

2020 Python 포트폴리오

컴퓨터정보공학과 20192611 김은진

Python?

- 오픈소스 프로그래밍 언어
- 풍부한 라이브러리
- 다양한 개발 환경
- 쉽고 빠른 소프트웨어 개발 가능!

배운 목적

- 기본적인 프로그래밍 언어
- 전 세계적으로 가장 많이 가르치는 프로그래밍 언어
- 아직도 계속해서 빠르게 상승 추세인 언어
- 4차 산업혁명시대에 맞는 컴퓨터 사고력을 위해

목차

- 파이썬 프로그래밍을 위한 기초다지기
- 일상에서 활용되는 문자열과 논리 연산
- 일상생활과 비유되는 조건과 반복
- 항목의 나열인 리스트와 튜플
- 키와 값의 나열인 딕셔너리와 집합

파이썬 프로그래밍을 위한 기초

- 다양한 자료: 문자열과 수
- 변수와 키워드, 대입 연산자
- 자료의 표준 입력과 자료 변환 함수

다양한 자료:문자열과 수

- 문자열 : 문자 하나 또는 문자가 모인 단어나 문장
 - ex)A, 12, 파이썬
 - ' ' 또는 " "를 앞뒤에 동일하게 사용
- 정수 : 15,7,20 등, integer
- 실수 : 3.14, 2.718 등, real 또는 float

다양한 자료:문자열과 수

문자열의 출력

print()를 이용



print('문자열')

문자열의 연결

print('문자열1' '문자열2')
print('String1'+'String2')



문자열1문자열2
String1String2

문자열의 반복

print(4*'반복')
print("반복"*3)



반복반복반복반복
반복반복반복

삼중 따옴표

print('''삼중 따옴표는
여러줄에 걸쳐 긴 문장을 쓸때 사용합니다''')



삼중 따옴표는
여러줄에 걸쳐 긴 문장을 쓸때 사용합니다

다양한 자료:문자열과 수

- 다양한 연산자

연산자	명칭	의미	예
*	곱하기	두 피연산자 곱하기	$3 * 4 \rightarrow 12$
/	나누기	왼쪽을 오른쪽 피연산자로 나누기	$10 / 4 \rightarrow 2.5$
%	나머지	왼쪽을 오른쪽 피연산자로 나눈 나머지	$21 \% 4 \rightarrow 1$
//	몫 나누기	왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수	$10 // 3 \rightarrow 3$
**	거듭제곱	왼쪽을 오른쪽 피연산자로 거듭제곱	$2 ** 3 \rightarrow 8$

다양한 자료:문자열과 수

- 언더스코어(_)

```
4
>>> _
4
>>> 25*_
100
```

- 대화형 모드에서 마지막에 실행된 결과 값을 대입해준다.
- 숫자,문자열 등 유효한 연산식이면 모두 가능하다.

- eval()

```
print(eval('3+15/2'))
print(eval(' "python"*3'))
```



```
10.5
pythonpythonpython
```

- 표현식 문자열 실행 함수
- 함수 인자는 반드시 연산 가능한 문자열이어야 함
- 문자열 연산도 가능
- 결과도 문자열

다양한 자료:문자열과 수

예제)

```
print(1,2,-5,3.14,2.71828)
print('Hi','Python!')
print(type(1))

print('23000원 은 ','5000원?개' "1000원?개")
print('5000원 ',23000 // 5000,'개 ')
print('1000원 ',(23000%5000) // 1000,'개 ')
```



```
1 2 -5 3.14 2.71828
Hi Python!
<class 'int'>
23000원 은 5000원?개 1000원?개
5000원 4 개
1000원 3 개
```

- 줄을 한 줄 비워도 결과에선 나타나지 않는다.
- 숫자는 ' / ' " 처리를 하지 않으면 문자열이
- 아닌 정수나 실수형태로 저장된다.

변수와 키워드,대입 연산자

- 변수
 - 자료를 담을 수 있는 그릇!
 - 영문자와 숫자 , ' _ '사용 가능
 - 키워드는 사용이 불가능 하다
 - 숫자는 맨 앞에 올 수 없다

변수이름 = 30

대입 연산자 오른쪽값을 왼쪽변수에 저장

변수와 키워드, 대입 연산자

다양한 대입 연산자

연산자	형태	의미
=	$a = b$	b의 결과값을 변수 a에 저장
+=	$a += b$	$a + b$ 의 결과 값을 변수 a에 저장
-=	$a -= b$	$a - b$ 의 결과 값을 변수 a에 저장
*=	$a *= b$	$a * b$ 의 결과 값을 변수 a에 저장
/=	$a /= b$	a / b 의 결과 값을 변수 a에 저장
%=	$a \% b$	$a \% b$ 의 결과 값을 변수 a에 저장
//=	$a //= b$	$a // b$ 의 결과 값을 변수 a에 저장
**=	$a ** b$	$a ** b$ 의 결과 값을 변수 a에 저장

변수와 키워드,대입 연산자

- `type()` 자료 유형을 알아볼 수 있는 함수

```
>>> type(3)
<class 'int'>
>>> type('hello')
<class 'str'>
>>> type(3.14)
<class 'float'>
```
- `divmod()` //연산과 %연산을 함께 수행 해 2개의 결과를 반환한다

```
>>> divmod(28,3)
(9, 1)
```

변수와 키워드,대입 연산자

동일한 변수에 값을 수정하는 다양한 대입 연산자

```
int = 1  
int = int+1  
print(int)  
int *=5  
print(int)
```



2
10

- 하지만 `int+1 = int` 처럼 대입연산자 왼쪽에 변수가 두개 이상 온다면 오류가 난다.
- 선언 하지 않은 변수는 사용이 불가능 하다.


자료의 표준 입력과 자료 변환 함수

- 표준입력 : 사용자의 입력을 받아 처리하는 방식.
- 16진법 : 단위가 16이다. 앞이 0x로 숫자가 시작한다.
- 8진법 : 단위가 8이다. 앞이 0o로 숫자가 시작한다.
- 2진법 : 단위가 2이다. 앞이 0b로 숫자가 시작한다.

