파이썬 포트폴리오

컴퓨터정보공학과 C반
20191787 이민경

목차

Part 1
파이썬 언어란?

파이썬의 특징 (1)

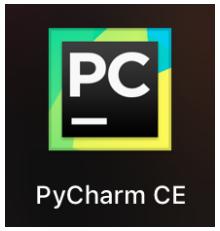
- 전 세계적으로 가장 많이 가르치는 프로그래밍 언어이다.
- 1991년 네덜란드 개발자 귀도 반 로섬이 만든 언어이다.
- 현재는 비영리 단체인 파이썬 소프트웨어 재단이 관리하고 있다.
- 기업의 실무에도 사용된다.
- 구글은 파이썬을 많이 사용하는 대표적인 기업으로 알려져 있다.
- 머신 러닝, 그래픽, 웹 개발 등 여러 업계에서 선호한다.
- 빅데이터 분석 처리 분야에 사용된다.
- 서버에서부터 라즈베리 파이까지 어떤 플랫폼에서도 사용이 가능하다.

파이썬의 특징 (2)

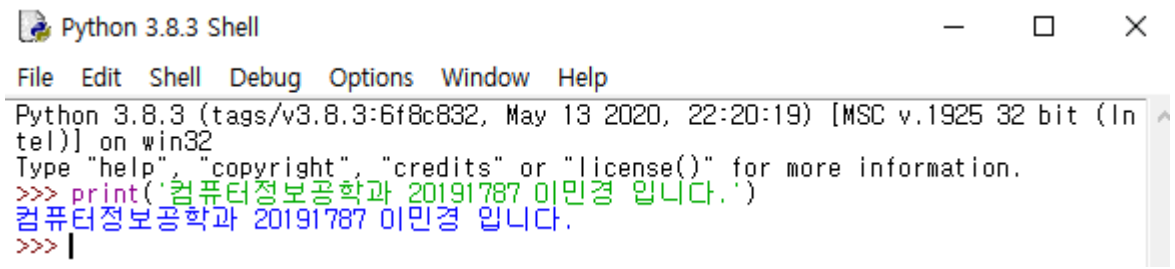
- 간결한 문법으로 입문자가 이해하기 쉽다.
- 대화형 모드(인터프리터 모드)를 지원한다.
- 다른 컴퓨터 언어(C, C++, JAVA 등)도 처리한다.
- 라이브러리가 풍부하고 다양한 개발 환경을 제공하고 있다.
- 모바일 앱 개발 환경에서는 사용하기 어렵다.
- 다른 언어에 비해 연산 속도가 떨어진다.
- 멀티 코어를 활용하기에 쉽지 않다.

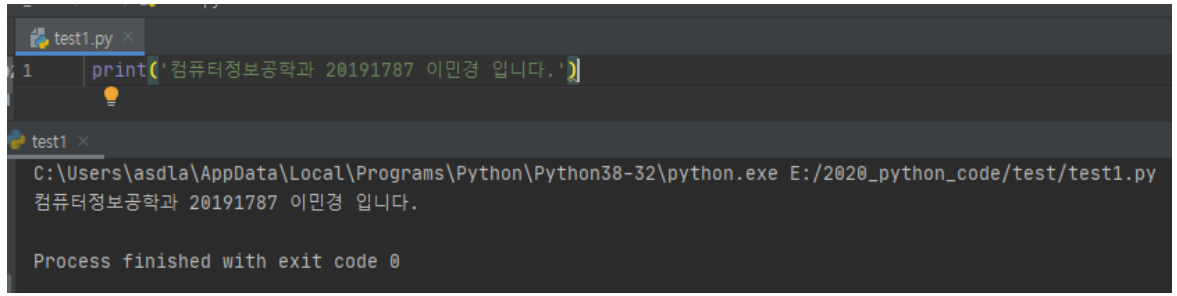
파이썬 개발도구

파이썬 쉘 IDLE



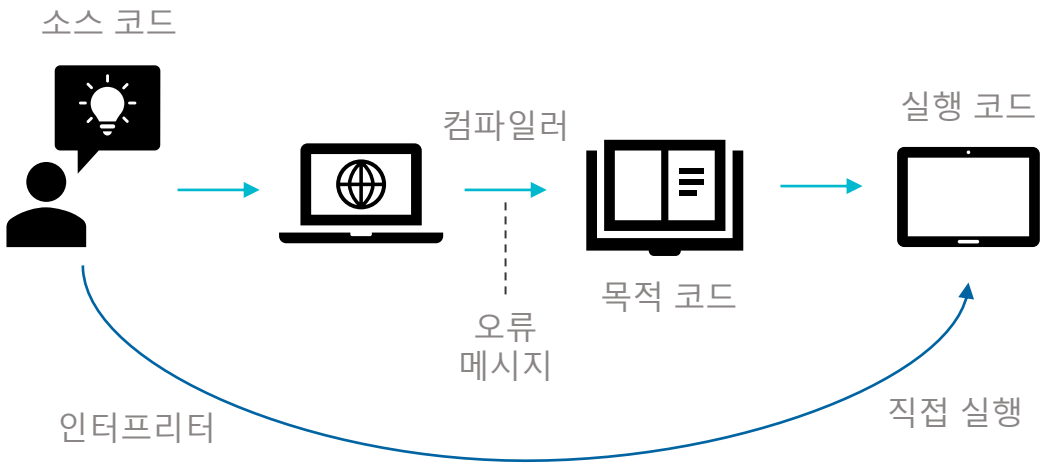
파이참(PyCharm)





인터프리터/컴파일러

"파이썬은 인터프리트 방식의 언어이다"



○ 인터프리터

- 사람이 이해할 수 있는 고급 언어로 작성된 코드를 한 단계씩 해석하여 실행 시키는 방법이다.
- 프로그램을 한 단계씩 기계어로 해석한다.

○ 컴파일러

- 고급 언어로 쓰인 프로그램을 그와 의미적으로 동등하며 컴퓨터에서 즉시 실행될 수 있는 형태의 목적 프로그램으로 바꾸어 주는 번역 프로그램이다.
- 자바와 C는 컴파일 방식의 언어이다.

Part 2

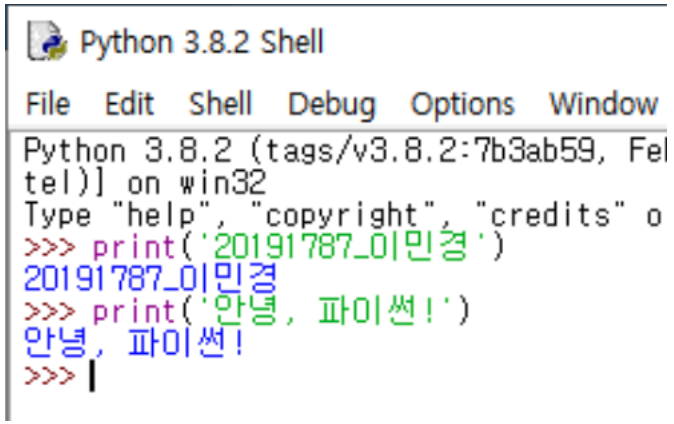
문자열/표준 입력/ 자료 변환 함수

문자열이란?

- 문자 하나 또는 문자가 모인 단어나 문장 또는 단락 등을 문자열(string)이라 한다.
- 작은따옴표나 큰따옴표로 둘러싸 'string' 또는 "string" 이라고 표현하므로, 따옴표로 둘러싸면 모두 문자열이라 할 수 있다.
- 작은따옴표로 둘러싼 문자열의 내부에는 작은따옴표를 문자로 사용할 수 없고, 큰따옴표도 이와 마찬가지이다.
- 문자 하나도 문자열로 취급하고, 따옴표로 둘러싼 숫자도 문자열로 취급한다.
- 문자열에서 따옴표는 앞뒤를 동일하게 사용해야 한다.

ex) "bts' (x), 'bts' (o)

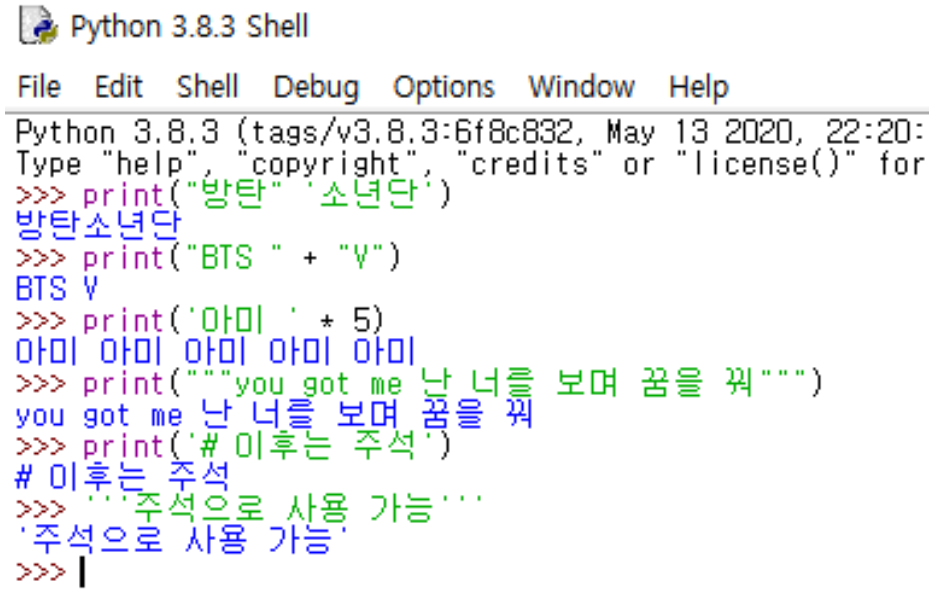
함수 print( )를 사용하여 문자열을 출력한 화면



함수 print( )란?
- 문자열이나 숫자 등의 자료를 콘솔에 출력하는 일을 수행한다.
- 출력 이후에 다음 줄로 이동하여 출력을 준비한다.

