

# SW교육 담당교원 역량강화 연수교재

SW교육과 파이썬(Python) 프로그래밍



**부산광역시교육청**

BUSAN METROPOLITAN CITY OFFICE OF EDUCATION

인재개발과



# ■ ■ ■ ■ 목 차 ■ ■ ■ ■

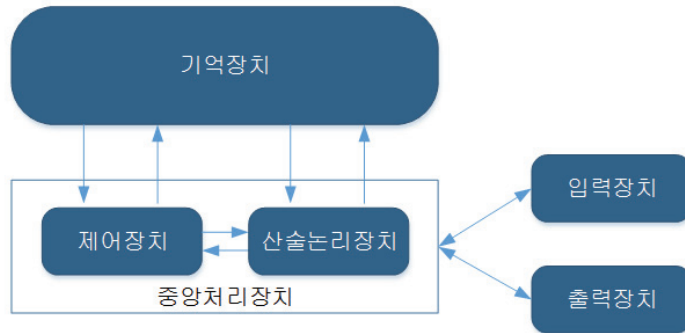
0. 컴퓨터 알기 .....	1
1. 파이썬(Python)의 개요 .....	4
2. 파이썬(Python) 설치하기(윈도우) .....	6
3. 기본 자료형(문자, 숫자) .....	9
4. 리스트 자료형 .....	15
5. 튜플자료형 .....	18
6. 딕셔너리자료형 .....	19
7. 집합자료형 .....	21
8. 참, 거짓 자료형 .....	23
9. 변수 .....	24
10. if .....	26
11. while .....	29
12. for .....	32
13. 함수 .....	35
14. 사용자 입력과 출력 .....	40
15. 파일 읽기 쓰기 .....	42
16. 클래스 .....	44
17. 모듈 .....	49
18. 예외처리 .....	52
19. 응용 예제 .....	54



## 0 컴퓨터 알기

### ■ 폰 노이만 구조

- 오늘날의 컴퓨터들은 모습과 크기, 성능은 다르지만 기본적인 구조는 거의 똑같음.
- 1945년에 존 폰 노이만이 발표한 논문 <EDVAC에 관한 보고서>에서 기술한 컴퓨터의 구조를 그대로 따르고 있기 때문



- EDVAC은 프로그램을 수정하는 일이 ENIAC에 비해 자유로움.
- EDVAC은 명령어를 기억장치에 내장하고 있었기 때문임.
  - ➔ 기억장치에 명령어를 내장하는 컴퓨터를 **내장식 컴퓨터**라고 함.
  - ➔ 내장식 컴퓨터는 폰 노이만의 논문에서 본격적으로 소개되었기 때문에 **폰 노이만 구조(von Neumann Architecture)**라는 이름으로 알려져 있음

### ■ PC의 폰노이만 구조

- 입력 장치 : 키보드와 마우스
- 출력 장치 : 모니터와 프린터
- 중앙처리장치 : 인텔 i7
- 기억장치 : DDR3 DRAM

### ■ 명령 주기(Instruction Cycle)

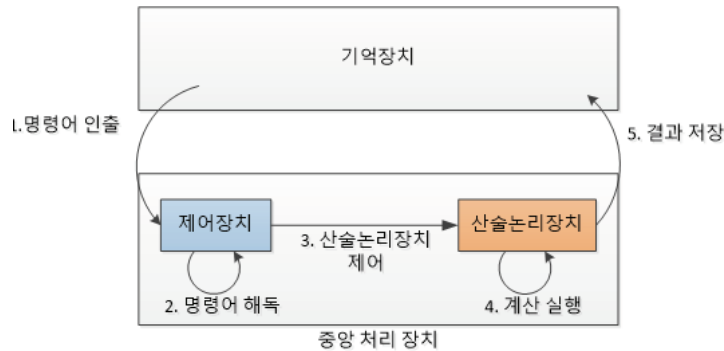
- 기억장치로부터 명령어를 불러오고 해독하며, 실행하는 주기
- CPU는 클럭(Clock)이라 부르는 시계를 갖고 있어서
  - 이 클럭이 움직일 때마다 신호를 보내 명령 주기를 반복시킴.
- CPU가 명령 주기를 빠르게 반복 할수록
  - 컴퓨터는 주어진 시간 내에 할 수 있는 일이 많아짐.
- Hz(헤르츠)라는 단위를 이용하여 표현.
  - CPU의 클럭 주파수가 1GHz : 그 CPU는 1초에 10억회의 명령 주기를 수행하는 성능

### ■ 중앙처리장치

- 컴퓨터가 수행하는 계산은 모두 이 CPU를 통해 이루어짐.
- CPU를 이루는 요소
  - 산술 논리 장치(ALU : Arithmetic Logic Unit)
  - 제어 장치(CU : Control Unit)

## ■ 제어장치와 산술논리장치의 동작 과정

1. 먼저 제어장치가 기억장치로부터 명령어를 가져옵니다(Fetch).
2. 제어장치는 가져온 명령어를 해독(Decode)합니다.
3. 제어장치는 해독한 명령어에 따라  
산술논리장치에 데이터를 옮기고 어떤 연산을 수행할지를 지시합니다.
4. 산술논리장치는 제어장치가 지시한 대로 계산을 수행(Execute)합니다.
5. 그리고 수행한 결과를 기억장치에 다시 저장(Store)합니다.



## ■ 산술논리장치 : 덧셈, 뺄셈, 곱셈, 나눗셈 등의 산술 연산과

참과 거짓(또는 0과 1)을 다루는 논리 연산을 수행하는 회로를 가짐.

## ■ 제어장치 : 산술논리장치를 통제하여 계산을 수행함.

## ■ 기억장치

- 기억장치(Memory) : 데이터와 함께 명령어를 저장하는 역할 수행.
- 메모리에 데이터를 기록하기 위해서는 CPU가 “쓰기 요청”을 보내고 반대로 메모리로부터 데이터를 가져오기 위해서는 “읽기 요청”을 보냄.  
메모리의 성능은 이 요청에 응답하기까지의 시간과 이러한 읽기/쓰기 요청을 연속적으로 처리하는 주기에 의해 결정
- 캐시(Cache) 메모리 : CPU와 가장 가까이 있는 기억장치. 성능이 좋지만 가격이 비쌈.
- 주기억 장치 : 캐시 메모리 아래에 위치한 기억장치. 대개 램(RAM)으로 구성
- 보조기억장치  
CPU로부터 가장 멀리 떨어져 있는 기억장치. 성능이 낮지만 가격도 낮음.  
하드디스크가 대세였지만, 최근 플래시 메모리와 SSD(Solid State Drive)가 부상.

## ■ 입력과 출력

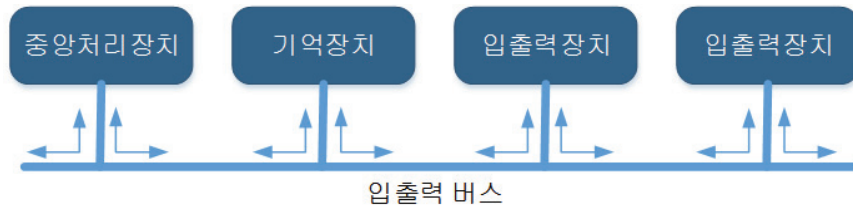
입력장치 : 마우스, 키보드, 터치스크린, 마이크, 게임패드, 동작 인식 장치

출력장치 : 모니터, 프린터, 3D 프린터, 스피커, 빔 프로젝터, E-Ink 전자 종이

## ■ 입출력 버스(I/O BUS)

버스 : 컴퓨터의 정보 전송 회로

중앙처리장치와 기억장치, 그리고 입출력 장치는 아래와 같이 버스로 연결되어 있음.

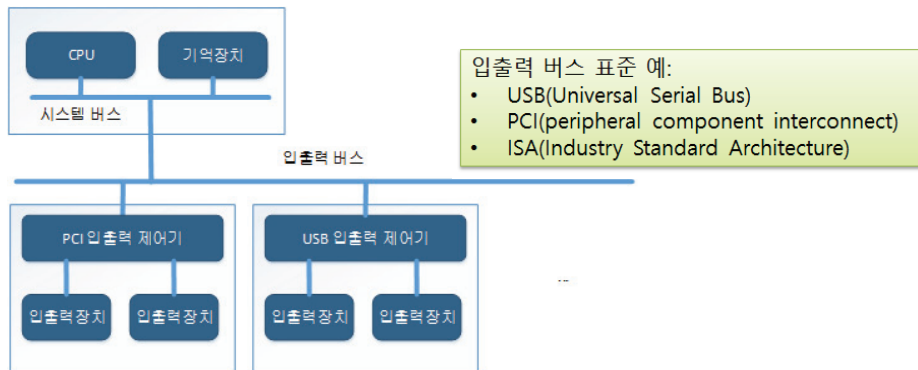


### ■ 입출력 버스의 문제점

중앙처리장치, 기억장치, 입출력장치가 동일한 입출력버스를 사용함으로써 빠르게 동작하는 중앙처리장치가 가장 느린 입출력 장치 때문에 제 성능을 낼 수 없는 문제가 발생  
CPU가 빠른 속도로 진화하면서 더욱 문제가 불거짐.

### ■ 개선된 입출력 버스

CPU와 기억장치는 시스템 버스(System Bus)로 묶고, 그 다음 다양한 입출력장치들은 입출력 버스로 묶어 CPU의 입출력 모듈에 연결



### ■ 운영체제(Operating System)

애플리케이션이 다양한 하드웨어(CPU, 기억장치, 입출력장치 등등) 위에서 동작할 수 있도록 하는 시스템 소프트웨어

운영체제는 애플리케이션에게 API(Application Programming Interface)를 제공하여 운영체제가 제어하고 있는 하드웨어를 이용할 수 있게 함.



### ■ 소프트웨어는 무엇으로 만드는가? : 프로그래밍 언어(Programming Language)

프로그램을 작성하기 위해 만들어진 인공 언어 체계

## ■ 컴파일 방식과 인터프리트 방식 프로그래밍 언어

	언어	실행 속도	이식성
컴 파 일	C, C++, 파스칼, 에이다	CPU가 실행할 수 있는 기계 코드로 컴파일되므로 실행 속도 빠름.	원래 애플리케이션이 개발된 CPU/운영체제와 다른 CPU/운영체제로 옮기는 경우, 대체로 코드를 그 대로 사용할 수 없음. 이식성 낮음. ※C#, 자바 등 가상머신 기반의 언어는 이식성 높음
인 터 프 리 트	베이직, 파이썬, 루비, PHP, 펄	애플리케이션을 실행할 때 마다 인터프리터가 소스 코 드를 기계 코드로 번역하는 절차를 거치므로 실행속도 느림.	인터프리터만 대상 CPU/운영체제를 지원한다면 애플 리케이션 코드는 변경할 필요 없이 어떤 환경에서나 실행 가능. 이식성 높음.

### TIP 용어 알고 가기

- 코딩 : 컴퓨터가 일하는 절차를 적는 행위
- 코드 : 코딩으로 완성된 문장, 컴퓨터 프로그래밍
- 초기의 코딩은 종이에 구멍을 뚫는 천공 카드 사용
- 포트란 : 높은 정밀도 필요한 계산에 쓰임, 과학기술언어
- SQL : 데이터베이스에서 무언가를 하고 싶을 때 사용
- 인터프리터 : 한 줄씩 소스 코드를 해석하여 실행해 결과를 바로 확인 하는 언어.
- 컴파일러 : 소스코드를 번역하여 목적프로그램으로 저장시켜 결과를 나중에 확인하는 언어

## 1 파이썬(Python)의 개요

- 1990년 암스테르담의 귀도반로성이 개발한 인터프리터 언어
- BBC의 코미디 프로그램인 ‘몬티 파이튼의 비행 서커스’에서 이름을 따옴
- 파이썬은 우리나라에서는 아직 대중적으로 사용되지 않음
- 외국에서는 교육 목적뿐 아니라 실무에서도 많이 사용 됨.

### ■ 파이썬으로 개발된 예

- NASA나 유튜브, 구글 앱
- 구글에서 만들어진 소프트웨어의 50% 이상이 파이썬으로 만들어졌다.
- 파일 동기화 서비스인 드롭박스(Dropbox)
- 쉽고 빠르게 웹 개발을 할 수 있도록 도와주는 프레임워크인 장고(Django)

### ■ 파이썬의 특징

- 프로그램을 모르더라도 직관적으로 무엇을 뜻하는지 알 수 있는 언어이다.
- 문법이 쉬워 빠르게 배울 수 있다.
- 무료이지만 강력하다.



- 시스템프로그래밍, 하드웨어 제어와 같은 매우 복잡한 프로그램은 불가능.  
→ 다른 언어로 만든 프로그램을 파이썬 프로그램에 포함 시킬 수 있다.
- 파이썬 프로그램은 공동 작업과 유지 보수가 매우 쉽고 편하다.
- 다른 언어로 작성된 많은 프로그램과 모듈들이 파이썬으로 재구성 됨.
- 국내에서도 그 가치를 인정받아 사용자층이 더욱 넓어지고 있다.
- 파이썬을 이용해 프로그램을 개발하는 기업체들 또한 늘어 가고 있다.

#### ■ 파이썬으로 할 수 있는 일

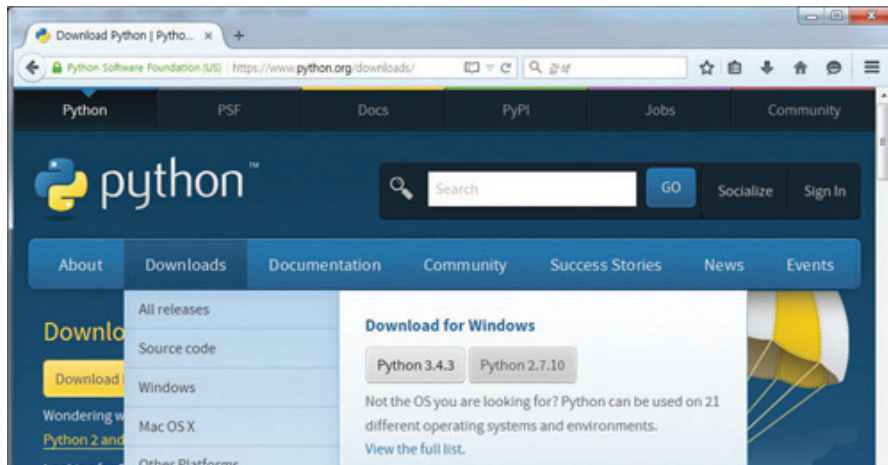
- (1) 시스템 유틸리티 제작 : 컴퓨터 이용에 도움이 되는 여러 소프트웨어를 제작
- (2) GUI 프로그래밍
  - 윈도우 창처럼 화면을 보며 마우스나 키보드로 조작할 수 있는 프로그램을 만드는 것
  - Tkinter(티케이인터) 모듈을 이용해 GUI 프로그램을 만든다.
  - 그 외 파이썬에는 wxPython, PyQt, PyGTK 등과 같이 Tkinter보다 빠른 속도와 보기 좋은 인터페이스도 있다.
- (3) C/C++와의 결합
- (4) 웹 프로그래밍 : 게시판이나 방명록과 같은 웹 프로그램을 만든다.
- (5) 수치 연산 프로그래밍 : Numeric Python이라는 수치 연산 모듈이 제공
- (6) 데이터베이스 프로그래밍 : 사이베이스(Sybase), 인포믹스(Infomix), 오라클(Oracle), 마이에스큐엘(MySQL), 포스트그레스큐엘(PostgreSQL) 등의 데이터베이스에 접근할 수 있게 해주는 도구들을 제공
- (7) 피클(pickle) 모듈: 피클은 파이썬에서 사용되는 자료들을 변형없이 그대로 파일에 저장하고 불러오는 일들을 한다.
- (8) 데이터 분석, 사물 인터넷
  - 판다스(Pandas)라는 모듈을 이용
  - 사물 인터넷 분야에서 라즈베리파이를 이용하면 홈시어터나 아주 작은 게임기 등 여러 가지 재미있는 것들을 만들 수 있는데 파이썬은 이 라즈베리파이를 제어하는 도구로 사용된다. (라즈베리파이에 연결된 모터를 작동시키거나 램프에 불이 들어오게 하는 일들을 파이썬으로 할 수 있다.)

