

파이썬 포트폴리오

유지인

20180652

2020.05.31

목차

01 파이썬 개요

- 1-1 파이썬 언어란?
- 1-2 파이썬 설치와 프로그래밍 방법
- 1-3 파이썬의 특징, 활용분야

02 파이썬 기초

- 2-1 문자열과 수
- 2-2 변수와 키워드, 대입 연산자
- 2-3 표준 입력과 자료 변환 함수

03 문자열과 논리 연산

- 3-1 문자열
- 3-2 문자열 관련 메소드
- 3-3 논리 자료, 연산

04 조건문과 반복문

- 4-1 조건문 if ~ else
- 4-2 반복문 for, while
- 4-3 난수, 반복 제어 break, continue

목차

05 항목의 리스트, 튜플

5-1 리스트 개요

5-2 리스트 부분 참조와 항목 삽입, 삭제

5-3 튜플

06 문자열과 논리 연산 2

6-1 딕셔너리

6-2 집합

6-3 함수 zip(), enumerate(), 시퀀스 간의 변화

01 파이썬 개요

-
- 1-1 파이썬 언어란?
 - 1-2 파이썬 설치와 프로그래밍 방법
 - 1-3 4차 산업실행의 파이썬

파이썬 언어란?

1. 파이썬 언어란?

'파이썬'은 배우기 쉽고 무료로 사용할 수 있는 오픈 소스 프로그래밍 언어다.

현재 미국과 우리나라의 대학 등 전 세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나다.

2. 제 4차산업에서 파이썬

4차 산업 혁명 시대에는 모든 사물이 연결되 초연결 사회이며, 모든 사업은 컴퓨터 과학 공학 기술을 기반으로 융합될 것이다.

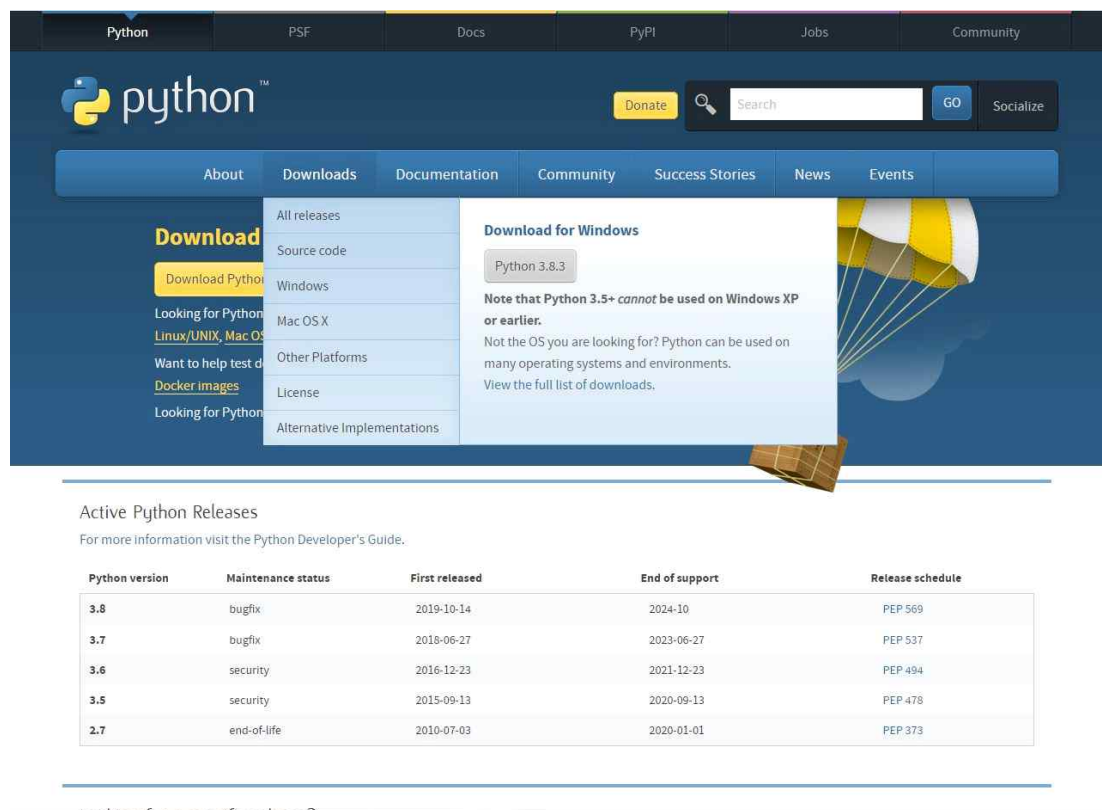
모든 산업 분야에서 과학 기술을 필요로 한다.

그러므로 미래 인재는 컴퓨터 과학 원리와 개념을 자신의 영역과 융합할 수있는 역량 즉 '**컴퓨팅 사고력**'이 필요하다.

'**컴퓨팅 사고력**'은 컴퓨팅의 기본 개념과 원리를 기반으로 문제를 효율적으로 해결하는 사고 능력으로

파이썬을 이용해 전문가는 프로그래밍 절차와 기술을 교육할 수 있고 비전문가는 컴퓨팅 사고력을 교육할 수 있다.

파이썬 설치와 프로그래밍 방법

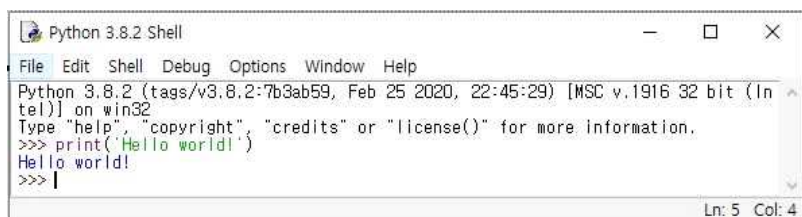


The screenshot shows the Python.org website with the 'Downloads' menu open. The 'Download for Windows' section is highlighted, showing the 'Python 3.8.3' download link. Below this, there is a table titled 'Active Python Releases' with columns for Python version, Maintenance status, First released, End of support, and Release schedule.

| Python version | Maintenance status | First released | End of support | Release schedule |
|----------------|--------------------|----------------|----------------|------------------|
| 3.8 | bugfix | 2019-10-14 | 2024-10 | PEP 569 |
| 3.7 | bugfix | 2018-06-27 | 2023-06-27 | PEP 537 |
| 3.6 | security | 2016-12-23 | 2021-12-23 | PEP 494 |
| 3.5 | security | 2015-09-13 | 2020-09-13 | PEP 478 |
| 2.7 | end-of-life | 2010-07-03 | 2020-01-01 | PEP 373 |

www.python.org/downloads 홈페이지로 이동후 Python 3.8.3 을 눌러 파일을 다운 받아 설치한다.

