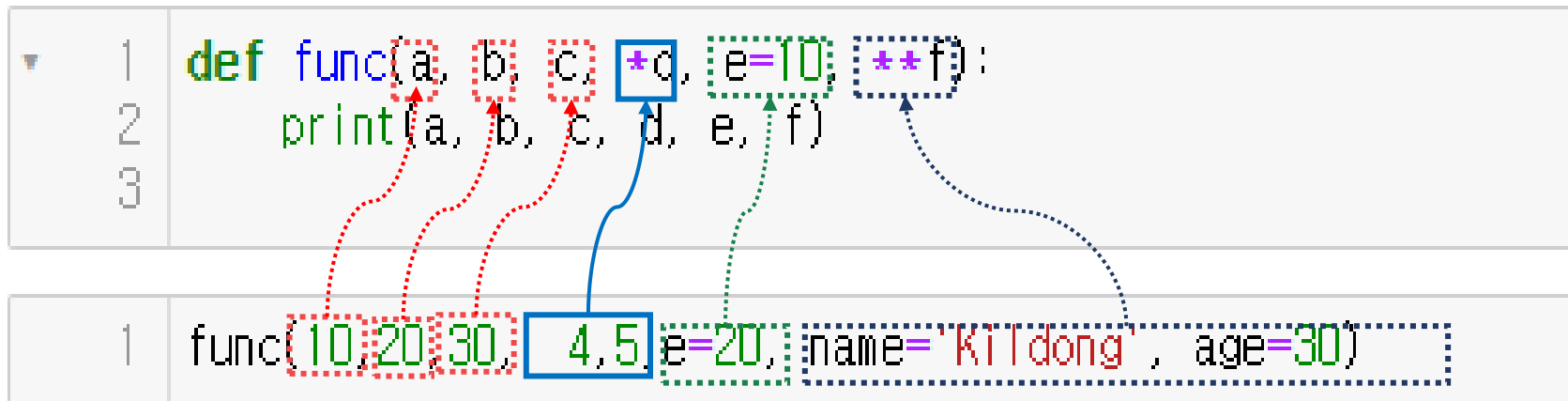
```
1 func6(10, 1, 2, 3, name='KiIDong', age=20)
```

```
10  
(1, 2, 3)  
{'name': 'KiIDong', 'age': 20}
```

매개변수의 순서

3절. 함수 매개변수

- 함수 선언 시 매개변수의 순서는 순서 인자, 튜플 인자, 키워드 인자, 딕셔너리 인자를 받을 수 있는 순서로 정의



10 20 30 (4, 5) 20 {'name': 'Kildong', 'age': 30}

1) 튜플 인수 언패킹

3절. 함수 매개변수 > 3.6. 인수 언패킹

- 인수가 이미 목록이나 튜플에 있지만 별도의 위치 인수가 필요한 함수 호출에 대해 압축을 풀어야하는 경우에는 반대 상황이 발생
- 예를 들어 내장 된 range() 함수는 별도의 start 및 stop 인수를 필요
- 별도로 사용할 수 없는 경우에는 * 연산자로 함수 호출을 작성하여 인

```
1 def add(*args):  
2     sum = 0  
3     for num in args:  
4         sum = sum + num  
5     return sum  
6
```

```
1 numbers = (1,2,3,4,5,6,7,8,9,10)
```


1) 튜플 인수 언패킹

3절. 함수 매개변수 > 3.6. 인수 언패킹

```
1 add(numbers)
```

numbers 변수를 add() 함수의 인자로
직접 넣을 수 없음

TypeError

Traceback (most recent call last)

```
<ipython-input-48-283aadd9c8e6> in <module>()
----> 1 add(numbers)
```

```
----> 1 add(numbers)
```

```
<ipython-input-46-8e24878e813e> in add(*args)
```

```
2     sum = 0
```

```
3     for num in args:
```

```
----> 4         sum = sum + num
```

```
5     return sum
```

TypeError: unsupported operand type(s) for +: 'int' and 'tuple'

```
1 add(*numbers)
```