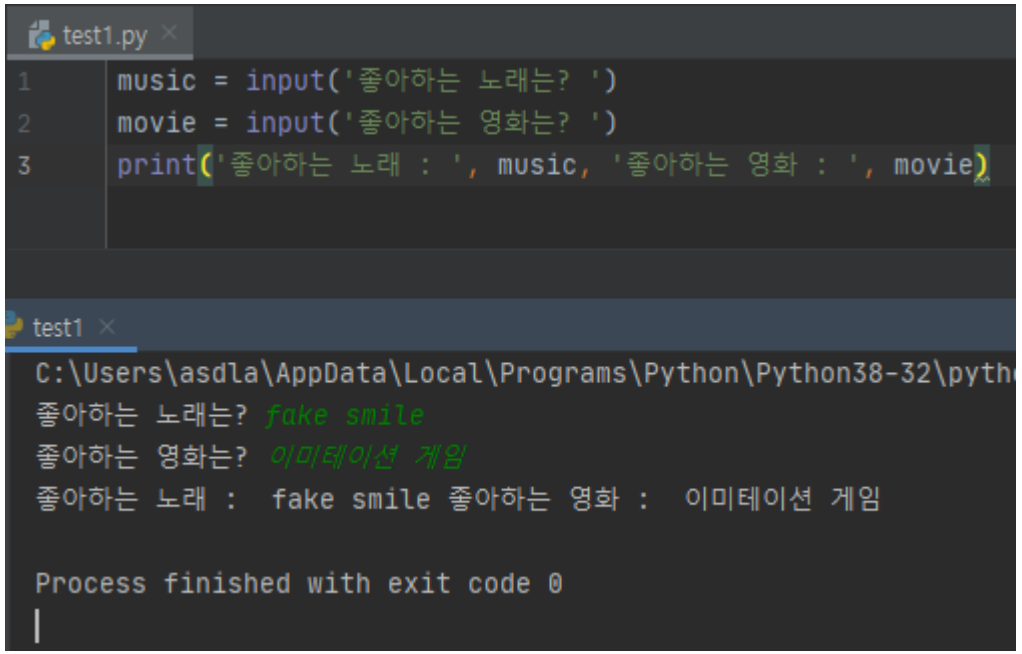
문자열 연산자/주석

- +는 문자열에서 문자열을 연결할 때 사용한다.
- *는 문자열에서 문자열을 지정된 수만큼 반복하는 연산을 수행한다. ex) 'hello' * 'hi' (x), 3 * 'hi'(o)
- *는 +보다 먼저 수행한다.
- 여러 줄에 걸쳐 문자열을 처리할 때, 작은따옴표나 큰따옴표를 연속적으로 3개씩 문자열 앞뒤에 둘러싸는 삼중 따옴표를 사용하기도 한다. (삼중따옴표는 주석이 아니다!)
- 파이썬 주석은 #으로 시작하고 그 줄의 끝까지 유효하다.



표준 입력 함수 input( )

- 프로그램 과정에서 쉘이나 콘솔에서 사용자의 입력을 받아 처리하는 방식을 표준 입력(standard input)이라 한다.
- 파이썬에서 기본적으로 입력을 받는 함수는 input( )함수이다.
- 값을 입력할 때까지 커서가 깜빡이면서 대기한다.
- 어떤 값을 입력하면 prompt에서 입력한 값이 작은따옴표 안에 나타난다.



자료 변환 함수 str( )

함수 str( )은 주로 정수와 실수를 문자열로 변환한다.

```
n1 = int(input("첫 번째 정수를 입력해주세요.:"))
n2 = int(input("두 번째 정수를 입력해주세요.:"))

sum = n1 + n2

print("두 수의 합은", sum, "입니다.")
```

```
첫 번째 정수를 입력해주세요.:123
두 번째 정수를 입력해주세요.:456
두 수의 합은 579 입니다.
```

→

```
n1 = int(input("첫 번째 정수를 입력해주세요.:"))
n2 = int(input("두 번째 정수를 입력해주세요.:"))

sum = n1 + n2

print("두 수의 합은", str(sum) + "입니다.")
```

```
첫 번째 정수를 입력해주세요.:123
두 번째 정수를 입력해주세요.:456
두 수의 합은 579입니다.
```

str( ) 함수를 사용하면 숫자를 문자로 변환시킬 수 있기 때문에, 위와 같이 579를 문자로 변환시켜 “579입니다.” 처럼 공백 없이 이어 적을 수 있다.

자료 변환 함수 int( )

함수 int( )는 정수 형태인 문자열을 정수,
실수를 정수로 변환한다.

```
n1 = input("첫 번째 정수를 입력해주세요.:")
n2 = input("두 번째 정수를 입력해주세요.:")

print(n1+n2)
```

```
첫 번째 정수를 입력해주세요.:123
두 번째 정수를 입력해주세요.:456
123456
```

```
n1 = int(input("첫 번째 정수를 입력해주세요.:"))
n2 = int(input("두 번째 정수를 입력해주세요.:"))

print(n1+n2)
```

```
첫 번째 정수를 입력해주세요.:123
두 번째 정수를 입력해주세요.:456
579
```

int( )함수는 정수 형태인 문자열을 정수로 변환시킬 수 있기 때문에, 위와 같이 입력 받은 n1과 n2를
int( )함수가 정수로 변환하여 123+456=579로 계산한다.

자료 변환 함수 float( )

함수 float( )는 소수점이 있는 실수 형태의 문자열을 실수, 정수를 실수로 변환한다.

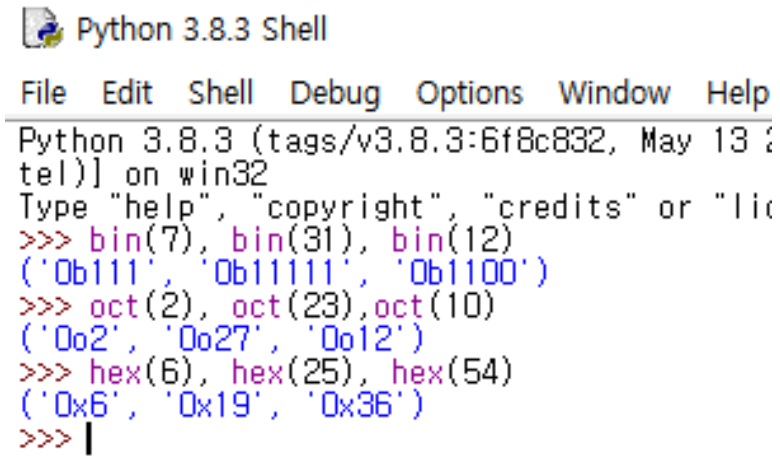
```
n1 = float(input("첫 번째 정수를 입력해주세요.:"))
n2 = float(input("두 번째 정수를 입력해주세요.:"))

print(n1+n2)
```

```
첫 번째 정수를 입력해주세요.:123
두 번째 정수를 입력해주세요.:456
579.0
```

float( )함수는 정수를 실수로 변환시키기 때문에, 123+456=579.0로 계산한다.

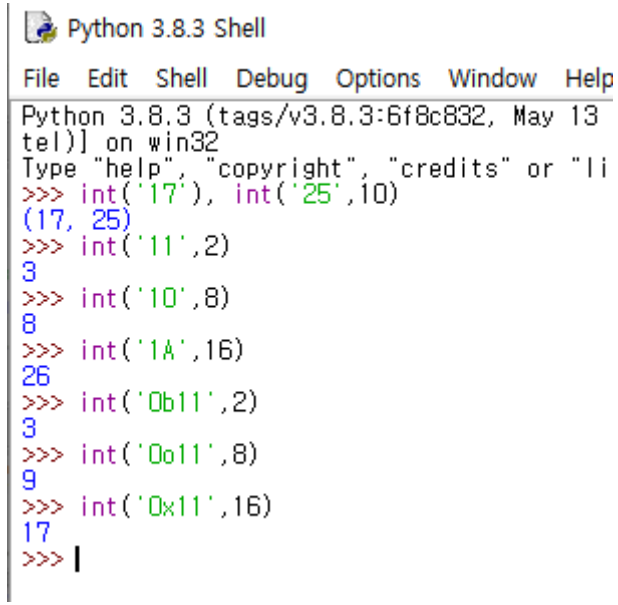
16/8/2진수 상수 표현 및 변환 함수 (1)

- 16진법 표현
  - 0x1f, 0X1E, 0Xa
  - hex( ) 함수를 통해 10진수를 16진수로 출력한다.
- 8진법 표현
  - 0o17, 0O16, 0O12
  - oct( ) 함수를 통해 10진수를 8진수로 출력한다.
- 2진법 표현
  - 0b11, 0B10, 0B1010
  - bin( ) 함수를 통해 10진수를 2진수로 출력한다.



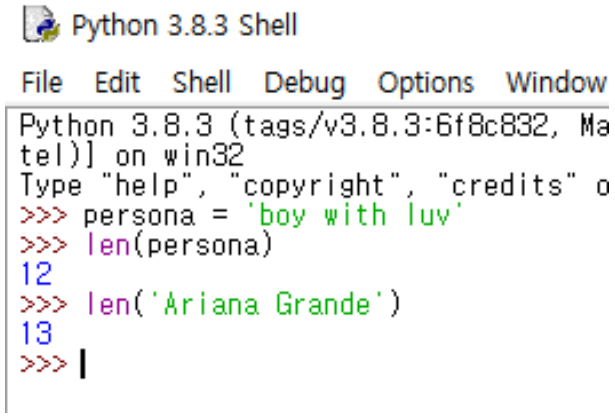
16/8/2진수 상수 표현 및 변환 함수 (2)

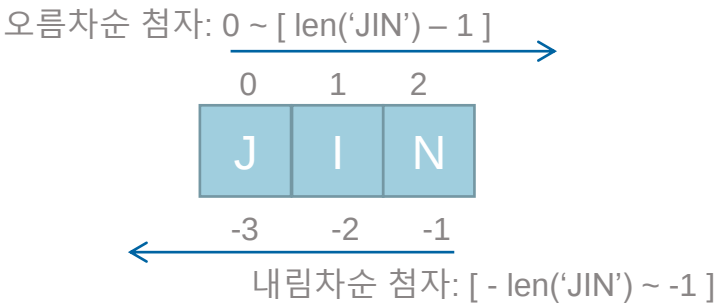
- int('strnum') 또는 int('strnum',10)은 10진수 형태의 문자열을 10진수 정수로 변환한다.
- int('strnum',n)을 사용할 때, n은 변환하려는 문자열 진수의 숫자를 2,8,10,16 등으로 넣는다.
- int('strnum',0)을 사용하면 strnum의 맨 앞이 0b, 0o, 0x에 따라 각각의 문자열을 자동으로 2진수, 8진수, 16진수로 인지해 10진수로 변환한다.



문자열 str 클래스/문자 참조

- 파이썬에서 문자열은 '문자의 나열'로, 텍스트 시퀀스(text sequence)라고도 한다.
- 문자열의 자료형은 class str 이기 때문에, 'JIN'과 같은 문자열 상수는 클래스 str의 객체이다.