파이썬 설치와 프로그래밍 방법

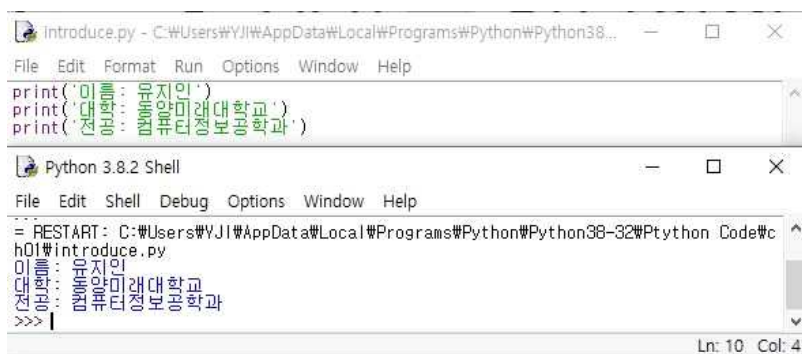


```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('Hello world!')
Hello world!
>>>
```

1. 파이썬 셸 실행

Hello World!를 출력한다.

- 첫 칸에 공간이 있으면 안된다.
- 문자열 앞뒤로 작은 따옴표를 붙인다. 'Hello World'



```
Introduce.py - C:\Users#YJ\#AppData#Local#Programs#Python#Python38...
File Edit Format Run Options Window Help
print('이름: 유지인')
print('대학: 연세대학교')
print('전공: 컴퓨터정보공학')

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
= RESTART: C:\Users#YJ\#AppData#Local#Programs#Python#Python38-32#Python Code#ch01#Introduce.py
이름: 유지인
대학: 연세대학교
전공: 컴퓨터정보공학
>>>
```

1. 파이썬 실행

파이썬 셸에서 [File] - [New File] 을 선택해 새 창을 만든다.

작성이 끝난 후에 저장하여 F5를 누르면 오류가 없다면 파이썬 셸에 표시된다.

파이썬의 특징, 활용분야

1. 파이썬의 특징

- ① 간결하고, 인간의 사고 체계와 닮아 사용하기 쉽다.
- ② 무료이고, 다양한 자료 구조의 제공으로 생산성이 높다.
- ③ 라이브러리가 독보적이고 강력하고, 데이터과학과 머신 러닝 등과 같은 다양한 분야에 적용할 수 있다.
- ④ 다른 모듈을 연결해 사용할 수 있는 풀 언어로도 자주 활용된다.

2. 파이썬 활용 분야

- 1) 교육과 학술 실무 분야에 사용
 - 파이썬의 외부에 풍부한 라이브러리가 있어 다양한 용도로 확장하기 좋기 때문이다.
- 2) 인공지능과 빅데이터 분석 처리 분야에 사용
 - 인공 지능의 머신 러닝과 딥러닝, 빅데이터 처리를 위한 통계 및 분석 방법의 라이브러리가 풍부하게 제공한다.

02 파이썬 기초

-
- 2-1 문자열과 수
 - 2-2 변수와 키워드, 대입 연산자
 - 2-3 표준 입력과 자료 변환 함수

문자열과 수

```

2-1.py - C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
print("원의 원주율 " + '3.141592')
print("python" 'programming' + ' language')
print('파이썬' + 3 * " 안녕")

= RESTART: C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/c
h02/2-1.py
원의 원주율 3.141592
pythonprogramming language
파이썬 안녕 안녕 안녕
>>>

```

```

2-2.py - C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
'''
01-02
| 2019 3. by KHS
'''
print('# 이후는 주석') #한 줄에서 문장 이후에도 주석 사용 가능
print('string: "python"') # 큰따옴표 내부에서 작은따옴표는 문자열
print('number: 1 5 3.14') # 문자열 내부에서 숫자도 문자열
print('string: 'python') # 작은따옴표 내부에서 작은따옴표는 문자열

= RESTART: C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/c
h02/2-2.py
# 이후는 주석
string: "python"
number: 1 5 3.14
string: 'python'
>>>

```

1. 문자열 연산자 +, *

더하기 기호인 +는 문자열에서 문자열을 연결하는 역할을 한다.

별표 *는 문자열에서 문자열을 지정된 수만큼 반복하는 연산을 수행한다.

- 반복 횟수인 정수가 하나 있어야 한다. '파이썬' * '언어' 이면 오류가 발생한다.

2. 주석

문자열이 길거나 여러줄에 걸쳐 문자열을 처리하기 위해서는

삼중 따옴표를 사용 한다.


삼중 따옴표는 주석으로 사용이 가능하다

#은 주석으로 사용된다. #은 한줄만 주석으로 유효하다.

문자열과 수

1. 정수와 실수

정수와 실수 모두 print()로 바로 표현이 가능하다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> print(15)
15
>>> print(3.14)
3.14
>>> 0
0
>>> 03
SyntaxError: leading zeros in decimal integer literals are not permitted; use an
    0o prefix for octal integers
>>> 03.14
3.14
Ln: 28 Col: 12
```

1. 정수 표현

정수는 0은 0이 여러개라도 가능하다.

0이 앞에 나오면 구문 오류가 발생한다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> 2.7834e4
27834.0
>>> 2.784e-4
0.0002784
>>> 2.7834E5
278340.0
>>>
Ln: 35 Col: 1
```

2. 실수 표현

실수는 문자 e를 사용해 2.7834e4처럼 지수승으로 표현할 수 있다.

문자 e는 대문자 E도 가능하다.

문자열과 수

※ 연산 예제

1) 예금의 단리와 복리 계산

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> 1000000 + (1000000 * 2.3/100) # 1년차 총액
1023000.0
>>> 1000000 + (1000000 * 2.3/100) + (1000000 * 2.3/100) #2년차 총액
1046000.0
>>> 1000000 * (1 + 2.3/100 * 3) # 3년차 총액
1069000.0
>>> 1000000 * (1 + 2.3/100 * 4) # 4년차 총액
1092000.0
>>> 1000000 * (1 + 2.3/100 * 5) # 5년차 총액
1115000.0
>>> 1000000 * (1 + 2.3/100) ** 5 # 복리 5년차 총액
1120413.0756413424
Ln: 117 Col: 1
```

2) 여러 자료 출력

```
2-3.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
print(1, 2, -5, 3.14, 2.71828)
print('Hi', 'Python')

print('23000원은 '5000원 ?개', '1000원 ?개')
print('5000원', 23000 // 5000, '개')
print('1000원', (23000 % 5000) // 1000, '개')