변수 앞에 *를 붙여 튜플을 언패킹 하여
전달

55

2) 딕셔너리 인수 언패킹

3절. 함수 매개변수 > 3.6. 인수 언패킹

● 딕셔너리 데이터는 ** 연산자로 키워드 인수를 전달

```
1 def func5(**data):
2     for item in data.items():
3         print(item)
4
```

```
1 custInfo = {"name": "Ki IDong", "age": 30, "address": "서울시 강남구"}
```

```
1 func5(custInfo)
```

```
TypeError                                Traceback (most recent call last)
<ipython-input-52-920d5b2e63d7> in <module>()
----> 1 func5(custInfo)
```

TypeError: func5() takes 0 positional arguments but 1 was given

```
1 func5(**custInfo)
```

```
('name', 'Ki IDong')
('age', 30)
('address', '서울시 강남구')
```

4.1. 람다식

4절. 람다식

- 람다식은 작은 익명함수를 의미함
- 작은 익명 함수는 lambda 키워드로 만들 수 있음

```
lambda variable_define : statements
```

- *variable_define*
함수의 인수를 정의
- *statement*
함수가 실행할 문장을 작성
한 문장만 작성할 수 있음
return 구문이 없어도 statement의 결과를 반환

람다 식 사용 예

4절. 람다식

- 람다식은 함수 객체가 필요한 곳이면 어디서든지 사용할 수 있음
- 람다식은 한 개의 문장(표현식)만 작성할 수 있음
- 중첩 된 함수 정의와 마찬가지로 람다 함수는 포함 된 범위(scope)의 변수들을 참조 할 수 있음

```
1 def add(a, b):  
2     return a+b  
3
```

```
1 add(1,2)
```

3

```
1 add2 = lambda a, b: a+b
```

```
1 add2(1,2)
```

3

4.2. 리턴문에 람다식 사용

4절. 람다식

- 리턴문에 함수의 이름을 사용
- 리턴문의 함수는 반드시 지역함수일 필요는 없음

```
1 def make_incrementor1(n):  
2     def add(a):  
3         return a+n  
4     return add  
5
```

```
1 f2 = make_incrementor1(10)
```

```
1 f2(5)
```

15

```
1 def make_incrementor2(n):  
2     return lambda x: x+n  
3
```

```
1 f = make_incrementor2(10)
```

```
1 f(5)
```

15

4.3. 함수 인수에 람다식 사용

4절. 람다식

- 람다식의 다른 용도는 작은 함수를 인수로 전달하는 것

```
1 data = [1, 2, 3, 4, 5]
2 def odd2(data):
3     if data%2 == 0:
4         return True
5     else:
6         return False
7
```

```
1 r = filter(odd2, data)
```

```
1 type(r)
```

filter

```
1 list(r)
```

[2, 4]

```
1 data = [1, 2, 3, 4, 5]
2 list(filter(lambda x: x%2==1, data))
```

[1, 3, 5]

5.1. 파이썬 내장 함수

5절. 파이썬 내장 함수

- 파이썬 내장함수는 **import 하지 않고 즉시 사용 가능한 함수**
- 내장 함수명은 일종의 키워드로 간주되므로 식별자로 사용하는 것은 피하여야 함

입출력 관련 함수

5절. 파이썬 내장 함수

| 함수명 | 기능 |
|--|--|
| <code>print(x)</code> | 객체를 문자열로 표시한다. |
| <code>input([prompt])</code> | 사용자 입력을 문자열로 반환한다. |
| <code>help([x])</code> | x에 대한 도움말을 출력한다. |
| <code>globals()</code> | 전역 변수의 리스트를 반환한다. |
| <code>locals()</code> 또는 <code>vars()</code> | 지역 변수의 리스트를 반환한다. <code>__dict__</code> 어트리뷰트 ²⁵⁾ 를 반환한다. |
| <code>del(x)</code> 혹은 <code>del x</code> | 객체를 변수 공간에서 삭제한다. |
| <code>eval(expr)</code> | 값을 구한다. |
| <code>exec(obj)</code> | 파이썬 명령을 실행시킨다. |
| <code>open(filename[,mode]))</code> | 파일을 연다 |