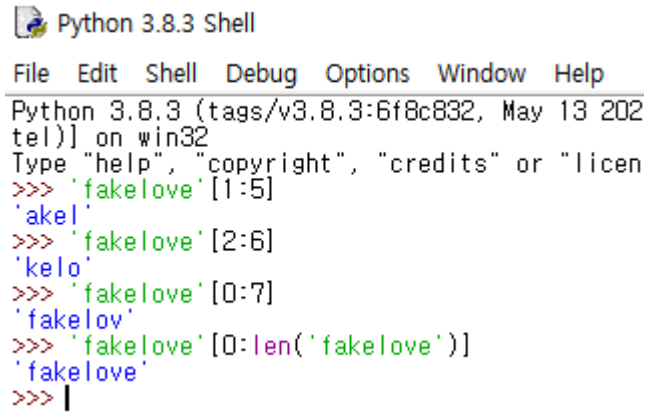




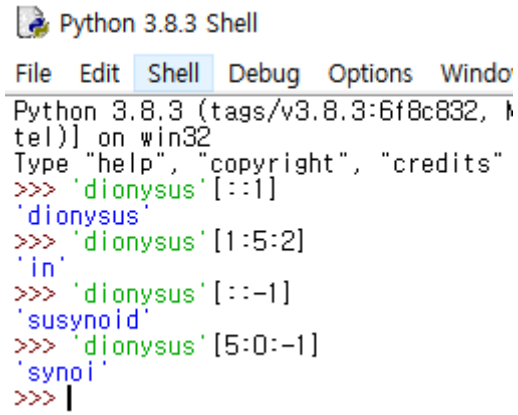
- 문자열을 구성하는 문자는 0부터 시작되는 첨자를 대괄호 안에 기술해 참조가 가능하다.
- -1부터 시작해서 -2, -3으로 작아지는 첨자도 역순으로 참조한다.
- 첨자가 유효 범위를 벗어나게 되면 에러가 발생한다.

문자열의 부분 문자열 참조 방식

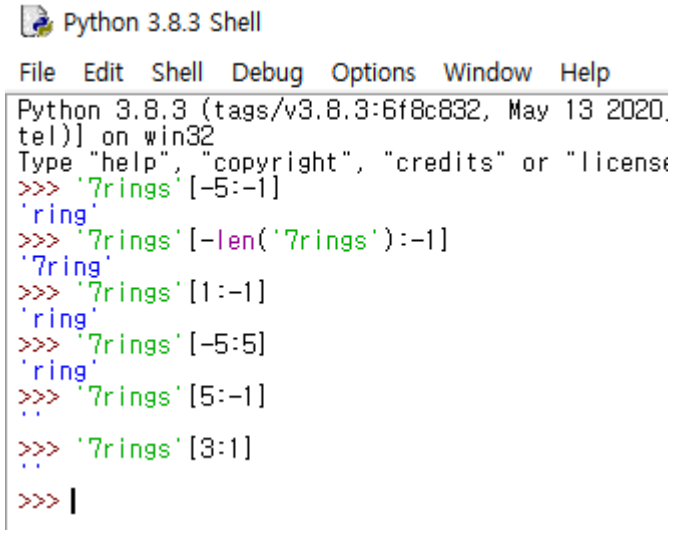
- 0과 양수를 이용한 문자열 슬라이싱



- str[start : end : step]로 문자 사이의 간격을 step으로 조정

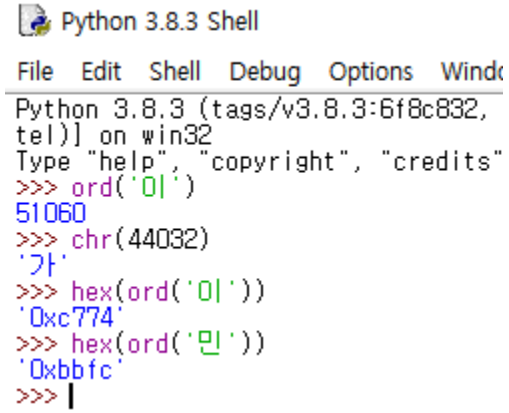


- 음수를 이용한 문자열 슬라이싱

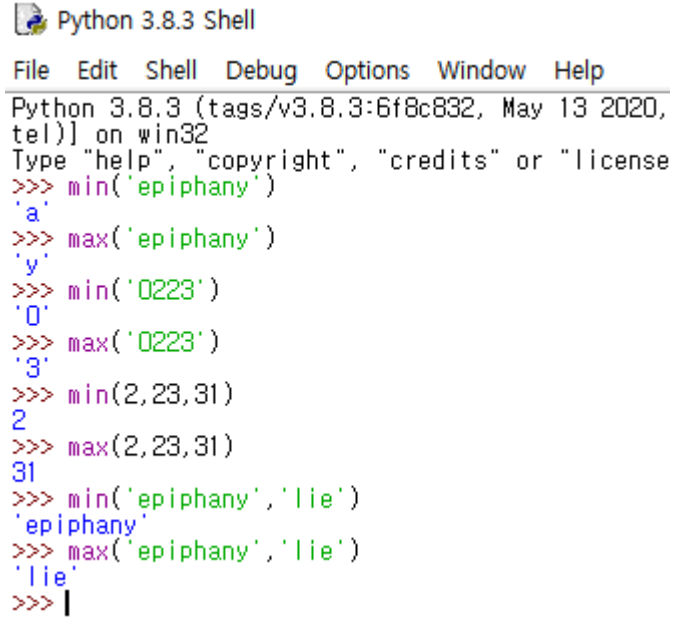


문자 함수

함수 ord( )는 문자의 유니코드 번호를 반환하고, 함수 chr( )는 유니코드 번호를 문자로 반환한다.



함수 max( ) / min( )는 인자의 최댓값과 최솟값을 반환하는 함수이다.



문자열 메소드 (1)

○ 문자열을 바꿔 반환하는 메소드 replace( )

- 메소드 replace(apple, mango, count)는 문자열 apple을 mango로 대체하는데, 옵션이 count는 대체 횟수를 지정한다.
- count가 없으면 모두 바꾸고, 있으면 앞에서부터 지정 횟수만큼 바꾼다.

```
letters = 'abcdefghijklmnopqrstuvwxyz'
```

```
new_letters = letters.replace('z', 'q')
print(letters)
print(new_letters)
```

```
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyq
```

new_letters는 letters에서 z를 q로 대체한다.

○ 부분 문자열 출현 횟수를 반환하는 메소드 count( )

```
poem = '''So "it is" quite different, then, if in a mountain town
the mountains are close, rather than far. Here

they are far, their distance away established,
consistent year to year, like a parent's

or sibling's. They have their own music.
So I confess I do not know what it's like,

listening to mountains up close, like a lover,
the silence of them known, not guessed at.'''
```

```
poem.count('if')
```

```
1
```

```
poem.count('like')
```

```
3
```

```
poem.count('apple')
```

```
0
```

문자열 메소드 (2)

- 문자와 문자 사이에 문자열을 삽입하는 메소드 join( )

```
a = ['a','b','c','d']
text = ''.join(a)
text2 = ','.join(a)
print(text)
print(text2)
```

```
abcd
a,b,c,d
```

text는 a b c d 사이에 공백을 제거하고, text2는 a b c d 사이에 , 을 추가한다.

- 문자열을 여러 문자열로 나누는 split( ) 메소드

```
>>> text2=" 문자열 양 옆으로 공백을 두도록 하겠습니다 "

>>> text2.split()
['문자열', '양', '옆으로', '공백을', '두도록', '하겠습니다']
>>> text2.split(' ')

['', '문자열', '양', '옆으로', '공백을', '두도록', '하겠습니다', '']
```

split(‘ ’)은 양 옆에 위치한 공백까지 리스트에 추가시킨다.

문자열 메소드 (3)

- 문자열을 찾는 메소드 find( )와 index( )

```
>>> seven = 'on 진구 시차 innerchild'
>>> seven.find('on')
0
>>> seven.index('on')
0
>>> seven.find('moon')
-1
>>> seven.index('moon')
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    seven.index('moon')
ValueError: substring not found
>>>
```

- 폭을 지정하고 중앙에 문자열 배치하는 메소드 center( )

```
>>> ' 아리아나 그란데 '.center(30, '*')
'********** 아리아나 그란데 **********'
>>> ' 아리아나 그란데 '.center(30)
'           아리아나 그란데           '
>>> ' 아리아나 그란데 '.center(30, '_')
'__________ 아리아나 그란데 __________'
>>>
```

- 폭을 지정하고 왼쪽 또는 오른쪽 정렬하는 메소드 ljust( )와 rjust( )

```
>>> '이민경'.ljust(10)
'이민경      '
>>> '이민경'.ljust(10, '*')
'이민경*******'
>>>
```

- 문자열 앞뒤의 특정 문자들을 제거하는 strip( ) 메소드

```
>>> '   sweetener   '.strip()
'sweetener'
>>> '   sweetener   '.lstrip()
'sweetener   '
>>> '   sweetener   '.rstrip()
'   sweetener'
>>>
```

- 제로 0을 채워 넣는 zfill( ) 메소드

```
>>> '223'.zfill(4)
'0223'
>>> '+1231'.zfill(8)
'+0001231'
>>>
```

문자열 메소드 (4)

○ 문자열의 format( ) 메소드

```
# 34, 56, 34
"{1} {2} {1}".format(12, 34, 56)
```

```
'34 56 34'
```

순서를 지정해서
format( ) 함수 사용

```
"{0} {0} {1} {1}".format(12, 34, 56)
```

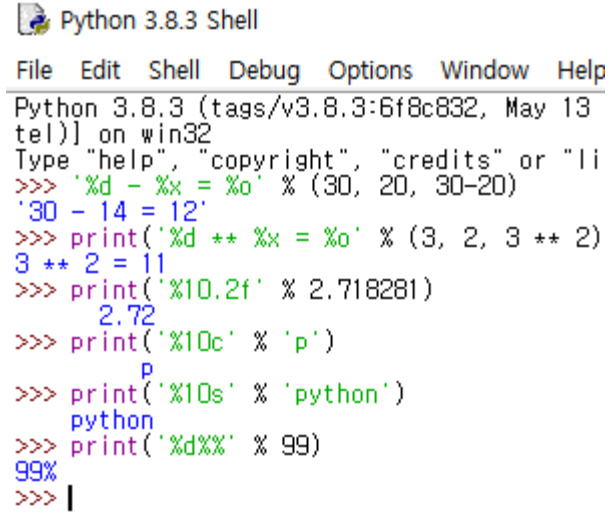
```
'12 12 34 34'
```

```
"{} / {} = {:2.2f}".format(2, 5, 2/5)
```

```
'2 / 5 = 0.40'
```

{:2.2f}에서 :2의 의미는 결과값의 정수 자리에 2칸을 할당하라는 의미이고, ".2f "의 의미는 소수점 둘째 자리까지 표현하라는 의미이므로, 결과값은 "0.40"이다.

