

파이썬 포트폴리오

20191791 김상희

## A. 파이썬 개요

파이썬 언어와 컴퓨팅 사고력

파이썬 설치와 파이썬 셸 실행

## B. 파이썬 프로그래밍을 위한 기초 다지기

다양한 자료: 문자열과 수

변수와 키워드, 대입 연산자

자료의 표준 입력과 자료 변환 함수

## C. 문자열과 논리 연산

문자열 다루기

문자열 관련 메소드

논리 자료와 다양한 연산

## D. 조건과 반복

If ... else

For / while

Break / continue

## E. 리스트와 튜플

리스트란?

리스트의 부분 참조와 항목의 삽입과 삭제

항목의 순서나 내용을 수정할 수 없는 튜플

## F. 문자열과 논리 연산

키와 값인 쌍의 나열인 딕셔너리

중복과 순서가 없는 집합

내장 함수 zip()과 enumerate(), 시퀀스 간의 변환

## A.파이썬 개요 - 파이썬 언어와 컴퓨팅 사고력

### 파이썬이란 무엇일까?

파이썬은 1980년대 후반에 네덜란드의 귀도 반 로섬(Guido Van Rossum)이 개발했다. 반 로섬은 영국의 인기 코미디 그룹, 몬티 파이썬(Monty Python)의 열렬한 팬이었다. 파이썬은 거기서 이름을 따왔다. 반 로섬은 여전히 언어 개발에 활발히 참여하고 있으며 파이썬 커뮤니티에서 자비로운 종신 독재자(BDFL, Benevolent Dictator For Life)라는 애칭으로 불리고 있다.

출시 이후 여러 번에 걸쳐 새로운 버전이 탄생했으면 현재는 버전 3이 쓰이고 있다. 이는 웹 기반 앱과 데스크톱 앱 모두에서 널리 사용된다. 또한 아두이노(Arduino, 오픈 소스를 지향하는 마이크로 컨트롤러를 내장한 기기 제어용 기판-웁킨이)를 이용한 사물 인터넷 기기를 위해서도 사용된다. 매우 강력하고 유연한 언어다.

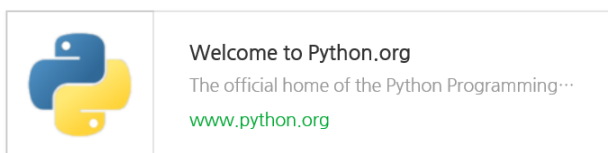
### 파이썬을 배우는 이유가 뭘까?

- 파이썬은 간단하면서도 직관성이 높은 언어다. 자바스크립트보다 더 편리하다.
- 파이썬은 서버에서부터 라즈베리 파이(Raspberry Pi)까지 어떤 플랫폼에서도 사용할 수 있다. 웹사이트에서부터 로봇까지 무엇이든 구축할 수 있다.
- 파이썬은 일상 업무를 자동화하는 데도 유용하다. 웹사이트에서 정보를 찾거나 자동적으로 자신의 파일에 이름을 붙이는 작업 같은 것을 할 수 있다.

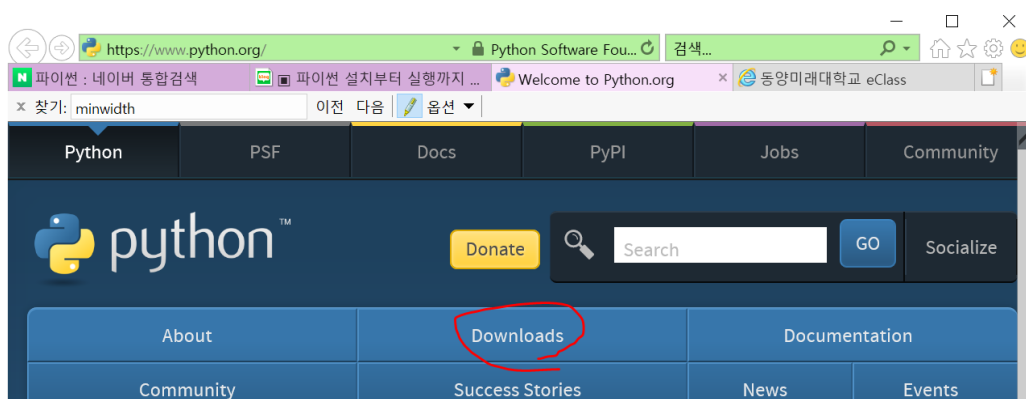
## A. 파이썬 개요 - 파이썬 설치와 파이썬 웹 실행

### 파이썬 설치 방법

<https://www.python.org/> <<파이썬 홈페이지



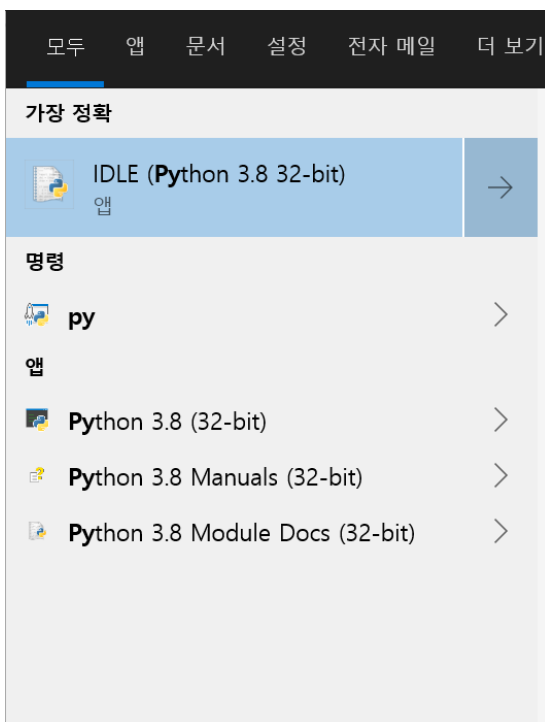
#### 1.파이썬 홈페이지에 들어간다



2. 최신버전의 파이썬을 다운받는다.



3. 설치가 잘 된 것을 확인한다.



## B. 파이썬 프로그래밍을 위한 기초 다지기 – 다양한 자료: 문자열과 수

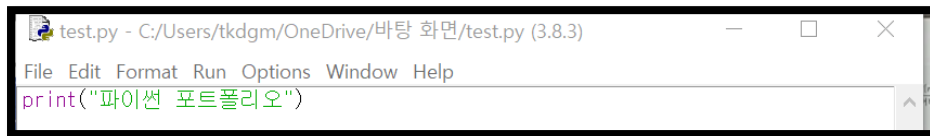
### 문자열

문자열이란 문자, 단어 등으로 구성된 문자들의 집합을 의미

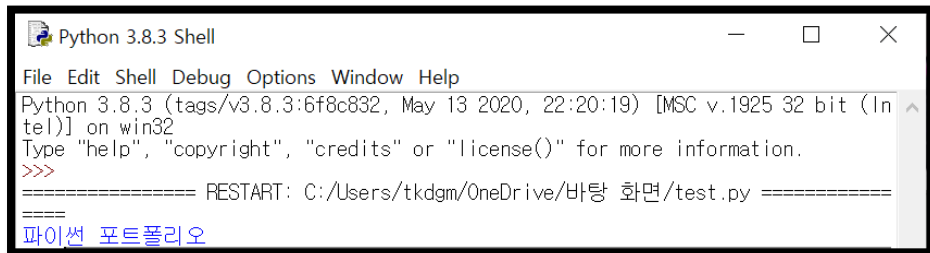
문자열은 어떻게 만들고 사용할까?

1. 큰따옴표(")로 양쪽 둘러싸기	"Life is too short"
2. 작은따옴표(')로 양쪽 둘러싸기	'Life is too short'
3. 큰따옴표 3 개로 양쪽 둘러싸기	"""Life is too short"""
4. 작은따옴표 3 개로 양쪽 둘러싸기	'''Life is too short'''

문자열을 출력하고 싶으면 print()메소드를 사용하면 된다.