#### ■ 파이썬으로 할 수 없는 일

- (1) 시스템과 밀접한 프로그래밍 영역 : 윈도우, 리눅스와 같은 운영체제, 엄청난 횟수의 반복과 연산을 필요로 하는 프로그램, 데이터 압축 알고리즘 개발 프로그램 등을 만드는 것은 어렵다.
- (2) 시모바일 프로그래밍 : 안드로이드 앱(App)을 개발하는 것은 불가능하다. 안드로이드에서 파이썬으로 만든 프로그램들이 실행되도록 지원은 함.

## 2 파이썬(Python) 설치하기(윈도우)

1. <http://www.python.org> 홈페이지에 접속
2. Downloads 메뉴 클릭 - 윈도우용 Python 설치 파일 내려 받기(3.x)

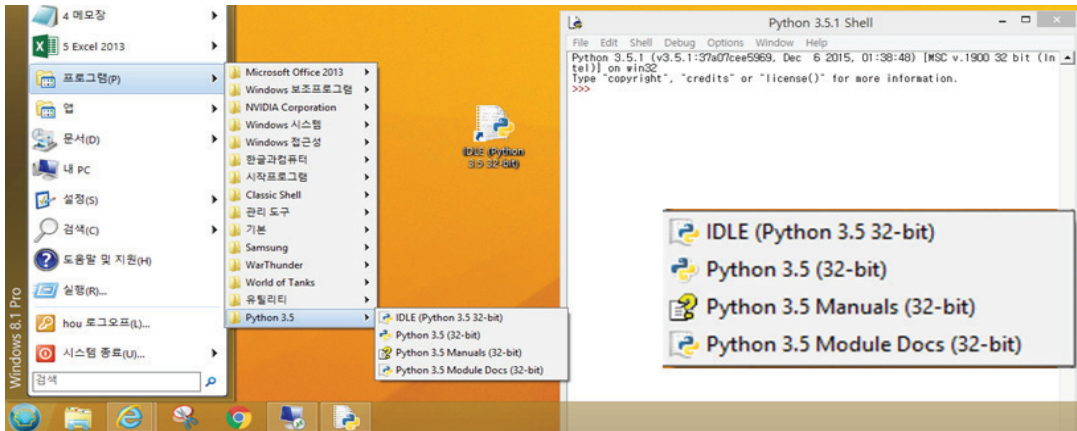


3. 내려 받은 실행파일 더블 클릭해 설치
4. 파이썬 사용자 선택은 '모든 사용자'로 설정



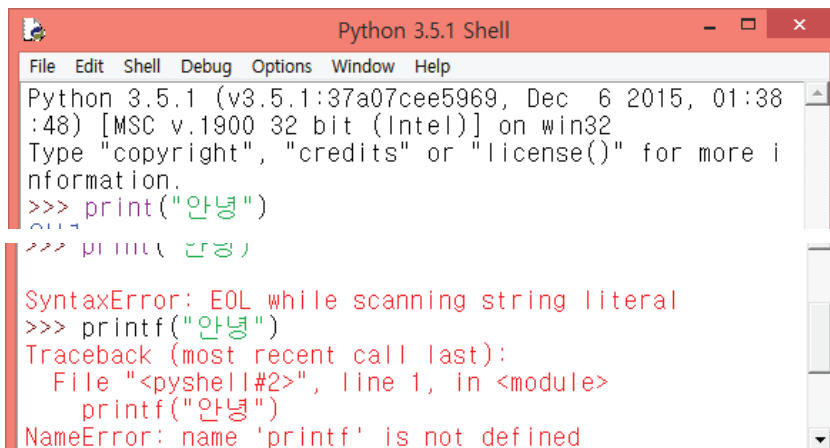
※ "Add Python 3.5 to PATH" 옵션을 누락할 경우 오류가 발생할 수 있다.  
만약 python이 설치되는 경로와 PATH에 대한 사전지식이 있는 사용자라면 이 옵션을 생략해도 됨

## 5. 직접 실행과 단축 아이콘을 이용한 실행



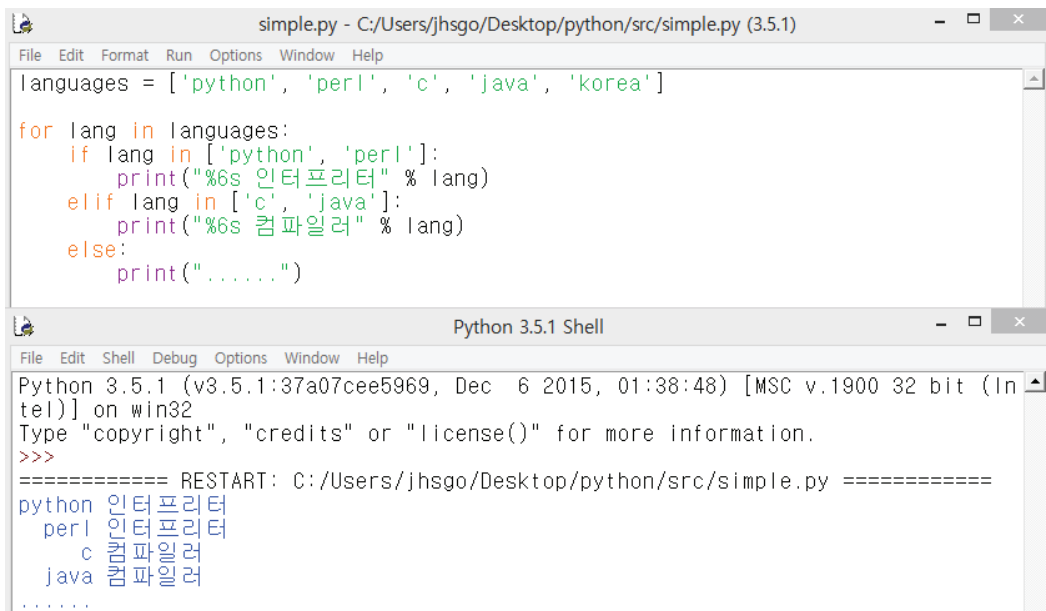
### ■ IDLE

- 파이썬에서 기본적으로 사용하는 텍스트 편집기
- 코드를 제대로 작성했는지 확인해주는 기능도 탑재
- IDLE 자체도 파이썬으로 제작되어 있음
- 다른 컴퓨터 환경(애플, 윈도우, 라즈베리파이)에서도 같은 인터페이스로 작업 가능
- IDLE 실행 후 코드를 프롬프트에 입력하고 엔터를 치면 결과 확인



- IDLE를 이용하여 새 파일에 소스 작성 후 Run 메뉴를 이용하여 결과 확인

```
1. languages = ['python', 'perl', 'c', 'java', 'korea']
2.
3. for lang in languages:
4.     if lang in ['python', 'perl']:
5.         print("%6s 인터프리터" % lang)
6.     elif lang in ['c', 'java']:
7.         print("%6s 컴파일러" % lang)
8.     else:
9.         print(" ")
```



```

simple.py - C:/Users/jhsgo/Desktop/python/src/simple.py (3.5.1)
File Edit Format Run Options Window Help
languages = ['python', 'perl', 'c', 'java', 'korea']

for lang in languages:
    if lang in ['python', 'perl']:
        print("%6s 인터프리터" % lang)
    elif lang in ['c', 'java']:
        print("%6s 컴파일러" % lang)
    else:
        print(".....")

Python 3.5.1 Shell
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/jhsgo/Desktop/python/src/simple.py =====
python 인터프리터
perl 인터프리터
c 컴파일러
java 컴파일러
.....
  
```

### 3 기본 자료형(문자, 숫자)

- 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것
- 프로그램의 기본이자 핵심 단위
- 자료형을 충분히 이해하지 않고 프로그래밍을 시작하려는 것은 기초 공사가 마무리되지 않은 상태의 빌딩으로 부실공사와 같다.
  1. 숫자형
  2. 문자열 자료형
  3. 리스트 자료형
  4. 튜플 자료형
  5. 딕셔너리 자료형
  6. 집합 자료형
  7. 자료형의 참과 거짓
  8. 자료형의 값을 저장하는 공간, 변수

#### ■ 숫자형

항목	예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
복소수	1 + 2j, -3j
8진수	0o34, 0o25
16진수	0x2A, 0xFF

(1) 정수형 : 양의 정수와 음의 정수, 숫자 0을 변수 a에 대입

```
>>> a = 123
>>> a = -123
>>> a = 0
```

(2) 실수형 : 소수점이 포함된 숫자

```
>>> a = 1.234
>>> a = -1.234
>>> a = 1.24E10
>>> a = 1.24e-10
```

(3) 8진수와 16진수

8진수(Octal)를 만들기 위해서는

숫자가 0o 또는 00(숫자 0 + 소문자 o 또는 대문자 O)로 시작하면 된다.

```
>>> a = 0o177
```

16진수(Hexadecimal)를 만들기 위해서는 0x로 시작하면 된다.

```
>>> a = 0xABC
```

(4) 복소수 : i 대신 j를 사용한다. 대소문자 구별 안 함.

```
>>> a = 1+2j
>>> b = 3-4j
>>> a = 1+2j
>>> a.real          ➡ 복소수.real은 복소수의 실수 부분을 리턴
>>> a.imag          ➡ 복소수.imag는 복소수의 허수 부분을 리턴
>>> a.conjugate()   ➡ 복소수.conjugate는 복소수의 켤레복소수를 리턴
>>> abs(a)          ➡ abs(복소수)는 복소수의 절댓값을 리턴
```

■ 사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b
>>> a * b
>>> a / b
>>> a ** b          ➡ a의 b제곱 값을 리턴
>>> 7 % 3           ➡ 나눗셈 후 나머지를 반환하는 % 연산자
>>> 7 // 4          ➡ 나눗셈 후 소수점 아랫자리를 버리는 // 연산자
```

연산자	의미	예제	결과값
*	곱하기	2*3	6
/	나누기(일반)	20/8	2.5
//	나누기(정수값만)	20//8	2
%	나머지	20%8	4
+	더하기	2+3	5
-	빼기	7-3	4

■ 문자형

문자열(String)이란 문자, 단어 등으로 구성된 문자들의 집합을 의미

■ 문자열 만드는 방법

- (1) 큰따옴표로 양쪽 둘러싸기  
"Hello World"
- (2) 작은따옴표로 양쪽 둘러싸기  
'Hello World'
- (3) 큰따옴표 3개를 연속으로 써서 양쪽 둘러싸기  
"""Hello World"""
- (4) 작은따옴표 3개를 연속으로 써서 양쪽 둘러싸기  
'''Hello Wor ld'''

## ■ 작은따옴표, 큰따옴표 사용하는 방법

(1) 문자열에 작은따옴표 (') 포함시키기

```
>>> na = "I' m a girl"
```

```
>>> na = 'I' m a girl' ➡ 구문 오류(SyntaxError)가 발생
```

(2) 문자열에 큰따옴표 (") 포함시키기

```
>>> say = '"파이썬" 쉽다. '
```

(3) W(백슬래시)를 이용해서 작은따옴표와 큰따옴표를 문자열에 포함시키기

```
>>> na = 'I\W'am a girl'
```

```
>>> say = "W" 파이썬 W" 쉽다."
```

## ■ 변수에 여러 줄인 문자열을 입력시키는 방법

(1) 줄을 바꾸기 위한 이스케이프 코드 \n 삽입하기

```
>>> multiline = "1줄 \n 2줄"
```

(2) 연속된 작은따옴표 3개('') 또는 큰따옴표 3개(""") 이용

```
>>> multiline='''
```

```
1줄
```

```
2줄
```

```
'''
```

## TIP 이스케이프 코드

- 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 “문자 조합”
- 주로 출력물을 보기 좋게 정렬하는 용도로 이용

코드	설명	코드	설명
\n	개행 (줄바꿈)	\r	캐리지 리턴
\t	수평 탭	\f	폼 피드
\\	문자 "\"	\a	벨 소리
\'	단일 인용부호(')	\b	백 스페이스
\"	이중 인용부호( )	\000	널문자

## ■ 문자열 연산하기

1) 문자열 더해서 연결하기(Concatenation)

```
>>> head = "파이썬 "
```

```
>>> tail = "안녕"
```

```
>>> head + tail
```

2) 문자열 곱하기

```
>>> a = "파이썬"
```

```
>>> a * 2
```

➡ a를 두 번 반복하라는 뜻

```
1. print("=" * 50)
2. print("파이썬 프로그램 1")
3. print("=" * 50)
```

## ■ 문자열 인덱싱과 슬라이싱

```
>>> charater = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
>>> charater[3]           ➡ 첫글자부터 0으로 인덱싱 되어 D가 출력
>>> charater[-1]          ➡ 뒷글자부터 세어서 인덱싱 되어 Z가 출력
>>> charater[0] + charater[1] + charater[2] + charater[3]
>>> charater[0:4]         ➡ [시작 번호:끝 번호]를 지정하면
                                끝 번호에 해당하는 것은 포함되지 않는다.