○ %d와 %f를 사용한 출력 처리



Part 3
변수와 다양한 연산

정수와 실수의 다양한 연산 (1)

- 산술 연산자의 계산 우선순위

연산자	명칭	의미	우선순위	ex
**	거듭제곱, 지수승	왼쪽을 오른쪽 피연산자로 거듭제곱	1	2 ** 7 = 2^7
//	몫 나누기	왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수	3	10 // 3 = 3
%	나머지	왼쪽을 오른쪽 피연산자로 나눈 나머지	3	22 % 5 = 2
/	나누기	왼쪽을 오른쪽 피연산자로 나누기	3	10 / 4 = 2.5
*	곱하기	두 피연산자 곱하기	3	3 * 8
+	더하기	두 피연산자를 더하거나 수의 부호	4, 2	7+5, +2
-	빼기	두 피연산자를 빼거나 수의 부호	4, 2	9-2, -7

정수와 실수의 다양한 연산 (2)

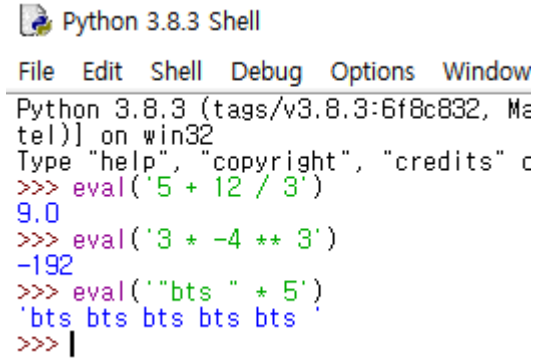
○ 언더스코어 (_)

- 대화형 모드에서 마지막에 실행된 결과값은 특별한 저장 공간인 _에 대입된다.

```
>>> 10
10
>>> _
10
>>> _ * 3
30
>>> _ * 20
600
```

_ 에는 마지막에 실행된 결과값이 저장되어 있지만, 파일 실행에서는 사용할 수 없고 대화형 모드에서만 사용할 수 있다.

○ 표현식 문자열 실행 함수 eval( )



변수와 대입 연산자 (1)

○ 변수 : 변하는 자료를 저장하는 메모리 공간이다.

ex) a = 7 , a = b = c = 3

(파이썬의 =은 수학에서 쓰이는 =과는 다르다.)

○ 키워드 : 프로그래밍 언어 문법에서 사용하는 이미 예약된 단어로 파이썬에서 사용하는 키워드의 개수는 약 33개 이다.

ex) False, None, while, elif, def, raise, return 등

○ 변수 이름을 붙일 때의 규칙

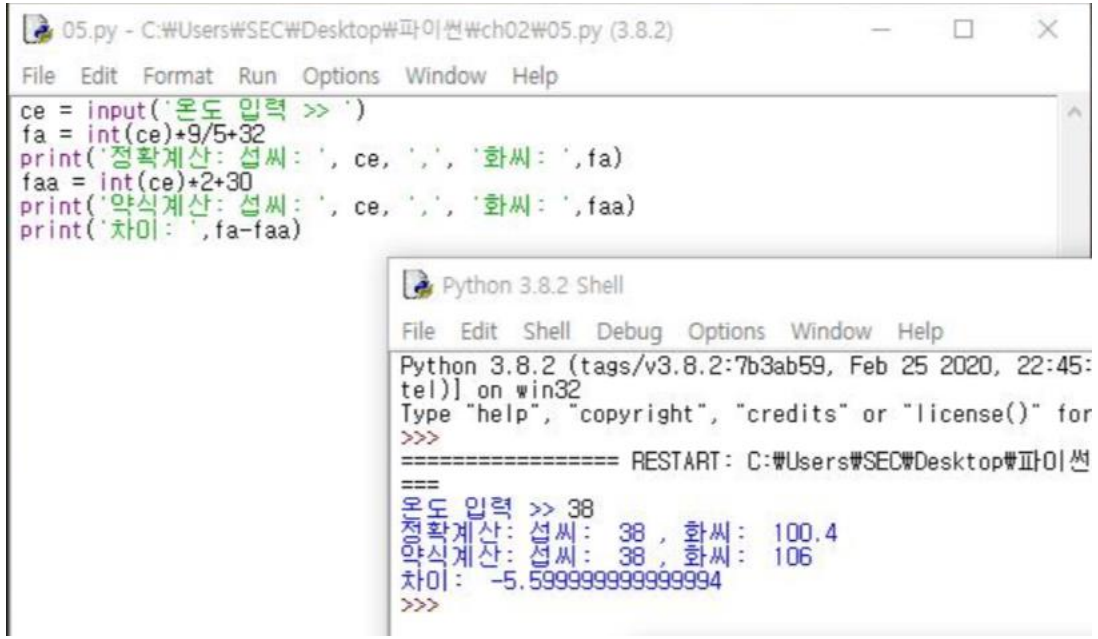
- 문자는 영문자, 숫자, _를 사용할 수 있고, 대소문자는 구별된다.
- 숫자는 맨 앞에 올 수 없다.
- 키워드는 사용할 수 없다.

○ 다양한 대입 연산자

대입 연산자	형태	의미
=	a=b	a=b : b의 결과값을 변수 a에 저장
+=	a+=b	a=a+b : a+b의 결과값을 변수 a에 저장
-=	a-=b	a=a-b : a-b의 결과값을 변수 a에 저장
*=	a*=b	a=a*b : a*b의 결과값을 변수 a에 저장
/=	a/=b	a=a/b : a/b의 결과값을 변수 a에 저장
%=	a%=b	a=a%b : a%b의 결과값을 변수 a에 저장
//=	a//=b	a=a//b : a//b의 결과값을 변수 a에 저장
**=	a**b	a=a**b : a**b의 결과값을 변수 a에 저장

변수와 대입 연산자 (2)

- 섭씨 온도와 화씨 온도의 관계 및 변환 예제



- 함수 divmod(a,b)는 연산자 //와 %를 함께 수행해 2개의 결과를 반환

```
In [1]: divmod(27,4)
Out[1]: (6, 3)

In [2]: 27//4          #몫
Out[2]: 6

In [3]: 27 % 4         #나머지
Out[3]: 3
```

논리 값과 논리 연산 (1)

○ 논리 유형 bool과 함수 bool( )

- 파이썬은 논리 값으로 True와 False를 제공한다.
- 정수의 0, 실수의 0.0, 빈 문자열 ‘’ 등은 False이고, 음수와 양수, 뭔가가 있는 ‘DNA’ 등의 문자열은 True 이다.

```
>>> bool(0), bool(0.0), bool(' ')
(False, False, False)
>>> bool(10), bool(3.14), bool('bts')
(True, True, True)
```

○ 논리곱과 논리합 연산자 and와 or

- and와 & 연산자는 두 항이 모두 참이어야 True이고, 하나라도 거짓이면 False이다.
- Or과 | 연산자는 두 항이 모두 거짓이어야 False이고, 하나라도 참이면 True이다.

```
>>> True & True, True and True
(True, True)
>>> True & False, True and False
(False, False)
```

```
>>> True | True, True or True
(True, True)
>>> True | False, True or False
(True, True)
```

논리 값과 논리 연산 (2)

○ 배타적 논리합 연산자 ^와 not 연산자

- 배타적 논리합 연산자인 ^은 두 항이 다르면 True, 같으면 False이다.
- 연산자 not은 뒤에 위치한 논리 값을 바꾼다.

```
>>> True ^ True
False
>>> True ^ False
True
>>> not True , not False
(False, True)
```

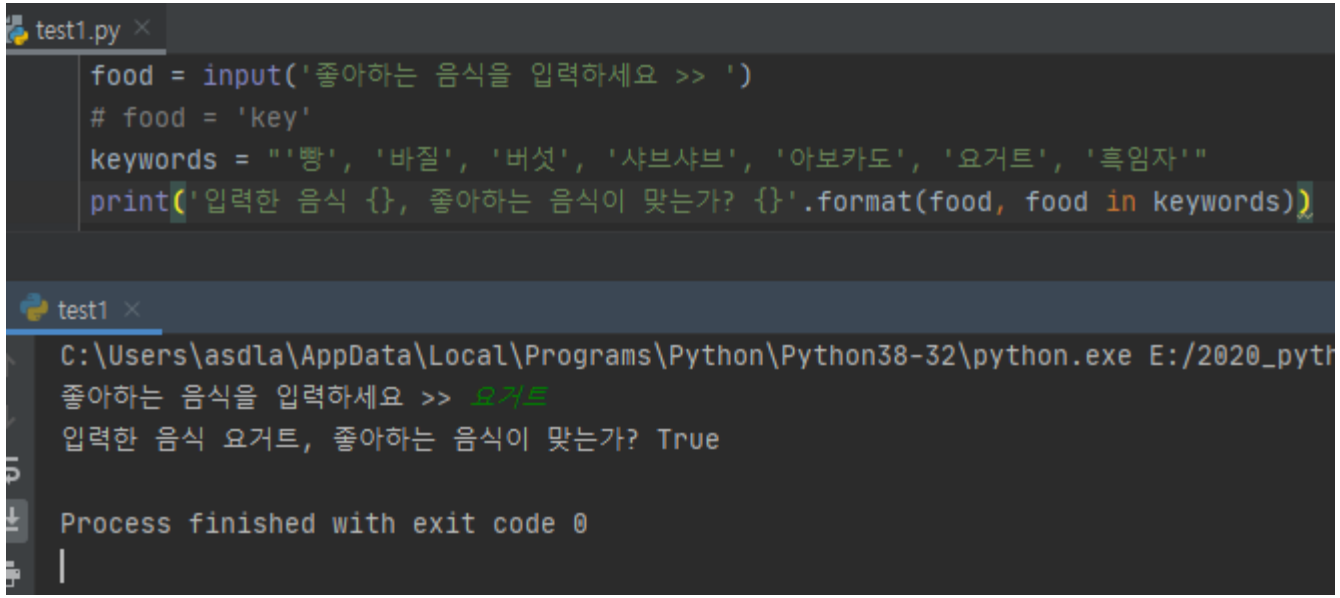
○ 논리 연산 우선순위 not / and / or

- 논리 연산은 not 연산, 논리곱 and, 논리합 or 순이다.

```
>>> not False and True
True
>>> not True or True and False
False
```

멤버십 연산 in

- 파이썬 키워드인 in은 멤버의 소속을 알 수 있는 멤버십 연산으로, 결과는 True, False의 논리 값이다.
- x in s : 문자열 s에 부분 문자열 x가 있으면 True, 없으면 False
- x not in s : 문자열 s에 부분 문자열 x가 없으면 True, 있으면 False



비트 배타적 논리합 ^을 사용해 암호화

- a ^ a == 0, a ^ 0 == a, a ^ a == ~a
- a ^ b == b ^ a, (a ^ b) ^ c == a ^ (b ^ c)
- (a ^ b) ^ b == a ^ (b ^ b) == a ^ 0 == a

- 세 번째 식을 활용하면 ^연산으로 간단한 암호화에 사용할 수 있다.
- 정수 a를 key와 연산식 a^key한 결과를 암호화 결과로 저장한 후, 필요 시 다시 이 저장된 암호화 결과인 a^key를 사용해 다시 a^key^key하면 원래 a로 복호화 할 수 있다.