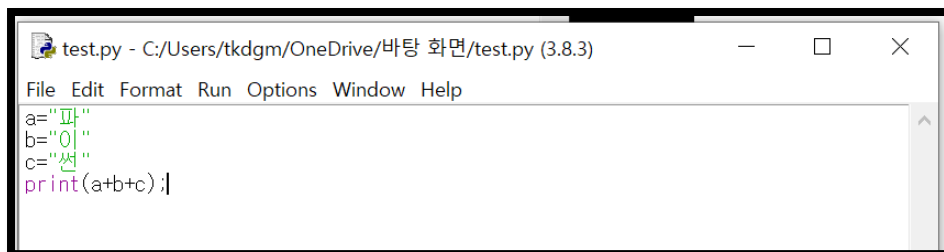
```
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
print('파이썬 포트폴리오')
```



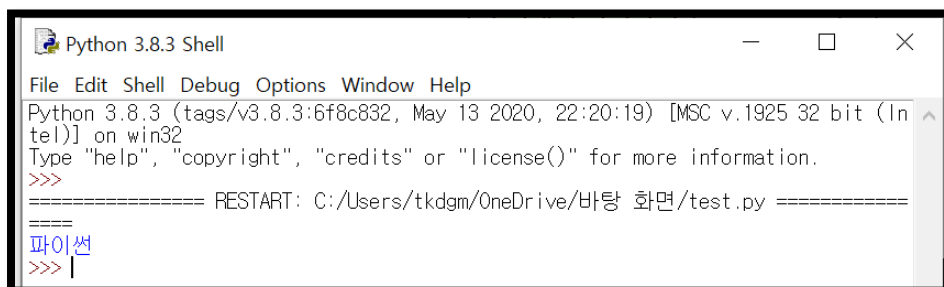
```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
=====
파이썬 포트폴리오
>>>
```

## 문자열(String) 연산자

+: 문자열의 연결 연산자

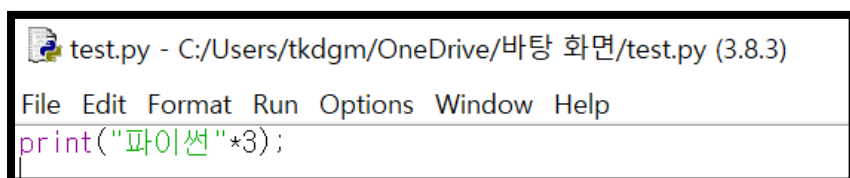


```
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
a='파'
b='이'
c='션'
print(a+b+c);
```

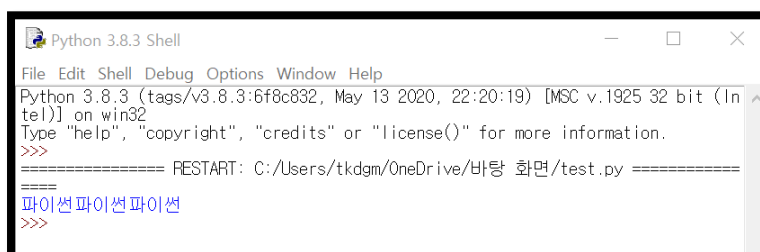


```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
=====
파이썬
>>> |
```

\* : 문자열 반복 연산자



```
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
print('파이썬'*3);
```



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
=====
파이썬파이썬파이썬
>>>
```

## 주석

실행되는 코드가 아니라 설명하고 기억하기 위한 것이다.

한 줄 주석 - #

여러 줄 주석 - """ / '''

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
>>> 주석이 아닙니다
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
#이것은 주석입니다
'''이것도 주석입니다
이것도 주석입니다
이것도 주석입니다
이것도 주석입니다'''
print("주석이 아닙니다")
```

## 정수와 실수

숫자는 간단히 정수(integer)와 실수(real or float)로 나뉜다.

15,7,20 등은 정수, 소수점이 있는 3.14 등은 실수이다.

➤ 실수 표현 방법

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
>>> 27834.0
>>> 0.00027834
>>> 278340.0
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
print(2.7834e4)
print(2.7834e-4) #2.7834 × 10의 -4승
print(2.7834E5) #2.7934 × 10의 -5승
```

## 산술연산자 우선순위

연산자	명칭	의미	우선순위
*	곱하기	두 피연산자 곱하기	3
/	나누기	왼쪽을 오른쪽 피연산자로 나누기	3
%	나머지	왼쪽을 오른쪽 피연산자로 나눈 나머지	3
//	몫 나누기	왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수	3
**	거듭제곱, 지수승	왼쪽을 오른쪽 피연산자로 거듭제곱	1

## B.파이썬 프로그래밍을 위한 기초 다지기 – 변수와 키워드, 대입 연산자

### 파이썬 자료형 Data Type

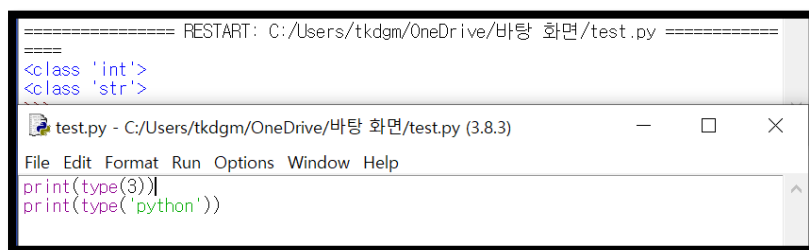
데이터의 타입, 데이터의 종류를 말한다.

종류	데이터 타입
문자열 타입	Str
숫자 타입	Int, float, complex
시퀀스 타입	List, tuple, range
매핑 타입	Dict
세트 타입	Set, frozenset
불린 타입	Bool
바이너리 타입	Bytes, bytearray, memoryview

### 데이터 타입 확인

데이터 타입을 확인하기 위해서는 type()함수를 사용한다.

변수를 이 함수의 인자로 주면 데이터 타입을 반환한다.



```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
>>>
<class 'int'>
<class 'str'>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
print(type(3))
print(type('python'))
```

### 변수 값 저장

변수란 '변하는 자료를 저장하는 메모리 공간'이다.

A = 10 처럼 변수명 = 값의 형식으로 저장한다.

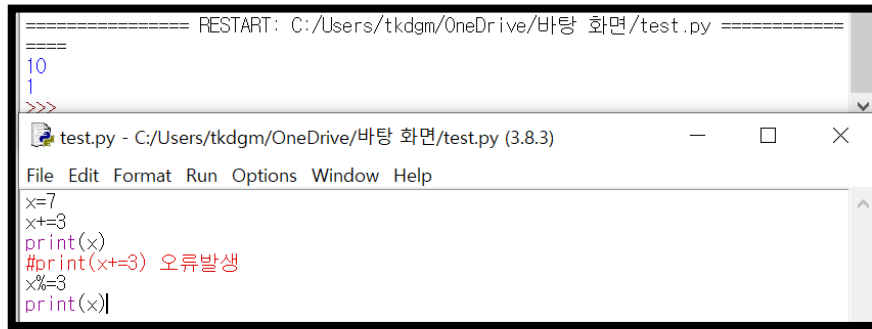
### 변수명 생성규칙

- 영문 문자와 숫자 사용 가능
- 대소문자 구분
- 한글도 지원하지만 권장 X
- 숫자부터 시작할 수 X
- 특수문자 사용 불가능
- 변수명 띄어쓰기 불가능
- 파이썬의 내장 키워드 사용 불가능 (if, for, while, or, and, true 등)

## 다양한 대입 연산자

연산자	축약형	기본형
+=	X+=7	X=X+7
-=	X-=7	X=X-7
*=	X*=7	X=X*7
/=	X/=7	X=X/7

### ➤ 연산자예제



```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====  
10  
1  
>>>  
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)  
File Edit Format Run Options Window Help  
x=7  
x+=3  
print(x)  
#print(x+=3) 오류발생  
x%=3  
print(x)
```

## B.파이썬 프로그래밍을 위한 기초 다지기 – 자료의 표준 입력과 자료 변환 함수

### 함수란?

함수의 기본 3 가지: 함수명, 인자 값, 반환 값

$$f(x)=y$$

중학교 때 배웠던 수학과 비슷하다. x 는인자값, y 는 반환값이다.

### 함수를 사용하는 방법

- 1.만들어진 함수를 사용하는 방법
- 2.직접 함수를 정의하는 방법

### 파이썬 표준 입력 함수 input()

input()은 파이썬에서 미리 정의해놓아 바로 사용할 수 있는 내장함수다.

입력받은 표준 입력을 문자열로 읽어 반환하는 함수이며,

입력받은 값을 '='을 통해 변수에 할당하여 사용한다.