자료의 표준 입력과 자료 변환 함수




`input()` 입력되는 표준 입력을 문자열로 읽어 반환하는 함수

```
num = input('숫자는?')  
print(num)
```



숫자는?5
5

자료 변환 함수

함수	역할	예
<code>str()</code>	문자열로 변환	<pre>name = 1234 print(type(name)) print(type(str(name)))</pre>  <pre><class 'int'> <class 'str'></pre>
<code>int()</code>	정수로 변환	<pre>name = '1234' print(type(name)) print(type(int(name)))</pre>  <pre><class 'str'> <class 'int'></pre>
<code>float()</code>	실수로 변환	<pre>name = '1.234' print(type(name)) print(type(float(name)))</pre>  <pre><class 'str'> <class 'float'></pre>

자료의 표준 입력과 자료 변환 함수

- 상수의 표현

```
num = 36  
print('2진 수:', bin(num))  
print('8진 수:', oct(num))  
print('16진 수:', hex(num))
```




2진 수: 0b100100
8진 수: 0o44
16진 수: 0x24

- bin() : 2진수 변환 함수
- oct() : 8진수 변환 함수
- hex() : 16진수 변환 함수
- 10진수가 아닌 다른 진수의 수도 변환 가능

자료의 표준 입력과 자료 변환 함수


- int()의 사용

```
num = '11'  
print(int(num,10))  
print(int(num,2))  
print(int(num,8))  
print(int(num,16))
```



```
11  
3  
9  
17
```

```
print(int('0b11',0))  
print(int('0o11',0))  
print(int('0x11',0))
```



```
3  
9  
17
```

- int('문자열',n)
:n진수인 문자열을 10진수로 변환
- int('문자열',0)
:문자열에 맞는 진수로 인지해 10진수로 변환

파이썬 프로그래밍을 위한 기초

- 문자열의 정의와 간단한 활용을 할 수 있음
- 간단한 정수와 실수의 연산가능
- 표준입력의 정의와 활용을 배움
- 정수와 실수의 변환 함수를 실습함
- 문자열의 정수,실수 변환하는 법을 알게 됨

파이썬 프로그래밍을 위한 기초

```
num1 = float(input('첫 번째 수 입력 >>'))
num2 = float(input('두 번째 수 입력 >>'))
print('합:', num1+num2)
print('차:', num1-num2)
print('곱:', num1*num2)
print('나누기:', num1/num2)
expression = input('연산식 입력')
print('연산식:', expression, '결과:', eval(expression))
```



```
첫 번째 수 입력 >>52
두 번째 수 입력 >>4
합: 56.0
차: 48.0
곱: 208.0
나누기: 13.0
연산식 입력 15/2+3
연산식: 15/2+3 결과: 10.5
```

```
base = int(input('입력할 정수의 진수는?'))
invar = input(str(base) + '진수 정수 입력>>')
data = int(invar, base)
```

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



```
입력할 정수의 진수는?10
10진수 정수 입력>>9
2진수: 0b1001
8진수: 0o11
10진수: 9
16진수: 0x9
```

일상에서 활용되는 문자열과 논리연산

- 문자열 다루기
- 문자열 관련 메소드
- 논리 자료와 다양한 연산

문자열 다루기

- len() : 문자열의 길이를 알 수 있다.
- 문자열의 문자 참조

```
str = 'Hello python'  
Le = len(str)  
print(str[0])  
print(str[Le-1])  
print(str[-Le])
```



H
n
H

- 문자열을 구성 하는 문자는 0부터 시작한다.
- -를 붙이면 역순으로 참조한다.

문자열 다루기

문자열 슬라이싱

```
str = 'Hello python'
print(str[0:3])
print(str[1:2])
print(str[-9:-2])
print(str[-4:-1])
```



```
Hel  
e  
lo pyth  
tho
```

```
str = 'Hello python'
print(str[1:-1])
print(str[0:-1])
print(str[2:-1])
print(str[1:-2])
```



```
ello pytho  
Hello pytho  
llo pytho  
ello pyth
```

- 양수 : n1:n2 라면 n1부터 n2전 까지 반환
- 음수 : n1:n2 라면 n1부터 n2전 까지 반환
- 순방향을 나타내는 0
- 1:-1 : 1부터 -1전 까지 -2까지 반환
- 0:-1 : 0부터 -2까지 반환

문자열 다루기

문자열 슬라이싱

- 공백의 의미

```
str = 'Hello python'  
print(str[:3])  
print(str[1:])
```



```
Hel  
ello python
```

'처음부터'와 '끝까지' 의미

- 문자 사이의 간격 조정

```
str = 'Hello python'  
print(str[1:2:3])  
print(str[1:5:2])  
print(str[::1])
```



```
e  
el  
Hello python
```

해당 숫자만큼 건너뛰고 반환

문자열 다루기

- `ord()` 유니코드를 알려주는 함수
 - ```
>>> ord('가')
44032
>>> ord('A')
65
>>> hex(ord('가'))
'0xac00'
```
- `chr()` 유니코드 번호를 문자로 반환 해 주는 함수
  - ```
>>> chr(44032)
'가'
>>> chr(65)
'A'
>>> chr(0xac00)
'가'
```
- `min()` 최소값 반환
 - ```
>>> min('hello,python')
','
>>> min('123456')
'1'
>>> min(4,7,3,9)
3
```
- `max()` 최대값 반환
  - ```
>>> max('hello,python')
'y'
>>> max('123456')
'6'
>>> max(4,7,3,9)
9
```

문자열 관련 메소드

- `replace()` 문자열안의 문자를 대체해주는 메소드

```
>>> str = '파이썬은 자바이다'
>>> str.replace('자바', '파이썬')
'파이썬은 파이썬이다'
>>> str.replace('파이썬', '사과')
'사과는 자바이다'
```

- `join()` 문자와 문자사이에 원하는 문자열을 삽입해주는 메소드

```
>>> str = '20200520'
>>> '-'.join(str)
'2-0-2-0-0-5-2-0'
```

문자열 관련 메소드

- `find()` 문자가 맨 처음 위치한 첨자를 반환 하는 메소드

```
>>> str = '사과는 배보다 맛있다'
>>> str.find('사과')
0
>>> str.find('배')
4
>>> str.find('수박')
-1
```



없을 경우 0을 반환!

- `index()` 문자가 맨 처음 위치한 첨자를 반환 하는 메소드

```
>>> str = '사과는 배보다 맛있다'
>>> str.index('사과')
0
>>> str.index('배')
4
>>> str.index('수박')
Traceback (most recent call last):
  File "<pyshell#41>", line 1, in <module>
    str.index('수박')
ValueError: substring not found
```



없을 경우 오류!