Ln: 10 Col: 1
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h02/2-3.py
1 2 -5 3.14 2.71828
Hi,Python
23000원은5000원 ?개 1000원 ?개
5000원 4 개
1000원 3 개
>>>
```

변수와 키워드, 대입 연산자



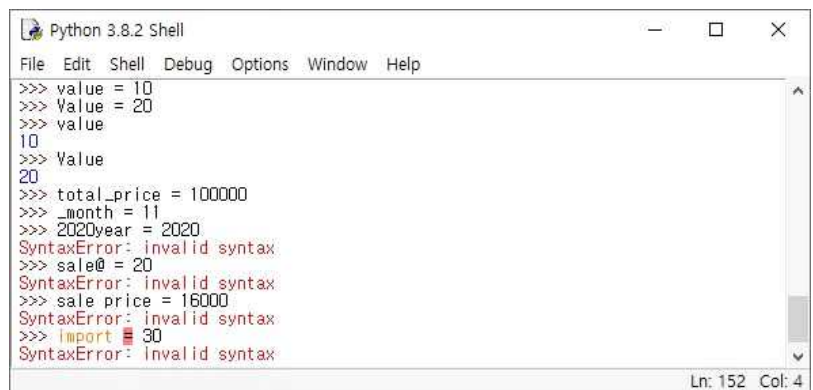
```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> type(3)
<class 'int'>
>>> pi = 3.14
>>> type(pi)
<class 'float'>
>>>
>>> unit = 3
>>> 3 = unit
SyntaxError: cannot assign to literal
>>>
```

1. type() 함수, 대입 연산자 =

type(3)과 같이 괄호 사이에 상수를 넣으면 지정된 상수의 자료형을 알 수 있다.

값을 저장하기 위해서는 대입 연산자(=)가 필요하다.

오른쪽 값을 왼쪽에 저장하는 모양으로 대입 연산자 왼쪽으로는 변수가 와야 한다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> value = 10
>>> Value = 20
>>> value
10
>>> Value
20
>>> total_price = 100000
>>> _month = 11
>>> 2020year = 2020
SyntaxError: invalid syntax
>>> sale@ = 20
SyntaxError: invalid syntax
>>> sale price = 16000
SyntaxError: invalid syntax
>>> import 30
SyntaxError: invalid syntax
>>>
```

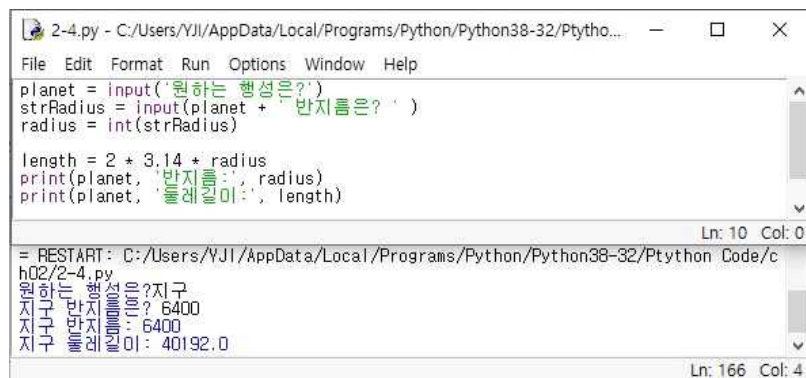
2. 변수 이름 규칙

- 문자는 대소문자의 영문자, 숫자 _로 구성
- 대소문자는 구별이 된다.
- 숫자는 맨 앞에 올 수 없다.
- 공백 문자는 사용 할 수 없다.
- import, True 등과 같은 키워드는 사용할 수 없다.

표준 입력과 자료 변환 함수

1. 표준 입력이란?

프로그램 과정에서 셸이나 콘솔에서 사용자의 입력을 받아 처리하는 방식



```

2-4.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
planet = input('원하는 행성은? ')
strRadius = input(planet + ' 반지름은? ')
radius = int(strRadius)

length = 2 * 3.14 * radius
print(planet, '반지름:', radius)
print(planet, '둘레길이:', length)

Ln: 10 Col: 0

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h02/2-4.py
원하는 행성은? 지구
지구 반지름은? 6400
지구 반지름: 6400
지구 둘레길이: 40192.0
Ln: 166 Col: 4

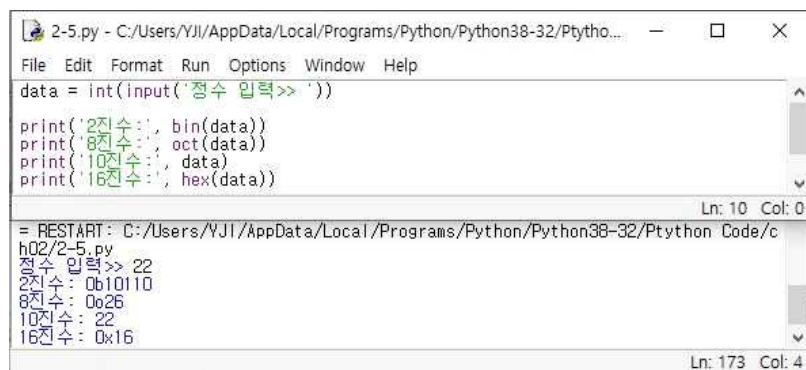
```

2. 함수 input() 표준 입력, 변환 함수 int()

- 함수 input()은 입력되는 표준 입력을 문자열로 읽어 반환하는 함수이다.
- input()에서 반환되는 입력 문자열을 변수에 저장하려면 대입 연산자 =을 사용해 변수에 저장해야 한다.
- 만일 문자열이 정수나 실수 형태가 아니면 오류가 발생한다.
- input()으로 받은 문자열을 int() 함수를 사용하여 타입을 변환한다.

3. 16진수, 10진수, 8진수, 2진수

- 16진수 : 맨앞에 0x(숫자 0과 알파벳 x)로 시작한다.
- 8진수 : 맨앞에 0o(숫자 0과 알파벳 o)로 시작한다.
- 2진수 : 맨앞에 0b(숫자 0과 알파벳 b)로 시작한다.
- 진수를 표현하는 알파벳은 대 소문자 모두 사용이 가능하다.



```

2-5.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
data = int(input('정수 입력>> '))

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

Ln: 10 Col: 0

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h02/2-5.py
정수 입력>> 22
2진수: 0b10110
8진수: 0o26
10진수: 22
16진수: 0x16
Ln: 173 Col: 4

```

03 문자열과 논리 연산

-
- 3-1 문자열
 - 3-2 문자열 관련 메소드
 - 3-3 논리 자료, 연산

문자열



```

3-1.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
str = '일요일 기러기'
print(str[:3], str[4:])
print(str[:-4], str[-3:])
print(str[:], str[:,], str[:,1])
print(str[:2])
print(str[:3])
print(str[:-2])
print(str[::-1])
Ln: 5 Col: 0

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h03/3-1.py
일요일 기러기
일요일 기러기
일요일 기러기 일요일 기러기 일요일 기러기
일요일 기러기
일요일 기러기
일요일 기러기
일요일 기러기
일요일 기러기
Ln: 182 Col: 4

```

1. 문자열 슬라이싱

문자열에서 일부분을 참조하는 방법을 슬라이싱이라고 한다.