관련 예제

03-12encryption.py | 비트 배타적 논리합 ^으로 ID 암호화 | 난이도 응용

```
orgPwd = int(input('ID로 사용할 여덟자리의 정수를 입력하세요 >> '))
keyMask = 27182818 # 키로 사용할 정수 하나를 저장
encPwd = orgPwd ^ keyMask # ID를 암호화시켜 저장
print('입력한 ID: %d' % orgPwd)
print('암호화해 저장된 ID: %d' % encPwd)
inPwd = int(input('로그인할 ID를 입력하세요 >> '))
result = encPwd ^ keyMask # 키로 암호화된 것을 복호화
print('로그인 성공: {}'.format(inPwd == result))
```

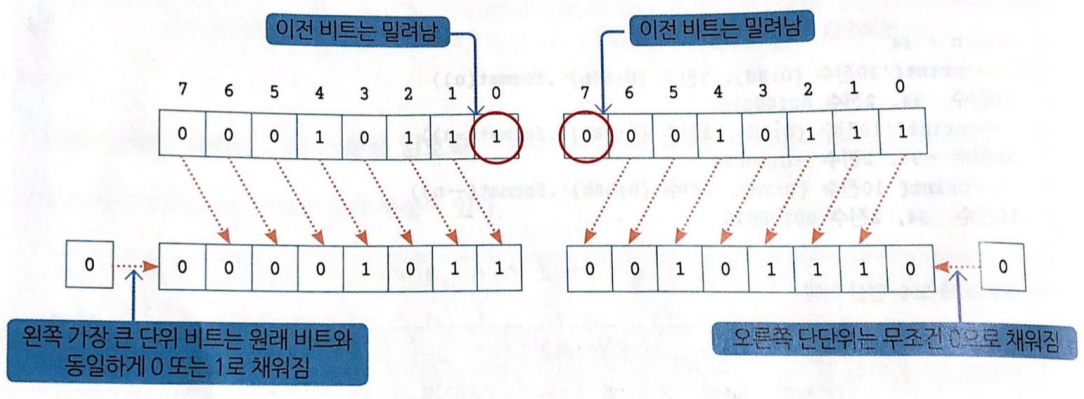
```
ID로 사용할 여덟자리의 정수를 입력하세요
>> 87652877
입력한 ID: 87652877
암호화해 저장된 ID: 78101743
로그인할 ID를 입력하세요 >> 87652877
로그인 성공: True
```

```
ID로 사용할 여덟자리의 정수를 입력하세요
>> 45678298
입력한 ID: 45678298
암호화해 저장된 ID: 52836408
로그인할 ID를 입력하세요 >> 45679299
로그인 성공: False
```

비트 이동 연산

○ 비트 이동 연산자 >>와 <<

- 비트 이동 연산자 >>와 <<는 연산자의 방향인 오른쪽 또는 왼쪽으로, 비트 단위로 뒤 피연산자인 지정된 횟수만큼 이동시킨다.



○ 연산자 우선순위

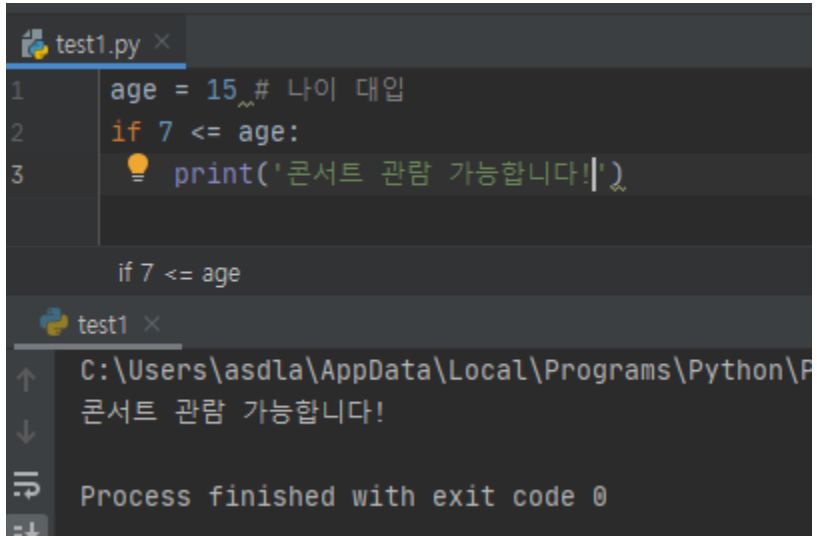
순위	연산자	설명	부류
1	**	지수승	지수 연산
2	~	비트 보수	단항 연산
3	+x, -x	부호	
4	* / // %	곱, 나누기, 몫, 나머지	산술 연산
5	+ -	더하기, 빼기	
6	>> <<	비트 이동	비트 연산
7	&	비트 and	
8	^	비트 xor	
9	|	비트 or	
10	< <= > >= == != not in is is not	관계, 소속, id 비교	관계 소속

Part 4

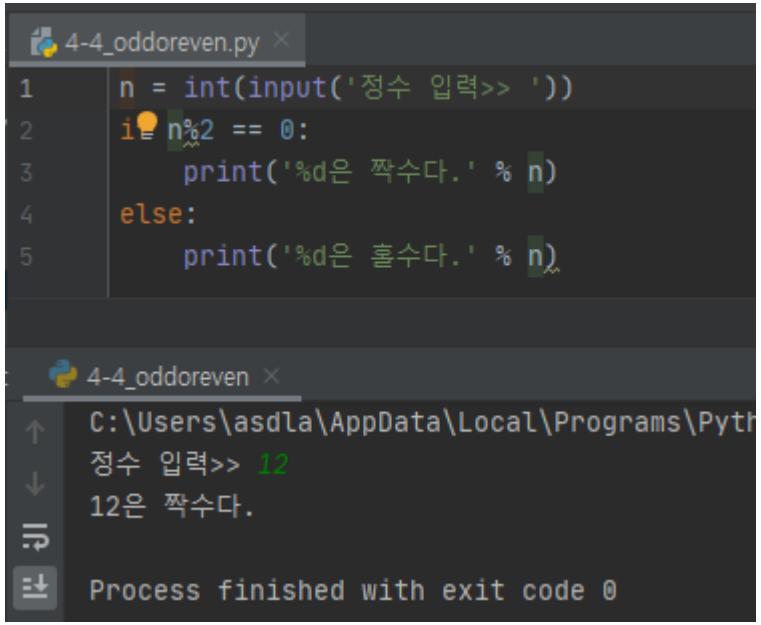
조건문/반복문/제어문

if문 / if ~ else문

- 만 7세 이상이면 콘서트 관람 가능



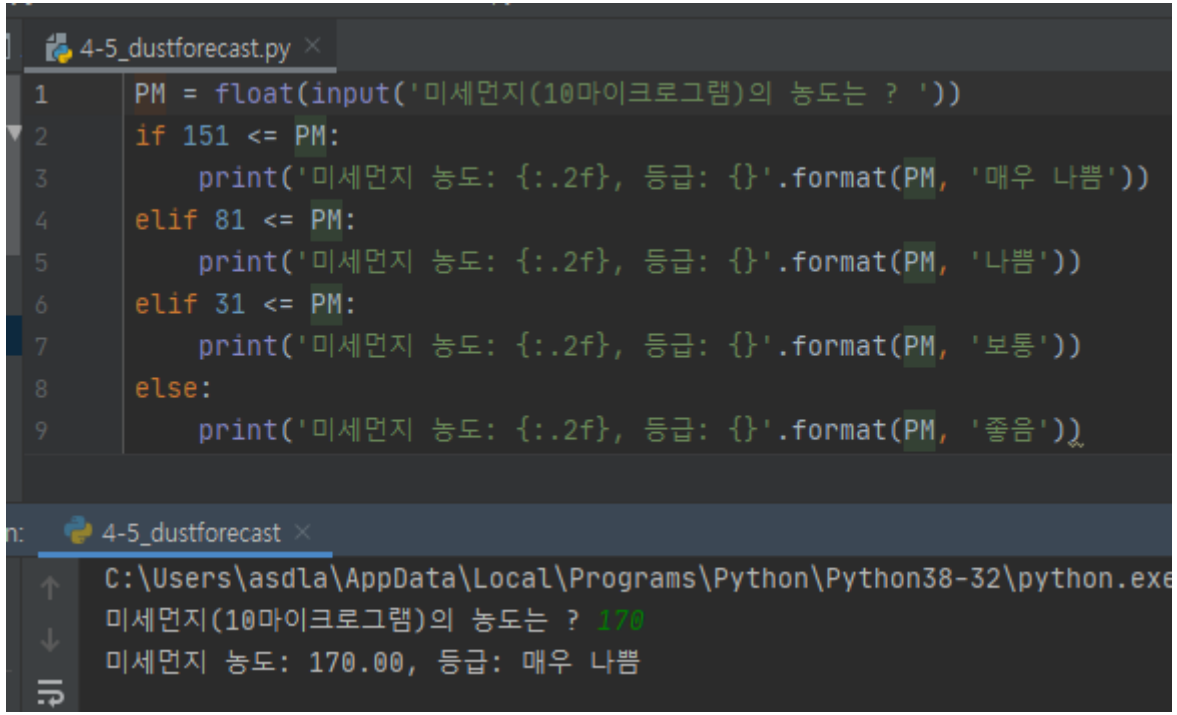
- 나머지 연산자 %를 사용해 정수의 홀수와 짝수 판별



if ~ else문에서 else: 블록은 논리 표현식 결과가 False일 때 실행된다.

if ~ elif문

- 미세먼지 농도에 따른 미세먼지 경고 예보



if ~ elif문은 여러 경로 중에서 하나를 선택하는 것인데, 논리 표현식 1이 True이면 문장1, 문장2의 블록을 실행하고 종료된다. 논리 표현식 1이 False여야 논리 표현식 2로 진행되고, 그 이후도 마찬가지이다. elif는 필요한 만큼 늘릴 수 있고, 마지막 else는 생략이 가능하다.