문자열 관련 메소드

메소드	메소드 예	결과
center()	'파이썬 강좌'.center(10,'*')	'**파이썬 강좌**'
	'파이썬 강좌'.center(10,)	' 파이썬 강좌 '
	'파이썬 강좌'.center(10,'=')	'==파이썬 강좌=='
ljust()/rjust()	'python'.ljust(10,'*')	'python*****'
	'python'.rjust(10,'*')	'*****python'
strip()	'python '.lstrip()	'python '
	'python '.rstrip()	' python'
	'python '.strip()	'python'
	'***python==='.strip()	'python'
zfill()	'234'.zfill(5)	'00234'

문자열 관련 메소드

- split()

```
str = '사과, 배, 수박, 참외'
print(str.split())
print(str.split(','))
print(str.split(', '))
```



```
['사과', '배', '수박', '참외']
['사과', '배', '수박', '참외']
['사과', '배', '수박', '참외']
```

- 공백을 기준으로 문자열을 나눠주는 메소드
- 나눈 항목은 리스트로 저장
- 특정 문자로 나눌 수도 있음
- 문자와 공백을 합하여 나눌 수 있음

문자열 관련 메소드

- format()

```
a,b,c, = 15,7,2
print('{}+{}={}'.format(3,4,3+4))
print('{2},{1},{0}'.format(a,b,c))
print('{:3d},{:f},{:4.3f}'.format(a,a,a))
print('{0:3d} % {1:2d} = {2:4.4f}'.format(a,b,a/b))
print('{:b},{:o},{:x}'.format(a,a,a))
print('{0:10f}{0:10F}'.format(3.5E1000))
print('{0:10f}{0:10F}'.format(float('nan')))
```



```
3+4=7
2,7,15
15,15.000000,15.000
15 % 7 = 2.1429
1111,17,f
           inf      INF
           nan      NAN
```

- {}안에 값을 넣어 주는 함수
- {n}이라면 format()에 n번째를 반환
- {:d}/{:f} 각각 정수와 실수로 반환
- {:nd} n만큼의 폭 지정
- {:n.md} n만큼의 폭에서 m만큼의 소수점 폭
- 무한대는 inf로 반환
- 계산이 불가할때는 nan으로 반환

논리 자료와 다양한 연산

- bool과 bool() : 논리 값의 자료형, 논리 값을 반환해 주는 함수
- and(&) : 두 항이 모두 참 일때 true 반환, 하나라도 거짓이면 false 반환 하는 연산자
- or(|) : 두 항이 모두 거짓일때 false 반환, 하나라도 참이면 true 반환 하는 연산자
- ^ : 두항이 같으면 true 다르면 false를 반환하는 연산자
- not:뒤에 위치한 논리 값을 바꾼다.

```
>>> True&True, True&False, True|False    >>> not True, not False  
(True, False, True)                      (False, True)
```

논리 자료와 다양한 연산

- 다양한 관계 연산자

연산자	연산 사용	의미	연산자	연산 사용	의미
>	$a > b$	크다	<=	$a <= b$	작거나 같다
>=	$a >= b$	크거나 같다	==	$a == b$	같다
<	$a < b$	작다	!=	$a != b$	다르다

- 멤버십 연산 in

연산	결과
$x \text{ in } s$	문자열에 s 에 부분 문자열 x 가 있으면 True 없으면 False반환
$x \text{ not in } s$	문자열에 s 에 부분 문자열 x 가 없으면 True 있으면 False반환

논리 자료와 다양한 연산

- 비트 논리 연산 :비트 연산은 정수로 저장된 메모리에서 비트와 비트의 연산을 말한다.

m	n	논리합	논리합	배타적 논리합
		$m \& n$	$m n$	$m \wedge n$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

논리 자료와 다양한 연산

- &mask 연산 : 특정 비트 값을 알아 낼 수 있음.

```
mask = 0b1111
a = int(input())
print('a의 오른쪽 4자리 비트는 {0:04b}'.format(a&mask))
```



182
a의 오른쪽 4자리 비트는 0110

```
mask = 0b11111111
a = int(input())
print('a의 오른쪽 8자리 비트는 {0:08b}'.format(a&mask))
```



678
a의 오른쪽 8자리 비트는 10100110

논리 자료와 다양한 연산

- 비트 이동 연산자 >> 와 <<

:연산자의 방향으로 비트 단위로 뒤 피연산자인
지정된 횟수만큼 이동시키는 연산자

```
print('10진 수 {0:3d}, 2진 수 {0:08b}'.format(20))  
print('10진 수 {0:3d}, 2진 수 {0:08b}'.format(20 >> 1))  
print('10진 수 {0:3d}, 2진 수 {0:08b}'.format(20 >> 4))  
print('10진 수 {0:3d}, 2진 수 {0:08b}'.format(20 << 1))  
print('10진 수 {0:3d}, 2진 수 {0:08b}'.format(20 << 4))
```



10진 수	20,	2진 수	00010100
10진 수	10,	2진 수	00001010
10진 수	1,	2진 수	00000001
10진 수	40,	2진 수	00101000
10진 수	320,	2진 수	101000000

- <<에 의해 생기는 빈자리는 0으로 채워진다
- >>에 의한 빈자리는 원래의 정수 부호에 따라 양수면0,음수면1로 채워진다

일상에서 활용되는 문자열과 논리연산

```
price = int(input('총 가격(원 가격) 입력 >>'))  
rate1 = (10000 <= price < 20000) * 1/100  
rate2 = (20000 <= price < 40000) * 2/100  
rate3 = (40000 <= price) * 4/100  
rate = rate1 + rate2 + rate3  
discount = price * rate  
discPrice = price - discount  
print("원가격:", price, '할인된 가격:', discPrice)  
print("할인율:", rate, '할인액:', discount)
```

→ 가격에 따라
할인율이 달라짐



총 가격(원 가격) 입력 >>67000
원가격: 67000 할인된 가격: 64320.0
할인율: 0.04 할인액: 2680.0

```
str = input('콤마로 구분된 단어 3개 입력 >>')  
str = str.replace(' ', ',')  
w1,w2,w3 = str.split(',')  
print('단어: {}, 역순: {}, 회문: {}'.format(w1, w1[::-1], w1==w1[::-1]))  
print('단어: {}, 역순: {}, 회문: {}'.format(w2, w2[::-1], w2==w2[::-1]))  
print('단어: {}, 역순: {}, 회문: {}'.format(w3, w3[::-1], w3==w3[::-1]))
```

w1==w1[::-1]
w2==w2[::-1]
w3==w3[::-1]