`str[srat:end]` : 문자열 srt에서 srart 첨자에서 end-1 첨자까지 문자열 반환

1) 양수를 이용한 슬라이싱

`'python' [1:5]`

`'ytho'`

2) 음수를 이용한 슬라이싱

`'python' [-5:-1]`

`'ytho'`

3) 양수와 음수 혼합한 슬라이싱

`'python' [1:-1]`

`'ytho'`

문자열 관련 메소드



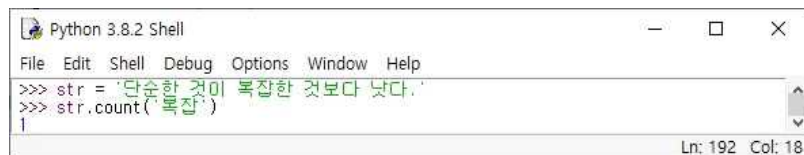
```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> str = '파이썬 파이썬 파이썬'
>>> str.replace('파이썬', 'Python', 2)
'Python Python 파이썬'
Ln: 208 Col: 31
```

1. 문자열 바꿔 반환하는 메소드 replace()

replace(old, new, count)는 문자열 old를 new로 대체하는데,

옵션인 count 대체 횟수를 지정한다.

옵션인 count가 없으면 모두 바꾸고, 있으면 앞에서부터 지정한 횟수만큼 바꾼다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> str = '단순한 것이 복잡한 것보다 낫다.'
>>> str.count('복잡')
1
Ln: 192 Col: 18
```

2. 부분 문자열 출현 횟수를 반환하는 메소드 count()

문자열 str에서 문자나 부분 문자열의 출현 횟수를 알려준다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> num = '12345'
>>> '->'.join(num)
'1->2->3->4->5'
Ln: 199 Col: 1
```

3. 문자와 문자 사이에 문자열을 삽입하는 메소드 join()

문자열과의 문자와 문자 사이에 원하는 문자열을 삽입하여 반환한다.

문자열 관련 메소드

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> str = '자바 c 파이썬'
>>> str.find('자바')
0
>>> str.index('자바')
0
>>> str.find('c++')
-1
>>> str.index('c++')
Traceback (most recent call last):
  File "<pyshell#76>", line 1, in <module>
    str.index('c++')
ValueError: substring not found
Ln: 229 Col: 33
```

4. 문자열을 찾는 메소드 find()와 index()

str에서 부분 문자열 sub가 매 처음에 위치한 첨자를 반환한다.
메소드 index()는 찾는 문자열이 없으면 오류를 발생하고,
메소드 find()는 오류를 발생시키지 않고 -1을 반환한다.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> '데스크톱 1000000, 노트북 1800000, 스마트폰 1200000'.split(',')
['데스크톱 1000000', '노트북 1800000', '스마트폰 1200000']
Ln: 214 Col: 8
```

5. 문자열을 여러 문자열로 나누는 메소드 split()

문자열 str에서 공백을 기준으로 문자열을 나눠준다.
만약(',')처럼 괄호 안에 특정한 문자열이 있을 경우 이 부분 문자열 값을
구분자를 이용해 문자열을 나눠 준다.

문자열 관련 메소드



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> 'python'.upper()
'PYTHON'
>>> 'PYTHON'.lower()
'python'
>>> 'python lecture'.capitalize()
'Python lecture'
>>> 'python lecture'.title()
'Python Lecture'
>>> 'PyThOn'.swapcase()
'pYtHoN'
```



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> '파이썬 강좌'.center(30, '*')
'*****파이썬 강좌*****'
```

6. 영문자 알파벳의 다양한 변환 메소드

.upper() : 모두 대문자로 변환해 반환

.lower() : 모두 소문자로 변환해 반환

.capitalize() : 첫 문자만 대문자로 변환해 반환

.title() : 단어마다 첫 문자를 대문자로 변환해 반환

.swapcase() : 소문자와 대문자를 서로 변환해 반환

7. 폭을 지정하고 중앙에 문자열 배치하는 메소드 center()

center(폭, 채울 문자) 전체 문자열의 폭을 지정한 후에 중앙에 문자열을 배치하고, 좌우의 나머지는 두번째 인자로 채운다.

두 번째 인자는 반드시 하나의 채울 문자로 선택적이며, 없으면 공백 문자로 채워진다.

문자열 관련 메소드

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> 'python'.ljust(10, '*')
'python****'
>>> 'python'.rjust(10, '*')
'****python'
```

Ln: 246 Col: 25

8. 폭을 지정하고 왼쪽, 오른쪽 정렬하는 메소드 ljust(), rjust()

.ljust() : 문자열의 폭을 지정하고 왼쪽에 붙여 정렬

.rjust() : 문자열의 폭을 지정하고 오른쪽에 붙여 정렬

- 빈 공간을 문자로 채우려면 두 번째 인자로 입력한다.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> 'python'.lstrip()
'python'
>>> 'python'.rstrip()
'python'
>>> 'python'.strip()
'python'
>>> '***python---'.strip('* -')
'python'
```

Ln: 277 Col: 4

9. 문자열 앞뒤의 특정 문자들을 제거하는 메소드 strip()

str.strip() : 문자열 str에서 맨 앞뒤의 공백 문자를 제거해 반환하는 메소드이다.