기본 자료형의 생성과 변환 함수

5절. 파이썬 내장 함수

| 함수명 | 기능 |
|----------------------------------|---------------------------------|
| <code>object()</code> | 새로운 object (모든 객체의 base)를 생성한다. |
| <code>bool(obj)</code> | 객체의 진리값을 반환한다. |
| <code>int(obj)</code> | 문자열 형태의 숫자나 실수를 정수로 변환한다. |
| <code>float(obj)</code> | 문자열 형태의 숫자나 정수를 실수로 변환한다. |
| <code>complex(re [, img])</code> | 문자열이나 주어진 숫자로 복소수를 생성한다. |

기본 자료형의 정보를 얻는 함수

5절. 파이썬 내장 함수

| 함수명 | 기능 |
|---|---|
| <code>type(obj)</code> | 객체의 형을 반환한다. |
| <code>dir(obj)</code> | 객체가 가진 함수와 변수들을 리스트 형태로 반환한다. |
| <code>repr(obj)</code> <code>ascii(obj)</code> | <code>eval()</code> 함수로 다시 객체를 복원할 수 있는 문자열 생성 <code>repr()</code> 과 유사하나 non-ascii문자는 escape한다. |
| <code>id(obj)</code> | 객체의 고유번호(int형)을 반환한다. |
| <code>hash(obj)</code> | 객체의 해시값(int형)을 반환. (같은 값이면 해시도 같다.) |
| <code>chr(num)</code> <code>ord(str)</code> | ASCII 값을 문자로 반환 한 문자의 ASCII 값을 반환 |
| <code>isinstance(obj, className)</code> | 객체가 클래스의 인스턴스인지를 판단한다. |
| <code>issubclass(class, classinfo)</code> | <code>class</code> 가 <code>classinfo</code> 의 서브클래스 일때 True 반환 |

열거형 정보를 얻는 함수

5절. 파이썬 내장 함수

- enumerate() 함수는 인덱스와 아이템을 하나씩 튜플 형식으로 반환
- filter() 함수는 iterable 객체의 아이템들 중에서 func() 함수의 결과가 True인 경우의 아이템들만 묶어서 반환

| 함수명 | 기능 |
|--|--|
| len(seq) | 시퀀스형을 받아서 그 길이를 반환한다. |
| iter(obj [,sentinel]) next(iterator) | 객체의 이터레이터(iterator)를 반환한다. 이터레이터의 현재 요소를 반환하고 포인터를 하나 넘긴다. |
| enumerate(iterable, start=0) | 이터러블에서 enumerate 형을 반환한다. 입력값으로 시퀀스자료형(리스트, 튜플, 문자열)을 입력을 받는다. |
| sorted(iterable[,key][,reverse]) | 정렬된 *리스트*를 반환 |
| reversed(seq) | 역순으로 된 *iterator*를 반환한다. |
| filter(func, iterable) | iterable의 각 요소 중 func()의 반환 값이 참인 것만을 묶어서 이터레이터로 반환. |
| map(func, iterable) | iterable의 각 요소를 func()의 반환값으로 매핑해서 이터레이터로 반환. |

산술/논리 연산과 관련된 함수

5절. 파이썬 내장 함수

| 함수명 | 기능 |
|---------------------------------------|--|
| hex(n) | 정수 n의 16진수 값을 구해서 '문자열'로 반환한다. |
| oct(n) | 정수 n의 8진수 값을 구해서 '문자열'로 반환한다. |
| bin(n) | 정수 n의 2진수 값을 구해서 '문자열'로 반환한다. |
| abs(n) | 절대값을 구한다. 복소수의 경우 크기를 구한다. |
| pow(x,y[,z]) | 거듭제곱을 구한다. pow(x,y)은 $x**y$ 와 같다. |
| divmod(a,b) | a를 b로 나눈 (몫, 나머지)를 구한다. 튜플 반환. |
| all(iterable) | iterable의 모든 요소가 True 일 경우 True를 반환. |
| any(iterable) | iterable의 하나 이상의 요소가 True 일 경우 True를 반환. |
| max(iterable) max(arg1, arg2, ...) | 최대값을 구한다. |
| min(iterable) min(arg1, arg2, ...) | 최소값을 구한다. |
| round() | 반올림을 한다. |