단어의 역순이 같은지 판정



콤마로 구분된 단어 3개 입력 >>오징어, 갈치, octopus
단어: 오징어, 역순: 어징오, 회문: False
단어: 갈치, 역순: 치갈, 회문: False
단어: octopus, 역순: supotco, 회문: False

일상에서 비유되는 조건과 반복

- 조건에 따른 선택 if---else
- 반복을 제어하는 for문과 while문
- 임의의 수인 난수와 반복을 제어하는 break문,continue문

조건에 따른 선택 if---else

- if -- else: 조건이 True일 때 블록 구문을 실행한다.

```
grade = float(input('1학기 평균 평점은?'))  
if 3.8 <= grade:  
    print('축하합니다! 장학금 지급 대상자이다.')  
print('당신의 1학기 평균 평점은 %.2f이다.'%(grade))
```

→ 블록 구문

↓
1학기 평균 평점은?3.9
축하합니다! 장학금 지급 대상자이다.
당신의 1학기 평균 평점은 3.90이다.

```
grade = float(input('1학기 평균 평점은?'))  
if 3.8 <= grade:  
    print('축하합니다! 장학금 지급 대상자이다.')  
print('당신의 1학기 평균 평점은 %.2f이다.'%(grade))
```

→ 블록 구문

↓
1학기 평균 평점은?2.5
당신의 1학기 평균 평점은 2.50이다.

→ 조건 문을 만족하지 않아 블록 구문이 실행 되지 않음

조건에 따른 선택 if---else

여러가지 조건을 가질 경우

```
category = int(input('원하는 음료는? 1.커피 2.주스'))
```

```
if category ==1:
    menu = int(input('번호 선택? 1.아메리카노 2.카페라테 3.카푸치노'))
    if menu == 1:
        print('1.아메리카노 선택')
    elif menu == 2:
        print('2.카페라테 선택')
    elif menu ==3:
        print('3.카푸치노 선택')
```

→ 벗어남

```
else:
    menu = int(input('번호 선택? 1.키워주스 2.토마토주스 3.오렌지주스'))
    if menu ==1:
        print('1.키워주스 선택')
    elif menu == 2:
        print('2.토마토주스 선택')
    elif menu ==3:
        print('3.오렌지주스 선택')
```



원하는 음료는? 1.커피 2.주스2
번호 선택? 1.키워주스 2.토마토주스 3.오렌지주스1
1.키워주스 선택

반복을 제어하는 for문과 while문

- for문은 항목의 나열인 <시퀀스>의 구성 요소인 모든 항목이 순서대로 변수에 저장 돼 반복을 수행한다.

for 변수 in <시퀀스>:

반복 몸체인_문장들

- while문은 <반복조건>에 따라 반복을 수행한다.

while <반복조건>:

반복 몸체인_문장들

반복을 제어하는 for문과 while문

for문 예제)

```
for i in range(3):
    coffee = input("주문하세요! [아메리카노] [카페라테] [카푸치노] >>")
    if coffee == '아메리카노':
        print('%s 주문' %coffee)
    elif coffee == '카페라테':
        print('%s 주문' %coffee)
    elif coffee == '카푸치노':
        print('%s 주문' %coffee)
    else:
        print('모르겠어요.')
else:
    print('주문을 마치겠습니다.')
```



```
주문하세요! [아메리카노] [카페라테] [카푸치노] >>카페라테
카페라테 주문
주문하세요! [아메리카노] [카페라테] [카푸치노] >>아메리카노
아메리카노 주문
주문하세요! [아메리카노] [카페라테] [카푸치노] >>카푸치노
카푸치노 주문
주문을 마치겠습니다.
```

- range(n)은 0에서 n-1까지의 수를 항목인 정수로 구성되는 시퀀스를 만든다.
- 따라서 0부터 2까지,
0,1,2가 항목인 시퀀스가 생성된다.

반복을 제어하는 for문과 while문

while문 예제)

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



```
2 * 1 = 2 2 * 2 = 4 2 * 3 = 6 2 * 4 = 8 2 * 5 = 10  
2 * 6 = 12 2 * 7 = 14 2 * 8 = 16 2 * 9 = 18  
3 * 1 = 3 3 * 2 = 6 3 * 3 = 9 3 * 4 = 12 3 * 5 = 15  
3 * 6 = 18 3 * 7 = 21 3 * 8 = 24 3 * 9 = 27  
4 * 1 = 4 4 * 2 = 8 4 * 3 = 12 4 * 4 = 16 4 * 5 = 20  
4 * 6 = 24 4 * 7 = 28 4 * 8 = 32 4 * 9 = 36  
5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25  
5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45  
6 * 1 = 6 6 * 2 = 12 6 * 3 = 18 6 * 4 = 24 6 * 5 = 30  
6 * 6 = 36 6 * 7 = 42 6 * 8 = 48 6 * 9 = 54  
7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35  
7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63
```

■
■
■

- range(n,m)을 이용해 n부터 m-1까지 반복
- 중첩 반복 사용
- 들여쓰기가 중요하다
- else이후의 블록문장은 반복이 종료된 후
마지막에 실행된다

난수와 제어문 break,continue

- 난수: 임의의 수를 발생한다
- break문:반복 내부에서 블록을 실행시키지 않고 종료 해 주기 위한 제어문이다
- continue:continue 이후의 반복 몸체를 실행하지 않고 다음 번 복을 위해 논리 조건을 수행한다

난수와 제어문 break, continue

break문 예제)

```
from random import randint
LUCKY = 7

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



```
1,7 나올 때까지 계속!
4,7 나올 때까지 계속!
5,7 나올 때까지 계속!
드디어 7, 종료!
```

continue문 예제)

```
days = ['monday', 'tuesday', 'wendnesday']

while True:
    user = input('월,화,수 중 하나 영어 단어 입력 >> ')
    if user not in days:
        print('잘못 입력했어요!')
        continue
    print('입력: %s, 철자가 맞습니다,' %user)
    break
print('종료'.center(15, '*'))
```



```
월,화,수 중 하나 영어 단어 입력 >> friday
잘못 입력했어요!
월,화,수 중 하나 영어 단어 입력 >> monday
입력: monday, 철자가 맞습니다,
*****종료*****
```

일상에서 비유되는 조건과 반복