```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
아무거나 입력하시오 툴툴툴
툴툴툴

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
a = input('아무거나 입력하시오')
print(a)
```

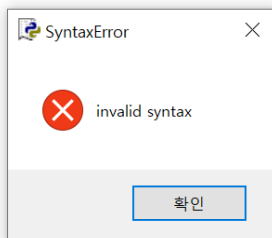
## 변환함수 *str(),int(),float()*

각각 문자열,정수,실수형으로 변환 가능하다.

문자열이 아닌 객체를 print()로 호출할 때, 파이썬은 디폴트값으로 str()함수를 사용한다.

변수나 상수를 문자열 안에 삽입하는 문자열 보간시에도 내부적으로 사용한다.

```
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/
File Edit Format Run Options Window Help
(int)'python'
```



int()변환시 문자열이 정수나 실수 형태가 아니면 ValueError 가 발생한다.

## 변환함수 *bin(),oct(),hex()*

10 진수를 바로 16 진수, 8 진수, 2 진수로 표현되는 문자열로 변환하려면

각각 함수 bin(), oct(), hex()를 사용해야 한다.

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
정수 입력 >>16
2진수 : 0b10000
8진수 : 0o20
16진수 : 0x10
10진수 : 16

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
data = int(input('정수 입력 >>'))

print('2진수 : ',bin(data))
print('8진수 : ',oct(data))
print('16진수 : ',hex(data))
print('10진수 : ',data)
```

## int(정수,진법기수)

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
6
3
8
26

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
#int('strnum',n) 에서 n은 변환하려는 문자열 진수의 숫자
print(int('6',10)) #모두 10진수로 변환
print(int('11',2)) #2진수 문자열을 2진수로 변환
print(int('10',8)) #8진수 문자열을 10진수로 변환
print(int('1A',16)) #16진수 문자열을 10진수로 변환
```

## C.문자열과 논리연산 – 문자열 다루기

### str 클래스

문자열은 하나 이상의 문자들로 구성된 문자들의 집합을 의미한다.

파이썬의 문자열은 str 클래스로 구현되어 있다.

str 자료형은 한글을 포함한 유니코드로 포함되는 전 세계의 모든 문자를 표현할 수 있으며, 변경불가능 자료형이다.

### 함수 len()

함수 len(문자열)으로 문자열의 길이를 알 수 있다.

### 문자열 인덱싱 & 슬라이싱

- 1)배열의 시작은 0 부터
- 2)범위를 지정할 경우 마지막 숫자 바로 앞까지만 자른다
- 3)음수를 사용하여 뒤에서부터 자를 수 있다

S[i] i 번째 항

S[i : j] i 번째 항 이상 j 번째 미만

S[i : j : k] i 번째 항 이상 j 번째 미만 k 만큼 건너뛰면서

S[-i] -index

S[ : ] 처음부터 끝까지

S[start : ] 처음부터

S[ : end] 끝까지

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
====
i like
y
n
t
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
str = 'i like python'
print(str[0:6])
print(str[8])
print(str[len(str)-1])
print(str[-4]) #역참조
```

## ord()와 chr()

파이썬에서 ord()함수는 문자의 아스키코드 값을 얻는 함수이다.

파이썬에서 chr()함수는 아스키코드 값을, 아스키 코드 값에 해당하는 문자로 반환하는 함수이다.

```
===== RESTART: C:/Users/tkdgm/OneDrive/바탕 화면/test.py =====
====
65
A
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
print(ord('A'))
print(chr(65))
```

## ord()와 chr()를 이용한 암호화

```
암호를 입력하시오 : good
입력한 암호는 [good] 입니다. 암호화를 진행합니다.
암호화된 비밀번호 : iqaf
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
# 암호화

pw = input('암호를 입력하시오 : ')
pw_s = "" #암호화된 비밀번호 저장
pw2 = "" #복호화된 암호, 첫번째 비밀번호랑 같은지 확인
print("입력한 암호는 [%s] 입니다. 암호화를 진행합니다." %pw)

c='' #아무문자
ac = 0 #아스키 코드로 변환
for i in range(len(pw)):
    c = pw[i]
    ac = ord(c) #아무문자의 아스키 코드
    ac += 2 #코드값 2 증가
    c = chr(ac) #아스키 코드를 문자로 변환
    pw_s += c #암호화된 암호 저장

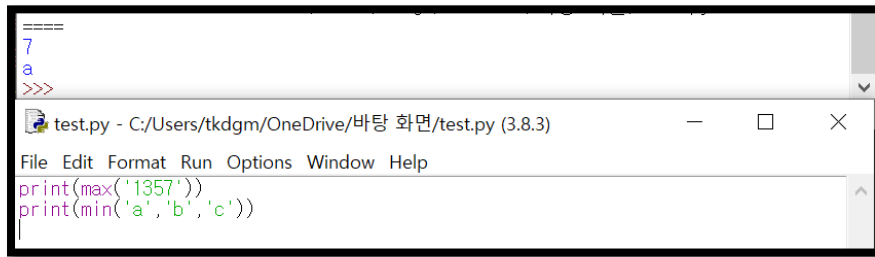
print("암호화된 비밀번호 : %s" %pw_s)
```

## 문자열의 최대 최소

내장 함수 min()과 max()는 인자의 최댓값과 최솟값을 반환하는 함수다.

인자가 1 개이면 문자열 내에서 최대와 최소인 문자를 반환하고,

인자가 2 개 이상이면 문자열 중 최대와 최소인 문자열을 반환한다.

A screenshot of a Python 3.8.3 interpreter window. The title bar reads 'test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor contains the following lines: 

```
====  
7  
a  
>>>  
print(max('1357'))  
print(min('a','b','c'))
```

### C. 문자열과 논리연산 – 문자열 관련 메소드

#### str.replace('문자열 1','문자열 2')

#전체 문자열 중 포함된 문자열 1 을 문자열 2 로 변경. 문자일부도 변경 가능

#문자열 1 이 존재하지 않을 때는 전체 문자열 그대로 리턴

#### str.count('문자')

# str 문자열에 포함된 문자의 개수를 리턴. 없을 때는 0 리턴

#### '삽입할 문자열'.join(str)

#인자 문자열에 삽입할 문자열이 중간중간 삽입됨

#### str.find('문자')

#문자열 내에서 문자의 포함 위치를 인덱스 번호로 리턴. 맨 처음 값으로 리턴. 없으면 -1 리턴

#### str.index('문자')

#find()와 동일한 리턴. 없으면 ValueError

#### str.rfind('문자')

#문자열 내에서 문자의 포함 위치를 뒤에서부터 찾아 인덱스 번호로 리턴. 값이 없으면 -1 리턴

#### str.rindex('문자')

#rfind()와 동일한 값 리턴. 없으면 ValueError

#### str.split('문자열')

#문자열 기준으로 str 을 분리하고 리스트로 반환. 문자열에는 공백이 가능하며

아무것도 없을 시 공백으로 인식. 공백에는 줄바꿈도 포함

#### str.capitalize()

#단어 (혹은 문장의 경우 첫글자) 의 첫 글자만을 대문자로

**str.title()**

#단어마다 첫 문자를 대문자로 변환해 반환

**str.upper()/lower()**

#모두 대문자/모두 소문자로 변환해 반환

**str.swapcase()**

#소문자와 대문자를 서로 변환하여 반환

**str.center(n)**

#숫자만큼의 칸에 가운데 정렬. center(숫자,문자) => 공백을 문자로 채움

**str.ljust()/rjust()**

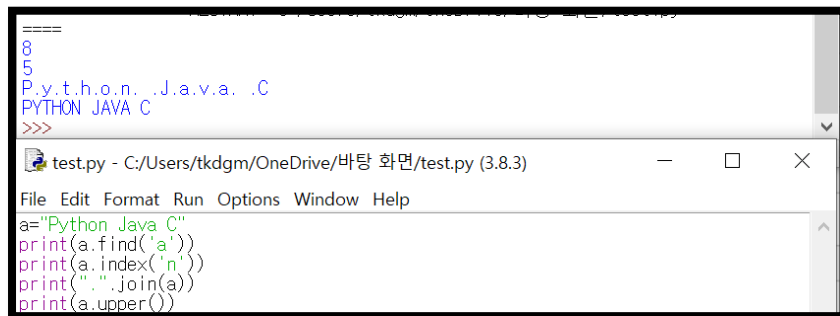
#문자열의 폭을 지정하고 왼쪽에 붙여 정렬/오른쪽에 붙여 정렬

**str.strip()**

#문자열 str 에서 맨 앞뒤의 공백을 제거해 반환,괄호안에 문자를 쓰면 그 문자 제거

**str.zfill(n)**

#문자열의 지정한 폭 앞 빈 부분에 0 을 채워넣고 싶다면 zfill(폭)



```
=====  
8  
5  
P.y.t.h.o.n. .J.a.v.a. .C  
PYTHON JAVA C  
>>>  
test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)  
File Edit Format Run Options Window Help  
a="Python Java C"  
print(a.find('a'))  
print(a.index('n'))  
print(".".join(a))  
print(a.upper())
```