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> '234'.zfill(5)
'00234'
```

Ln: 285 Col: 4

10. 0을 채워 넣는 메소드 zfill()

.zfill() : 문자열의 지정한 폭 앞 빈 부분에 0을 채워 넣을 수 있다.

문자열 관련 메소드

11. 출력을 정형화하는 함수 format()

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> a, b = 10, 3
>>> print('{0:d} / {1:d} = {2:f}'.format(a, b, a / b))
10 / 3 = 3.333333
>>> print('{0:5d} / {1:5d} = {2:10.3f}'.format(a, b, a/b))
10 / 3 = 3.333
>>> a, b = 19, 8
>>> print('{0:5b} // {1:5o} = {2:5d}'.format(a, b, a // b))
Traceback (most recent call last):
  File "<pysHELL#136>", line 1, in <module>
    print('{0:5b} // {1:5o} = {2:5d}'.format(a, b, a // b))
ValueError: cannot switch from manual field specification to automatic field numbering
Ln: 305 Col: 27
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> print('1234567890' * 2)
12345678901234567890
>>> print('{0:10f}{0:10.3f}'.format(3.1415))
3.141500 3.142
>>> print('{0:10f}{0:10.3f}'.format(3.5E1000))
inf inf
>>> print('{0:10f}{0:10.3f}'.format(float('nan')))
nan nan
Ln: 315 Col: 4
```

1) 메소드 format()를 이용해 출력 처리

- 문자열 '{} + {} = {}' 내부에서 중괄호 {}이 위치한 부분에
- format(3, 4, 3+4) 인자인 3, 4, 7이 순서대로 출력된다.
- 문자열에서 {}을 제외한 나머지 부분인 '+='은 쓰여 있는 그대로 출력된다.
- 2진수, 8진수, 16진수로 출력하려면 각각 b, o, x로 형식 유형을 지정해 출력할 수 있다.
- 인자의 순번은 모두 적어야 하며, 빠려면 모두 빼야 한다. 부분적으로 적으면 다음과 같이 오류가 발생한다.

2) 문자열의 중괄호 {n:mf}로 실수를 형식 유형 출력 처리

- 폭을 10.3f로 지정하면 전체 폭은 10, 반올림해 소숫점 이하 세 자리까지 출력한다
- 다만 F는 무한대를 의미하는 INF와
- 계산이 불가능한 수를 표현하는 NAN을 대문자로 표시한다.

논리 자료, 연산

1. 논리 값과 논리 연산

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> print(True, False)
True False
>>> type(True)
<class 'bool'>
>>> bool(0), bool(0.0), bool(' ')
(False, False, True)
>>> bool(0), bool(0.0), bool('')
(False, False, False)
>>> bool(10), bool(3.14), bool('python')
(True, True, True)
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> True & True, True and True
(True, True)
>>> True & False, True and False
(False, False)
>>> False & True, False and True
(False, False)
>>> False & False, False and False
(False, False)
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> True | True, True or True
(True, True)
>>> True | False, True or False
(True, True)
>>> False | True, False or True
(True, True)
>>> False | False, False or False
(False, False)
```

1) 논리 유형 bool과 함수 bool()

파이썬은 논리 값으로 참과 거짓을 의미하는 True와 False를 키워드로 제공한다.
 정수의 0, 실수의 0.0, 빈 문자열 "" 등은 False이며,
 음수, 양수, 뭔가가 있는 'java' 등의 문자열은 모두 True다.
 int(True) = 1 int(False) = 0 이다.

2) 논리곱과 논리합 연산자 and

논리곱인 and와 & 연산자는 두 항이 모두 참이어야 True다.
 하나라도 거짓이면 False 이다.

3) 논리곱과 논리합 연산자 or

논리합이라는 or과 | 연산자는 두 항이 모두 거짓이어야 False다.
 하나라도 참이면 True 이다.

논리 자료, 연산



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> True ^ True
False
>>> True ^ False
True
>>> False ^ True
True
>>> False ^ False
False
>>> not True, not False
(False, True)
Ln: 396 Col: 24
```

4) 배타적 논리합 연산자^와 not 연산자

배타적 논리합 연산자인 ^은 두 항이 다르면 True, 같으면 False다.

연산자 not은 뒤에 위치한 논리 값을 바꾼다.



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> not False and True
True
>>> (not False) and True
True
>>> not True or True and False
False
>>> (not True) or (True and False)
False
Ln: 403 Col: 4
```

5) 논리 연산 우선순위 not and or

논리 연산은, not 연산, 논리 곱 and, 논리합 or 순이다.

논리 자료, 연산

```

3-2.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
#전기 사용량의 기본 요금 계산
usage = float(input('가정의 전기 사용량(kWh)은 >> '))
less200 = usage <= 200
less400 = 200 < usage <= 400
greater400 = 400 < usage

base = 730 * less200 + 1260 * less400 + 6060 * greater400
print('전기 사용량(kw): %d, 기본 요금(원): %d' % (usage, base))

Ln: 12 Col: 0
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h03/3-2.py
가정의 전기 사용량(kWh)은 >> 180
전기 사용량(kw): 180, 기본 요금(원): 730
Ln: 411 Col: 4

```

2. 관계 연산

| 연산자 | 의미 | 문자열 관계 연산 |
|----------|--------|-----------------------------|
| $a > b$ | 크다 | 사전 순서에서 뒤에 (코드 값이 크다) |
| $a >= b$ | 크거나 같다 | 사전 순서에서 뒤에 (코드 값이 크다)있거나 동일 |
| $a < b$ | 작다 | 사전 순서에서 앞에 (코드 값이 작다) |
| $a <= b$ | 작거나 같다 | 사전 순서에서 앞에 (코드 값이 작다)있거나 동일 |
| $a == b$ | 같다 | 사전 순서에서 동일 |
| $a != b$ | 다르다 | 사전 순서에서 다름 |

3. 멤버십 연산 in

```

*3-3.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Pyth...
File Edit Format Run Options Window Help
inkey = input('배운 파이썬 키워드를 입력하세요 >>')
keywords = "False", "True", "and", "in", "is", "not", "or"
print('입력 단어 {}, 키워드인가? {}'.format(inkey, inkey in keywords))

Ln: 6 Col: 0
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h03/3-3.py
배운 파이썬 키워드를 입력하세요 >>and
입력 단어 and, 키워드인가? True
Ln: 407 Col: 2

```

1) 메소드 format()를 이용해 출력 처리

멤버의 소속을 알 수 있는 멤버십 연산으로, 결과는 True, False의 논리 값이다.

$x \text{ in } s$: 문자열 s 에 부분 문자열 x 가 있으면 True, 없으면 False

$x \text{ not in } s$: 문자열 s 에 부분 문자열 x 가 없으면 True, 있으면 False

논리 자료, 연산

```

3-4.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
a = int(input('정수 하나를 입력하세요 >> '))
mask = 0b1111
print('정수 {0} 2진수로는 {0:b}'.format(a))
print('가장 오른쪽 4비트: {0:04b} 정수로는 {0}'.format(a & mask))

Ln: 8 Col: 0

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h03/3-4.py
정수 하나를 입력하세요 >> 195
정수 195 2진수로는 11000011
가장 오른쪽 4비트: 0011 정수로는 3