```
import random
answer = random.randint(1,10)
indata = int(input('1에서 10 사이의 수를 맞춰주세요 >>'))
while True:
    if indata == answer:
        print('축하합니다. {}: 정답입니다'.format(indata))
        break;
    elif indata < answer:
        str = '{}보다 더 큰 수로 다시 입력 >>'.format(indata)
    else:
        str = '{}보다 더 작은 수로 다시 입력 >>'.format(indata)
    indata = int(input(str))

print("종료".center(30,"*"))
```



```
1에서 10 사이의 수를 맞춰주세요 >>7
7보다 더 작은 수로 다시 입력 >>1
1보다 더 큰 수로 다시 입력 >>3
축하합니다.3: 정답입니다
*****종료*****
```

항목의 나열인 리스트와 튜플

- 여러 자료 값을 편리하게 처리하는 리스트
- 리스트의 부분 참조와 항목의 삽입과 삭제
- 항목의 순서나 내용을 수정 할 수 없는 튜플

여러 자료 값을 처리하는 리스트

- 리스트: 여러 항목을 하나의 단위로 묶어 손쉽게 사용하는 복합 자료형
- 항목의 나열인 시퀀스
- 항목이 같은 자료형일 필요는 없다
- 자료값이 중복 일 수 있다

여러 자료 값을 처리하는 리스트

```
coffee = []  
print(type(coffee))  
print(coffee)  
coffee = ['에스프레소', '아메리카노']  
print(type(coffee))  
print(coffee)  
coffee.append('카페라떼')  
print(coffee)  
print(len(coffee))
```



```
<class 'list'>  
[]  
<class 'list'>  
['에스프레소', '아메리카노']  
['에스프레소', '아메리카노', '카페라떼']  
3
```

- 이름 = [] : 빈 리스트 생성
- .append() : 리스트 맨뒤에 항목을 추가한다
- len(): 리스트의 항목 수를 반환 하는 함수

여러 자료 값을 처리하는 리스트

```
from random import choice
lista = [1,2,3,4,5]
print(lista[2])
print(lista[-1])
print(choice(lista))
```



3
5
2

- 리스트이름 [n]: 리스트의 n-1번째 항목을 참조한다
- 리스트의 항목은 0부터 시작한다
- 음수는 역순으로 -1부터 시작한다
- choice()는 리스트의 항목 중 하나를 임의로 선택해 준다

여러 자료 값을 처리하는 리스트

- count()

```
lista = ['사과', '복숭아', '사과', '배', '수박', '배', '사과']  
print(lista.count('사과'))  
print(lista.count('배'))
```



3
2

- 리스트의 해당 항목의 개수를 알려준다

- index()

```
lista = ['사과', '복숭아', '사과', '배', '수박', '배', '사과']  
print(lista.index('사과'))  
print(lista.index('배'))
```



0
3

- 리스트에서 해당 항목이 제일 처음 나온 순서를 알려준다

- 항목수정

```
lista = ['사과', '복숭아', '사과', '배', '수박', '배', '사과']  
lista[3] = '복숭아'  
print(lista)
```



['사과', '복숭아', '사과', '복숭아', '수박', '배', '사과']

여러 자료 값을 처리하는 리스트

```
animal = [['사자', '코끼리', '호랑이'], '조류', '어류']  
print(animal)
```

```
for s in animal:  
    print(s)  
print()
```

```
bird = ['독수리', '참새', '까치']  
fish = ['갈치', '붕어', '고등어', '참치']  
animal[1:] = [bird, fish]  
print(animal)
```

```
for lst in animal:  
    for item in lst:  
        print(item, end = ' ')  
    print()  
print()
```



```
['사자', '코끼리', '호랑이'], '조류', '어류']  
['사자', '코끼리', '호랑이']  
조류  
어류
```

```
['사자', '코끼리', '호랑이'], ['독수리', '참새', '까치'],  
['갈치', '붕어', '고등어', '참치']  
사자 코끼리 호랑이  
독수리 참새 까치  
갈치 붕어 고등어 참치
```

- 리스트안에 리스트가 항목으로 올 수 있다
- [n:] = [리스트] :n번째 항목부터 리스트로 대체해준다
- 중첩 반복문으로 리스트 안의 리스트를 참조 할 수 있다

부분참조와 항목의 삽입과 삭제

리스트 슬라이싱

```
lista = [1,2,3,4,5,6,7,8,9]
print(lista[:])
print(lista[::])
print(lista[::-1])
print()
print((lista[1:5]))
print((lista[1:5:2]))
print((lista[6::-1]))
print((lista[6::-2]))
```



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

```
[2, 3, 4, 5]
[2, 4]
[7, 6, 5, 4, 3, 2, 1]
[7, 5, 3, 1]
```

- [:],[::] : 항목 전체를 참조
- [::-1] : 항목 전체를 역순으로 참조
- [n1:n2] : n1부터n2까지 참조
- [n1:n2:n3] : n1부터n2까지 n3간격으로 참조
- n3이 음수라면 역순으로 참조한다
- n3이 -1보다 작으면 간격은 절대값만큼이다

부분참조와 항목의 삽입과 삭제

```
lista = ['문학', '비문학', '영어', '불어', '중국어', '외국어', '수학', '과학']  
lista[0:2] = ['국어']  
print(lista)  
lista[1:4] = lista[4:4]  
print(lista)  
lista.insert(4, '기술과가정')  
print(lista)
```



```
['국어', '영어', '불어', '중국어', '외국어', '수학', '과학']  
['국어', '외국어', '수학', '과학']  
['국어', '외국어', '수학', '과학', '기술과가정']
```

- 리스트는 = 로 수정이 가능하다
- 일정 구간을 바꿀 수도 있다
- 다른 리스트의 슬라이스도 가능하다
- insert(n, '항목'):
n번째에 항목을 삽입 해 준다

부분참조와 항목의 삽입과 삭제

```
lista = ['밥', '국어', '외국어', '수박', '수학', '과학', '키위', '기술과가정']
lista.remove('수박')
print(lista)
lista.pop(4)
print(lista)
del lista[0]
print(lista)
lista.clear()
print(lista)
del lista
print(lista)
```



```
['밥', '국어', '외국어', '수학', '과학', '키위', '기술과가정']
['밥', '국어', '외국어', '수학', '키위', '기술과가정']
['국어', '외국어', '수학', '키위', '기술과가정']
[]
```