>>> charater[5:]
>>> charater[:17]
>>> charater[:]
>>> charater[19:-7]
```

```
1. a = "20160217Rainy"
2. date = a[:8]
3. year = a[:4]
4. day = a[4:8]
5. weather = a[8:]
```

## ■ 문자값을 변경하는 방법

```
>>> a = "Python"
>>> a[1] = 'y'
```

## ■ 문자열 포매팅

코드	설명	코드	설명
%s	문자열 (String)	%o	8진수
%c	문자 1개(character)	%x	16진수
%d	정수 (Integer)	%%	Literal % (문자 % 자체)
%f	부동소수 (floating-point)		

### (1) 숫자 바로 대입

```
>>> "I eat %d apples." % 3
```

### (2) 문자열 바로 대입

```
>>> "I eat %s apples." % "five"
```

### (3) 숫자 값을 나타내는 변수로 대입

```
>>> number = 3
>>> "I eat %d apples." % number
```

### (4) 2개 이상의 값 넣기

```
>>> number = 10           >>> day = "three"
>>> "I ate %d apples. so I was sick for %s days." % (number, day)
>>> "Error is %d%." % 100           ➡ 오류 발생
>>> "Error is %d%." % 100
```



## ■ 포맷 코드와 숫자 함께 사용하기

### (1) 정렬과 공백

```
>>> "%10s" % "korea"    ➡ 전체 길이가 10개인 문자열 공간에서
                        글자를 오른쪽으로 정렬하고 그 앞의 나머지는 공백으로 남김.
>>> "%-10s one." % 'korea' ➡ 왼쪽 정렬하고 나머지는 공백으로 채움
```

### (2) 소수점 표현하기

```
>>> "%0.4f" % 3.142134234 ➡ 소수점 네 번째 자리까지만 표시
>>> "%10.4f" % 3.42134234 ➡ 전체 길이 10개인 문자열 공간에
                        소수점 네 번째 자리까지만 표시하고, 오른쪽으로 정렬
```

## ■ 문자열 관련 함수들

### (1) count : 문자 개수 세기

```
>>> a = "korea"          >>> a.count('k')    ➡ 문자열 중 문자 k의 개수를 반환
```

### (2) find : 위치 알려주기

```
>>> a = "Python is Good"
>>> a.find('i')           ➡ 문자열 중 문자 i가 처음으로 나온 위치를 반환
➡ 만약 찾는 문자나 문자열이 존재하지 않는다면 -1을 반환
➡ 파이썬은 숫자를 0부터 센다.
```

### (3) index : 위치 알려주기

```
>>> a = "I love You"
>>> a.index('v')          ➡ 문자열 중 문자 v가 맨 처음으로 나온 위치를 반환
➡ 만약 찾는 문자나 문자열이 존재하지 않는다면 오류를 발생
```

### (4) join : 문자열 삽입

```
>>> a = ","
>>> a.join('abcd')        ➡ abcd의 각각의 문자 사이에 변수 a의 값인 ','를 삽입한다.
```

### (5) upper : 소문자를 대문자로 바꾸기

```
>>> a = "korea"          >>> a.upper()      ➡ 소문자를 대문자로 바꾸어 준다
```

### (6) lower : 대문자를 소문자로 바꾸기

```
>>> a.lower()             ➡ 대문자를 소문자로 바꾸어 준다.
```

### (7) lstrip : 왼쪽 공백 지우기

```
>>> a.lstrip()            ➡ 가장 왼쪽에 있는 한 칸 이상의 연속된 공백들을 모두 지움
```

### (8)rstrip : 오른쪽 공백 지우기

```
>>> a.rstrip()            ➡ 가장 오른쪽의 한 칸 이상의 연속된 공백들을 모두 지움
```

### (9) strip : 양쪽 공백 지우기

```
>>> a.strip()             ➡ 양쪽에 있는 한 칸 이상의 연속된 공백들을 모두 지운다.
```

### (10) replace : 문자열 바꾸기

```
>>> a = "I love You"
>>> a.replace("Love", "Hate") ➡ replace(바뀌게 될 문자열, 바꿀 문자열)처럼 사용
```

### (11) split : 문자열 나누기

```
>>> a = "I love You"      >>> a.split()    ➡ 공백을 기준으로 문자열을 나눔
>>> a = "a:b:c:d"         >>> a.split(':') ➡ 괄호 안의 값으로 문자열을 나눔
```

## ■ 고급 문자열 포매팅

### (1) 숫자 바로 대입하기

>>> "I eat {0} apples".format(3)      ➡ {0} 부분이 숫자 3으로 바뀌었다.

(2) 문자열 바로 대입하기

>>> "I eat {0} apples".format("one")      ➡ {0} 항목이 one문자열로 바뀜

(3) 숫자 값을 가진 변수로 대입하기

```
>>> number = 3
```

>>> "I eat {0} apples".format(number)    ➔ {0} 항목이 number 변수의 값으로

(4) 2개 이상의 값 넣기

```
>>> number = 10
```

```
>>> day = "three"
```

```
>>> "I ate {0} apples. so I was sick for {1} days.".format(number, day)
```

➔ 2개 이상의 값을 넣을 경우 문자열의 {0}, {1}과 같은 인덱스 항목들이 format 함수의 입력값들로 순서에 맞게 바뀐다.

(5) 이름으로 넣기

```
>>> "I ate {number} apples. so I was sick for {day} days."W
      .format(number=10, day=3)
```

➔ {0}, {1} 대신 {name} 형태를 이용

(6) 인덱스와 이름을 혼용해서 넣기

```
>>> "I ate {0} apples. so I was sick for {day} days.".format(10, day=3)
```

➡ 인덱스 항목과 name=value 형태를 혼용하는 것도 가능

(7) 왼쪽 정렬

>>> "{0:<10}".format("hi")      ➡ 10자리수로 왼쪽으로 정렬

(8) 오른쪽 정렬

>>> "{0:>10}".format("hi")      ➡ 10자리수로 오른쪽으로 정렬

(9) 가운데 정렬

```
>>> "{0:^10}".format("hi")      ➡ 가운데 정렬
```

(10) 공백 채우기

➡ 채워 넣을 문자 값은 정렬 문자인 <, >, ^ 바로 앞에 넣어야 한다.

>>> "{0:=^10}".format("hi")      ➔ 가운데 정렬, 빈 공간을 =문자로 채움

>>> "{0: !<10}".format("hi")      ➡ 왼쪽 정렬, 빈 공간을 !문자로 채움

### (11) 소수점 표현하기

```
>>> y = 3.42134234
```

>>> "{0:0.4f}".format(y)      ➡ 소수점을 4자리까지만 표현

>>> "{0:10.4f}".format(y)      ➡ 10 자릿수로 맞춰서 소수점 4자리만 표현

(12) { 또는 } 문자 표현하기

```
>>> "{{ and }}" .format()
```

➔ { }와 같은 중괄호(brace) 문자를 포매팅 문자가 아닌

문자 그대로 사용하고 싶은 경우에는 {{와 }}처럼 2개를 연속해서 사용

## 4 리스트 자료형

### ■ 리스트 만드는 방법

```
>>> odd = [1, 3, 5, 7, 9]
➔ 대괄호([ ])로 감싸 주고 각 요소값들은 쉼표(,)로 구분
➔ 리스트명 = [요소1, 요소2, 요소3, ... ]

>>> a = [ ] ➔ 비어 있는 리스트는 a = list()로 생성
>>> b = ['python', 'is', 'very', 'good']
>>> c = [1, 2, 'python', 'is']
>>> d = [1, 2, ['python', 'is']]
```

### ■ 리스트 인덱싱

```
>>> a = [1, 2, 3]
>>> a[0] ➔ a[0]은 리스트 a의 첫 번째 요소값
>>> a[0] + a[2] ➔ 첫번째 요소인 a[0]과 세번째 요소인 a[2]의 값을 더한 것
>>> a[-1] ➔ a[-1]은 a의 마지막 요소값을 말한다.

>>> a = [1, 2, 3, ['a', 'b', 'c']]
>>> a[0]
>>> a[-1] ➔ a[-1]은 마지막 요소값인 ['a', 'b', 'c']를 나타냄.
>>> a[3]
>>> a[-1][0] ➔ ['a', 'b', 'c'] 안의 첫 번째 요소를 나타냄.
>>> a[-1][1]
>>> a[-1][2]
```

### ■ 삼중 리스트에서 인덱싱하기

```
>>> a = [1, 2, ['a', 'b', ['python', 'is']]]
>>> a[2][2][0]
```

### ■ 리스트의 슬라이싱

```
>>> a = [1, 2, 3, 4, 5]
>>> a[0:2]
>>> a = "12345"
>>> a[0:2]
>>> a = [1, 2, 3, 4, 5]
>>> b = a[:2] ➔ b는 a의 처음 요소부터 두 번째 요소인 a[2]까지 나타냄
➔ a[2] 값인 3은 포함되지 않는다.
>>> c = a[2:] ➔ c라는 변수는 리스트 a의 세 번째 요소부터 끝까지 나타냄
```

## ■ 중첩된 리스트에서 슬라이싱하기

```
>>> a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]
>>> a[2:5]                ➡ [3, ['a', 'b', 'c'], 4]
>>> a[3][:2]              ➡ ['a', 'b']
```

## ■ 리스트 연산자

### (1) 리스트 더하기(+)

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b                ➡ 리스트 사이에서 + 기호는 2개의 리스트를 합치는 기능
```

### (2) 리스트 반복하기(\*)

```
>>> a = [1, 2, 3]
>>> a * 3                ➡ [1, 2, 3] 리스트가 세 번 반복되어 새 리스트를 생성
```

## ■ 리스트 연산 오류

```
>>>a = [1, 2, 3]
>>>a[2] + "hi"           ➡ 숫자 + 문자열 : 형오류 발생
>>>str(a[2]) + "hi"      ➡ 숫자를 문자로 변환하여 문자+문자로 처리
```

## ■ 리스트의 수정, 변경과 삭제

### (1) 리스트에서 하나의 값 수정하기

```
>>> a = [1, 2, 3]
>>> a[2] = 4             ➡ a[2]의 요소값 3이 4로 바뀜
```

### (2) 리스트에서 연속된 범위의 값 수정하기

```
>>> a[1:2]
>>> a[1:2] = ['a', 'b', 'c']
➡ a[1:2]는 a[1]부터 a[2]까지이며 a[2]는 포함하지 않으므로 a = [1, 2, 4]에서
a[1]의 값인 2만을 의미한다. 즉, a[1:2]를 ['a', 'b', 'c']로 바꾸었으므로
a 리스트에서 2라는 값 대신에 ['a', 'b', 'c']라는 값이 대입되었다.
```

```
>>> a[1] = ['a', 'b', 'c']
>>> a                ➡ [1, ['a', 'b', 'c'], 4]
```

### (3) [ ] 사용해 리스트 요소 삭제하기

```
>>> a[1:3] = [ ]
```

### (4) del 함수 사용해 리스트 요소 삭제하기

```
>>> del a[1]           ➡ del a[x]는 x번째 요소값을 삭제
```

## ■ 리스트 관련 함수들

### (1) 리스트에 요소 추가(append)

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a.append([5,6])
```

### (2) 리스트 정렬(sort)

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a = ['a', 'c', 'b']
>>> a.sort()
```

### (3) 리스트 뒤집기(reverse)

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
```

### (4) 데이터의 위치값 반환(index)

```
>>> a = [1,2,3]
>>> a.index(3)
>>> a.index(1)
>>> a.index(0) ➔ 0값은 a 리스트에 존재하지 않아서 값 오류(ValueError) 발생
Traceback (innermost last):
  File "", line 1, in ?
    a.index(0)
ValueError: list.index(x): x not in list
```

### (5) 리스트에 요소 삽입(insert)

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4) ➔ 0번째 요소(a[0]) 위치에 4를 삽입
>>> a.insert(3, 5)
```

### (6) 리스트 요소 제거(remove)

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3) ➔ 3이 2개 있을 경우 첫 번째 3만 제거
```

### (7) 리스트 요소 끄집어내기(pop)

```
>>> a = [1,2,3]
>>> a.pop() ➔ 마지막 요소 추출
>>> a.pop(1) ➔ 1번째 요소 추출
```

### (8) 리스트에 포함된 요소 x의 개수 세기(count)

➔ count(x)는 리스트 내에 x가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수

```
>>> a = [1,2,3,1]
>>> a.count(1) ➔ 1이라는 값이 리스트 a에 2개 들어 있으므로 2를 돌려준다.
```

### (9) 리스트 확장(extend)

a 리스트에 x 리스트를 더하게 된다.

```
>>> a = [1,2,3] >>> a.extend([4,5])
>>> b = [6, 7] >>> a.extend(b) ➔ a += [4,5]와 동일
```

## 5 튜플 자료형

### ■ 튜플이란?

- 리스트와 거의 비슷
- 리스트는 [과 ]으로 둘러싸지만 튜플은 (과 )으로 둘러싼다.
- 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.
  - ➔ 튜플의 항목값은 변화가 불가능
  - ➔ 튜플의 요소값은 한번 정하면 지우거나 변경할 수 없다

### ■ 튜플 만드는 방법

```
>>> t1 = ()
>>> t2 = (1,) ➔ 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙임
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3 ➔ 괄호()를 생략해도 무방
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

### ■ 인덱싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0]
>>> t1[3] ➔ 리스트와 마찬가지로 t1[0], t1[3]처럼 인덱싱이 가능
```

### ■ 슬라이싱하기

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[1:] ➔ t1[1]부터 튜플의 마지막 요소까지 슬라이싱 가능
```

### ■ 튜플 더하기

```
>>> t2 = (3, 4)
>>> t1 + t2 ➔ 튜플을 더하는 방법
```

### ■ 튜플 곱하기

```
>>> t2 * 3 ➔ (3, 4, 3, 4, 3, 4) 반복하는 방법
```

#### ※ 튜플 요소값 삭제 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> del t1[0]
```

Traceback (innermost last):

File "", line 1, in ?del t1[0]

TypeError: object doesn't support item deletion

#### ※ 튜플 요소값 변경 시 오류

```
>>> t1 = (1, 2, 'a', 'b')
>>> t1[0] = 'c'
```

Traceback (innermost last):

File "", line 1, in ?t1[0] = 'c'

TypeError: object doesn't support item assignment

## 6 딕셔너리 자료형

### ■ 딕셔너리 자료형이란?

- 이름 = "홍길동", 생일 = "1월 1일" 등으로 구분하는 대응 관계를 나타낼 수 있는 자료형
- 다른 언어 : 연관 배열(Associative array) 또는 해시(Hash)라고 함.
- 딕셔너리는 리스트나 튜플처럼 순차적으로 해당 요소값을 구하지 않음  
     ➔ Key를 통해 Value를 얻는다.
- {"김연아":"피겨스케이팅", "류현진":"야구", "박지성":"축구", "귀도":"파이썬"}

### ■ 딕셔너리 만드는 방법

- {Key1:Value1, Key2:Value2, Key3:Value3      }
- Key와 Value의 쌍 여러 개가 {과 }로 둘러싸여 있다.
- 각각의 요소는 Key : Value 형태로 이루어져 있고 쉼표(,) 로 구분되어 있다.
- Key에는 변하지 않는 값을 사용
- Value에는 변하는 값과 변하지 않는 값 모두 사용
- 딕셔너리 예  
     >>> dic = {'name':'HONG', 'phone':'1234', 'birth': '0314'}  
     >>> a = {1: 'hi'}           ➔ Key로 정수값 1, Value로 'hi'라는 문자열을 사용  
     >>> a = { 'a': [1,2,3]} ➔ Value에 리스트도 넣을 수 있음

### ■ 딕셔너리 쌍 추가, 삭제하기

#### (1) 딕셔너리 쌍 추가하기

```
>>> a = {1: 'a'}
>>> a[2] = 'b'                   ➔ {2: 'b', 1: 'a'}
➔ {1: 'a'}라는 딕셔너리에 a[2] = 'b'와 같이 입력하면
딕셔너리 a에 Key와 Value가 각각 2와'b'인 2 : 'b'라는 딕셔너리 쌍이 추가
>>> a['name'] = 'pey'           ➔ {'name':'pey', 2: 'b', 1: 'a'}
>>> a[3] = [1,2,3]              ➔ {'name': 'pey', 3: [1, 2, 3], 2: 'b', 1: 'a'}
```

#### (2) 딕셔너리 요소 삭제하기

```
>>> del a[1]                   ➔ {'name': 'pey', 3: [1, 2, 3], 2: 'b'}
➔ del a[key]처럼 입력하면 지정한 key에 해당하는 {key : value} 쌍이 삭제
```

### ■ 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'name':'HONG', 'phone':'1234', 'birth': '0314'}
>>> grade['name']           ➔ "딕셔너리 변수[Key]"를 사용
>>> a = {1:'a', 2:'b'}
>>> a[1]
>>> a = {'a':1, 'b':2}
>>> a['a']
```

## ■ 딕셔너리 만들 때 주의할 사항

- Key는 고유 값 : 중복되는 Key 값을 설정해 놓으면 하나를 제외한 나머지 것들이 모두 무시
  - ➔ 중복되는 Key를 사용하지 말라는 것

```
>>> a = {1:'a', 1:'b'}
```

```
>>> a
```
- Key에 리스트는 쓸 수 없다는 것
  - ➔ 튜플은 Key로 쓸 수 있다.
  - ➔ 리스트를 Key로 설정하면 형 오류(TypeError)가 발생
 

```
>>> a = {[1,2] : 'today'}
```

```
Traceback (most recent call last):
```

```
File "", line 1, in ?
```

```
TypeError: unhashable type
```

## ■ 딕셔너리 관련 함수들

### (1) Key 리스트 만들기(keys)

```
>>> a = {'name':'HONG', 'phone':'1234', 'birth': '0314'}
```

```
>>> a.keys() ➔ dict_keys(['name', 'phone', 'birth'])
```

➔ a.keys()는 딕셔너리 a의 Key만을 모아서 dict\_keys라는 객체를 리턴

※ 리턴값으로 리스트가 필요한 경우에는 "list(a.keys())"를 사용  
dict\_keys, dict\_values, dict\_items 등은 리스트로 변환하지 않더라도  
기본적인 반복성(iterate) 구문(예: for문)들을 실행할 수 있다.