연습문제(실습형)

1. 함수의 인자로 리스트를 받은 후 리스트 내에 있는 모든 정수값에 대한 최대값과 최소값을 리턴하는 함수를 작성하세요
 - `def get_max_min(data_list):`
 - `get_max_min = lambda ~`
2. 체질량 지수(Body Mass Index, BMI)는 체중과 키를 이용해 비만도를 나타내는 지수로 아래의 수식에 의해 계산됩니다. 함수의 인자로 체중(kg)과 신장(m)를 입력받은 후 BMI값에 따라 ‘마른 체형’, ‘표준’, ‘비만’, ‘고도비만’ 중 하나의 상태를 출력하는 함수를 구현해 보세요.
 - `getBMI(kg, m):`
 - BMI지수 = 체중(kg) / 신장(m)의 제곱
 - BMI < 18.5 : 마른체형
 - 18.5 ≤ BMI < 25 : 표준
 - 25 ≤ BMI < 30 : 비만
 - BMI ≥ 30 : 고도비만

연습문제 (실습형)

3. 직각삼각형의 밑변과 높이를 입력받은 후 삼각형의 면적과 둘레를 계산하는 함수를 작성하세요
 - 리턴값은 면적과 둘레를 return하도록 구현하세요
 - `math.sqrt()` : 제곱근을 구하는 함수이용 (ex)`math.sqrt(25)` : 5.0
 - `def get_triangle(width, height):`
4. 함수의 인자로 시작과 끝 숫자가 주어질 때 시작부터 끝까지의 모든 정수값의 합을 리턴하는 함수를 작성하세요(시작값과 끝값 포함).
 - `def mysum(from, end):`
5. 함수의 인자로 문자열을 포함하는 리스트가 입력될 때 각 문자열의 첫 세글자로만 구성된 리스트를 리턴하는 함수와 람다식을 작성하세요
 - 예를 들어 함수의 입력으로 `['Seoul', 'Daegu', 'Kwangju', 'Jeju']`가 입력될 때 함수의 리턴값은 `['Seo', 'Dae', 'Kwa', 'Jej']`
 - `def get_abbrs(lst):`

연습문제(실습형)

6. 다음 코드를 람다 함수 형태로 수정할 때 알맞은 코드를 작성하시오

```
def f(x, y):  
    return x ** y
```

7. ex = [1,2,3,4,5]를 [1,4,9,16,25]의 결과를 얻을 수 있도록 람다함수와 map()함수를 사용하여 구현과 리스트 컴프리헨션으로 구현하시오

```
>>> ex = [1, 2, 3, 4, 5]  
>>> _____  
[1, 4, 9, 16, 25]
```

```
>>> ex = [1, 2, 3, 4, 5]  
>>> _____  
[1, 4, 9, 16, 25]
```

연습문제 (실습형)

8. 다음 코드를 각각 실행하면 서로 다른 결과가 나온다. 이런 결과가 나오는 이유를 서술하시오

```
>>> a = [1, 2, 3]
>>> print(*a)
1 2 3
>>> print(a)
[1, 2, 3]
```

9. 다음 코드의 실행 결과는?

```
>>> date_info={'year':2024, 'month':9, 'day':6}
>>> result = '{year}-{month}-{day}'.format(**date_info)
>>> result
```

10. n개의 벡터(리스트나 튜플, 셋등)의 크기들이 같은지 여부를 reutrn하는
vector_size_check(*vector _var) 를 한 줄의 코드(리스트 컴프리헨션 이용)로 작성하시오.