Ln: 412 Col: 89

```

```

3-5.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
num = int(input('이동 연산 num << cnt를 수행할 정수 num 입력 ? '))
cnt = int(input('이동 연산 num << cnt를 수행할 정수 cnt 입력 ? '))

print('{0:032b} {0:8d} : num'.format(num))
print('{2:032b} {2:8d} : {0} << {1}'.format(num, cnt, num << cnt))
print('{2:032b} {2:8d} : {0} * 2 ** {1}'.format(num, cnt, num * 2 ** cnt))

Ln: 10 Col: 0

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h03/3-5.py
이동 연산 num << cnt를 수행할 정수 num 입력 ? 25
이동 연산 num << cnt를 수행할 정수 cnt 입력 ? 3
00000000000000000000000000000001001 25 : num
000000000000000000000000000000011001000 200 : 25 << 3
000000000000000000000000000000011001000 200 : 25 * 2 ** 3

Ln: 449 Col: 43

```

4. 비트 논리 연산

비트 연산은 정수로 저장된 메모리에서 비트와 비트의 연산을 말한다.

| m | n | 논리합 | 논리합 | 배타적 논리합 |
|---|---|-----|-----|---------|
| | | m&n | m n | m^n |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

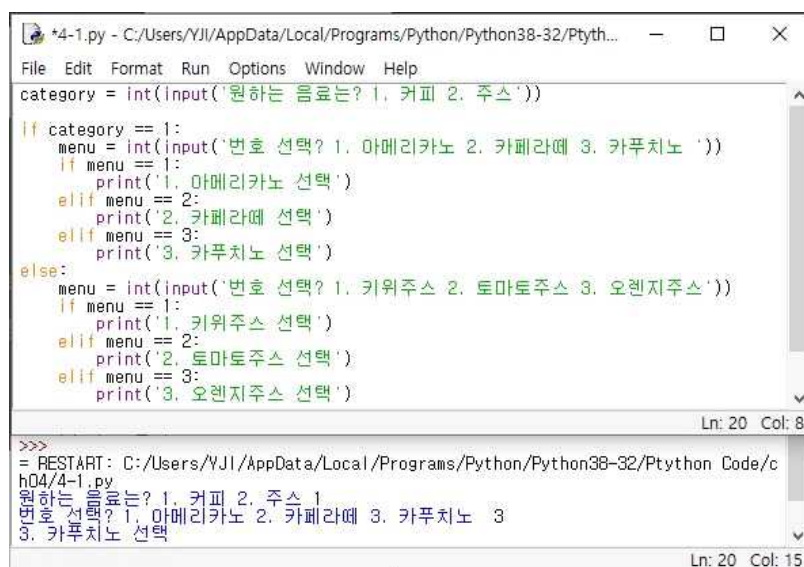
5. 비트 이동 연산

- >> 와 << 는 연산자의 방향인 오른쪽 또는 왼쪽으로 ,
비트 단위로 뒤 피연산자인 지정된 횟수만큼 이동시키는 연산자다.
- <<에 의해 생기는 오른쪽 빈 자리는 모두 0으로 채워지며,
>>에 의한 빈 자리는 원래 정수 부호에 따라
0이나 양수이면 0, 음수이면 1이 채워진다.

04 조건문과 반복문

-
- 4-1 조건문 if ~ else
 - 4-2 반복문 for, while
 - 4-3 난수, 반복 제어 break, continue

조건문 if ~ else



```

*4-1.py - C:/Users/YJl/AppData/Local/Programs/Python/Python38-32/Pyth...
File Edit Format Run Options Window Help
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')

Ln: 20 Col: 8

>>>
= RESTART: C:/Users/YJl/AppData/Local/Programs/Python/Python38-32/Python Code/c
h04/4-1.py
원하는 음료는? 1. 커피 2. 주스 1
번호 선택? 1. 아메리카노 2. 카페라떼 3. 카푸치노 3
3. 카푸치노 선택

Ln: 20 Col: 15
  
```

조건문 if, if ... else, if ... elif

1) if

- 조건인 논리 표현식의 결과가 True이면 이후에 구성된 블록 구문인 문장 1과 문장 2를 실행한다. 결과가 False이면 실행하지 않는다.
- if문에서 논리 표현식 이후에는 반드시 콜론이 있어야 한다.
- 콜론 이후 다음 줄부터 시작되는 블록은 반드시 들여쓰기를 해야 한다.

2)if ... else

- 조건 if의 논리표현식 결과는 True 또는 False이며, if ... else 문에서 else: 블록은 논리 표현식 결과가 False일 때 실행된다.
- if문에서 논리 표현식 결과가 True이면 논리 표현식 콜론 이후 블록을 실행한다
- 논리 표현식의 결과가 False이면 else: 이후의 블록을 실행한다.

3)if ... elif

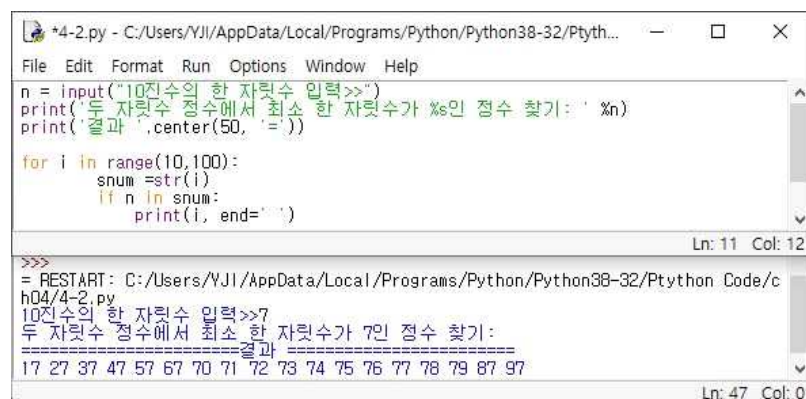
- 논리 표현식 1이 True이면 문장1, 문장 2의 블록을 실행하고 종료된다.
- 논리 표현식 1이 False여야 elif 논리 표현식 2로 진행된다.
- elif는 필요한 만큼 늘릴 수 있으며, 마지막 else는 선택적으로 생략할 수 있다.

반복문 for, while

1. for문

for 변수 in <시퀀스>:

반복 몸체인 _문장들



```

*4-2.py - C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
n = input("10진수의 한 자릿수 입력>>")
print('두 자릿수 정수에서 최소 한 자릿수가 %s인 정수 찾기:' % n)
print('결과', center(50, '='))

for i in range(10,100):
    snum =str(i)
    if n in snum:
        print(i, end=' ')