### ※ dict\_keys 객체

- 리스트를 사용하는 것과 차이가 없음
- 리스트 고유의 함수인 append, insert, pop, remove, sort등의 사용할 수는 없다.

```
>>> for k in a.keys():
```

```
    print(k)
```

- dict\_keys 객체를 리스트로 변환

```
>>> list(a.keys())
```

### (2) Value 리스트 만들기(values)

```
>>> a.values() ➔ dict_values(['HONG', '1234', '0314'])
```

### (3) Key, Value 쌍 얻기(items)

```
>>> a.items() ➔dict_items([('name','HONG'),('phone','1234'),('birth','0314')])
```

➔ items 함수는 key와 value의 쌍을 튜플로 묶은 값을 dict\_items 객체로 돌려준다.

### (4) Key: Value 쌍 모두 지우기(clear)

```
>>> a.clear()
```

- ➔ clear() 함수는 딕셔너리 안의 모든 요소를 삭제한다.

빈 리스트를 [ ], 빈 튜플을 ()로 표현하는 것과 마찬가지로 빈 딕셔너리도 {}로 표현



### (5) Key로 Value얻기(get)

```
>>> a = {'name': 'HONG', 'phone': '1234', 'birth': '0314'}
```

```
>>> a.get('name')
```

```
>>> a.get('phone')
```

➡ get(x) 함수는 x라는 key에 대응되는 value를 돌려준다.

```
>>> a.get('nokey')
```

```
>>> a['nokey']
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

KeyError: 'nokey'

➡ a['nokey']처럼 존재하지 않는 키(nokey)로 값을 가져오려고 할 경우  
a['nokey']는 Key 오류를 발생  
a.get('nokey')는 None을 리턴

```
>>> a.get('No', 'yes')
```

➡ key 값이 없을 경우 디폴트 값을 대신 가져오게 함

➡ get(x, '디폴트 값')을 사용

### (6) 해당 Key가 딕셔너리 안에 있는지 조사하기(in)

```
>>> a = {'name': 'HONG', 'phone': '1234', 'birth': '0314'}
```

```
>>> 'name' in a ➡ True
```

```
>>> 'email' in a ➡ False
```

➡ 'name'이라는 문자열은 a 딕셔너리의 key 중 하나이므로 참(True)을 리턴

➡ 'email'은 a 딕셔너리 안에 존재하지 않는 key이므로 거짓(False)을 리턴

## 7 집합 자료형

### ■ 집합 자료형 만드는 방법

- 집합에 관련된 것들을 쉽게 처리하기 위해 만들어진 자료형

- 집합 자료형은 set 키워드를 이용해 만듦

```
>>> s1 = set([1,2,3])
```

```
>>> s2 = set("Hello") ➡ set()의 괄호 안에 리스트 또는 문자열을 입력 가능
```

```
>>> s2 ➡ {'e', 'l', 'o', 'H'}
```

➡ 중복을 허용하지 않는다.

➡ 순서가 없다(Unordered).

※ 리스트, 튜플 : 순서가 있음 ➡ 인덱싱을 통해 자료형의 값을 얻을 수 있음

set 자료형 : 순서가 없음 ➡ 인덱싱으로 값을 얻을 수 없음 (딕셔너리와 비슷)

## ■ set 자료형에 저장된 값을 인덱싱으로 접근하는 방법

- 리스트나 튜플로 변환한 후 사용

```
>>> s1 = set([1,2,3])
>>> l1 = list(s1)
>>> l1[0]
>>> t1 = tuple(s1)
>>> t1
>>> t1[0]
```

➡ 중복을 허용하지 않는 set의 특징은

자료형의 중복을 제거하기 위한 필터 역할로 종종 사용되기도 함

## ■ 교집합, 합집합, 차집합 구하기

```
>>> s1 = set([1, 2, 3, 4, 5, 6])
>>> s2 = set([4, 5, 6, 7, 8, 9])
```

### (1) 교집합

```
>>> s1 & s2           ➡ "&" 기호를 이용하면 교집합을 간단히 구함
>>> s1.intersection(s2) ➡ intersection 함수를 사용해도 동일한 결과를 리턴
>>> s2.intersection(s1) ➡ 같은 결과 도출
```

### (2) 합집합

```
>>> s1 | s2           ➡ "|" 기호를 이용한 방법
>>> s1.union(s2)       ➡ union 함수를 이용
```

### (3) 차집합

```
>>> s1 - s2           ➡ -(빼기) 기호를 이용한 방법
>>> s1.difference(s2) ➡ difference 함수를 이용

>>> s2 - s1
>>> s2.difference(s1)
```

## ■ 집합 자료형 관련 함수들

### (1) 값 1개 추가하기(add)

```
>>> s1 = set([1, 2, 3])
>>> s1.add(4)           ➡ 이미 만들어진 set 자료형에 값을 추가
```

### (2) 값 여러 개 추가하기(update)

```
>>> s1 = set([1, 2, 3])
>>> s1.update([4, 5, 6]) ➡ 여러 개의 값을 한꺼번에 추가(update)
```

### (3) 특정 값 제거하기(remove)

```
>>> s1 = set([1, 2, 3])
>>> s1.remove(2)        ➡ 특정 값을 제거
```

## 8 참, 거짓 자료형

### ■ 자료형의 참과 거짓을 구분하는 기준

값	참 or 거짓	값	참 or 거짓
"python"	참	{}	거짓
""	거짓	1	참
[1, 2, 3]	참	0	거짓
[]	거짓	None	거짓
()	거짓		

- 문자열, 리스트, 튜플, 딕셔너리 등의 값이 비어 있으면(" ", [], (), {}) ➡ 거짓
- 문자열, 리스트, 튜플, 딕셔너리 등의 값이 비어있지 않으면 ➡ 참
- 숫자는 그 값이 0일 때 ➡ 거짓

```
>>> a = [1, 2, 3, 4]
>>> while a:           ➡ 조건문이 참인 동안 조건문 안에 있는 문장을 반복 수행
    a.pop()           ➡ 리스트 a의 마지막 요소를 꺼내는 함수
    ➡ 꺼낼 것이 없으면 a가 빈 리스트로 거짓이 되어 반복문이 중지 됨
```

while 조건문:  
수행할 문장

```
>>> if []:             ➡ []는 비어 있는 리스트로 거짓
    print("True")
else:
    print("False")
```

```
>>> if [1, 2, 3]:      ➡ [1, 2, 3]은 요소값이 있는 리스트이기 때문에 참
    print("True")
else:
    print("False")
```

## 9 변수

### ■ 변수 만드는 방법

- =(assignment) 기호를 사용
- 다른 언어처럼 변수의 자료형을 함께 쓸 필요는 없다.
  - ➔ 파이썬은 변수에 저장된 값을 기준으로 자동으로 자료형이 결정
- 변수명 = 변수에 저장할 값
- 파이썬의 모든 자료형은 객체
  - ➔ 숫자 3은 상수값이 아닌 하나의 "정수형 객체"
- a=3과 같이 선언 ➔ a.real처럼 내장 함수를 바로 사용 가능

```
>>> type(3)
<class 'int'>
```

```
>>> a = 3
```

```
>>> b = 3
```

```
>>> a is b ➔ True : a = 3을 입력하는 순간 3이라는 정수형 객체가 생성
➔ 변수 a는 3이라는 객체의 메모리 주소를 가리킨다.
➔ 다음에 변수 b가 동일한 객체인 3을 가리킨다.
➔ 3이라는 정수형 객체를 가리키는 변수가 2개가 됨.
➔ 이 두 변수는 가리키고 있는 대상이 동일
```

```
>>> import sys
```

```
>>> sys.getrefcount(3) ➔ 30 : 자료형에 대한 참조 개수를 알려주는 함수
```

```
>>> a = 3
```

```
>>> sys.getrefcount(3) ➔ 31
```

```
>>> b = 3
```

```
>>> sys.getrefcount(3) ➔ 32
```

```
>>> c = 3
```

```
>>> sys.getrefcount(3) ➔ 33
```

### ■ 변수를 만드는 여러 가지 방법

```
>>> a, b = ('python', 'life') ➔ 튜플로 a, b에 값을 대입
➔ 튜플은 괄호를 생략해도 됨.
>>> (a, b) = 'python', 'life'
>>> [a,b] = ['python', 'life'] ➔ 리스트로 변수를 만들 수도 있음
>>> a = b = 'python' ➔ 여러 개의 변수에 같은 값을 대입 가능
>>> a = 3
>>> b = 5
>>> a, b = b, a ➔ 두 변수의 값을 아주 간단히 바꿀 수 있음
```

## ■ 메모리에 생성된 변수 없애기

- 레퍼런스 카운트 : 3이라는 객체를 가리키는 변수들의 개수
- 레퍼런스 카운트가 0이 되는 순간 3이라는 객체는 자동으로 사라짐.
  - ➔ 가비지 콜렉션(Garbage collection, 쓰레기 수집)

```
>>> a = 3
>>> b = 3
>>> del(a)
>>> del(b)
```

- ➔ 레퍼런스 카운트가 0이 되어 정수형 객체를 메모리에서 사라지게 함
- ➔ del 명령어를 이용하여 일일이 삭제할 필요는 없음
- ➔ 파이썬이 이 모든 것을 자동으로 해주기 때문

## ■ 리스트를 변수에 넣고 복사하고자 할 때

```
>>> a = [1,2,3]
>>> b = a      ➔ b라는 변수에 a가 가리키는 리스트를 대입
>>> a[1] = 4    ➔ a[1]을 4라는 값으로 바꾸면 a 리스트와 b 리스트도 함께 바뀜
                ➔ a, b 모두 같은 리스트인 [1, 2, 3]을 가리키고 있었기 때문
>>> a           ➔ [1, 4, 3]
>>> b           ➔ [1, 4, 3]
```

## ■ 다른 리스트를 만드는 방법

b 변수를 생성할 때 a와 같은 값을 가지도록 복사해 넣으면서  
a가 가리키는 리스트와는 다른 리스트를 가리키게 하는 방법

### (1) [:] 이용

```
>>> a = [1, 2, 3]
>>> b = a[:]      ➔ 리스트 전체를 가리키는 [:]을 이용해서 복사
>>> a[1] = 4
>>> a             ➔ [1, 4, 3]
>>> b             ➔ [1, 2, 3]
```

### (2) copy 모듈 이용

```
>>> from copy import copy
>>> b = copy(a)    ➔ b = copy(a)는 b = a[:]과 동일
>>> b is a
    ➔ is : 두 변수가 같은 값을 가지면서 다른 객체를 생성했는지 확인하는 함수
    true : 서로 같은 객체,
    false : 서로 다른 객체
```

## 10 if

➡ if : 프로그래밍에서 조건을 판단하여 해당 조건에 맞는 상황을 수행하는 데 쓰이는 것

```
>>> money = 1
>>> if money:
    print("커피")
else:
    print("물")
```

### ■ if문의 기본 구조

```
if 조건문:
    수행할 문장1

else:
    수행할 문장A
```

➡ 유의사항 : 들여쓰기  
공백 또는 탭으로 통일해서 사용

### ■ 조건문이란?

if 조건문에서 "조건문"이란 참과 거짓을 판단하는 문장

자료형	참	거짓
숫자	0이 아닌 숫자	0
문자열	"abc"	""
리스트	[1,2,3]	[]
터플	(1,2,3)	()
딕셔너리	{"a":"b"}	{}

### ■ 비교연산자

비교연산자	설명	비교연산자	설명
x < y	x가 y보다 작다	x != y	x와 y가 같지 않다
x > y	x가 y보다 크다	x >= y	x가 y보다 크거나 같다
x == y	x와 y가 같다	x <= y	x가 y보다 작거나 같다

```
>>> x = 3
>>> y = 2
>>> x > y      ➡ True
>>> x < y      ➡ False
>>> x != y     ➡ True
```

### ■ and, or, not

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

"돈이 3000원 이상 있거나 카드가 있다면 커피를 마시고 그렇지 않으면 물을 마셔라."