중첩된 조건문

- 커피와 주스의 메뉴 선택

커피, 주스와 같은 음료의 부류를 선택한 후에 다시 구체적인 음료를 선택하는 것을 구현하려면 옆의 예제와 같이 if문의 내부에 다시 if문을 사용해야 한다.

4-6_nestedmenu.py

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

if category == 1:
    menu = int(input('번호 선택? 1. 아메리카노 2. 카페라떼 3. 카푸치노 '))
    if menu == 1:
        print('1. 아메리카노 선택')
    elif menu == 2:
        print('2. 카페라떼 선택')
    elif menu == 3:
        print('3. 카푸치노 선택')
else:
    menu = int(input('번호 선택? 1. 키위주스 2. 토마토주스 3. 오렌지주스 '))
    if menu == 1:
        print('1. 키위주스 선택')
    elif menu == 2:
        print('2. 토마토주스 선택')
    elif menu == 3:
        print('3. 오렌지주스 선택')
```

4-6_nestedmenu

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020
원하는 음료는? 1. 커피 2. 주스 1
번호 선택? 1. 아메리카노 2. 카페라떼 3. 카푸치노 2
2. 카페라떼 선택

Process finished with exit code 0
```

반복을 실행하는 for문 (1)

- 여러 개의 값을 갖는 시퀀스에서 변수에 하나의 값을 순서대로 할당한다.
- 할당된 변수 값을 갖고 블록의 문장들(문장1,문장2)를 순차적으로 실행한다.
- 반복 몸체인 문장1, 문장2에서 변수를 사용할 수 있다.
- 시퀀스의 그 다음 값을 변수에 할당해 다시 반복 블록을 실행한다.
- 이러한 과정을 시퀀스의 마지막 항목까지 수행한다.
- 시퀀스의 마지막 항목까지 실행한 후 선택 사항인 else: 블록을 실행하고 반복을 종료한다.

```
for 변수 in <시퀀스> :
문장1
문장2
else :
```

반복을 실행하는 for문 (2)

- 시퀀스에서 for 구문으로 섭씨 온도에서 3도씩 증가하면서 화씨로 변환하여 출력

```
c=20
for i in 0,3,3,3,3,3,3,3:
    c+=i
    f=c*9/5+32
    ff=c*2+30
    print("섭씨: ",c," 화씨: ",f," 화씨(약식): ",ff," 차이: {0:.2f}".format(ff-f))
else:
    print()
```

```
Python 3.8.2 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:/Users/SEC/Desktop/파이썬/ch04/5.py ==========
===
섭씨:  20  화씨:  68.0  화씨(약식):  70  차이: 2.00
섭씨:  23  화씨:  73.4  화씨(약식):  76  차이: 2.60
섭씨:  26  화씨:  78.8  화씨(약식):  82  차이: 3.20
섭씨:  29  화씨:  84.2  화씨(약식):  88  차이: 3.80
섭씨:  32  화씨:  89.6  화씨(약식):  94  차이: 4.40
섭씨:  35  화씨:  95.0  화씨(약식):  100  차이: 5.00
섭씨:  38  화씨:  100.4  화씨(약식):  106  차이: 5.60
섭씨:  41  화씨:  105.8  화씨(약식):  112  차이: 6.20
```

반복 구조가 간단한 while문 (1)

- 논리 표현식이 True이면 반복 몸체인 문장1, 문장2를 실행한 후 다시 논리 표현식을 검사해 실행한다.
- 논리 표현식이 False이면 반복 몸체를 실행하지 않고, 선택 사항인 else : 이후의 블록을 실행한 후 반복을 종료한다.

```
while 논리 표현식 :
    문장1
    문장2
else :
    문장3
```

반복 구조가 간단한 while문 (2)

- 어린이를 위한 놀이 기구 탑승 처리

논리 변수 more로 while 반복의 여부를 결정하고, 키가 130 미만이면 현재 탑승한 인원이 저장된 변수 cnt를 증가시키고, 탑승 인원 cnt가 최대 인원 4명이 되면 while문의 조건 검사인 논리 변수 more를 False로 저장해 빠져나오면서 else: 블록에서 다음 번에 이용하라는 메시지를 출력

4-10_checkrides.py

```
MAXNUM = 4
MAXHEIGHT = 130

more = True
cnt = 0
while more:
    height = float(input("키는 ? "))
    if height < MAXHEIGHT:
        cnt += 1
        print('들어가세요.', '%d명' % cnt)
    else:
        print('커서 못 들어갑니다.')
    if cnt == MAXNUM:
        more = False
else:
    print('%d명 모두 찼습니다. 다음 번에 이용하세요' % cnt)
```

4-10_checkrides

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\py
키는 ? 120
들어가세요. 1명
키는 ? 111
들어가세요. 2명
키는 ? 150
커서 못 들어갑니다.
키는 ? 125
들어가세요. 3명
키는 ? 102
들어가세요. 4명
4명 모두 찼습니다. 다음 번에 이용하세요

Process finished with exit code 0
```

임의의 수를 발생하는 난수

- 파이썬에서는 모듈 random의 함수 randint(시작, 끝)를 사용해 정수 시작과 끝 수 사이에서 임의의 정수를 얻을 수 있다.

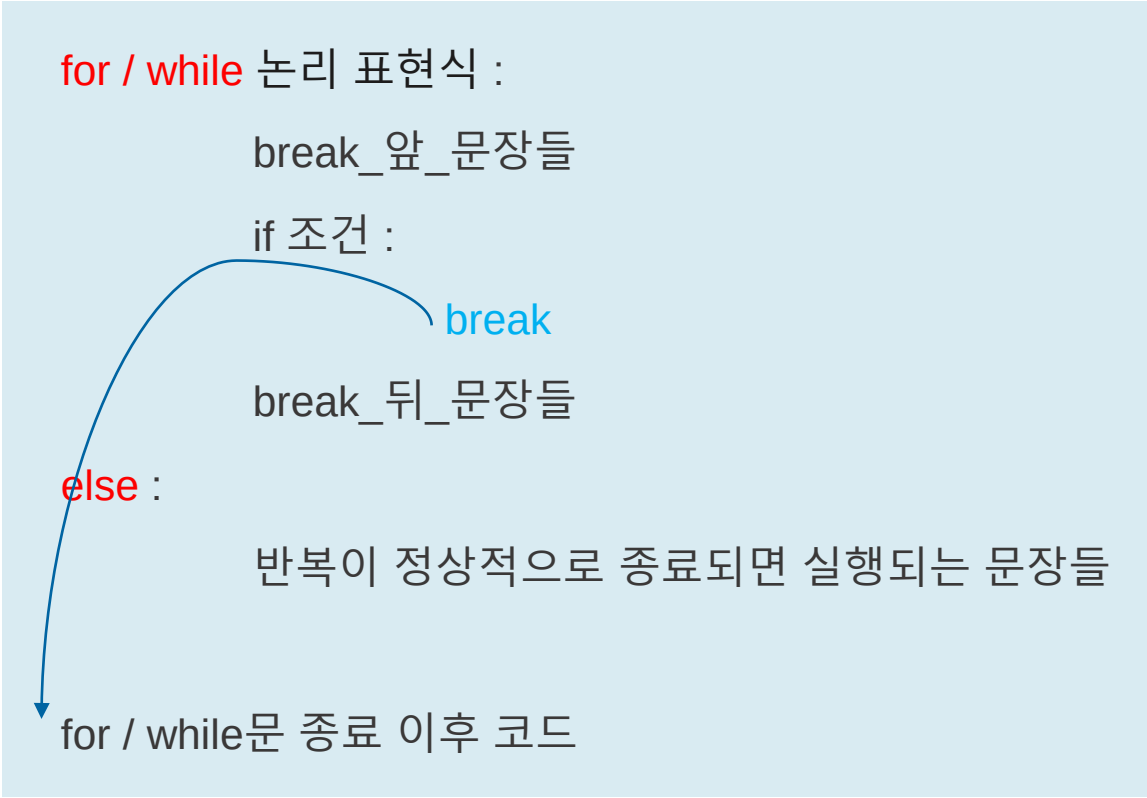
```
import random
list=[random.randint(1,99) for i in range(3)]
m,n=(list[0],)*2
s=0
for e in list:
    if m<e:m=e