## 출력을 정형화하는 함수 *format()*

포매팅 = 문자열 중간중간 변수나 상수를 함께 출력하는 방법

**%d** -> 정수형

**%f** -> 실수형

**%s** -> 문자열

```
=====
3 + 4 = 7
>>>

test.py - C:/Users/tkdgm/OneDrive/바탕 화면/test.py (3.8.3)
File Edit Format Run Options Window Help
str = '{} + {} = {}'.format(3,4,3+4)
print(str)
```

문자열 '{} + {} = {}' 내부에서 중괄호 부분에 format 인자인 3,4,3+4 가 순서대로 출력된다.

인자는 상수 뿐만 아니라 변수도 가능하다.

## *{n:md}로 정수를 형식 유형으로 출력 처리*

{인자의 순번:출력 형식}

```
=====
10 / 3 = 3.333333
>>> |

Ln: 80 Col: 4

File Edit Format Run Options Window Help
a, b=10, 3
print('{0:d} / {1:d} = {2:f}'.format(a,b,a/b))
#d는 정수 , f는 부동소수 표기
```

## *{n:mf}로 실수를 형식 유형 출력 처리*

형식유형 f와 F 는 모두 실수를 소수점 형태로 출력한다.

기본적으로 소수점 이하 여섯 자리이다.

```
=====
3.141500      3.142
inf           INF
nan           NAN
>>> |

Ln: 85 Col: 4

File Edit Format Run Options Window Help
print('{0:0f} {0:10.3F}'.format(3.1415))
print('{0:0f} {0:10.3F}'.format(3.5E1000))
print('{0:0f} {0:10.3F}'.format(float('nan')))
```

## *문자열의 다양한 형식 유형*

d,n	10 진수 정수, n 은 국가에 맞는 구분자 추가
c	유니코드 수에 대응하는 문자 출력
f,F	기본적으로 소수점 여섯 자리까지 실수로 출력하며, F 인 경우는 'inf','nan'표시를 대문자 'INF','NAN'로 표시
b	2 진수 정수
o	8 진수 정수
x,X	16 진수 표현으로 a~f 까지 소문자와 대문자로 각각 표시
e,E	지수 형식 3.141592e+00 으로 지수 표현이 각각 소문자 e 와 대문자 E
g,G	실수를 일반적으로는 소수점 형식으로 출력하지만 커지면 지수승으로 표시
%	퍼센트 형식, 인자의 100 배를 소수점으로 출력하고 맨 뒤에 % 출력
s	문자열 형식이며 기본적으로 왼쪽 정렬, 그러나 수 형식은 모두 기본이 오른쪽 정렬

➤ 예제소스

```
====
30 - 14 = 12
      2.72
      p
99%
>>> |

Ln: 91 Col: 4

File Edit Format Run Options Window Help
print('%d - %x = %o' %(30,20,30-20))
print('%10.2f' %2.718281)
print('%10c' %'p')
print('%d%%' % 99)
```

C.문자열과 논리연산 – 논리 자료와 다양한 연산

논리유형 bool 과 함수 bool()

파이썬에는 ==(같다), !=(다르다), >(크다), <(작다), >=(크거나 같다), <=(작거나 같다) 관계연산자가 있다. 관계연산자는 불 값(True, False)을 리턴한다. 결과가 참이면 True, 거짓이면 False 를 리턴한다.

bool()함수를 이용하면 True,False 여부를 알 수 있다. None,정수 0,실수 0.0,빈 문자열(공백은 True),빈 리스트,빈 집합 등은 False 이다.

```

>>> bool(0)
False
>>> bool(0.0)
False
>>> bool('')
False
>>> bool(' ')
True
>>> bool([])
False
>>> bool({})
False
>>> |

```

Ln: 103 Col: 4

## 파이썬 논리연산자

파이썬 논리연산자에는 and, or, not 이 있다. **and 연산(&)**은 둘 다 True 일 때 True 를 리턴한다.

**or()** 연산은 둘다 False 일 때 False 를 리턴한다.(둘 중 하나만 참이면 참)

**not(^)** 연산은 True 는 False 로 False 는 True 로 변환한다.(거꾸로)

```

>>> not True
False
>>> True and False
False
>>> True or False
True
>>> |

```

Ln: 118 Col: 4

## 파이썬 논리연산자 우선순위

not, and, or 순이다.

```

>>> not False and True
True
>>> (not False) and True
True
>>> not True or True and False
False
>>> (not True) or (True and False)

```

## 논리 값 True 와 False 의 산술 연산

True 와 False 를 각각 1 과 0 으로 산술 연산에 활용할 수 있다.



## 멤버십 연산 in

파이썬 키워드인 in 은 멤버의 소속을 알 수 있는 멤버십 연산이다.

연산자	예제	설명
in	1 in(1,2,3)	왼쪽 피연산자의 값이 오른쪽 피연산자 멤버 중 일치하는 값이 존재하면 True
not in	1 not in(1,2,3)	왼쪽 피연산자의 값이 오른쪽 피연산자 멤버 중 일치하는 값이 존재하지 않으면 True

## 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^

연산자	예제	설명
&	10&5	두 피연산자의 and 비트연산을 수행함
	10 5	두 피연산자의 or 비트연산을 수행함
^	10^5	두 피연산자의 xor 비트연산을 수행함
<<	10<<2	왼쪽 피연산자의 비트를 왼쪽으로 2 개 비트이동
>>	10>>2	왼쪽 피연산자의 비트를 오른쪽으로 2 개 비트이동

```
>>> bin(0b10101010 & 0b11000011) #둘다1이면1, 아니면0
'0b10000010'
>>> bin(0b10101010 | 0b11000011) #둘중하나만 1이면 1
'0b11101011'
>>> bin(0b10101010 ^ 0b11000011) #다르면1, 같으면 0
'0b1101001'
>>> bin(~0b10101010) #not 연산, 1은 0 0은 1로 변환
'-0b10101011'
```

(비트이동이 눈에 띄게 bin 사용)

기본적으로 비트 연산은 0, 1 로 이루어진 2 진수를 연산하는 방식이다.

파이썬에서는 0b 를 사용하여 2 진수를 입력한다.(0o => 8 진수 0x =>16 진수)

bin()함수는 정수를 이진수 문자열로 변환한다.

```
>>> bin(0b10101010 << 2) #왼쪽비트이동: 2를 곱하는 효과
'0b1010101000'
>>> 0b10101010 << 2
680
>>> 0b10101010
170
>>> bin(0b10101010 >> 2) #오른쪽 비트이동 : 2로 나누는 효과
'0b101010'
>>> 0b10101010 >> 2
42
>>> 0b10101010
170
```

## D.조건과 반복 – if...else

**if 조건부분:**                   #조건부분 => 결과값으로 불린(true,false)값이 나오는 식

                  수행부분               #조건부분이 true 일 때 실행하고 싶은 명령들

**else:**                               #수행부분을 다른 코드와 비교하기 위해 들여쓰기를 해주어야 함

                  수행부분               #조건부분이 false 일 때 실행하고 싶은 명령들

<b>if</b> 온도가 5 도 이하다 :	<b>if</b> 온도가 5 도 이하다 :
<b>else :</b>	<b>elif</b> 온도가 10 도 이하다 :
<b>if</b> 온도가 10 도 이하다 :	<b>elif</b> 온도가 15 도 이하다 :
<b>else :</b>	
<b>if</b> 온도가 15 도 이하다 :	
<b>else :</b>	