```
>>> money = 2000
>>> card = 1
>>> if money >= 3000 or card:
    print("커피")
else:
    print("물")
```

### ■ in, not in

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

```
>>> 1 in [1, 2, 3]           ➡ True
>>> 1 not in [1, 2, 3]      ➡ False
>>> 'a' in ('a', 'b', 'c') ➡ True
>>> 'j' not in 'python'     ➡ True
```

"만약 주머니에 돈이 있으면 커피를 마시고 그렇지 않으면 물을 마셔라."

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
    print("커피")
else:
    print("물")
```

### ■ 조건문에서 아무 일도 하지 않게 설정하는 방법

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket:
    pass           ➡ 아무 일도 하지 않는다.
else:
    print("물")
```

### ■ if문을 한 줄로 작성

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket : pass
else: print("카드를 꺼내라")
```

## ■ 다양한 조건을 판단하는 elif

```

If <조건문>:
    <수행할 문장>

elif <조건문>:
    <수행할 문장>

else:
    <수행할 문장>
..

```

```

>>> pocket = ['paper', 'handphone']
>>> card = 1
>>> if 'money' in pocket:
        print("커피")
    else:
        if card:
            print("코코아")
        else:
            print("커피")

```

```

>>> pocket = ['paper', 'cellphone']
>>> card = 1
>>> if 'money' in pocket:
        print("커피")
    elif card:
        print("코코아")
    else:
        print("물")

```



# 11 while

## ■ while문의 기본 구조

```
while <조건문>:
    <수행할 문장1>
    ....
```

## ■ "열 번 찍어 안 넘어 가는 나무 없다" 라는 속담

```
>>> treeHit = 0
>>> while treeHit < 10:
    treeHit = treeHit + 1
    print("나무를 %d번 찍었습니다." % treeHit)
    if treeHit == 10:
        print("나무 넘어갑니다.")
```

treeHit	조건문	조건판단	수행하는 문장	while문
0	0 < 10	참	나무를 1번 찍었습니다.	반복
1	1 < 10	참	나무를 2번 찍었습니다.	반복
..				
9	9 < 10	참	나무를 10번 찍었습니다. 나무 넘어갑니다.	반복
10	10 < 10	거짓		종료

⇒ while문의 조건문은 treeHit < 10 이다. 즉, treeHit가 10보다 작은 동안에 while문이 동작

## ■ number가 4가 아닌 동안 prompt를 출력하고 사용자로부터 번호를 입력받기

```
>>> number = 0
>>> while number != 4:
    print(prompt)
    number = int(input())
```

⇒ 사용자의 숫자 입력을 받아들이는 것

## ■ while문 강제로 빠져나가기

```
1. >>> coffee = 10
2. >>> money = 300
3. >>> while money:
4.     print("돈을 받고 커피를 줍니다.")
5.     coffee = coffee - 1
6.     print("남은 커피의 양은 %d개입니다." % coffee)
7.     if not coffee:
8.         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
9.         break
```

■ break문 이용해 자판기 작동 과정 만들기

```
# coffee.py

1. coffee = 10
2. while True:
3.     money = int(input("돈을 넣어 주세요: "))
4.     if money == 300:
5.         print("커피를 줍니다.")
6.         coffee = coffee -1
7.     elif money > 300:
8.         print("거스름돈 %d를 주고 커피를 줍니다." % (money -300))
9.         coffee = coffee -1
10.    else:
11.        print("돈을 다시 돌려주고 커피를 주지 않습니다.")
12.        print("남은 커피의 양은 %d개 입니다." % coffee)
13.        if not coffee:
14.            print("커피가 다 떨어졌습니다. 판매를 중지 합니다.")
15.            break
```

C:\WPython>python coffee.py

돈을 넣어 주세요:

입력란에 여러 숫자를 입력해 보면서 결과를 확인하자.

돈을 넣어 주세요: 500

거스름돈 200를 주고 커피를 줍니다.

돈을 넣어 주세요: 300

커피를 줍니다.

돈을 넣어 주세요: 100

돈을 다시 돌려주고 커피를 주지 않습니다.

남은 커피의 양은 8개입니다.

돈을 넣어 주세요:

<<UTF-8코드를 이용하여 한글 지원>>

파이썬 2.7 버전을 사용시

```
# -*- coding: utf-8 -*-
```

파이썬 3 버전을 사용 : 파일을 저장할 때 파일 인코딩을 utf-8로 맞춰 저장

C:\WPython>python coffee.py

File "coffee.py", line 9

SyntaxError: (unicode error) 'utf-8' codec can't decode byte

0xb9 in position 0:

invalid start byte

### ■ 조건에 맞지 않는 경우 맨 처음으로 돌아가기

→ while문을 빠져나가지 않고 while문의 맨 처음(조건문)으로 다시 돌아가게 함

→ **continue**문

→ 1부터 10까지의 숫자 중 홀수만을 출력하는 예제

```
>>> a = 0
>>> while a < 10:
    a = a+1
    if a % 2 == 0: continue    → 짝수일때마다 while문으로 이동함.
    print(a)
```

### ■ 무한 루프

```
while True:
    수행할 문장1
    .....
```

```
>>> whileTrue:
    print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
```

### ■ 피보나치 수열 구하기

```
array = [0,1]

deffibonach(n):
    i=0
    while array[i]+array[i+1] <= n:
        array.append(array[i]+array[i+1])
        i+=1
```

### ■ 예시

<pre>number=1 while number&lt;11:     print(number*2)     number=number+1</pre>	<pre>number=1 while number&lt;11:     print(number, "x 5 =", number*5)     number=number+1</pre>
<pre>number=1 while number&lt;=100:     print(number)     number=number+1</pre>	<pre>number=1 while 101&gt;number:     print(number)     number=number+1</pre>

## 12 for

### ■ for문의 기본 구조

```
for 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
```

➡ 리스트나 튜플, 문자열의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 "수행할 문장1", "수행할 문장2" 등이 수행된다.

### ■ for문 예제

#### (1) 전형적인 for문

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:
    print(i)
```

➡ ['one', 'two', 'three']라는 리스트의 첫 번째 요소인 'one'이 먼저 i 변수에 대입된 후 print(i)라는 문장을 수행한다. 다음에 'two'라는 두 번째 요소가 i 변수에 대입된 후 print(i) 문장을 수행하고 리스트의 마지막 요소까지 이것을 반복한다.

#### (2) 다양한 for문의 사용

```
>>> a = [(1,2), (3,4), (5,6)]
>>> for (first, last) in a:
    print(first + last)      ➡ 3 7 11
```

➡ a 리스트의 요소값이 튜플이기 때문에 각각의 요소들이 자동으로 (first, last)라는 변수에 대입된다. >>> (first, last) = (1, 2))

#### (3) for문의 응용

"총 5명의 학생의 시험 점수가 60점 이상 합격, 그렇지 않으면 불합격"

```
1. marks = [90, 25, 67, 45, 80]
2. number = 0
3. for mark in marks:
4.     number = number + 1
5.     if mark >= 60:
6.         print("%d번 학생은 합격입니다." % number)
7.     else:
8.         print("%d번 학생은 불합격입니다." % number)
```

### ■ for문에서의 continue

```
1. marks = [90, 25, 67, 45, 80]
2. number = 0
3. for mark in marks:
4.     number = number + 1
5.     if mark < 60: continue
6.     print("%d번 학생 축하합니다. 합격입니다. " % number)
```

➔ 점수가 60점 이하인 학생일 경우에는 mark < 60이 참이 되어 continue문이 수행된다.  
따라서 축하 메시지를 출력하는 print문을 수행하지 않고 for문의 처음으로 돌아가게 됨.

### ■ range함수

range라는 함수 : 숫자 리스트를 자동으로 만들어 주는 함수

```
>>> a = range(10)      ➔ 0부터 10 미만의 숫자를 포함하는 range객체를 만들기
>>> a = range(1, 11)   ➔ range(시작 숫자, 끝 숫자) 형태, 끝 숫자는 포함되지 않음
```

### ■ 1~10까지 합계 구하는 예시

```
>>> sum = 0
>>> for i in range(1, 11):
>>>     sum = sum + i
>>> print(sum)
```

### ■ 성적처리 예시

```
marks = [90, 25, 67, 45, 80]
for number in range(len(marks)):      ➔ len 함수 : 리스트 내 요소의 개수를 출력
    if marks[number] < 60: continue
    print("%d번 학생 축하합니다. 합격입니다." % (number+1))
```

### ■ range를 이용한 구구단

```
>>> for i in range(2,10):
>>>     for j in range(1, 10):
>>>         print(i*j, end=" ")
>>>     print('')
```

➔ end=" " : 해당 결과값을 출력할 때 다음 줄로 넘기지 않고 그 줄에 계속해서 출력

### ■ 리스트 안에 for문 포함하기

```
>>> a = [1,2,3,4]
>>> result = []
>>> for num in a:
>>>     result.append(num*3)
>>> print(result)      ➔ [3, 6, 9, 12]
```

## ■ 표현식을 사용

```
>>> result = [num * 3 for num in a]
>>> print(result)           ➡ [3, 6, 9, 12]
```

➡ 짝수에만 3을 곱하여 담고 싶다면 "if 조건"을 사용  
[표현식 for 항목 in 반복가능객체 if 조건]

```
>>> result = [num * 3 for num in a if num % 2 == 0]
>>> print(result)           ➡ [6, 12]
```

➡ 표현식 for 항목1 in 반복가능객체1 if 조건1  
for 항목2 in 반복가능객체2 if 조건2

## ■ 구구단

```
>>> result = [x*y for x in range(2,10)
               for y in range(1,10)]
>>> print(result)
```

## ■ 예시 : 1~1000에서 각 숫자의 개수 구하기

```
def process(n):
    r = {}
    for x in str(n):
        r[x] = r.get(x, 0) + 1
    return r

result = dict()
for i in range(1, 1001):
    for k, v in process(i).items():
        result[k] = result.get(k, 0) + v
print("\n".join("{0}:{1}".format(*x) for x in sorted(result.items())))
```

## ■ 예시 : 소수 구하기

```
def sieve(n):
    if n < 2:
        return []
    s = [0, 0] + [1] * (n - 1)
    for i in range(2, int(n**.5)+1):
        if s[i]:
            s[i*2::i] = [0] * ((n - i)//i)
    return [i for i, v in enumerate(s) if v]

print(len(sieve(1000)))
```

## 13 함수

### ■ 함수란 무엇인가?

- 프로그램에서 자주 사용되는 코드 블록을 따로 한번만 만들어 두고 필요할 때마다 불러서 사용하는 기능
- 자신이 만든 프로그램을 함수화하면 프로그램의 흐름을 파악하기 쉬움
- 코드가 중복되지 않고 간결해짐
- 한 번 작성해둔 코드를 여러 번 사용하므로 코드의 재사용성이 증가 됨
- 기능별로 함수를 작성해서 사용하므로 프로그램의 모듈화가 증대 됨
- 프로그램의 기능을 함수 단위로 나누어 묶어 두었기 때문에 코드를 수정하기 쉬워짐

### ■ 파이썬 함수의 구조

```
def 함수명(입력 인수):
    <수행할 문장1>
    <수행할 문장2>
```

```
>>> def sum(a, b):
        return a + b
>>> a = 3
>>> b = 4
>>> c = sum(a, b)
>>> print(c)
```

➡ 함수명 : sum, 인수 : a, b  
➡ return : 함수의 결과값을 돌려주는 명령어  
➡ 사용자 함수 sum 호출하여 사용

### ■ 함수의 형태

#### (1) 일반적인 함수

- 입력 값이 있고 결과 값이 있는 함수

```
def 함수이름(입력인수):
    <수행할 문장>
    return 결과값
```

```
def sum(a, b):
    result = a + b
    return result
```

```
>>> a = sum(3, 4)
>>> print(a)
```

➡ 결과값 받을 변수 = 함수명(입력 인수1, 입력 인수2,...)

#### (2) 입력값이 없는 함수

- 입력값이 없는 함수

```
def 함수이름():
    <수행할 문장>
    return 결과값
```

```
>>> def say():
        return 'hi'
```

```
>>> a = say()
>>> print(a)
```

➡ 결과값을 받을 변수 = 함수명()

### (3) 결과 값이 없는 함수

def 함수이름(입력인수): <수행할 문장>	defsum(a, b): print("%d, %d의 합은 %d" % (a, b, a+b))
-----------------------------	---

>>> sum(3, 4)                      ➡ 함수명(입력 인수1, 입력 인수2, )

※ 결과값은 오직 return 명령어로만 돌려받을 수 있다.

### (4) 입력값도 결과값도 없는 함수

def 함수이름(): <수행할 문장>	>>> defsay(): print('Hi')
-------------------------	------------------------------

>>> say()                      ➡ 함수명()

### ■ 입력값이 여러 개일 때

def 함수이름(*입력변수): <수행할 문장>	>>> def sum_many(*args): sum = 0 for i in args: sum = sum + i return sum
------------------------------	--

```
>>> result = sum_many(1,2,3)
>>> print(result)
>>> result = sum_many(1,2,3,4,5,6,7,8,9,10)
>>> print(result)
```

```
>>> defsum_mul(choice, *args):
    if choice == "sum":
        result = 0
        for i in args:
            result = result + i
    elif choice == "mul":
        result = 1
        for i in args:
            result = result * i
    return result
```

```
>>> result = sum_mul('sum', 1,2,3,4,5)
>>> print(result)
```

```
>>> result = sum_mul('mul', 1,2,3,4,5)
>>> print(result)
```



## ■ 출력값이 여러 개일 때

```
>>> def sum_and_mul(a,b):
    return a+b, a*b
>>> result = sum_and_mul(3,4)      ➡ a+b와 a*b는 튜플값으로 (a+b, a*b)로 돌려줌
                                   ➡ result = (7, 12)
>>> sum, mul = sum_and_mul(3, 4)    ➡ sum, mul = (7, 12)이 됨. sum=7, mul=12

>>> def sum_and_mul(a,b):
    return a+b                      ➡ return문을 만나면 결과값을 돌려주고 함수를 종료
    return a*b                     ➡ 2번째 return문은 실행되지 않음.
>>> def say_nick(nick):
    if nick == "장독":
        return                      ➡ 리턴값은 없음(함수 리턴값은 return문에 의해서만 수행)
    print("나의 별명 = %s " % nick)
>>> say_nick('김치')
>>> say_nick('장독')
```

## ■ 입력 인수에 초기값 미리 설정하기

```
def say_myself(name, old, man=False):      ➡ man=False 미리 입력 인수에 초기값 설정
    print("나의 이름은 %s " % name)
    print("나이는 %d살 " % old)
    if man:
        print("남자")
    else:
        print("여자")

>>> say_myself("김태희", 27) >>> say_myself("김태희", 27, False)
>>> say_myself("장동건", 27, True)
```

※ 함수 입력 인수에 초기값을 설정시 주의할 사항 : 초기값이 들어가는 변수는 마지막에 위치

```
>>> def say_myself(name, man=True, old):
    SyntaxError: non-default argument follows default argument
```

## ■ 함수 안에서 선언된 변수의 효력 범위

- 함수 안에서 사용한 변수의 이름을 함수 밖에서도 동일하게 사용 가능

<pre># test.py a = 1 def test(a):     a = a + 1    ➡ 함수안의 a변수 test(a) print(a) ➡ 결과값 : 1 (함수밖의 a변수)</pre>	<pre># test_error.py def test(a):     a = a + 1 test(3) print(a) ➡ 오류 발생</pre>
---	--

## ■ 함수 안에서 함수 밖의 변수를 변경하는 방법

### (1) return 이용하기

```
a = 1
def test(a):
    a = a + 1
    return a
```

```
a = test(a)
print(a)
```

➡ test 함수 안의 a 변수는 함수 밖의 a와는 다른 것이다.

### (2) global 명령어 이용하기

```
a = 1
def test():
    global a
    a = a + 1
```

➡ 함수 안에서 함수 밖의 a 변수를 직접 사용한다는 의미

➡ 외부 변수에 종속적인 함수는 그다지 좋은 함수가 아니다

```
test()
print(a)
```

## ■ 예시

```
import random
```

```
computer_number = random.randint(1, 100)
```

```
def is_same(target, number):
```

```
    if target == number:
```

```
        result="Win"
```

```
    elif target > number:
```

```
        result="Low"
```

```
    else:
```

```
        result="High"
```

```
    return result
```

```
print("안녕\n난 1부터 100 중 숫자 하나를 골랐어요.")
```

```
guess = int(input("뭔지 쓰고 엔터 키를 누르세요."))
```

```
higher_or_lower = is_same(computer_number, guess)
```

```
while higher_or_lower != "Win":
```

```
    if higher_or_lower == "Low":
```

```
        guess = int(input("그것 보단 높습니다. 다시 생각해보세요."))
```

```
    else:
```

```
        guess = int(input("그것 보단 낮습니다. 다시 생각해보세요."))
```

```
    higher_or_lower = is_same(computer_number, guess)
```

```
input("정답!\n잘 했어요.\n\n\n\n마치려면 엔터 키를 누르세요.")
```

### ■ 계산기 예제

```
from tkinter import *
from decimal import *

def click(key):
    entry.insert(END, key)

window = Tk()
window.title("나의 계산기")

top_row = Frame(window)
top_row.grid(row=0, column=0, columnspan=2, sticky=N)

entry = Entry(top_row, width=22, bg="light green")
entry.grid()

num_pad = Frame(window)
num_pad.grid(row=1, column=0, sticky=W)

num_pad_list = ['7', '8', '9', '4', '5', '6', '1', '2', '3', '0', '.', '=']
r = 1
c = 0
for b in num_pad_list:
    Button(num_pad, text=b, width=1, command=click).grid(row=r, column=c)
    c = c+1
    if c > 2:
        c = 0
        r = r+1

operator = Frame(window)
operator.grid(row=1, column=1, sticky=E)
operator_list = ['*', '/', '+', '-', '(', ')', 'C']

r = 2
c = 0
for b in operator_list:
    Button(operator, text=b, width=1, command=click).grid(row=r, column=c)
    c = c+1
    if c > 1:
        c = 0
        r = r+1

window.mainloop()
```

## ※ 내장함수

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()	

## 14 사용자 입력과 출력

### ■ 키보드로부터 사용자 입력

- input의 사용
 

```
>>> a = input() ➔ input은 입력되는 모든 것을 문자열로 취급
```
- 안내 문구 또는 질문이 나와서 사용자에게 입력을 받을 때
 

```
input("질문 내용")
>>> number = input("숫자를 입력하세요: ")
```

### ■ 출력하기

- 큰따옴표("")로 둘러싸인 문자열은 + 연산과 동일
 

```
>>> print("life" "is" "too short")
>>> print("life"+"is"+"too short")
```

- 문자열 띄어쓰기는 콤마로 함.