>>>
= RESTART: C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Python Code/c
h04/4-2.py
10진수의 한 자릿수 입력>>7
두 자릿수 정수에서 최소 한 자릿수가 7인 정수 찾기:
=====결과=====
17 27 37 47 57 67 70 71 72 73 74 75 76 77 78 79 87 97
  
```

for

- 여러 개의 값을 갖는 시퀀스에서 변수에 하나의 값을 순서대로 할당한다.
- 할당된 변수값을 갖고 블록의 문장들(문장 1, 문장 2)을 순차적으로 실행한다.
- 반복 몸체인 문장 1, 문장 2에서 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.

반복문 for, while

1. for문 예제

The screenshot shows a Python IDE window titled '*4-3.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Ptyth...'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

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

The status bar at the bottom right of the code editor indicates 'Ln: 6 Col: 0'. Below the code editor, the output of the program is displayed, showing a grid of multiplication results. The output starts with 'RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Ptython Code/c' and 'h04/4-3.py'. The results are organized by the outer loop variable 'i' (from 2 to 9) and the inner loop variable 'j' (from 1 to 9). The status bar at the bottom right of the output window indicates 'Ln: 61 Col: 19'.

```
>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Ptython Code/c
h04/4-3.py
2 * 1 = 2 4 * 1 = 4 6 * 1 = 6 8 * 1 = 8
2 * 2 = 4 4 * 2 = 8 6 * 2 = 12 8 * 2 = 16
2 * 3 = 6 4 * 3 = 12 6 * 3 = 18 8 * 3 = 24
2 * 4 = 8 4 * 4 = 16 6 * 4 = 24 8 * 4 = 32
2 * 5 = 10 4 * 5 = 20 6 * 5 = 30 8 * 5 = 40
2 * 6 = 12 4 * 6 = 24 6 * 6 = 36 8 * 6 = 48
2 * 7 = 14 4 * 7 = 28 6 * 7 = 42 8 * 7 = 56
2 * 8 = 16 4 * 8 = 32 6 * 8 = 48 8 * 8 = 64
2 * 9 = 18 4 * 9 = 36 6 * 9 = 54 8 * 9 = 72
3 * 1 = 3 5 * 1 = 5 7 * 1 = 7 9 * 1 = 9
3 * 2 = 6 5 * 2 = 10 7 * 2 = 14 9 * 2 = 18
3 * 3 = 9 5 * 3 = 15 7 * 3 = 21 9 * 3 = 27
3 * 4 = 12 5 * 4 = 20 7 * 4 = 28 9 * 4 = 36
3 * 5 = 15 5 * 5 = 25 7 * 5 = 35 9 * 5 = 45
3 * 6 = 18 5 * 6 = 30 7 * 6 = 42 9 * 6 = 54
3 * 7 = 21 5 * 7 = 35 7 * 7 = 49 9 * 7 = 63
3 * 8 = 24 5 * 8 = 40 7 * 8 = 56 9 * 8 = 72
3 * 9 = 27 5 * 9 = 45 7 * 9 = 63 9 * 9 = 81
```

반복문 for, while

2. while문

`while` < 논리 표현식 >:

반복문체인_문장들

`else`:

문장



```

*4-4.py - C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python
File Edit Format Run Options Window Help
MAXNUM =4
MAXHEIGHT =130
more = True
cnt = 0
while more:
    height = float(input("키는 ? "))
    if height < MAXHEIGHT:
        cnt += 1
        print('들어가요. ', '%d명' %cnt)
    else:
        print('커서 못 들어갑니다.')
        if cnt == MAXNUM:
            more = False
        else:
            print('%d명 모두 찾습니다. 다음 번에 이용하세요.' % cnt)
Ln: 18 Col: 12

>>>
= RESTART: C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/c
h04/4-4.py
키는 ? 120
들어가요. 1명
키는 ? 110
들어가요. 2명
키는 ? 125
들어가요. 3명
키는 ? 150
커서 못 들어갑니다.
3명 모두 찾습니다. 다음 번에 이용하세요.
키는 ?
Ln: 144 Col: 9
  
```

while

- while 문은 for문에 비해 간결하며 반복 조건인 논리 조건인 논리 표현식의 값에 따라 반복을 수행한다.
- while 문은 헛스르 정해놓지 않고 어떤 조건이 False가 될 때까지 반복을 수행하는 데 적합하다.
- 논리 표현식이 True이면 반복 몸체인 문장 1, 문장 2를 실행한 후 다시 논리 표현식을 검사해 실행한다.
- 논리 표현식이 False이면 반복 몸체를 실행하지 않고, 선택 사항인 `else`: 이후의 블록을 실행한 후 반복을 종료한다.

난수, 반복 제어 break, continue

```

4-5.py - C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Ptytho...
File Edit Format Run Options Window Help
winnumber = 11, 17, 28, 30, 33, 35

print(' 모의 로또 당첨 번호 '.center(28, '='))
print(winnumber)
print()
print(' 내 번호 확인 '.center(30, '-'))

cnt = 0
import random
for i in range(6):
    n = random.randint(1,45)
    if n in winnumber:
        print(n, 'O', end = ' ')
        cnt += 1
    else:
        print(n, 'X', end = ' ')
print()
print(cnt, '개 맞음')

Ln: 22 Col: 0

>>>
= RESTART: C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Ptython Code/c
h04/4-5.py
===== 모의 로또 당첨 번호 =====
(11, 17, 28, 30, 33, 35)

----- 내 번호 확인 -----
6 X 4 X 1 X 35 O 29 X 44 X
1 개 맞음

Ln: 152 Col: 16

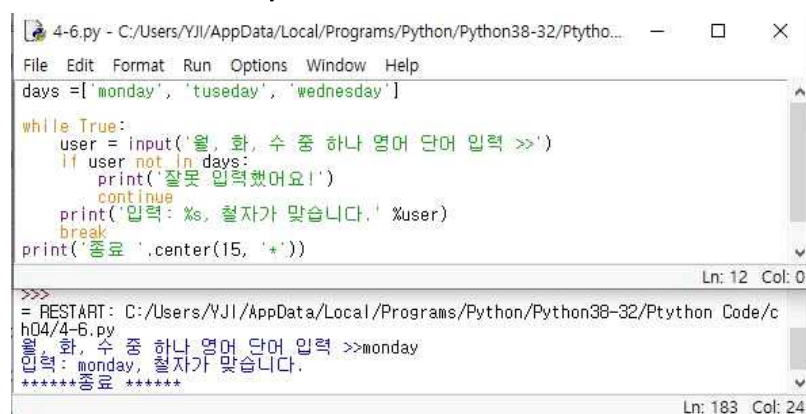
```

1. 난수

- 모듈 random의 함수 randint(시작, 끝)를 사용해 정수 시작과 끝 수 사이에서 임의의 정수를 얻을 수 있다. (시작과 끝을 모두 포함한다.)
- random.ranfint(1,3)는 import random으로 모듈 활용을 선언한 후 1, 2, 3 중 한가지 수를 임의로 얻을 수 있다.

난수, 반복 제어 break, continue

2. 반복 제어 break, continue