```
Traceback (most recent call last):
  File "C:\wpyhton\문자열연결.py", line 14, in <module>
    print(lista)
NameError: name 'lista' is not defined
```

- remove('값') : 리스트에서 값을 삭제 한다
- pop(n) : n항목을 삭제한다
- pop() : 맨 마지막 항목을 삭제한다
- del 리스트[n] : 리스트의 n을 삭제
- del 리스트 : 리스트를 아예 삭제
- clear() : 빈 리스트로 만들어 준다

부분참조와 항목의 삽입과 삭제

- reverse()

```
listnum = [1,2,3,4,5,6,7]  
listnum.reverse()  
print(listnum)
```



[7, 6, 5, 4, 3, 2, 1]

- 항목 순서를 뒤집어준다

- sort()

```
listnum = [4,6,2,7,3,1,5]  
listnum.sort()  
print(listnum)
```



[1, 2, 3, 4, 5, 6, 7]

- 항목을 오름차순으로 정렬한다

- sorted()

- 항목을 오름차순으로 정렬한 것을 새로운 리스트로 반환한다.

부분참조와 항목의 삽입과 삭제

- 컴프리헨션 : 구문을 함축하여 코딩하는 것을 일컫는다

예제)

```
a = []  
for i in range(10):  
    a.append(i)  
print(a)  
  
seq = [i for i in range(10)]  
print(seq)
```



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

```
for i in range(10):  
    a.append(i)
```



```
seq = [i for i in range(10)]
```


부분참조와 항목의 삽입과 삭제

- 리스트의 복사

```
lista = ['자바', 'C언어', '파이썬']  
listb = lista  
print(listb)  
print(lista is listb)  
listb.pop() is 는 동일 객체 여부를 검사한다  
print(lista)
```



```
['자바', 'C언어', '파이썬']  
True  
['자바', 'C언어']
```

- 두 변수는 같은 객체

- 리스트의 깊은 복사

```
lista = ['자바', 'C언어', '파이썬']  
listb = lista[:]  
print(listb)  
print(lista is listb)  
listb.pop() is 는 동일 객체 여부를 검사한다  
print(lista)
```



```
['자바', 'C언어', '파이썬']  
False  
['자바', 'C언어', '파이썬']
```

- 두 변수는 다른 객체

순서나 내용을 수정 할 수 없는 튜플

- 튜플 : 항목의 나열인 시퀀스

- 수정이 불가능하다
- 콤마로 구분된 리스트로 표현된다
- 항목의 제한이 없다

- 튜플의 생성과 참조

```
tuplea = ()  
print(type(tuplea))  
tuplea = ('가', '나', '다')  
print(tuplea[1])
```



```
<class 'tuple'>  
나
```

순서나 내용을 수정 할 수 없는 튜플

- 튜플의 연결과 반복, 정렬과 삭제

```
day1 = ('monday', 'tuesday', 'wendnesday')
day2 = ('thursday', 'friday', 'saturday', 'sunday')
day = day1 + day2
print(day)
print(sorted(day, reverse=True))
print(day)
del day
print(day)
```

reverse = True 역순 내림차순으로 반환

삭제



```
('monday', 'tuesday', 'wendnesday', 'thursday', 'friday', 'saturday', 'sunday')
['wendnesday', 'tuesday', 'thursday', 'sunday', 'saturday', 'monday', 'friday']
('monday', 'tuesday', 'wendnesday', 'thursday', 'friday', 'saturday', 'sunday')
Traceback (most recent call last):
  File "C:\python\5-13.py", line 11, in <module>
    print(day)
NameError: name 'day' is not defined
```

sorted()는 새로운 리스트를 반환하기
때문에 기존의 튜플은 변하지 않는다

항목의 나열인 리스트와 튜플

```

menu = ('주문 종료', '올인원팩', '투게더팩', '트리오팩', '패밀리팩')
price = (0, 6000, 7000, 8000, 10000)

msg = '주문할 콤보 번호와 수량을 계속 입력하세요!'
for i in range(len(menu)):
    msg += '\n\n\t %d %s' % (i, menu[i])
    if i != 0:
        msg += ' %5d원' % (price[i])

more = True
total = 0
while more:
    print(msg)
    order = input('콤보번호 입력>>')
    order = int(order)
    if order == 0:
        print()
        print('주문 종료'.center(20, '*'))
        more = False
    elif 1 <= order <= 4:
        cnt = input('수량 입력>>')
        cnt = int(cnt)
        print('%s, %d개 주문' % (menu[order], cnt))
        sub = price[order] * cnt
        total += sub
        print('%s 주문 가격 : %d, 총 가격 %d' % (menu[order], sub, total))
    else:
        print('모르겠어요, 다시 주문해주세요')
else:
    print('총 주문 가격 %d원' % total)
    print('주문을 마치겠습니다.')

```



```

주문할 콤보 번호와 수량을 계속 입력하세요!
0 주문 종료
1 올인원팩 6000원
2 투게더팩 7000원
3 트리오팩 8000원
4 패밀리팩 10000원
콤보번호 입력>>1
수량 입력>>10
올인원팩, 10개 주문
올인원팩 주문 가격 : 60000, 총 가격 60000
주문할 콤보 번호와 수량을 계속 입력하세요!
0 주문 종료
1 올인원팩 6000원
2 투게더팩 7000원
3 트리오팩 8000원
4 패밀리팩 10000원
콤보번호 입력>>2
수량 입력>>5
투게더팩, 5개 주문
투게더팩 주문 가격 : 35000, 총 가격 95000
주문할 콤보 번호와 수량을 계속 입력하세요!
0 주문 종료
1 올인원팩 6000원
2 투게더팩 7000원
3 트리오팩 8000원
4 패밀리팩 10000원
콤보번호 입력>>0

*****주문 종료*****
총 주문 가격 95000원
주문을 마치겠습니다.

```

키와 값의 나열인 딕셔너리와 집합

- 키와 값인 쌍의 나열인 딕셔너리
- 중복과 순서가 없는 집합
- 내장함수 `zip()`과 `enumerate()`, 시퀀스 간의 변환

키와 값이 쌍의 나열인 딕셔너리

- 딕셔너리: 키와 값의 쌍인 항목을 나열한 시퀀스이다
- 키와 값은 콜론으로 구분한다
- 항목 값으로 리스트나 튜플 등이 가능하다
- 키는 수정이 불가능하다