# else 가 많아져야 할 때는 elif 을 사용하면 코드가 간결해진다.if... elif 는 다중 택일 결정 구조이다.

# if 문에서 논리 표현식 이후에는 반드시 콜론(:)이 있어야 한다.

```
f.py =====
1학기 평균 평점은?4.5
축하합니다! 장학금 지급 대상자이다.
당신의 1학기 평균 평점 4.50이다.

04-02gradeif.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\파이썬예제...
File Edit Format Run Options Window Help
grade = float(input('1학기 평균 평점은?'))
if 3.8 <= grade:
    print('축하합니다! 장학금 지급 대상자이다.')
    print('당신의 1학기 평균 평점 %.2f이다.' %(grade))
```

if 사용 예제

```
q.py =====
1.1 1.1
2.5 3.6
3.6 7.2
4.2 11.4
5.4 16.8
합:16.80, 평균:3.36
```

```
04-07numseq.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\파이썬예...
File Edit Format Run Options Window Help
sum = 0
for i in 1.1,2.5,3.6,4.2,5.4:
    sum += i
    print(i,sum)
else:
    print('합:%.2f,평균:%.2f' %(sum,sum/5))
```

else 사용 예제

```
cast.py ===
미세먼지(10마이크로그램)의 농도는?88
미세먼지 농도: 88.000000, 등급: 나쁨
```

```
04-05dustforecast.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\파이...
File Edit Format Run Options Window Help
PM = float(input('미세먼지(10마이크로그램)의 농도는?'))
if 151 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '매우 나쁨'))
elif 81 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '나쁨'))
elif 31 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '보통'))
else:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '좋음'))
```

elif 사용 예제

## D.조건과 반복 – for / while

반복문이란 연산 혹은 어떠한 기능을 반복적으로 실행하기 위해 사용하는 구문이다. 대표적으로는 for 과 while 문이 있다. for 문의 경우 반복되는 부분이나 범위를 코드 작성자가 구체적으로 지정하는 반면, while 문의 경우 참과 거짓을 기준으로 조건이 거짓이 되기 전까지 무한 반복을 실행한다.

### for 문의 기본 구조

**for** 변수 **in** 리스트(또는 튜플, 문자열) :

수행할 문장 1

수행할 문장 2

.....

혹은

**for** 반복횟수 **in** 시퀀스(반복 절차) :

statements(s)

```
>>> for i in ["a","b","c"]:
    print(i)

a
b
c
```

for 문의 핵심은 순회할 수 있는 (iterable) 자료를 사용한다는 것이다.

순회 가능 자료 : 문자열, 리스트, 딕셔너리, range() 함수 등...

정수형/실수형의 경우 순회할 수 없다. 단 문자열의 경우 문자 순서에 따라 index 번호가 부여 가능하므로 사용 가능하다 ( index 가 있으면 순회 가능)

단 딕셔너리의 경우 변수를 하나로 설정하여 해당 변수를 출력하면 'key'값만 가져온다.

따라서 값만 필요할 경우 'for l in a.values()'의 형태로 작성한다.

키,값 모두 변수로써 필요하다면 'for l,k in a.items()'의 형태로 작성한다.

## *range()*

범위를 지정해주는 함수로, 위의 예제에서처럼 a~z까지의 문자

혹은 1-1000까지의 숫자를 사용해야 한다면 그것들을 모두 입력하기엔 시간과 수고가 든다.

그럴 때 사용할 수 있는 함수가 range()이다.

```
>>> for i in range(5) :
        print(i, end=' ')

0 1 2 3 4
>>> |
```

내장 함수 range()의 인자가 range(5)처럼 한 개이면 반복이 5회 수행된다.

시퀀스의 항목은 1부터가 아니라 0부터이므로 마지막도 5가 아닌 4이다.

```
>>> for i in range(1,6) :
        print(i, end=' ')

1 2 3 4 5
>>> |
```

range(1,6)은 시작 정수인 1에서 5까지 5개의 정수를 반복할 수 있다.

range(10,0)은 동작하지 않는다.

range는 숫자가 증가하는 기본 값이 양수 1이기 때문에 감소시키고 싶다면

증가폭을 음수로 지정해야 한다. 음수로 지정하는 방법 말고도 reversed를 사용하면

숫자의 순서를 반대로 뒤집을 수 있다.

**for** 변수 **in** reversed(range(횟수))

**for** 변수 **in** reversed(range(시작,끝))

**for** 변수 **in** reversed(range(시작,끝,증가폭))

밑은 reversed와 음수 증가폭을 인자에 넣은 예제이다.

```
>>> for i in reversed(range(1,10)):
    print(i)
```

9  
8  
7  
6  
5  
4  
3  
2  
1

```
>>> for i in range(9,0,-1):
    print(i)
```

9  
8  
7  
6  
5  
4  
3  
2  
1

for 은 리스트, 튜플, 문자열 등 시퀀스 객체로 반복할 수 있다. 그것들이 무엇인지는 뒤에서 설명한다.

for 에 리스트를 넣으면 리스트의 요소를 꺼내면서 반복한다.

### 리스트 요소 출력

list = ['kim','sang','hee']                    #리스트, 딕셔너리, 튜플은 뒤에서 설명

for l in list :                                #리스트의 마지막 요소까지 출력

    print(i)

### 딕셔너리 출력

dic1 = {'kim':11,'sang':12,'hee':13}

for key in dic1:                            #key 값에 'kim','sang','hee'가 순차적으로 들어감

    print(key ,'=>',dic1[key])

### 리스트 요소가 가리키는 값이 여러 개인 경우

list = [('kim',1),('sang',2),('hee',3)]#리스트 안에 튜플이 있음. list[0] = ('kim',1)

for(name,nnum) in list:                    #list 의 요소가 튜플이기 때문에 각각 요소가 name,nnum 에 대입됨

    print(name, ':' ,nnum)

## while 문 기본구조

for 문은 정해진 횟수를 반복하는 기능이고 while 은 조건이 참인 동안 반복하는 기능이다.

while 뒤에 true 나 임의의 값을 넣어 무한반복 시킬 수도 있다.

**while** 논리 표현식 : #반드시 콜론이 필요하다

문장 1            #논리 표현식이 True 일 때 실행되는 반복 단위이다.

문장 2

**else :**

문장 3            #논리 표현식이 False 가 되어 반복이 종료된 마지막에 실행된다.

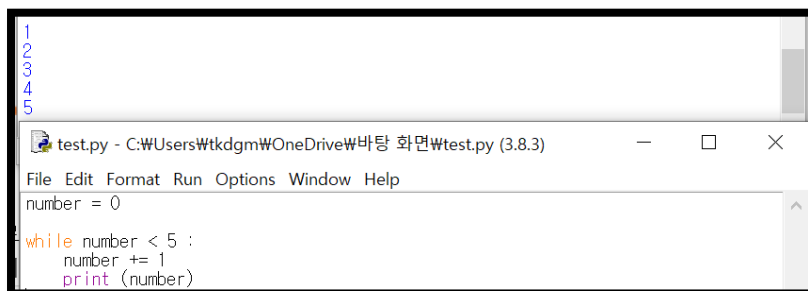
```
>>> while True :
    print("무한반복")
```



## while 의 조건 반복 기능

while 뒤에 비교연산 등을 이용하여 조건이 참인 동안 반복할 수 있다.

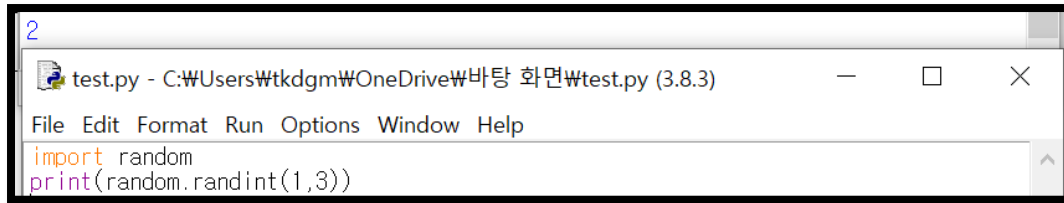
```
1
2
3
4
5
```



## D.조건과 반복 – break/continue

### 임의의 수를 발생하는 난수

파이썬에서는 모듈 random 의 함수 randint(시작,끝)을 사용해 정수 시작과 끝 수 사이에서 임의의 정수를 얻을 수 있다. 여기서는 시작과 끝을 모두 포함한다.즉, random.randint(1,3)는 import random 으로 모듈 활용을 선언한 후 1,2,3 중 한 가지 수를 임의로 얻을 수 있다.



```
2
test.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\test.py (3.8.3)
File Edit Format Run Options Window Help
import random
print(random.randint(1,3))
```

## 반복을 종료하는 break 문

앞에서 배운 for 문은 range() 함수에서 지정한 범위를 벗어나면 종료한다. while 문은 조건식이 거짓이 되면 종료하거나 무한반복해 Ctrl + C 를 누르면 종료한다. 하지만 계속되는 반복을 논리적으로 빠져나가는 방법도 있는데, 바로 break 문이다.