```
>>> print("life", "is", "too short")
```

- 한 줄에 결과값 출력하기 위해서는 end를 이용해 끝 문자를 지정

```
>>> for i in range(10):  
    print(i, end=' ')
```

## ■ 예시

```
import random
```

```
ans1="자! 해보세요!"
```

```
ans2="됐네요, 이사람아"
```

```
ans3="뭐라고? 다시 생각해보세요."
```

```
ans4="모르니 두려운 것입니다."
```

```
ans5="철쭉인가요?? 제 정신이 아니군요!"
```

```
ans6="당신이라면 할 수 있어요!"
```

```
ans7="해도 그만, 안 해도 그만, 아니겠어요?"
```

```
ans8="맞아요. 당신은 올바른 선택을 했어요."
```

```
want_more_advice="y"
```

```
print("MyMagic8Ball에 오신 것을 환영합니다.")
```

```
user_name = input("안녕. 당신의 이름을 입력하세요.\n")
```

```
print("만나서 반가워요.", user_name, ".", sep="")
```

```
while(want_more_advice == "y"):
```

```
    question = input("조언을 구하고 싶으면 질문을 입력하고 엔터 키를 누르세요.\n")
```

```
    print("고민 중 ...\n" * 4)
```

```
    choice=random.randint(1, 8)
```

```
    if choice==1:
```

```
        answer=ans1
```

```
    elif choice==2:
```

```
        answer=ans2
```

```
    else:
```

```
        answer=ans8
```

```
    print(answer)
```

```
    want_more_advice = input("\n좀 더 충고가 필요한가요? y/n\n")
```

```
print("\n안녕", user_name, ". 이용해주셔서 고마워요.", sep="")
```

```
input("\n\n끝내려면 엔터 키를 누르세요.")
```

## 15 파일 읽기 쓰기

### ■ 파일 생성하기

파일 객체 = open(파일 이름, 파일 열기 모드)

```
f = open("새파일.txt", 'w')
f.close()
```

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가 시킬 때 사용

➡ 만약 새파일.txt라는 파일을 C:/Python이라는 디렉터리에 생성하고 싶다면

```
f = open("C:/Python/새파일.txt", 'w')
f.close()
```

### ■ 파일을 쓰기 모드로 열어 출력값 적기

```
f = open("C:/Python/새파일.txt", 'w')
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

```
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    print(data)
```

C:\Python>python writedata.py

➡ 파일의 생성여부는 탐색기를 이용

### ■ 프로그램의 외부에 저장된 파일을 읽는 여러 가지 방법

#### (1) readline() 함수

```
f = open("C:/Python/새파일.txt", 'r')
line = f.readline()
print(line)
f.close()
```

```
f = open("C:/Python/새파일.txt", 'r')
while True:
    line = f.readline()
    if not line: break
    print(line)
f.close()
while 1:
    data = input()
    if not data: break
    print(data)
```

**(2) readlines() 함수 이용하기**

```
f = open("C:/Python/새파일.txt", 'r')
lines = f.readlines()
for line in lines:
    print(line)
f.close()
```

**(3) read() 함수 이용하기**

```
f = open("C:/Python/새파일.txt", 'r')
data = f.read()
print(data)
f.close()
```

**■ 파일에 새로운 내용 추가하기**

쓰기 모드('w') : 이미 존재하는 파일일 경우 기존 내용이 모두 사라지게 된다  
추가 모드('a') : 원래 있던 값을 유지하면서 단지 새로운 값만 추가

```
f = open("C:/Python/새파일.txt", 'a')
for i in range(11, 20):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

```
C:\Python>python adddata.py
```

**■ with문과 함께 사용하기**

- 파일을 열고 닫는 것을 자동으로 처리 : with문 이용

```
with open("foo.txt", "w") as f:
    f.write("Life is good")
```

**■ sys 모듈로 입력 인수 넣기**

도스 명령어 [인수1 인수2 ...]

```
#sys1.py
import sys
args = sys.argv[1:] ➔ sys 모듈의 argv는 명령창에서 입력한 인수
for i in args:
    print(i)
```

```
C:\Python>python sys1.py aaa bbb ccc
```

```
#sys2.py
import sys
args = sys.argv[1:]
for i in args:
    print(i.upper(), end=' ')
```

→ 소문자를 입력받아서  
→ 대문자로 바꾸어 줌

C:\WPython>python sys2.py life is good

#### ■ 파일에 기록된 탭을 공백문자로 바꾸기

```
filename=input("Enter your file name : ")
tempfile=open(filename)
tempfile=tempfile.read()
temp_str=tempfile.replace("\t"," ")
tempfile=open(filename,'w')
tempfile.write(temp_str)
tempfile.close()
```

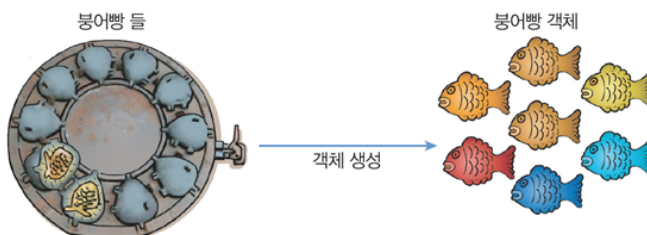
## 16 클래스

#### ■ 클래스란?

- 객체를 만들어내기 위해 정의된 설계도, 틀
- 클래스는 객체가 아님. 실체도 아님
- 멤버 변수와 멤버 함수 선언

#### ■ 객체

- 객체는 생성될 때 클래스의 모양을 그대로 가지고 탄생
- 멤버 변수와 멤버 함수로 구성
- 메모리에 생성, 실체(instance)라고도 부름
- 하나의 클래스 틀에서 찍어낸 여러 개의 객체 생성 가능
- 객체들은 상호 별도의 공간에 생성





## ■ 클래스의 구조

```
class 클래스이름[(상속 클래스명)]:
    <클래스 변수 1>
    <클래스 변수 2>

    def 클래스함수1(self[, 인수1, 인수2,...]):
        <수행할 문장 1>
        <수행할 문장 2>

    def 클래스함수2(self[, 인수1, 인수2,...]):
        <수행할 문장1>
        <수행할 문장2>
```

<pre>result = 0 def adder(num):     global result     result += num     return result</pre>	<pre>class Calculator:     def __init__(self):         self.result = 0     def adder(self, num):         self.result += num         return self.result</pre>
<pre>print(adder(3)) print(adder(4))</pre>	<pre>cal1 = Calculator() cal2 = Calculator() print(cal1.adder(3)) print(cal1.adder(4)) print(cal2.adder(3)) print(cal2.adder(7))</pre>
<pre>class Simple:     pass</pre>	<pre>a = Simple()</pre>

## ■ 클래스 변수

```
>>> class Service:
    secret = "나는 홍길동입니다."
```

➡ 클래스 이름은 Service

```
>>> pey = Service()
>>> pey.secret
```

➡ secret 변수를 '.'(도트 연산자)를 이용해서 호출

## ■ 클래스 함수

```
>>> class Service:
    secret = "나는 홍길동입니다."
    def sum(self, a, b):
        result = a + b
        print("%s + %s = %s입니다." % (a, b, result))
```

➡ self 변수를 클래스 함수의 첫 번째 인수로 받아야 함

```
>>> pey = Service()
>>> pey.sum(1,1)
```

```
>>> classService:
    secret = "나는 홍길동이다"
    def setname(self, name):
        self.name = name
    def sum(self, a, b):
        result = a + b
        print("%s님 %s + %s = %s" % (self.name, a, b, result))
```

```
>>> pey = Service()           ➡ self는 Service에 의해서 생성된 인스턴스(예: pey)를 지칭
>>> pey.setname("홍길동")
>>> pey.sum(1, 1)           ➡ 홍길동님 1 + 1 = 2
```

#### ■ 사칙연산 클래스 만들기

<pre>&gt;&gt;&gt; classFourCal:     defsetdata(self, first, second):         self.first = first         self.second = second     defsum(self):         result = self.first + self.second         return result     defmul(self):         result = self.first * self.second         return result     defsub(self):         result = self.first - self.second         return result     defdiv(self):         result = self.first / self.second         return result</pre>	<pre>&gt;&gt;&gt; a = FourCal() &gt;&gt;&gt; b = FourCal() &gt;&gt;&gt; a.setdata(4, 2) &gt;&gt;&gt; b.setdata(3, 7) &gt;&gt;&gt; a.sum() &gt;&gt;&gt; a.mul() &gt;&gt;&gt; a.sub() &gt;&gt;&gt; a.div() &gt;&gt;&gt; b.sum() &gt;&gt;&gt; b.mul() &gt;&gt;&gt; b.sub() &gt;&gt;&gt; b.div()</pre>
--	--

#### ■ 클래스의 상속

<pre>&gt;&gt;&gt; class Kim(Park):     lastname = "김"</pre>	<pre>&gt;&gt;&gt; class Park:     lastname = "박"     def__init__(self, name):         self.fullname = self.lastname + name     def travel(self, where):         print("%s, %s여행을 가다." % (self.fullname, where))</pre>
---	---

## ■ 메서드 오버라이딩

```
>>> class HouseKim(HousePark):
    lastname = "김"
    def travel(self, where, day):
        print("%s, %s여행 %d일 가네." % (self.fullname, where, day))
```

➡ 오버라이딩(Overriding) : 메서드 이름을 동일하게 다시 구현하는 메서드

## ■ 연산자 오버로딩 : 연산자(+, -, \*, /,, )를 객체끼리 사용할 수 있게 하는 기법

```
class Park:
    lastname = "박"
    def __init__(self, name):
        self.fullname = self.lastname + name
    def travel(self, where):
        print("%s, %s여행을 가다." % (self.fullname, where))
    def love(self, other):
        print("%s, %s 사랑에 빠졌네" % (self.fullname, other.fullname))
    def __add__(self, other):
        print("%s, %s 결혼했네" % (self.fullname, other.fullname))
```

```
class Kim(HousePark):
    lastname = "김"
    def travel(self, where, day):
        print("%s, %s여행 %d일 가네." % (self.fullname, where, day))
```

```
pey = Park("태환")
juliet = Kim("태희")
pey.love(juliet)
pey + juliet
```

## ■ 예시

```
from tkinter import *
canvas_height = 400
canvas_width = 600
canvas_color = "black"
p1_x=canvas_width/2
p1_y=canvas_height
p1_colour='green'
line_width=5
line_length=5
```

```

def p1_move(x, y):
    global p1_x
    global p1_y
    p1_new_x = p1_x + x
    p1_new_y = p1_y + y
    canvas.create_line(p1_x, p1_y, p1_new_x, p1_new_y, width=line_width, fill=p1_colour)
    p1_x = p1_new_x
    p1_y = p1_new_y
def p1_move_N(event):
    p1_move(0, -line_length)
def p1_move_S(event):
    p1_move(0, line_length)
def p1_move_E(event):
    p1_move(line_length, 0)
def p1_move_W(event):
    p1_move(-line_length, 0)
def erase_all(event):
    canvas.delete(ALL)
window = Tk()
window.title("MyEtchaSketch")
canvas=Canvas(bg=canvas_color, height=canvas_height, width=canvas_width, highlightthickness=0)
canvas.pack()
window.bind("<Up>", p1_move_N)
window.bind("<Down>", p1_move_S)
window.bind("<Left>", p1_move_W)
window.bind("<Right>", p1_move_E)
window.bind("u", erase_all)
window.mainloop()

```

## 17 모듈

### ■ 모듈 만드는 방법

- 모듈: 함수나 변수 또는 클래스 등을 모아 놓은 파일

#### # mod1.py

```
def sum(a, b):  
    return a + b
```

➡ 파일 mod1.py를 만들고 C:\WPython 디렉터리에 저장 : 모듈

```
C:\WPython>dir
```

```
C:\WPython>python
```

```
>>> import mod1
```

➡ import는 이미 만들어진 모듈을 사용

➡ import 모듈이름

```
>>> print(mod1.sum(3,4))
```

#### # mod1.py

```
def safe_sum(a, b):  
    if type(a) != type(b):  
        print("더할수 있는 것이 아닙니다.")  
        return  
    else:  
        result = sum(a, b)  
        return result
```

```
>>> import mod1
```

```
>>> print(mod1.safe_sum(3, 4))
```

```
>>> print(mod1.safe_sum(1, 'a'))
```

➡ return에 의해서 None 값을 돌려주게 됨

### ■ "from 모듈이름 import 모듈함수"를 사용

```
>>> from mod1 import sum
```

```
>>> sum(3, 4)
```

```
>>> from mod1 import sum, safe_sum
```

```
>>> from mod1 import *
```

### ■ if \_\_name\_\_ == "\_\_main\_\_":

```
# mod1.py
```

```
def sum(a, b):  
    return a+b
```

```
def safe_sum(a, b):  
    if type(a) != type(b):
```



```

        print("더할 수 있는 것이 아닙니다.")
        return
    else:
        result = sum(a, b)
        return result

print(safe_sum('a', 1))
print(safe_sum(1, 4))
print(sum(10, 10.4))

```

C:\WPython>python mod1.py