11. 다음과 같은 결과를 얻기 위해 하나의 스칼라값을 리스트나 튜플, 셋등 벡터에 곱하는 코드를 작성하시오.(단 입력되는 벡터의 크기는 일정하지 않음)

```
>>> scalar_vector_product(5, [1,2,3,4])
[5, 10, 15, 20]
```


연습문제(서술형)

1. 다음 코드의 실행 결과는?

```
pairs = [(1, 'd'), (2, 'c'), (3, 'b'), (4, 'a')]
pairs.sort(key=lambda pair: pair[1])
pairs
```

- ① [(1, 'd'), (2, 'c'), (3, 'b'), (4, 'a')]
- ② [(4, 'a'), (3, 'b'), (2, 'c'), (1, 'd')]
- ③ [(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
- ④ 프로그램 오류

연습문제(서술형)

2. 함수 영역과 변수들의 참조 영역에 따른 동작을 보여주기 위한 문제입니다. 오류가 발생하는 라인을 찾고 문제를 해결하기 위한 코드를 작성하며, 실행 결과를 쓰세요.

```
1. a = 10
2. def sub() :
3.     a += 1
4.     print(a, end='')
5.
6. def func():
7.     for i in range(2):
8.         a = 5
9.         a += 1
10.        print(a, end='')
11.        sub()
12.
13. a += 1
14. func()
```

- 오류가 발생하는 라인 :
- 문제 해결을 위한 코드 :
- 코드 수정 후 실행 결과 :

5. 연습문제(서술형)

3. 다음 구문을 실행한 결과는

```
var = 100  
def func(var):  
    var = 200  
func(var)  
print(var)
```

4. 다음 구문의 실행 결과는?

```
def my_func(func, *args):  
    return func(*args)  
import numpy as np  
my_func(np.add, 2, 3)
```

5. 연습문제(서술형)

5. 다음 구문의 실행결과는?

```
def my_func(func, *args):
```

```
    return func(*args)
```

```
my_func(lambda a, b : a**b, 3, 2)
```

6. 파이썬 함수에 대한 설명 중 잘못된 것은?

- ① 파이썬의 함수는 중복 정의해 사용할 수 있다.
- ② 파이썬의 함수 매개변수는 기본값을 가질 수 있다.
- ③ `**args` 형식의 매개변수가 있으면 키워드 인수는 딕셔너리 형식으로 받는다.
- ④ 함수를 호출할 때 매개변수 이름이 없는 인수는 매개변수 이름이 있는 인수보다 앞에 와야 한다.

5. 연습문제(서술형)

7. 다음과 같은 구문이 있을 경우 오류가 발생하는 함수 호출은?

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

- ① func(10, list_)
- ② func(20, L=list_)
- ③ func(a=30)
- ④ func([], a=40)

연습문제(서술형)

8. 다음 보기가 설명하는 것을 작성하기 위한 키워드를 쓰세요.(영문 소문자로 쓰세요)
- 작은 익명 함수를 의미합니다.
 - 함수가 실행할 문장이 한 문장일 경우에 만들어 사용할 수 있습니다.
 - 다른 함수의 인수 또는 리턴값에 사용할 수 있습니다.

1부 교육목표 체크리스트

1부의 내용을 공부한 전과 후에 표에 있는 항목들을 체크해 보세요. 체크표시가 늘어날수록 여러분의 실력도 늘어납니다

| 교육목표 | | ✓ |
|------|---|---|
| 1 | 파이썬 개발 환경을 구축하고 코드를 작성해서 테스트할 수 있다. | |
| 2 | 파이썬 언어의 특징을 3가지 이상 설명할 수 있다. | |
| 3 | 변수를 선언하고 사용할 수 있다. | |
| 4 | 파이썬 언어의 연산자를 사용하여 계산을 하고 그 결과를 변수에 대입할 수 있다. | |
| 5 | 파이썬 언어의 자료형(정수, 실수, 논리, 문자)을 구분하고 설명할 수 있다. | |
| 6 | 파이썬 언어의 명령문(if, for 등 제어문)의 흐름을 이해하고 기본 구문을 작성할 수 있다. | |
| 7 | 매개변수 선언이 없는 함수를 정의하고 호출할 수 있다. | |
| 8 | 매개변수, 지역변수, 전역변수를 구분할 수 있다. | |
| 9 | 값에 의한 호출과 참조에 의한 호출을 구분할 수 있다. | |
| 10 | 함수의 매개변수 종류를 이해하고 함수 정의 시 활용할 수 있다. | |