### for 에서의 break

**for** 변수 **in** 시퀀스 :

문장 1

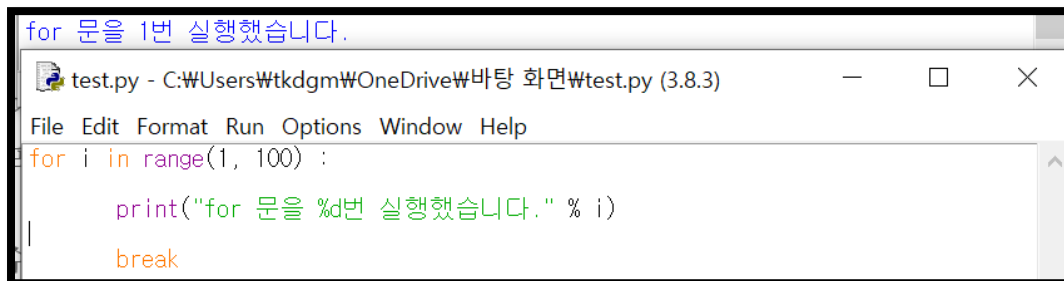
**if** 조건:

**break**

...문장 n

**else** :

#반복이 정상적으로 종료되면 실행되는 문장들



```
for 문을 1번 실행했습니다.
test.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\test.py (3.8.3)
File Edit Format Run Options Window Help
for i in range(1, 100) :
    print("for 문을 %d번 실행했습니다." % i)
    break
```

이 예제는 for 문이 99 번 실행되도록 했지만 break 문을 만나 한번만 실행하고 종료한다.

### while 에서의 break

**while** 논리 표현식 :

문장 1

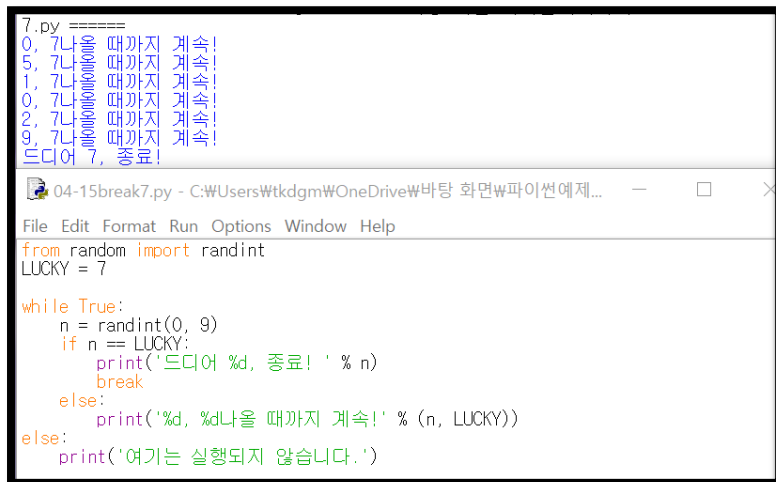
**if** 조건 :

**break**

...문장 n

**else :**

#반복이 정상적으로 종료되면 실행되는 문장들



```
7.py =====
0, 7나올 때까지 계속!
5, 7나올 때까지 계속!
1, 7나올 때까지 계속!
0, 7나올 때까지 계속!
2, 7나올 때까지 계속!
9, 7나올 때까지 계속!
드디어 7, 종료!

04-15break7.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\파이썬예제...
File Edit Format Run Options Window Help
from random import randint
LUCKY = 7

while True:
    n = randint(0, 9)
    if n == LUCKY:
        print('드디어 %d, 종료!' % n)
        break
    else:
        print('%d, %d나올 때까지 계속!' % (n, LUCKY))
else:
    print('여기는 실행되지 않습니다.')
```

반복에서 break 문에 의해 종료되면 반복 while 이나 for 의 else:블록은 실행되지 않는다.

### 반복문으로 다시 돌아가게 하는 *continue* 문

continue 문을 만나면 블록의 남은 부분을 무조건 건너뛰고 반복문의 처음으로 돌아간다. 즉 continue 문을 만나면 반복문을 처음부터 다시 실행한다.

#### *for* 에서의 *continue*

**for** 변수 **in** 시퀀스 :

...

**continue**

...

#### *while* 에서의 *continue*

**while** 논리 표현식 :

...

**continue**

...

다음은 1 ~ 100 의 합계를 구하되 3 의 배수를 제외하고 더하는 프로그램이다.



```
1~100의 합계 (3의 배수 제외) : 3367
test.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\test.py (3.8.3)
File Edit Format Run Options Window Help
a,b=0,0
for b in range(1,101):
    if b%3 ==0:
        continue
    a += b
print("1~100의 합계 (3의 배수 제외) : %d" %a)
```

b가 3의 배수일 때 continue 문으로 a에 b를 더하지 않고 반복문으로 돌아간다.

## E.리스트와 튜플 - 리스트란?

리스트는 항목의 나열인 시퀀스다. 즉 리스트는 콤마로 구분된 항목(원소)들의 리스트로 표현되며, 각 항목은 모두 같은 자료형일 필요는 없다. 즉 정수, 실수, 문자열, 리스트 등이 모두 가능하다. 항목 순서는 의미가 있으며 항목 자료 값은 중복되어도 상관없다.

### 리스트 생성

기본적인 리스트 만들기

a=[1,2,3]      # [ ] 사이에 데이터를 열거하여 리스트 생성

b= list()      # list()로 빈 리스트 생성

b= list((1,2,3))    # list() 안에 ( ) 사이에 데이터를 열거하여 리스트 생성

c= [0,0,0,0,0]    # 0이라는 값으로 초기화된 리스트 생성

d= [ ]      # 이 방법으로도 빈 리스트 생성 가능

```
[1, 2, 3]
[1, 2, 3]
[0, 0, 0, 0, 0]
test.py - C:\Users\Wtkdgm\OneDrive\바탕 화면\test.py (3.8.3)
File Edit Format Run Options Window Help
a=[1,2,3]
b=list()
b=list((1,2,3))
c=[0,0,0,0,0]
print(a)
print(b)
print(c)
```

### 리스트 표현식 - 반복문

[값 for 변수 in range(반복범위)]

➤ 다음과 같은 형태를 한 줄로 표현한 것

for 변수 in range(반복범위):  
    [값]

➤ 반복횟수만큼 값이 리스트의 항목으로 추가되어 리스트가 생성됨

```
a = [0 for _ in range(5)]
print (a)
```

Ln: 3 Col: 0

```
[0, 0, 0, 0, 0]
>>>
```

### range 값을 값으로 갖는 리스트 생성

```
a = [i for i in range(10)]
print (a)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
...
```

반복문을 수행하면서 바뀌는 i 값으로 리스트의 값이 초기화됨. 순차적으로 리스트 값 추가 가능

```
c=[ ]
for i in range(10):
    c.append(i)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
...
```

append()메소드를 사용하여 표현 가능

```
a = [(i+10) for i in range(0,20,3)]
print (a)
```

```
[10, 13, 16, 19, 22, 25, 28]
```

range()에 (start,stop,step)적용하여 리스트 생성 가능

```
a = list(x for x in range(5))
print(a)
```

```
[0, 1, 2, 3, 4]
```

list()의 인수로 표현식 사용하여 리스트 생성 가능

### 리스트 표현식 - 조건문

[변수 for 변수 in range(반복범위)if 조건식]

list(변수 for 변수 in range(반복범위)if 조건식)

- 다음과 같은 형태를 한 줄로 표현한 것

```
for 변수 in range(반복범위):
    if 조건식 :
        [값]
```

- 반복 횟수 안에서 조건식이 참일 때 변수가 리스트의 항목으로 추가되어 리스트가 생성됨

다음은 1 부터 100 까지 5 의 배수로만 이루어진 배열을 생성하여 출력하는 예제이다.

```
a = [i for i in range(1,101) if i%5 == 0 ]
#표현식에 반복문과 조건문을 모두 적용하여 리스트 생성

b = [i for i in range(5,101,5)]
#range()의 단계값을 적용하여 리스트 생성

c = list(i for i in range(1,101) if i%5 == 0)
#list() 함수 이용

print(a)
print(b)
print(c)
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
```

## 리스트의 항목 참조

리스트에서 첨자(index)를 사용해 항목 하나하나를 참조할 수 있다. 첨자는 정수여야 하며, 문자열에서 배운 첨자 방식과 동일하다.