```

더할 수 있는 것이 아닙니다.
None
5
20.4

```

C:\WPython>python

```

>>> import mod1
                                     ➡ mod1.py 파일의 sum과 safe_sum 함수를 사용하기 위해
                                     mod1.py 파일을 import하면 바로 실행이 되어 결과값 출력

더할 수 있는 것이 아닙니다.
None
5
20.4

```

➡ 이런 문제를 방지하기 위해 if \_\_name\_\_ == "\_\_main\_\_": 사용

```

if __name__ == "__main__":
    print(safe_sum('a', 1))
    print(safe_sum(1, 4))
    print(sum(10, 10.4))

```

## ■ 클래스나 변수 등을 포함한 모듈

```

# mod2.py
PI = 3.141592

class Math:
    def solv(self, r):
        return PI * (r ** 2)

def sum(a, b):
    return a+b

if __name__ == "__main__":

```

```
print(PI)
a = Math()
print(a.solve(2))
print(sum(PI , 4.4))
```

```
C:\Python>python mod2.py
```

```
C:\Python>python
```

```
>>> import mod2          ➡ __name__ == "__main__"이 거짓이 되어 아무런 값도 출력되지 않음
```

#### ■ 모듈에 포함된 변수, 클래스, 함수 사용하기

```
>>> print(mod2.PI)
>>> a = mod2.Math()
>>> print(a.solve(2))
>>> print(mod2.sum(mod2.PI, 4.4))
```

#### ■ 새 파일 안에서 이전에 만든 모듈 불러오기

```
# test.py
import mod2
result = mod2.sum(3, 4)
print(result)
```

#### ■ 모듈을 불러와서 사용하는 방법 (모듈을 저장한 디렉터리로 이동하지 않음)

##### (1) sys.path.append(모듈을 저장한 디렉터리) 사용하기

```
>>> import sys
>>> sys.path          ➡ sys 모듈은 파이썬 라이브러리가 설치되어 있는 디렉터리를 확인
['', 'C:\\Windows\\SYSTEM32\\python35.zip', 'c:\\Python35\\DLLs',
 'c:\\Python35\\lib', 'c:\\Python35', 'c:\\Python35\\lib\\site-packages']
➡ 소스코드 안에서는 반드시 / 또는 \\ 기호를 사용
>>> sys.path.append("C:/Python/Mymodules")
➡ sys.path에 C:\Python\Mymodules라는 디렉터리를 추가
>>> sys.path

>>> import mod2          ➡ 다른 디렉토리에 있는 모듈 호출 가능
>>> print(mod2.sum(3,4))
```

##### (2) PYTHONPATH 환경 변수 사용하기

```
C:\>set PYTHONPATH=C:\Python\Mymodules          ➡ set 도스 명령어를 이용하여 경로 추가
C:\>python
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more information.
>>> import mod2
>>> print(mod2.sum(3,4))
```

## 18 예외 처리

### ■ 오류 예시

#### (1) "FileNotFoundError" 오류

디렉터리 안에 없는 파일을 열려고 시도했을 때 발생하는 오류이다.

```
>>> f = open("나없는파일", 'r')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: '나없는파일'
```

➡ 가 발생

#### (2) ZeroDivisionError 오류 : 0으로 다른 숫자를 나눈 경우

```
>>> 4 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

#### (3) IndexError 오류

```
>>> a = [1,2,3]
>>> a[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

➡ 오류가 발생시 프로그램을 중단하고 오류메시지를 보여 준다.

### ■ 오류 예외 처리 기법

```
try:
    ...
except [발생 오류[as 오류 메시지 변수]]:
    ...
```

- ➡ try 블록 수행 중 오류가 발생하면 except 블록이 수행된다.
- ➡ try블록에서 오류가 발생하지 않는다면 except 블록은 수행되지 않는다.
- ➡ [ ] 기호 : 괄호 안의 내용을 생략 가능

#### (1) try, except만 쓰는 방법 : 오류가 발생하기만 하면 except 블록 수행

```
try:
    ...
except:
    ...
```



**(2) 발생 오류만 포함한 except문**

```
: except문에 미리 정해 놓은 오류와 일치할 때만 except 블록을 수행
try:
    ...
except 발생 오류:
    ...
```

**(3) 발생 오류와 오류 메시지 변수까지 포함한 except문**

```
: 오류 메시지의 내용까지 알고 싶을 때 사용하는 방법
try:
    ...
except 발생 오류 as 오류 메시지 변수:
    ...

try:
    4 / 0
except ZeroDivisionError as e:
    print(e)
```

**■ try .. else**

- else절은 예외가 발생하지 않은 경우에 실행
- 반드시 except절 바로 다음에 위치

```
try:
    f = open('foo.txt', 'r')
except FileNotFoundError as e: ➔ foo.txt라는 파일이 없다면 except절이 수행
    print(str(e))
else: ➔ foo.txt 파일이 있다면 else절이 수행
    data = f.read()
    f.close()
```

**■ try .. finally**

```
f = open('foo.txt', 'w')
try:
    # 무언가를 수행한다.
finally: ➔ finally절은 예외 발생 여부에 상관없이 항상 수행
    f.close()
```

**■ 오류 회피하기**

```
try:
    f = open("나없는파일", 'r')
except FileNotFoundError:
    pass ➔ FileNotFoundError가 발생할 경우 pass를 사용하여 오류를 그냥 회피
```

## ■ 오류 일부러 발생시키기

- raise라는 명령어를 이용해 오류를 강제로 발생

```
class Bird:
    def fly(self):
        raise NotImplementedError
        ➡ 꼭 작성해야 할 부분 구현되지 않은 경우 일부러 오류를 발생시키고자 사용

class Eagle(Bird):
    pass

eagle = Eagle() ➡ Eagle 클래스는 Bird 클래스를 상속
eagle.fly()
```

- ➡ Eagle 클래스에서 fly 함수를 구현하지 않았기 때문에 Bird 클래스의 fly 함수가 호출된다.  
raise문에 의해 다음과 같은 NotImplementedError가 발생

NotImplementedError가 발생되지 않게 하려면 Eagle 클래스에 fly 함수를 반드시 구현

```
class Eagle(Bird):
    def fly(self):
        print("very fast")
```

```
eagle = Eagle()
eagle.fly()
```

# 19 응용 예제

## ■ 버블 소트

```
1. classBubbleSort():
2.     defbsort(self, num):
3.         t = 0
4.         swap=0
5.         for i in range(len(num)):
6.             for j in range(len(num)-1):
7.                 if num[j]>num[j+1]:
8.                     t=num[j]
9.                     num[j]=num[j+1]
10.                    num[j+1]=t
11.                    swap+=1
12.                print num, swap
13. a=BubbleSort()
14. a.bsor([3, 1, 4, 1, 5, 9, 2, 6])
```

```
1. def bubble_sort(a):
2.     n_loop = 0
3.     n_swap = 0
4.     while True:
5.         i = 0
6.         n_loop += 1
7.         swap_hist = n_swap
8.         while i < len(a)-1:
9.             if a[i] > a[i+1]:
10.                 tmp = a[i]
11.                 a[i] = a[i+1]
12.                 a[i+1] = tmp
13.                 n_swap += 1
14.             i += 1
15.         if swap_hist == n_swap:
16.             print n_loop, n_swap
17.         return()
18.
19. a = [3, 1, 4, 1, 5, 9, 2, 6]
20. bubble_sort(a)
```

#### ■ 비밀번호 알아내기 프로그램

```
1. import random
2. response1 = "걱정 말고 다시 시도해보세요."
3. response2 = "그렇듯하지만 내 비밀번호는 아니에요. 다시 입력해보세요."
4. response3 = "내 비밀번호가 아니에요. 내 비밀번호는 정말 쉬워요."
5. response4 = "잘했어요!"
6. MY_PASSWORD = "내 비밀번호예요"
7. guess_correct = False
8. counter = 0
9. gave_in = False
10.
11. def is_correct(guess, password):
12.     return guess == password
13.
14. print("안녕.\n")
15. users_guess = input("추측한 내 비밀번호를 입력하세요. ")
16. true_or_false = is_correct(users_guess, MY_PASSWORD)
17.
18. while true_or_false == False:
19.     computer_response = random.randint(1, 3)
```

```

20.     if counter >= 2:
21.         yes_or_no = input("포기하시겠습니까? Y/N? ")
22.         if (yes_or_no == "Y" or yes_or_no == "y"):
23.             print("내 비밀번호가 ₩my password₩에 있습니다. 확인하시겠어요?")
24.             gave_in = True
25.             break
26.         else:
27.             print("₩n좋아요. 계속 하겠습니다. ...")
28.     if computer_response == 1:
29.         print(response1)
30.     elif computer_response == 2:
31.         print(response2)
32.     else:
33.         print(response3)
34.     users_guess = input("₩n다음 비밀번호는 무엇입니까? ")
35.     true_or_false = is_correct(users_guess, MY_PASSWORD)
36.     counter = counter+1
37. if gave_in == False: print(response4) # 이 줄은 새로운 줄에 적을 수 있습니다
38. input("₩n₩n₩n리턴 키를 눌러 끝내세요.")

```

#### ■ 비밀번호 알아내기 프로그램

```

1. f = open("c:/temp/test.txt")
2. data = f.read()
3. f.close()
4.
5. def decrypt(encrypted, key):
6.     result = []
7.     key = map(ord, list(key))
8.     total = 0
9.     for i, s in enumerate(encrypted):
10.         e = int(s)
11.         k = key[i%3]
12.         d = e ^ k
13.         total += d
14.         result.append(chr(d))
15.     return total, ''.join(result)
16.
17. alpha = "abcdefghijklmnopqrstuvwxyz"
18. encrypted = data.split(",")
19. for s1 in alpha:
20.     for s2 in alpha:
21.         for s3 in alpha:

```

```
22.         key = s1+s2+s3
23.         total, r = decrypt(encrypted, key)
24.         if r.find(" and ") != -1:
25.             print "key:", key
26.             print "sum:", total
27.             print r
```

#### ■ 단위 변환 프로그램

```
1. import unittest
2.
3. class U:
4.     def __init__(self, name):
5.         self.name = name
6.         self.parent = None
7.         self.rate_list = []
8.         self.value = 1.0
9.
10.    def set_rate(self, u, value):
11.        u.set_parent(self)
12.        u.value = value
13.        self.rate_list.append(u)
14.
15.    def set_parent(self, u):
16.        self.parent = u
17.
18.    def get_parent(self):
19.        return self.parent
20.
21.    def get_path(self, to):
22.        a = self._get_path(to, [to])
23.        b = to._get_path(self, [self])
24.        if len(a) >= len(b):
25.            return reversed(a)
26.        else:
27.            return b
28.
29.    def _get_path(self, to, path=[]):
30.        if self == to: return path
31.        parent = to.get_parent()
32.        if not parent:
33.            path.append(self)
34.        else:
```

```

35.         path.append(parent)
36.         self._get_path(parent, path)
37.         return path
38.
39.     def get_rate(self, to):
40.         rate = 1.0
41.         path = list(self.get_path(to))
42.         for i in range(len(path)):
43.             if i+1 < len(path): next = path[i+1]
44.             else: next = None
45.             if next in path[i].rate_list:
46.                 rate = rate * next.value
47.             elif next:
48.                 rate = rate / path[i].value
49.         return rate
50.
51.     def change(self, to, value):
52.         rate = self.get_rate(to)
53.         return value * rate
54.
55.     def __str__(self):
56.         return self.name
57.
58.
59. def exchange(fr, to):
60.     inch = U("inch")
61.     cm = U("cm")
62.     pt = U("pt")
63.     px = U("px")
64.     dxa = U("dxa")
65.     emu = U("emu")
66.     mm = U("mm")
67.
68.     inch.set_rate(cm, 2.54)
69.     inch.set_rate(pt, 72.0)
70.     inch.set_rate(px, 96.0)
71.     cm.set_rate(mm, 10.0)
72.     pt.set_rate(dxa, 20.0)
73.     dxa.set_rate(emu, 635.0)
74.
75.     fr_v, fr = fr.split()
76.     fr = eval(fr)

```

```
77.     to = eval(to)
78.     r = int(fr.change(to, int(fr_v)))
79.     return "%s %s" % (r, to.name)
80.
81.
82. class UnitTest(unittest.TestCase):
83.     def test1(self):
84.         self.assertEqual("10 cm", exchange("10 cm", "cm"))
85.         self.assertEqual("254 mm", exchange("10 inch", "mm"))
86.         self.assertEqual("768 pt", exchange("1024 px", "pt"))
87.         self.assertEqual("8 inch", exchange("768 px", "inch"))
88.         self.assertEqual("10 inch", exchange("9144000 emu", "inch"))
89.         self.assertEqual("800 px", exchange("12000 dxa", "px"))
90.
91.
92. if __name__ == "__main__":
93.     unittest.main()
94.
```

## ■ 볼링 프로그램

```
1. -*- coding: utf-8 -*-
2. 볼링 점수 계산 프로그램
3. 10 번째 프레임 부터 뒤로 몰아 입력받는다.
4. 12 번째 추가 기회가 생기면 득점을 앞으로 몰아 적는다.
5. 예)
6. 10 번째 프레임
7. (첫번째 투구) 쓰러트린 핀의 갯수를 입력하시오 : 0
8. (두번째 투구) 쓰러트린 핀의 갯수를 입력하시오 : 8
9. '''

10. FRAME_NUM = 12
11.
12. def chance():
13.     frist_collapsed_pin = input("(첫번째 투구) 쓰러트린 핀갯수를 입력하시오: ")
14.     if int(frist_collapsed_pin) == 10:
15.         return (10,0)
16.     second_collapsed_pin = input("(두번째 투구) 쓰러트린 핀갯수 입력하시오: ")
17.     return (int(frist_collapsed_pin) , int(second_collapsed_pin))
18.
19. def culc_frame(frame_count, strike = 0):
20.     global scores
21.     if frame_count is 9 and strike ==0:
```

```

22.         if scores[9][0] == 10 and scores[10][0] == 10 and scores[11][0] == 10:
23.             return 30
24.         elif scores[9][1] + scores[10][1] == 10:
25.             return scores[9][1] + scores[10][1] + scores[11][0]
26.         else:
27.             return scores[9][1] + scores[10][1]
28.
29.     score_point = scores[frame_count][0] + scores[frame_count][1]
30.     if score_point < 10:
31.         if strike == 2:
32.             return scores[frame_count][0]
33.         return score_point
34.     elif score_point == 10:
35.         if scores[frame_count][1] == 0:
36.             strike += 1
37.             if strike == 3:
38.                 return 10
39.             score_point += culc_frame(frame_count + 1, strike)
40.             return score_point
41.         else:
42.             if strike > 0:
43.                 score_point = (scores[frame_count][0] + scores[frame_count][1])
44.                 return score_point
45.             else:
46.                 score_point += scores[frame_count + 1][0]
47.                 return score_point
48.
49. scores = [None for _ in range(FRAME_NUM)]
50. for i in range(FRAME_NUM):
51.     print (i+1, '번째 프레임')
52.     scores[i] = chance()
53.
54. sum = 0
55. score_list = [0 for _ in range(FRAME_NUM)]
56. for i in range(10):
57.     sum += culc_frame(i)
58.     score_list[i] = sum
59. print ( '|---game-----|')
60.
61. for score, score_sum in zip(scores, score_list):
62.     print ('ball1, ball2 :', score, 'sum :', score_sum)

```



## ■ 공게임 프로그램