print(list,' 중에서 최대: ',m)
```

```
Python 3.8.2 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2
tel)] on win32
Type "help", "copyright", "credits" or "lic
>>>
========== RESTART: C:/Users/SEC/De
===
[11, 23, 86]  중에서 최대:  86
>>>
```

1부터 99까지의 난수인 임의의 정수 3개를 대상으로 가장 큰 정수를 출력

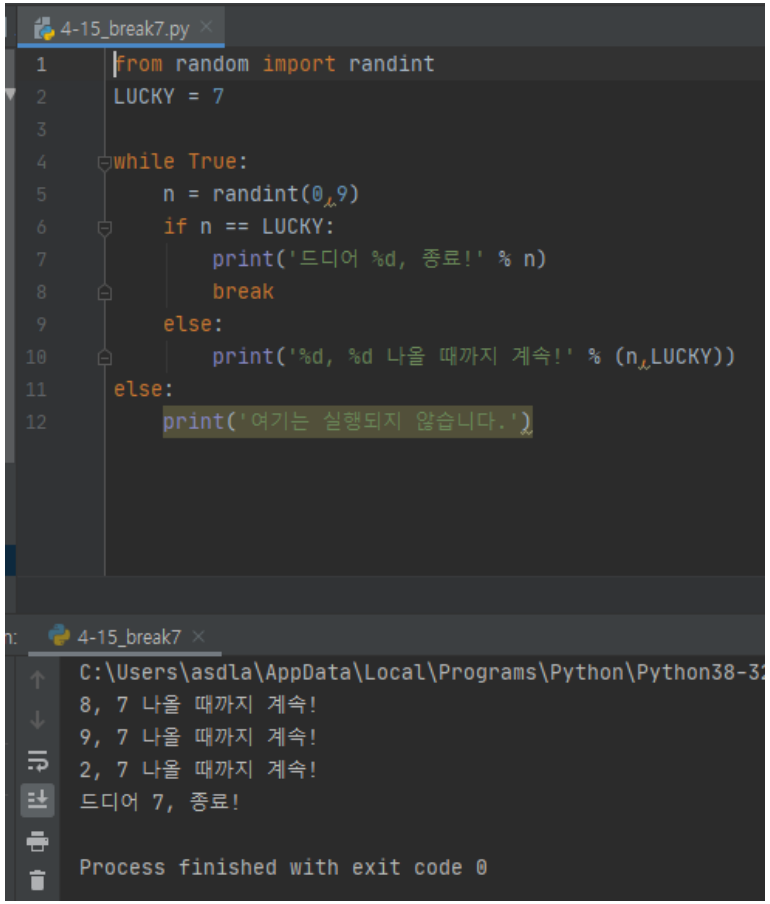
반복을 제어하는 break문 (1)

- 반복 while이나 for문에서 break 문장은 반복을 종료하고 반복 문장 이후를 실행한다.
  즉, break문은 특정한 조건에서 즉시 반복을 종료할 경우에 사용된다.
- break문으로 반복이 종료되는 경우, 반복의 else : 블록은 실행되지 않는다.



반복을 제어하는 break문 (2)

- 0에서 9 사이의 난수 중에서 7이 나오면 반복 종료



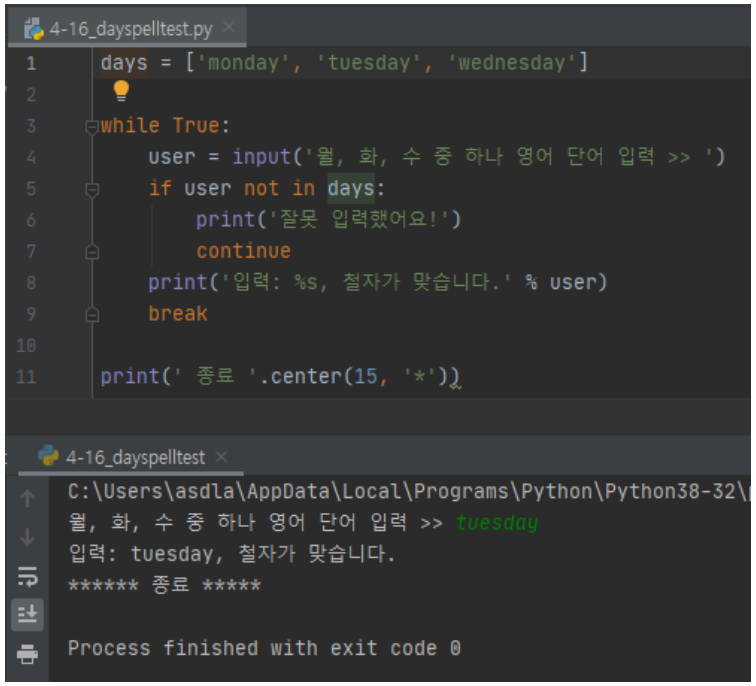
반복에서 break문에 의해 종료되면 반복 while이나 for의 else: 블록은 실행되지 않는다.

반복을 제어하는 continue문

- 반복 for와 while문 내부에서 continue 문장은 이후의 반복 몸체를 실행하지 않고 다음 반복을 위해 논리 조건을 수행한다.

```
while 논리 표현식 :
        continue
for 변수 in 시퀀스 :
        continue
```

- continue와 break를 사용한 예제



while문을 사용해 사용자에게 요일을 표준 입력으로 받은 후, 맞으면 정답을 출력하고 break로 while을 빠져나와 종료한다. 만일 맞지 않으면 continue문으로 계속 표준 입력을 받아 정답이 맞을 때까지 반복한다.

Part 5

리스트와 튜플

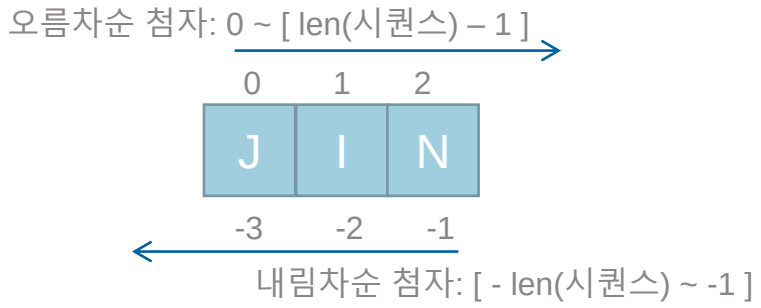
리스트의 개념과 생성

- 리스트는 항목의 나열인 시퀀스다.
- 리스트는 콤마로 구분된 항목들의 리스트로 표현되며, 각 항목은 모두 같은 자료형일 필요는 없다.
- 항목 순서는 의미가 있으며, 항목 자료 값은 중복되어도 상관없다.
- 리스트는 대괄호 [ ] 사이에 항목을 기술한다.

```
>>> bts = [‘RM’, ‘진’, ‘SUGA’, ‘J-HOPE’, ‘지민’, ‘V’, ‘정국’]
>>> print(bts)
[‘RM’, ‘진’, ‘SUGA’, ‘J-HOPE’, ‘지민’, ‘V’, ‘정국’]
>>> type(bts)
<class ‘list’>
```

리스트의 항목 참조 (1)

- 리스트에서 첨자를 사용해 항목 하나하나를 참조할 수 있다.
- 첨자는 첫 요소가 0부터 시작하며, 순차적으로 1씩 증가한다.
- 마지막 요소의 첨자는 역순으로 -1부터 시작하며, 반대로 1씩 감소한다.
- 첨자의 범위는 0에서 [len(시퀀스)-1]까지, -1에서 [-len(시퀀스)]까지 가능하다.



리스트의 항목 참조 (2)

- 첨자로 항목 참조 예제

5-5_rockgame.py

```
1  rsp = ['가위', '바위', '보']
2  for i in range(len(rsp)):
3      print(rsp[i], end=' ')
4  print()
5
6  from random import choice
7  print('컴퓨터의 가위 바위 보 5개')
8  for i in range(5):
9      print(choice(rsp))
```

5-5_rockgame

```
C:\Users\asdla\AppData\Local\Programs\Pyt
가위 바위 보
컴퓨터의 가위 바위 보 5개
보
바위
바위
보
가위
```

'가위', '바위', '보' 리스트인 rsp를 메소드 choice의 인자로 호출한다.

리스트의 항목 수정

- 리스트의 메소드 count(값)는 값을 갖는 항목의 수, index(값)는 인자인 값의 항목이 위치한 첨자를 반환한다.

5-6_foodorder.py

```
1   food = ['짜장면', '짬뽕', '우동', '울면']
2   print(food)
3   # 탕수육 주문 추가
4   food.append('탕수육')
5   print(food)
6   # 짬뽕을 굴짬뽕으로 주문 변경
7   food[1] = '굴짬뽕'
8   print(food)
9
10  # 우동을 물만두로 주문 변경
11  food[food.index('우동')] = '물만두'
12  print(food)
```

5-6_foodorder

```
C:\Users\asdla\AppData\Local\Programs\Python\Py
['짜장면', '짬뽕', '우동', '울면']
['짜장면', '짬뽕', '우동', '울면', '탕수육']
['짜장면', '굴짬뽕', '우동', '울면', '탕수육']
['짜장면', '굴짬뽕', '물만두', '울면', '탕수육']

Process finished with exit code 0
```

index()를 사용해서 우동을 물만두로 변경했다.

리스트 메소드

- 첨자 위치에 항목을 삽입하려면, 리스트.insert(첨자,항목)을 이용한다.
- 하나의 항목을 삭제하려면 메소드 remove(값) 또는 pop(첨자), pop( )을 이용한다.
- 문장 del은 뒤에 위치한 변수나 항목을 삭제한다.
- clear( )는 리스트의 모든 항목을 제거하여 빈 리스트로 만든다.
- 리스트.extend(list)는 리스트에 인자인 list를 가장 뒤에 추가한다.
- reverse( )는 항목 순서를 반대로 뒤집는다.
- sort( )는 리스트 항목의 순서를 오름차순으로 정렬한다.
- 내장 함수 sorted(리스트)는 리스트의 항목 순서를 오름차순으로 정렬한 새로운 리스트를 반환한다.

5-10_wordsort.py

```
1   word = list('삶꿈정')
2   word.extend('복빛')
3   print(word)
4   word.sort()
5   print(word)
6
7   fruit = ['복숭아', '자두', '골드키위', '귤']
8   print(fruit)
9   fruit.sort(reverse=True)
10  print(fruit)
11
12  mix = word + fruit
13  print(sorted(mix))
14  print(sorted(mix, reverse=True))
```

5-10_wordsort

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\pytho
['삶', '꿈', '정', '복', '빛']
['꿈', '복', '빛', '삶', '정']
['복숭아', '자두', '골드키위', '귤']
['자두', '복숭아', '귤', '골드키위']
['골드키위', '귤', '꿈', '복', '복숭아', '빛', '삶', '자두', '정']
['정', '자두', '삶', '빛', '복숭아', '복', '꿈', '귤', '골드키위']

Process finished with exit code 0
```

괄호로 정의하는 시퀀스 튜플

- 튜플은 문자열, 리스트와 같은 항목의 나열인 시퀀스이다.
- 리스트와 달리 항목의 순서나 내용의 수정이 불가능하다.
- 괄호 ( ) 사이에 항목을 기술한다.
- 빈 튜플은 ( )로 만든다.
- 튜플의 자료형 이름은 클래스 tuple이다.
- 항목이 하나인 튜플을 표현할 때는 마지막에 콤마를 반드시 붙여야 한다.

튜플 변수 singer에는 k-pop 가수가 저장되고, 튜플 변수 song에는 singer에 대응하는 주요 노래를 저장한다. 참조에 해당하는 메소드 count( )와 index( )는 튜플에서 사용할 수 있다.