- ✓ 첨자는 첫 요소가 0 부터 시작하며, 순차적으로 1 씩 증가한다.
- ✓ 마지막 요소의 첨자는 역순으로 -1 부터 시작하며, 반대로 1 씩 감소한다.
- ✓ 그러므로 첨자의 범위는 0 에서 [len(시퀀스)-1]까지, -1 에서 [-len(시퀀스)]까지 가능하다.

0	1	2	3	4	5	오름차순 첨자
p	y	t	h	o	n	
-6	-5	-4	-3	-2	-1	내림차순 첨자

리스트 항목을 참조하는 예제

```
pl = ['C', 'C++', 'Python', 'Java']
print(pl[0])
print(pl[2])
print()

for i in range(len(pl)):
    print(pl[i])
```

```
C
C++
Python
Java
...
```

## 중첩된 리스트 (2 차원 리스트)

2 차원 리스트란?

-행과 열이 있는 테이블 구조.

-리스트의 항목으로 리스트가 있는 구조[ ][ ][ ]

-리스트 안에 리스트 뿐 아니라 튜플을 넣어서 구성할 수도 있음.

행 0	[0],[0]	[0],[1]	[0],[2]	[0],[3]	[0],[4]
행 1	[1],[0]	[1],[1]	[1],[2]	[1],[3]	[1],[4]
행 2	[2],[0]	[2],[1]	[2],[2]	[2],[3]	[2],[4]
행 3	[3],[0]	[3],[1]	[3],[2]	[3],[3]	[3],[4]

## 2차원 리스트 만들기

n0 = [1,2,3]    #1 차원 리스트는 항목으로 리스트를 갖지 않음

nn = [[1,2,3],[4,5,6]]    #리스트의 항목을 리스트로 갖는 리스트가 2차원 리스트

```
nn = [[1,2,3], [4,5,6], [7,8,9]]
print(nn[0])
print(nn[0][1])
```

[1, 2, 3]  
2

## 2차원 리스트 출력

인덱스를 이용한 출력

```
a = [[1,2,3,4], [5,6,7,8], [9,10,11,12]]
for i in range(3):
    for j in range(4):
        print(i+1,j+1, a[i][j])
```

1 1 1  
1 2 2  
1 3 3  
1 4 4  
2 1 5  
2 2 6  
2 3 7  
2 4 8  
3 1 9  
3 2 10  
3 3 11  
3 4 12

리스트를 반복 범위에 그대로 적용하여 출력

```
a = [[1,2,3,4], [5,6,7,8], [9,10,11,12]]
for n in a:
    print(n)
```

[1, 2, 3, 4]  
[5, 6, 7, 8]  
[9, 10, 11, 12]

## E.리스트와 튜플 - 리스트의 부분 참조와 항목의 삽입과 삭제

### 첨자 3개로 리스트 일부분을 참조하는 슬라이싱

리스트[start:stop:step]

: 첨자 start 에서 첨자 stop-1 까지 step 간격으로 부분 리스트 반환

- ✓ 0 에서 시작하는 오름차순
- ✓ 마지막 요소 -1 에서 시작하는 내림차순 첨자도 가능
- ✓ 다소 복잡하지만 두 순차의 첨자를 함께 사용 가능

### 리스트의 슬라이스로 리스트 일부 수정

```
li = ['수', '정', '완', '료']  
li[0:2] = ['출력']  
print(li)  
['출력', '완', '료']
```

### 리스트 슬라이스에 슬라이스 대입

```
li = ['가', '다', '나', '야']  
li[0:1] = li[2:3]  
print(li)  
['나', '다', '나', '야']
```

### 리스트 항목 삽입

리스트 첨자 위치에 항목을 삽입하려면 **리스트.insert(첨자, 항목)**을 이용한다.

삽입되는 항목은 무엇이든 가능하며, 빈 리스트에도 삽입할 수 있다.

```
li=[]  
li.insert(0, 'good')  
print(li)  
['good']
```

### 리스트 항목 삭제

리스트에서 하나의 항목을 삭제하려면 메소드 **remove(값)** 또는 **pop(첨자), pop()**를 사용한다.

인자가 없는 pop()은 마지막 항목을 삭제하고 삭제된 값을 반환하며, 첨자를 사용한 pop()은 지정된 첨자의 항목을 삭제하고 반환한다.

```
li=[1,3,5,7]  
li.remove(3)  
print(li)  
print(li.pop())  
[1, 5, 7]  
7
```

문장 **del** 은 뒤에 위치한 변수나 항목을 삭제한다. 슬라이스로 리스트의 일부를 삭제할 수도 있다.

del 변수 는 아예 변수자체를 메모리에서 제거한다.

```
li=[1,3,5,7]
del li[0]
print(li)
```

[3, 5, 7]

메소드 **clear()**는 리스트의 모든 항목을 제거한다. 이후 리스트는 빈 리스트가 된다.

### 리스트의 추가,연결

리스트 메소드 **리스트.extend(list)**는 인자인 list 를 가장 뒤에 추가한다.

```
li=[1,3,5,7]
li.extend([9])
print(li)
```

[1, 3, 5, 7, 9]

### 리스트 항목의 순서와 정렬

```
li=[1,4,2,6]
|
li.reverse() #항목 순서를 뒤집는 메소드 reverse()
print(li)
li.sort()    #리스트 항목의 순서를 오름차순으로 정
print(li)
li.sort(reverse=True) #리스트 자체를 역순으로 정렬
print(li)
li2 = sorted(li)      #오름차순한 li를 새로운 리스트로 반환
print(li2)
```

[6, 2, 4, 1]  
[1, 2, 4, 6]  
[6, 4, 2, 1]  
[1, 2, 4, 6]

## E.리스트와 튜플 - 항목의 순서나 내용을 수정할 수 없는 튜플

### 튜플이란?

한 번 생성되면 항목의 추가, 변경, 삭제가 불가능한 리스트 형태 데이터 타입

리스트와 유사하지만, 값을 변경할 수 없음

**튜플명 = (값 1, 값 2, 값 3 , ...)**

- ✓ 소괄호 '('' 사이에 값들을 나열해서 튜플 변수명에 할당
- ✓ 튜플의 항목은 값들은 숫자형이나 문자형 데이터 타입을 가질 수 있고, 혼합하여 가질 수도 있음
- ✓ 항목값들은 콤마로 구분
- ✓ 항목의 순서인 인덱스는 0 부터 시작
- ✓ 항목이 한 개인 튜플은 항목 뒤에 콤마를 적어줌

## 튜플의 참조

```
singer = ('BTS', '불뽕간사춘기', 'BTS', '블랙핑크', '태연')
song = ('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
print(singer)
print(song)

print(singer.count('BTS'))
print(singer.index('불뽕간사춘기'))
print(singer.index('BTS')) #list와 사용하는 방법이 같다.
print()
#singer[0]='bts' #튜플의 값을 변경할 경우 에러 발생

for _ in range(len(singer)):
    print('%s: %s' % (singer[_], song[_]))

('BTS', '불뽕간사춘기', 'BTS', '블랙핑크', '태연')
('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
2
1
0

BTS: 작은 것들을 위한 시
불뽕간사춘기: 나만, 봄
BTS: 소우주
블랙핑크: Kill This Love
태연: 사계
```

## 튜플 메소드

튜플은 내용을 변경할 수 없는 데이터 타입이므로 메소드로도 항목의 내용을 변경할 수 없다.