```
1. # MyPong의 주된 파일을 만듭니다.
2.
3. from tkinter import *
4. import table, ball, bat, random
5.
6.
7. window = Tk()
8. window.title("MyBreakout")
9. my_table = table.Table(window)
10.
11. # 전역 변수 초기화
12. x_velocity = 4
13. y_velocity = 10
14. first_serve = True
15.
16. # Ball 공장에서부터 볼을 주문합니다
17. my_ball = ball.Ball(table = my_table, x_speed=x_velocity, y_speed=y_velocity,
18.                      width=24, height=24, colour="red", x_start=288, y_start=188)
19.
20. # Bat 공장에서부터 배트를 주문합니다
21. bat_B = bat.Bat(table = my_table, width=100, height=10,
22.                 x_posn=250, y_posn=370, colour="blue")
23.
24. # Bat 클래스로부터 배트를 주문하지만 이것은 벽돌을 호출하는 것은 아닙니다.
25. bricks = []
26. b=1
27. while b < 7:
28.     i=80
29.     bricks.append(bat.Bat(table = my_table, width=50, height=20,
30.                           x_posn=(b*i), y_posn=75, colour="green"))
31.     b = b+1
32.
33. ##### 함수:
34. def game_flow():
35.     global first_serve
36.     # 첫번째 서버를 기다립니다:
37.     if(first_serve==True):
38.         my_ball.stop_ball()
39.         first_serve = False
40.
41.     # 배트에 공이 충돌하는지 감지
42.     bat_B.detect_collision(my_ball, sides_sweet_spot=False,
                           topnbottom_sweet_spot=True)
```

```

43.
44.     # 벽돌에 공이 충돌하는지 감지
45.     for b in bricks:
46.         if(b.detect_collision(my_ball, sides_sweet_spot=False) != None):
47.             my_table.remove_item(b.rectangle)
48.             bricks.remove(b)
49.         if(len(bricks) == 0):
50.             my_ball.stop_ball()
51.             my_ball.start_position()
52.             my_table.draw_score("", " YOU WIN!")
53.
54.     # 아래쪽 벽에 공이 충돌하는지 감지
55.     if(my_ball.y_posn >= my_table.height - my_ball.height):
56.         my_ball.stop_ball()
57.         my_ball.start_position()
58.         first_serve = True
59.         my_table.draw_score("", " WHOOPS!")
60.
61.     my_ball.move_next()
62.     window.after(50, game_flow)
63.
64. def restart_game(master):
65.     my_ball.start_ball(x_speed=x_velocity, y_speed=y_velocity)
66.     my_table.draw_score("", "")
67.
68. # 배트를 제어할 수 있도록 키보드의 키와 연결
69. window.bind("<Left>", bat_B.move_left)
70. window.bind("<Right>", bat_B.move_right)
71.
72. # 스페이스 키를 게임 재시작 기능과 연결
73. window.bind("<space>", restart_game)
74.
75. game_flow()
76. window.mainloop()

```

1. # 이 파일은 원래 MyPong을 위해 만든 공 클래스입니다.
2. # 이 클래스는 게임 영역의 측면에 부딪치는 공을 정의합니다.
- 3.
4. import table, random
- 5.
6. class Ball:
7. ##### 생성자
8. def \_\_init\_\_(self, table, width=14, height=14, colour="red", x\_speed=6,

```
y_speed=4, x_start=0, y_start=0):
9.     self.width = width
10.    self.height = height
11.    self.x_posn = x_start
12.    self.y_posn = y_start
13.    self.colour = colour
14.
15.    self.x_start = x_start
16.    self.y_start = y_start
17.    self.x_speed = x_speed
18.    self.y_speed = y_speed
19.    self.table = table
20.    self.circle = self.table.draw_oval(self)
21.
22.    #### 메서드
23.    def start_position(self):
24.        self.x_posn = self.x_start
25.        self.y_posn = self.y_start
26.
27.    def start_ball(self, x_speed, y_speed):
28.        self.x_speed = -x_speed if random.randint(0,1) else x_speed
29.        self.y_speed = -y_speed if random.randint(0,1) else y_speed
30.        self.start_position()
31.
32.    def move_next(self):
33.        self.x_posn = self.x_posn + self.x_speed
34.        self.y_posn = self.y_posn + self.y_speed
35.        # 공이 왼쪽 벽에 부딪혔을 때:
36.        if(self.x_posn <= 3):
37.            self.x_posn = 3
38.            self.x_speed = -self.x_speed
39.        # 공이 오른쪽 벽에 부딪혔을 때:
40.        if(self.x_posn >= (self.table.width - (self.width - 3))):
41.            self.x_posn = (self.table.width - (self.width - 3))
42.            self.x_speed = -self.x_speed
43.        # 공이 위쪽 벽에 부딪혔을 때:
44.        if(self.y_posn <= 3):
45.            self.y_posn = 3
46.            self.y_speed = -self.y_speed
47.        # 공이 아래쪽 벽에 부딪혔을 때:
48.        if(self.y_posn >= (self.table.height - (self.height - 3))):
49.            self.y_posn = (self.table.height - (self.height - 3))
50.            self.y_speed = -self.y_speed
51.        # 마지막으로 원의 이동:
```

```

52.         x1 = self.x_posn
53.         x2 = self.x_posn+self.width
54.         y1 = self.y_posn
55.         y2 = self.y_posn+self.height
56.         self.table.move_item(self.circle, x1, y1, x2, y2)
57.
58.     def stop_ball(self):
59.         self.x_speed = 0
60.         self.y_speed = 0

```

```

1. # 이 파일은 원래 MyPong을 위해 만든 Table 클래스입니다.
2. # 이 클래스는 Table의 처리 공간을 2D 직사각형으로 정의합니다.
3.
4. from tkinter import *
5.
6. class Table:
7.     ##### 생성자
8.     def __init__(self, window, colour="black", net_colour="green",
9.                  width=600, height=400, vertical_net=False, horizontal_net=False):
10.         self.width = width
11.         self.height = height
12.         self.colour = colour
13.
14.         # tkinter 공장에서부터 캔버스 주문:
15.         self.canvas = Canvas(window, bg=self.colour, height=self.height,
16.                               width=self.width)
17.         self.canvas.pack()
18.
19.         # tkinter 공장의 메서드를 사용하여 캔버스에 네트 추가:
20.         if(vertical_net):
21.             self.canvas.create_line(self.width/2, 0, self.width/2, self.height,
22.                                     width=2, fill=net_colour, dash=(15, 23))
23.         if(horizontal_net):
24.             self.canvas.create_line(0, self.height/2, self.width, self.height/2,
25.                                     width=2, fill=net_colour, dash=(15, 23))
26.
27.         # 득점판 추가:
28.         font = ("monaco", 72)
29.         self.scoreboard = self.canvas.create_text(300, 65, font=font, fill =
"darkgreen")

```

```
30.     ##### 메서드
31.     # Canvas에 직사각형을 추가하는 도구:
32.     def draw_rectangle(self, rectangle):
33.         x1 = rectangle.x_posn
34.         x2 = rectangle.x_posn + rectangle.width
35.         y1 = rectangle.y_posn
36.         y2 = rectangle.y_posn + rectangle.height
37.         c = rectangle.colour
38.         return self.canvas.create_rectangle(x1, y1, x2, y2, fill=c)
39.
40.     # Canvas에 타원을 추가하는 도구:
41.     def draw_oval(self, oval):
42.         x1 = oval.x_posn
43.         x2 = oval.x_posn + oval.width
44.         y1 = oval.y_posn
45.         y2 = oval.y_posn + oval.height
46.         c = oval.colour
47.         return self.canvas.create_oval(x1, y1, x2, y2, fill=c)
48.
49.     # canvas의 아이템을 조작할 수 있는 도구:
50.     def move_item(self, item, x1, y1, x2, y2):
51.         self.canvas.coords(item, x1, y1, x2, y2)
52.
53.     def remove_item(self, item):
54.         self.canvas.delete(item)
55.
56.     def change_item_colour(self, item, c):
57.         self.canvas.itemconfigure(item, fill=c)
58.
59.     # canvas에 득점판을 추가하는 도구:
60.     def draw_score(self, left, right):
61.         scores = str(right) + " " + str(left)
62.         self.canvas.itemconfigure(self.scoreboard, text=scores)
```

```
1. # 이 파일은 원래 MyPong을 위해 만든 Bat 클래스입니다.
2. import table
3. class Bat:
4.     ##### 생성자
5.     def __init__(self, table, width=15, height=100, x_posn=50, y_posn=50,
6.         colour="green", x_speed = 23, y_speed = 23):
7.         self.width = width
8.         self.height = height
```

```

8.         self.x_posn = x_posn
9.         self.y_posn = y_posn
10.        self.colour = colour
11.
12.        self.x_start = x_posn
13.        self.y_start = y_posn
14.        self.x_speed = x_speed
15.        self.y_speed = y_speed
16.        self.table = table
17.        self.rectangle = self.table.draw_rectangle(self)
18.
19.        ##### 메서드
20.        def detect_collision(self, ball, sides_sweet_spot=True,
topnbottom_sweet_spot=False):
21.            collision_direction = ""
22.            collision = False
23.            feel = 5
24.            # 배트 변수:
25.            top = self.y_posn
26.            bottom = self.y_posn + self.height
27.            left = self.x_posn
28.            right = self.x_posn + self.width
29.            v_centre = top + (self.height/2)
30.            h_centre = left + (self.width/2)
31.            # 공 변수:
32.            top_b = ball.y_posn
33.            bottom_b = ball.y_posn + ball.height
34.            left_b = ball.x_posn
35.            right_b = ball.x_posn + ball.width
36.            r = (right_b - left_b)/2
37.            v_centre_b = top_b + r
38.            h_centre_b = left_b + r
39.
40.            if((bottom_b > top) and (top_b < bottom) and (right_b > left) and (left_b <
right)):
41.                collision = True
42.                if(collision):
43.                    if((top_b > top-r) and (bottom_b < bottom+r) and (right_b > right) and
(left_b <= right)):
44.                        collision_direction = "E"
45.                        ball.x_speed = abs(ball.x_speed)
46.
47.                    elif((left_b > left-r) and (right_b < right+r) and (bottom_b > bottom)
and (top_b <= bottom)):

```

```
48.         collision_direction = "S"
49.         ball.y_speed = abs(ball.y_speed)
50.
51.         elif((left_b > left-r) and (right_b < right+r) and (top_b < top) and
(bottom_b >= top)):
52.             collision_direction = "N"
53.             ball.y_speed = -abs(ball.y_speed)
54.
55.         elif((top_b > top-r) and (bottom_b < bottom+r) and (left_b < left) and
(right_b >= left)):
56.             collision_direction = "W"
57.             ball.x_speed = -abs(ball.x_speed)
58.
59.         else:
60.             collision_direction = "miss"
61.
62.         if((sides_sweet_spot == True) and (collision_direction == "W" or
collision_direction == "E")):
63.             # 배트의 중심에서 얼마나 먼 거리에서 충돌이 발생했는지 y 값을 찾습
니다.
64.             adjustment = -(v_centre - v_centre_b)/(self.height/2)
65.             ball.y_speed = feel * adjustment
66.
67.         if((topnbottom_sweet_spot == True) and (collision_direction == "N" or
collision_direction == "S")):
68.             # 배트의 중심에서 얼마나 먼 거리에서 충돌이 발생했는지 x 값을 찾습니다.
69.             adjustment = -(h_centre - h_centre_b)/(self.width/2)
70.             ball.x_speed = feel * adjustment
71.
72.         return (collision, collision_direction)
73.
74.     def move_up(self, master):
75.         self.y_posn = self.y_posn - self.y_speed
76.         if(self.y_posn <= 0):
77.             self.y_posn = 0
78.         x1 = self.x_posn
79.         x2 = self.x_posn+self.width
80.         y1 = self.y_posn
81.         y2 = self.y_posn+self.height
82.         self.table.move_item(self.rectangle, x1, y1, x2, y2)
83.
84.     def move_down(self, master):
85.         self.y_posn = self.y_posn + self.y_speed
86.         far_bottom = self.table.height - self.height
```

```

87.         if(self.y_posn >= far_bottom):
88.             self.y_posn = far_bottom
89.         x1 = self.x_posn
90.         x2 = self.x_posn+self.width
91.         y1 = self.y_posn
92.         y2 = self.y_posn+self.height
93.         self.table.move_item(self.rectangle, x1, y1, x2, y2)
94.
95.     def move_left(self, master):
96.         self.x_posn = self.x_posn - self.x_speed
97.         if(self.x_posn <= 0):
98.             self.x_posn = 0
99.         x1 = self.x_posn
100.        x2 = self.x_posn+self.width
101.        y1 = self.y_posn
102.        y2 = self.y_posn+self.height
103.        self.table.move_item(self.rectangle, x1, y1, x2, y2)
104.
105.    def move_right(self, master):
106.        self.x_posn = self.x_posn + self.x_speed
107.        far_right = self.table.width - self.width
108.        if(self.x_posn >= far_right):
109.            self.x_posn = far_right
110.        x1 = self.x_posn
111.        x2 = self.x_posn+self.width
112.        y1 = self.y_posn
113.        y2 = self.y_posn+self.height
114.        self.table.move_item(self.rectangle, x1, y1, x2, y2)
115.
116.    def start_position(self):
117.        self.x_posn = self.x_start
118.        self.y_posn = self.y_start

```

### 참고 문헌

- <https://docs.python.org/3/>
- <http://www.slideshare.net/sk8erchoi/presentations>
- Effective Python 파이썬 코딩의 기술 : 똑똑하게 코딩하는 법
- 뇌를 자극하는 파이썬 3 : 프로그래밍을 배우기에 가장 재미있는 언어
- 대학생들을 위한 파이썬 프로그래밍 : 누구나 할 수 있는 가장 쉬운 프로그래밍 입문서
- 누구나 쉽게 배우는 파이썬 프로그래밍
- Do it! 점프 투 파이썬
- 창의적 사고를 위한 Python 프로그래밍 정복하기 (IT Holic 103)





#### 업무 총괄 및 기획

- 노민구 부산광역시교육청 교육국장
- 안주태 부산광역시교육청 인재개발과장
- 배정철 부산광역시교육청 정보교육담당장학관



#### 기획, 편성, 집필

- 이재혁 부산광역시교육청 인재개발과 장학사
- 정근식 부산광역시교육청 인재개발과 장학사
- 김성울 부산광역시교육청 인재개발과 장학사
- 이숙령 부산광역시교육청 인재개발과 주무관
- 장희숙 동의대학교 ICT공과대학 컴퓨터소프트웨어공학과 교수

### SW교육 담당교원 역량강화 연수교재

#### SW교육과 파이썬(Python) 프로그래밍

발행번호 : 부산교육 2016-101

발 행 일 : 2016. 5.

발 행 처 : 부산광역시교육청

#### 〈본 자료 파일 탑재 위치〉

부산광역시교육청 홈페이지 -> 과홈페이지 -> 인재개발과 -> 정보교육