5-12_kpoptuple.py

```
singer = ('BTS', '볼빨간사춘기', '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('볼빨간사춘기'))
print(singer.index('BTS'))
print()

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

for _ in range(len(singer))

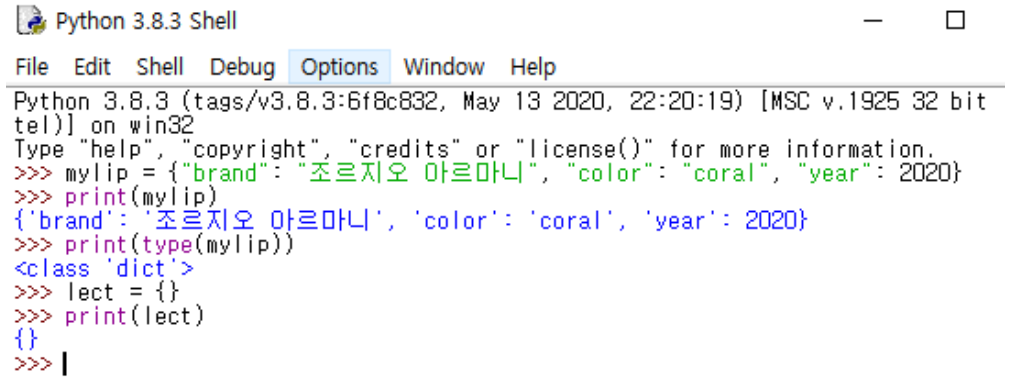
5-12_kpoptuple

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020_
('BTS', '볼빨간사춘기', 'BTS', '블랙핑크', '태연')
('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
2
1
0

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

Part 6
딕셔너리

딕셔너리의 개념과 생성

- 딕셔너리는 키와 값의 쌍인 항목을 나열한 시퀀스이다.
- 콤마로 구분된 항목들의 리스트로 표현된다.
- 각각의 항목은 키:값과 같이 키와 값을 콜론으로 구분하고 전체는 중괄호 {...}로 묶는다.
- 중괄호 {...} 사이에 키와 값의 항목을 기술한다.
- 항목 순서는 의미가 없으며, 키는 중복될 수 없다.
- 키는 수정될 수 없지만, 값은 수정될 수 있다.
- 값은 키로 참조된다.
- 빈 중괄호로 빈 딕셔너리를 만들 수 있다. (출력 시 {}로 표시됨)
- 내장 함수 dict() 호출로도 빈 딕셔너리를 만들 수 있다.



함수 dict( )로 생성하는 딕셔너리

- 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성 방법

6-3_btsdict.py

```
1   bts1 = {'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준'}
2   bts1['소속사'] = '빅히트 엔터테인먼트';
3   print(bts1)
4   bts2 = dict([['그룹명', '방탄소년단'], ['인원수',7]])
5   print(bts2)
6   bts3 = dict((('리더', '김남준'), ('소속사', '빅히트 엔터테인먼트')))
7   print(bts3)
8
9   bts = dict(그룹명='방탄소년단', 인원수=7, 리더='김남준', 소속사='빅히트 엔터테인먼트')
10  # 구성원 추가
11  bts['구성원'] = ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
12
13  print(bts) # 전체 출력
14  print(bts['구성원']) # 구성원 출력
```

6-3_btsdict

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020_python_code/ch06/6-3_btsdict.py
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔터테인먼트'}
{'그룹명': '방탄소년단', '인원수': 7}
{'리더': '김남준', '소속사': '빅히트 엔터테인먼트'}
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔터테인먼트', '구성원': ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']}
['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
```

딕셔너리의 생성 방법으로 중괄호에 '그룹명':'방탄소년단' 등의 항목들로 삽입하는 방법과 함수 dict( )에서 ['그룹명','방탄소년단'] 등의 리스트를 이용하는 방법 등을 활용한다.

딕셔너리의 키

- 딕셔너리의 키는 수정 불가능한 객체는 모두 가능하기 때문에, 정수는 물론 실수도 가능하다.
- 새로운 키로 대입하면 항상 새로운 키 값의 항목이 삽입된다.

```
>>> age = { 26 : '김태형' }
>>> age[29] = '김석진'
>>> print(age)
{ 26 : '김태형', 29 : '김석진'} // 새로운 항목 삽입
>>> print(age[29])
김석진
```

6-4_monthdictionary.py

```
month = {1: 'January', 2: 'February', 3: 'March', 4: 'April'}
month[5] = 'May'
month[6] = 'June'
month[7] = 'July'
month[8] = 'August'
month[9] = 'September'
print(month)
print()

from random import randint
# 임의로 5번의 월 단어 출력
for i in range(5):
    r = randint(1,9)
    print('%d: %s' % (r, month[r]))
```

6-4_monthdictionary

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe
{1: 'January', 2: 'February', 3: 'March', 4: 'April', 5: 'May', 6: '

8: August
8: August
1: January
6: June
3: March
```

1월에서 9월까지의 영어 단어를 표현하기 위해 1-'January'처럼 키-값을 월-영어 단어 항목으로 하는 딕셔너리 생성

딕셔너리 항목의 순회

- 메소드 keys( )는 키로만 구성된 리스트를 반환한다.
- 메소드 items( )는 (키, 값) 쌍의 튜플이 들어 있는 리스트를 반환한다.
- 메소드 values( )는 값으로 구성된 리스트를 반환한다.
- 반복 for문에서는 시퀀스 위치에 있는 딕셔너리 변수만으로도 모든 키를 순회할 수 있다.

6-5_seasondict.py

```
season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
print(season.keys())
print(season.items())
print(season.values())

# 메소드 keys()로 항목 순회
for key in season.keys():
    print('%s %s ' % (key, season[key]))

for item in season.items():
    print('{} {} '.format(item[0], item[1]), end=' ')
print()
# 메소드 items()의 반환 값인 튜플을 한 변수에 저장한 경우, 항목 순회 2
for item in season.items():
    print('{} {} '.format(*item), end=' ')
print()
```

6-5_seasondict

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020_python_code/
dict_keys(['봄', '여름', '가을', '겨울'])
dict_items([('봄', 'spring'), ('여름', 'summer'), ('가을', 'autumn'), ('겨울', 'winter')])
dict_values(['spring', 'summer', 'autumn', 'winter'])
봄 spring
여름 summer
가을 autumn
겨울 winter
봄 spring  여름 summer  가을 autumn  겨울 winter
봄 spring  여름 summer  가을 autumn  겨울 winter
```

엽의 예제에서 사계절의 딕셔너리에서 메소드 keys( ), items( ), values( )를 출력한 뒤에, 각각 메소드 key( )와 items( )를 이용해 사전의 전체 항목을 순회한 후 출력

딕셔너리 메소드 (1)

- 메소드 get(키)는 키의 해당 값을 반환한다.
- 메소드 get(키)는 딕셔너리에 키가 없어도 오류가 발생하지 않고 아무것도 없다는 의미인 None을 반환한다.
- 메소드 pop(키)는 키인 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.
- 메소드 popitem( )은 임의의 (키, 값)의 튜플을 반환하고 삭제한다.
- 딕셔너리를 문장 del에 이어 키로 지정하면 해당 항목이 삭제된다.
- 메소드 clear( )는 기존의 모든 키:값 항목을 삭제한다.
- 메소드 update(다른 딕셔너리)는 인자인 다른 딕셔너리를 합병한다.
- 문장 in으로 딕셔너리에 키가 존재하는지 검사할 수 있다.

딕셔너리 메소드 (2)

- 딕셔너리 메소드를 사용한 예제

6-6_colordict.py

```
1  color = dict(검은색='black', 흰색='white', 녹색='green', 파란색='blue')
2  print(color)
3
4  # 항목 조회
5  print(color.get('녹색'))
6  print(color.get('노란색'))
7  print()
8
9  # 항목 추가
10 color['노란색'] = 'yellow'
11 print(color)
12 print()
13
14 # 항목 삭제
15 c = '흰색'
16 print('삭제: %s %s' % (c, color.pop('흰색')))
17 print(color)
18 c = '빨간색'
19 print('삭제: %s %s' % (c, color.pop(c, '없어요')))
20
21 print('임의 삭제: {} '.format(color.popitem()))
22 print('임의 삭제 후: {} '.format(color))
23
24 c = '검은색'
25 del color[c]
26 print('{} 삭제 후: {}'.format(c, color))
27
28 # 모든 항목 삭제
29 color.clear()
30 print(color)
```

6-6_colordict

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020_python_code/
{'검은색': 'black', '흰색': 'white', '녹색': 'green', '파란색': 'blue'}
green
None

{'검은색': 'black', '흰색': 'white', '녹색': 'green', '파란색': 'blue', '노란색': 'yellow'}

삭제: 흰색 white
{'검은색': 'black', '녹색': 'green', '파란색': 'blue', '노란색': 'yellow'}
삭제: 빨간색 없어요
임의 삭제: ('노란색', 'yellow')
임의 삭제 후: {'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
검은색 삭제 후: {'녹색': 'green', '파란색': 'blue'}
{}

Process finished with exit code 0
```

메소드 get( ), pop( ), popitem( ), clear( )등을 사용해 조회를 하거나 항복을 삭제한다. 그리고 문장 del로 색상 항목 1개를 삭제한다.

연습 예제 (1) – 딕셔너리 복사

test1.py

```
1   album = dict(wings='begin', loveyouself='dna', mapofthesoul='home')
2
3   album['danger'] = 'rain'
4   album['danger'] = 'heavyrain'
5   print(album)
6
7   e = list(album)
8   print(e)
9   f = list(album.items())
10  print(f)
11  t = dict(f)
12  print(t)
```

test1

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:/2020_python_code/test/test1.py
{'wings': 'begin', 'loveyouself': 'dna', 'mapofthesoul': 'home', 'danger': 'heavyrain'}
['wings', 'loveyouself', 'mapofthesoul', 'danger']
[('wings', 'begin'), ('loveyouself', 'dna'), ('mapofthesoul', 'home'), ('danger', 'heavyrain')]
{'wings': 'begin', 'loveyouself': 'dna', 'mapofthesoul': 'home', 'danger': 'heavyrain'}

Process finished with exit code 0
```

연습 예제 (2) – if ~ elif 문을 이용한 긴급재난지원금 조회

test1.py

```
1  num = int(input('가구원 수는? '))
2  if 4 <= num:
3      print('가구원 수: {}, 긴급재난지원금: {}'.format(num, '100만원'))
4  elif num == 3:
5      print('가구원 수: {}, 긴급재난지원금: {}'.format(num, '80만원'))
6  elif num == 2:
7      print('가구원 수: {}, 긴급재난지원금: {}'.format(num, '60만원'))
8  else:
9      print('가구원 수: {}, 긴급재난지원금: {}'.format(num, '40만원'))
```

elif num == 3

test1

```
C:\Users\asdla\AppData\Local\Programs\Python\Python38-32\python.exe E:
가구원 수는? 3
가구원 수: 3, 긴급재난지원금: 80만원

Process finished with exit code 0
```

포트폴리오 작성 소감

사실 포트폴리오 작성을 시작할 때, 정말 막막했습니다. 강의를 듣기는 했지만 아직 파이썬에 대해 완벽히 이해도 하지 못했었고, 분량이 많아서 걱정을 했었는데 포트폴리오를 작성하면서 책을 꼼꼼히 읽고, 직접 코드도 돌려보고 하다보니 확실히 포트폴리오 작성이 파이썬 공부하는데 도움이 많이 되었습니다. 다른 과목도 이렇게 포트폴리오 작성하듯이 공부하면 더 빨리 이해가 될 것 같아서 전공 과목 뿐만 아니라 따로 하고 있는 공부들도 이런식으로 해볼까 합니다. 교수님께서 최소 분량이 30~40장 이상이라고 하셨을 때, 어떻게 저걸 다할까 생각했지만, 막상 분량을 생각 안 하고 공부하는 순서대로 작성을 하다보니 오히려 너무 분량이 많아지는 것은 아닐까 싶은 생각이 들기도 했습니다. 아무튼 저에게는 정말 좋은 경험이었습니다. 감사합니다!

Thank You ☺