리스트의 메소드 중 index(), count() 정도가 사용 가능하다.

```
li=('사과','귤','당근')
li2=sorted(li)
print(type(li2),li)

<class 'list'> ('사과', '귤', '당근')
```

## F.문자열과 논리연산 – 키와 값인 쌍의 나열인 딕셔너리

### 딕셔너리의 개념과 생성

#### 1)딕셔너리

키(key)와 값(value)의 쌍으로 이루어진 구조

딕셔너리의 키는 중복될 수 없고, 튜플처럼 변경할 수 없음, 값은 수정될 수 있다.

자료의 순서가 없음. 따라서 인덱스를 가지고 있지 않다.

#### 2)딕셔너리의 기본 구조

딕셔너리명 = {키 1:값 1,키 2:값 2,...}

✓ 중괄호 {} 사용

- ✓ 키와 값을 콜론 : 으로 구분
- ✓ 항목값들은 콤마 , 로 구분
- ✓ 키는 중복될 수 없고, 리스트를 사용할 수 없음 3)딕셔너리 생성

### 3)딕셔너리 생성

딕셔너리의 키는 문자열, 정수형, 실수형 ,튜플 등이 가능하다.

리스트는 딕셔너리의 키로 사용할 수 없다.

딕셔너리의 값에 해당하는 값은 문자열, 정수형, 실수형, 튜플, 리스트 등 모든 데이터 타입이 가능

```
a = {}
b = {1:'일',2:'이',3:'삼'}
print(b)
print(b[1])
```

```
{1: '일', 2: '이', 3: '삼'}
일
>>>
```

#b[1]에서 1 은 인덱스가 아닌 키값을 의미함

내장함수 dict() 함수에서 리스트나 튜플 1 개를 이용해 딕셔너리를 만들수 있다.

함수 dict()의 리스트나 튜플 내부에서 일련의 키-값 쌍으로 [키,값] 리스트 형식과

(키,값) 튜플 형식을 모두 사용할 수 있다.

```
day = dict ((( '월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday') ))
print(day)

{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
```

### 대표적인 딕셔너리의 멤버 함수

update(dict)	1.두개의 딕셔너리를 병합. 키 값이 다르면 새로운 키와 값을 추가해준다. 2.키 값이 중복되면 해당 키에 새로운 값으로 저장한다.
pop(key)	1.키 값을 입력하면 그 키에 해당하는 값과 함께 삭제+반환한다. 2.리스트처럼 'del a[키]'로도 지울 수 있다.
keys()	키를 리스트 형태로 출력한다.
values()	값을 리스트 형태로 출력한다.
clear()	딕셔너리의 모든 내용을 삭제하고 빈 딕셔너리 형태로 만든다.
get(key)	키에 대응하는 값을 불러온다. 만약 키가 존재하지 않는다면 None 을 반환한다.
items()	키와 값의 짝을 튜플로 묶어서 반환한다.



```

day = dict(((('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday'))))
print(day.items())
print(day.values())
print(day.get('월'))
print(day.popitem())

dict_items([('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')])
dict_values(['monday', 'tuesday', 'wednesday'])
monday
('수', 'wednesday')

```

## F. 문자열과 논리연산 – 중복과 순서가 없는 집합

파이썬의 Set 은 중복과 순서가 없는 집합이다. 수학에서의 집합과 비슷하다고 보면 된다.

### Set 자료형의 선언

```

set1 = set([1,2,3])      #{1,2,3} 출력

set2 = {4, 5, 6}        # {4, 5, 6} 출력

set3 = set("Hello")     # {'e', 'H', 'o', 'l'} 출력

set4 = frozenset([1, 2, 3]) # frozenset({1, 2, 3}) 출력

```

Set 자료형을 선언할 때는 set([ ]) 또는 { } 기호를 사용할 수 있다.

3 번째 줄에서 중복 및 순서가 없다는 특징을 보여준다.

frozenset 은 선언 후 변경 할 수 없는 집합이다.

### Set 자료형의 내부 함수

**add()** : 집합에 요소 추가

```

set.add(1)
set.add(2)
set.add('서울')
print(set)

# 배열의 항목을 추가하기
set.update(['부산', '인천'])
print(set)

# set의 항목을 추가하기
set.update({'대구', '대전', '광주'}) #set as iterable element
print(set)

{'서울', 1, 2}
{1, 2, '서울', '부산', '인천'}
{1, 2, '서울', '부산', '인천', '대구', '광주', '대전'}

```

### update() : 집합에 다른 집합 추가

```
set1 = set([1, 2, 3])
set1.update([4, 5, 6]) # {1, 2, 3, 4, 5, 6} 출력
set2 = {7, 8, 9}
set1.update(set2) # {1, 2, 3, 4, 5, 6, 7, 8, 9} 출력
```

### remove() : 집합의 해당 요소 제거

```
set.discard(1)
print(set)

# 다시 discard()함수를 이용해 1을 제거
#항목이 이미 제거되어 1이 없지만 예외가 발생하지 않는다.
set.discard(1)
print(set)

# remove()함수를 이용해 4를 제거
set.remove(4)
print(set)

# 다시 remove()함수를 이용해 4를 제거
#항목이 이미 제거되어 4가 없기 때문에 예외가 발생한다..
set.remove(4)
```

```
{2, 3, 4}
{2, 3, 4}
{2, 3}
```

```
Traceback (most recent call last):
  File "C:\Users\#tkdgm\OneDrive\바탕 화면\test.py", line 18, in <module>
    set.remove(4)
KeyError: 4
```

### clear() : 빈 집합으로 만듦

```
set1 = set([1, 2, 3])
set1.clear() # set() 출력(빈 집합)
```

### union() : 집합들의 합집합을 구함. | 연산자와 같음

```
set1 = set([1, 2, 3])
set2 = set([3, 4, 5])

set3 = set1.union(set2) # {1, 2, 3, 4, 5} 출력, set1 | set2 과 같음
```

### intersection() : 집합들의 교집합을 구함. & 연산자와 같음

```

set1 = set([1, 2, 3])
set2 = set([2, 3, 4])
set3 = set([1, 2, 5])

set4 = set1.intersection(set2).intersection(set3)
# {2} 출력, set1 & set2 & set3 과 같음

```

**difference()** : 집합들의 차집합을 구함. – 연산자와 같음

```

set1 = set([1, 2, 3])
set2 = set([2, 3, 4])

set3 = set1.difference(set2) # {1} 출력, set1 - set2와 같음

```

## F.문자열과 논리연산 – 내장함수 zip()과 enumerate(), 시퀀스 간의 변환

**zip()** 함수는 각기 다른 개수로 되어있는 자료형일지라도 동일한 개수로 쌍을 지어 묶어준다.

함수 zip()의 결과는 자료형 zip 이다. zip 은 간단히 리스트나 튜플로 변환할 수 있다.

```

name = ['a', 'b']
value = [1, 2]
for n, v in zip(name, value):
    print(n, v)

```

```

a 1
b 2

```

#두개의 리스트 name 과 value 에서 각각 하나씩 순서대로 받아와 튜플 형태로 넣는다.

for 문 안에 n,v 라는 2 개의 값을 넣었기 때문에 묶여서 나오지 않는다.

**enumerate()** 함수는 순서가 있는 자료형(리스트, 튜플, 문자열)을 입력으로 받아 인덱스 값을 포함하는 enumerate 객체로 돌려준다.

```

#for을 사용했을때
a = [1, 2, 3, 4, 5]
for i in a:
    print(i)

#enumerate() 사용
for index,value in enumerate(a):
    print(index, value)

```

```

1
2
3
4
5
0 1
1 2
2 3
3 4
4 5

```

enumerate()에서 index 값을 생략한다면?

```
a = [1, 2, 3, 4, 5]
for v in enumerate(a):
    print(v)
```

```
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
```

그래도 인덱스가 value 에 포함되어 함께 출력되는 것을 볼 수 있다.

## 시퀀스 간의 변환

튜플 > 리스트

```
space = '밤', '낮', '해', '달'
print(list(space))
#['밤', '낮', '해', '달']
```

리스트 > 집합

```
space = '밤', '낮', '해', '달'
print(set(space))
#{'밤', '해', '달', '낮'} # set은 순서가 없음
```

딕셔너리 > 리스트

```
game = dict(일='1',이='2',삼='3',사='4')
lgame = list(game)
print(lgame)
```

```
['일', '이', '삼', '사']
```

딕셔너리가 다른 자료형으로 변환될 때 키(Key) 값만 남게 된다.