```
groupnumber = {'잔나비':5, '트와이스':9, '방탄소년단':7}  
print(groupnumber)  
print(type(groupnumber))
```

➡ {'잔나비': 5, '트와이스': 9, '방탄소년단': 7}
<class 'dict'>

```
abc = {}  
abc['수박'] = '과일';  
abc['수확일'] = [2020,6];  
print(abc)
```

➡ {'수박': '과일', '수확일': [2020, 6]}

키와 값이 쌍의 나열인 딕셔너리

- dict()를 이용한 딕셔너리 생성

```
example = dict((( '1', '일번' ), ( '2', '이번' )))  
print(example)  
example2 = dict([ [ '1', '일번' ], ( '2', '이번' ) ] )  
print(example2)
```



```
{ '1': '일번', '2': '이번' }  
{ '1': '일번', '2': '이번' }
```

```
example = dict(일번='1',이번='2',삼번='3')  
print(example)
```



키가 문자열일때만 가능하다



```
{ '일번': '1', '이번': '2', '삼번': '3' }
```

키와 값이 쌍의 나열인 딕셔너리

```
season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
```

```
print(season.keys())
```

 → 키로만 구성된 리스트 반환

```
print(season.items())
```

 → 키,값의 쌍으로 구성된 리스트 반환

```
print(season.values())
```

 → 값으로만 구성된 리스트 반환

```
print(season[봄])  
print(season['여름'])  
print(season['가을'])  
print(season['겨울'])
```

→ 첨자가 아니라 키값이 들어간다
0을 넣어도 0이라는 키가 없기 때문에 오류가난다



```
dict_keys(['봄', '여름', '가을', '겨울'])  
dict_items([('봄', 'spring'), ('여름', 'summer'), ('가을', 'autumn'), ('겨울', 'winter')])  
dict_values(['spring', 'summer', 'autumn', 'winter'])  
spring  
summer  
autumn  
winter
```


키와 값이 쌍의 나열인 딕셔너리

- `get()` :키의 해당 값을 반환

```
>>> example = {'일번': '1', '이번': '2', '삼번': '3'}  
>>> print(example.get('이번'))  
2
```

- `pop()` :키인 항목을 삭제하고,삭제되는 키의 값을 반환한다

```
>>> example = {'일번': '1', '이번': '2', '삼번': '3'}  
>>> print(example.pop('이번'))  
2  
>>> example  
{'일번': '1', '삼번': '3'}
```

- `popitem()` :임의의 (키,값)을 반환하고 삭제한다

```
>>> example = {'일번': '1', '이번': '2', '삼번': '3'}  
>>> example.popitem()  
( '삼번', '3' )  
>>> example  
{ '일번': '1', '이번': '2' }
```

키와 값이 쌍의 나열인 딕셔너리

- del : 딕셔너리 항목 삭제한다

```
>>> example = {'일번': '1', '이번': '2', '삼번': '3'}
>>> del example['일번']
>>> example
{'이번': '2', '삼번': '3'}
```

- clear(): 빈 딕셔너리로 만들어준다

```
>>> example = {'일번': '1', '이번': '2', '삼번': '3'}
>>> example.clear()
>>> example
{}

```

- update(): 인자인 다른 딕셔너리를 합병한다

```
example1 = {'일번': '1', '이번': '2', '삼번': '3'}
example2 = {'가': '하나', '나': '둘', '다': '셋'}
example1.update(example2)
print(example1)
print('다' in example1)
```

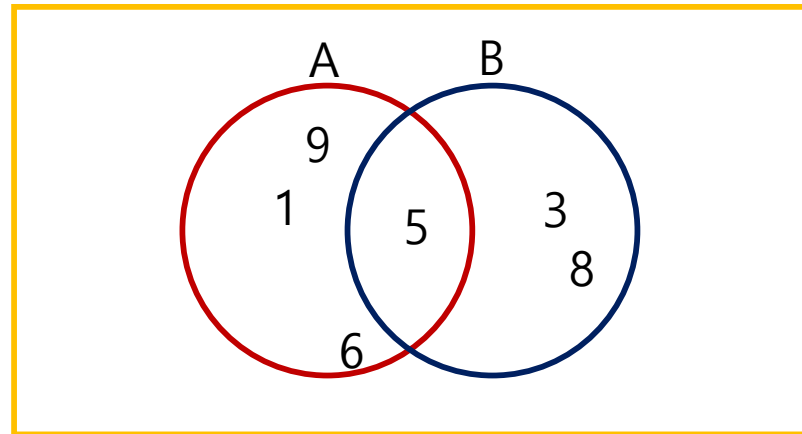


```
{'일번': '1', '이번': '2', '삼번': '3',
'가': '하나', '나': '둘', '다': '셋'}
True
```

in으로 딕셔너리에 키가 존재하는 지 검사한다

중복과 순서가 없는 집합

- 집합 : 수학에서의 집합과 비슷하다



- 중복 될 수 없다
- 순서는 의미가 없다
- 원소는 수정 불가능 한 것이어야 한다

중복과 순서가 없는 집합

- set(): 집합 생성

```
f = set()
print(type(f))
f = set([1, 'a', '가'])
print(f)
print(set('가나다')) → 각 문자가 원소인 집합이 된다
print(set([1, [2, 3]]))
```



```
<class 'set'>
{'가', 1, 'a'} → 순서는 상관없다
{'다', '가', '나'}
```

```
Traceback (most recent call last):
  File "C:\wpython\문자열연결.py", line 9, in <module>
    print(set([1, [2, 3]]))
TypeError: unhashable type: 'list'
```

수정 될 수 있는
리스트나 딕셔너리는 허용되지 않는다

중복과 순서가 없는 집합

```
f = {1,2,3,4,5,6,7}
f.add(8)
print(f)
f.discard(8)
print(f)
f.pop()
print(f)
f.clear()
print(f)
f.remove(8)
```



```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7}
{2, 3, 4, 5, 6, 7}
set()
```

```
Traceback (most recent call last):
  File "C:\python\문자열연결.py", line 13, in <module>
    f.remove(8)
KeyError: 8
```

- add(원소):집합에 원소를 추가한다
- discard(원소):집합에서 원소를 삭제한다
- pop():집합에서 원소를 임의로 삭제한다
- remove(원소):집합에서 원소를 삭제한다
- remove는 원소가 없으면 오류가 발생하고,
discard는 원소가 없어도 오류가 없다

중복과 순서가 없는 집합

- 합집합:

```
fx = {1,2,3,4}  
fy = {2,4,5,7}
```

```
print(fx|fy)
```

```
print(fx.union(fy))
```

```
fx.update(fy)
```

```
print(fx) → fx에 반영 됨
```

```
fx |= {6,8}
```

```
print(fx)
```

```
{1, 2, 3, 4, 5, 7}
```

```
{1, 2, 3, 4, 5, 7}
```

```
{1, 2, 3, 4}
```

```
{1, 2, 3, 4, 5, 7}
```

- 교집합:

```
fx = {1,2,3,4}
```

```
fy = {2,4,5,7}
```

```
print(fx&fy)
```

```
print(fx.intersection(fy))
```

```
print(fx)
```

```
fx.intersection_update(fy)
```

```
print(fx) → fx에 반영 됨
```

```
{2, 4}
```

```
{2, 4}
```

```
{1, 2, 3, 4}
```

```
{2, 4}
```

중복과 순서가 없는 집합

- 차집합: $fx = \{1, 2, 3, 4\}$
 $fy = \{2, 4, 5, 7\}$

```
print(fx-fy)
print(fx.difference(fy))
print(fy-fx)
print(fy.difference(fx))
```



```
{1, 3}
{1, 3}
{5, 7}
{5, 7}
```

- 여집합: $fx = \{1, 2, 3, 4\}$
 $fy = \{2, 4, 5, 7\}$

```
print(fx^fy)
print(fx.symmetric_difference(fy))
fx ^= fy
print(fx)
```



```
{1, 3, 5, 7}
{1, 3, 5, 7}
{1, 3, 5, 7}
```

내장함수와 시퀀스 변화

- 내장함수 zip(): 여러 개의 튜플 항목 시퀀스를 리스트로 묶어준다

```
>>> a=['가','나','다']
>>> b=[1,2,3,4]
>>> print(type(zip(a,b)))
<class 'zip'>
>>> list(zip(a,b))
[('가', 1), ('나', 2), ('다', 3)]
>>> tuple(zip(a,b))
(('가', 1), ('나', 2), ('다', 3))
```

```
>>> a=['가','나','다']
>>> b=[1,2,3]
>>> dict(zip(a,b))
{'가': 1, '나': 2, '다': 3}
>>> dict(zip('abcd','1234'))
{'a': '1', 'b': '2', 'c': '3', 'd': '4'}
>>> dict(zip(a,b,b))
Traceback (most recent call last):
  File "<pyshell#9>", line 1, in <module>
    dict(zip(a,b,b))
ValueError: dictionary update sequence element #0 has length 3; 2 is required
```

- 동일한 수로 이뤄진 튜플 항목 시퀀스를 묶어준다
- list의 인자로 사용하면 항목이 튜플인 리스트 생성
- tuple의 인자로 사용하면 항목이 튜플인 튜플 생성
- 짧은 쪽의 인자에 맞는 리스트 구성
- dict에서 zip의 인자 2개를 사용하면 딕셔너리 생성
- 2개가 아닐 경우 오류 발생

내장함수와 시퀀스 변화

- 내장함수 enumerate(): 0부터 시작하는 첨자와 항목 값의 튜플 생성

```
>>> ex = ['A', 'B', 'C', 'D']
>>> list(enumerate(ex))
[(0, 'A'), (1, 'B'), (2, 'C'), (3, 'D')]
>>> ex2 = 'example2'
>>> list(enumerate(ex2))
[(0, 'e'), (1, 'x'), (2, 'a'), (3, 'm'), (4, 'p'), (5, 'l'), (6, 'e'), (7, '2')]
>>> list(enumerate([10, 20, 30, 40], start=2))
[(2, 10), (3, 20), (4, 30), (5, 40)]
```

- start로 첨자의 시작을 정해 줄 수 있다

```
>>> sub = ['국어', '영어', '수학']
>>> for tp in enumerate(sub):
    print('list[{}]: {}'.format(tp[0], tp[1]))
    print('list[{}]: {}'.format(*tp))
```

```
list[0]: 국어
list[0]: 국어
list[1]: 영어
list[1]: 영어
list[2]: 수학
list[2]: 수학
```

- tp[0]은 첨자 tp[1]은 값
- *tp로 지정하면 자동으로 나뉘어 앞부분에는 첨자, 뒤에는 값이 출력된다

내장함수와 시퀀스 변환

• 시퀀스의 변환

```
word = '가', '나', '다', '라'
print(type(word))
print(type(list(word)))
fruit = ['수박', '귤', '복숭아', '사과']
print(type(tuple(fruit)))
```



```
<class 'tuple'>
<class 'list'>
<class 'tuple'>
```

```
>>> fruit = ['수박', '귤', '복숭아', '사과', '수박']
>>> print(set(fruit))
{'복숭아', '귤', '사과', '수박'}
```

- set은 리스트를 집합으로 변환
- 중복된 항목은 1개만 삽입
- 순서 무의미

```
fruit = dict(일번='수박',이번='귤',삼번='복숭아',사번='사과')
lfruit = list(fruit)
tfruit = tuple(fruit)
print(lfruit)
print(tfruit)
```



```
['일번', '이번', '삼번', '사번']
('일번', '이번', '삼번', '사번')
```

- 딕셔너리를 튜플이나 리스트로 변환하면 항목이 키로만 구성된 튜플과 리스트로 반환된다

키와 값의 나열인 딕셔너리와 집합

```
singer = ['BTS','불빨간사춘기','BTS','블랙핑크','태연','BTS']
song = ['작은 것들을 위한 시','나만 봄','소우주','Kill This Love','사계']

kpop = list(zip(singer,song))
print(kpop)
kpchart = dict(enumerate(kpop,start =1))

import pprint
pprint.pprint(kpchart)
```



```
[('BTS', '작은 것들을 위한 시'), ('불빨간사춘기', '나만 봄'), ('BTS', '소우주'),
 ('블랙핑크', 'Kill This Love'), ('태연', '사계')]
{1: ('BTS', '작은 것들을 위한 시'),
 2: ('불빨간사춘기', '나만 봄'),
 3: ('BTS', '소우주'),
 4: ('블랙핑크', 'Kill This Love'),
 5: ('태연', '사계')}
```

마무리

- 파이썬을 이해하고 활용 할 수 있게 되었다
- 연산을 계산 할 수 있게 되었다
- 간단한 게임 구현을 할 수 있게 되었다
- 주문프로그램을 작성 할 수 있게 되었다.