```

4-6.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
days = ['monday', 'tuesday', 'wednesday']

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, '*'))

Ln: 12 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h04/4-6.py
월, 화, 수 중 하나 영어 단어 입력 >>monday
입력: monday, 철자가 맞습니다.
*****종료*****
Ln: 183 Col: 24

```

1) break문

반복 while이나 for문에서 break 문장은 반복을 종료하고 반복 문장 이후를 실행한다.

특정한 조건에서 즉시 반복을 종료할 경우에 사용된다.

break문으로 반복이 종료된느 경우, 반복의 else: 블록은 실행되지 않는다.

2) contiune

반복 for와 while문 내부에서 continue 문장은 이후의 반복 몸체를 실행하지 않고

다음 반복을 위해 논리 조건을 수행한다.

05 항목의 리스트, 튜플

5-1 리스트 개요

5-2 리스트 부분 참조와 항목 삽입, 삭제

5-3 튜플

리스트 개요

```

5-1.py - C:\Users\YJ\AppData\Local\Programs\Python\Python38-32\...
File Edit Format Run Options Window Help
coffee = ['에스프레소', '아메리카노', '카페라떼', '카페모카']
print(coffee)
print(type(coffee))

num = 0
for s in coffee:
    num += 1
    print('%d, %s' % (num, s))

Ln: 11 Col: 0

>>>
= RESTART: C:\Users\YJ\AppData\Local\Programs\Python\Python38-32\Python Code\c
h05\5-1.py
['에스프레소', '아메리카노', '카페라떼', '카페모카']
<class 'list'>
1. 에스프레소
2. 아메리카노
3. 카페라떼
4. 카페모카
Ln: 6 Col: 14

```

1. 리스트의 개념

리스트는 항목의 나열인 시퀀스다.

리스트는 콤마로 구분된 항목들의 리스트로 표현되며, 각 항목은 모두 같은 자료형을
항목 순서는 의미가 있으며, 항목 자료값은 중복돼도 상관없다.

리스트는 대괄호 [] 사이에 항목을 기술한다.

[항목1, 항목2, 항목3, ...] # 리스트

```

5-1_2.py - C:/Users/YJ/AppData/Local/Programs/Python/Python38-32/Pty...
File Edit Format Run Options Window Help
goods = []
for i in range(3):
    item = input('구입할 품목은? ')
    goods.append(item)
    print(goods)
print('길이: %d' % len(goods))

Ln: 9 Col: 0

>>>
= RESTART: C:/Users/YJ/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05\5-1_2.py
구입할 품목은? 과자
['과자']
구입할 품목은? 우유
['과자', '우유']
구입할 품목은? 세면도구
['과자', '우유', '세면도구']
길이: 3
Ln: 22 Col: 12

```

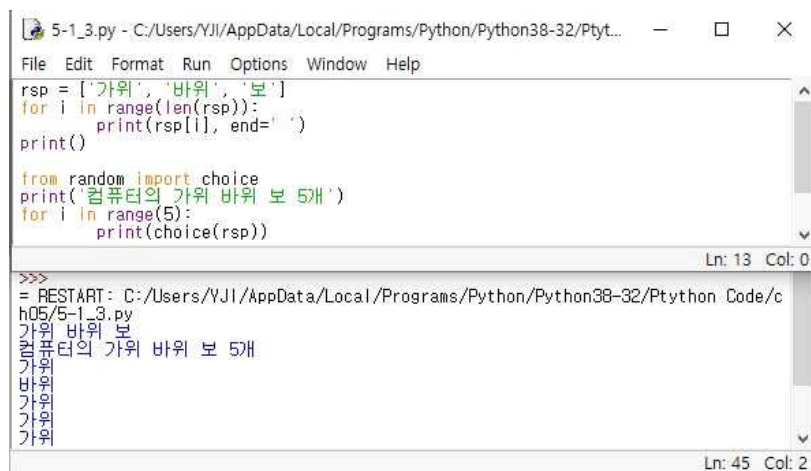
2. 빈 리스트 생성

빈 대괄호로 빈 리스트를 만들 수 있다.

list() 함수 : 문자열의 항목인 문자로 구성되는 리스트로 변환된다.

len() 함수 : 리스트의 항목 수를 알 수 있다.

리스트 개요



```
5-1_3.py - C:/Users/VJI/AppData/Local/Programs/Python/Python38-32/Ptyt...
File Edit Format Run Options Window Help
rsp = ['가위', '바위', '보']
for i in range(len(rsp)):
    print(rsp[i], end=' ')
print()

from random import choice
print('컴퓨터의 가위 바위 보 5개')
for i in range(5):
    print(choice(rsp))

Ln: 13 Col: 0

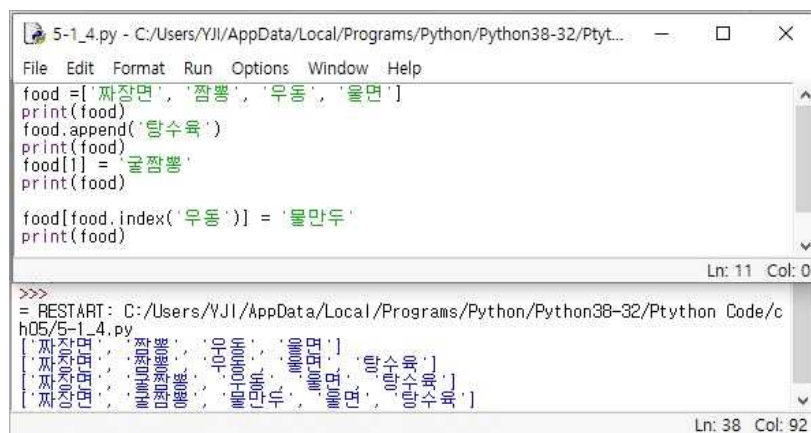
>>>
= RESTART: C:/Users/VJI/AppData/Local/Programs/Python/Python38-32/Ptython Code/c
h05/5-1_3.py
가위 바위 보
컴퓨터의 가위 바위 보 5개
가위
바위
가위
가위
가위
Ln: 45 Col: 2
```

3. 리스트의 항목 참조

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 $[\text{len}(\text{시퀀스})-1]$ 까지, -1에서 $[-\text{len}(\text{시퀀스})]$ 까지 가능하다.

리스트 개요



```
5-1_4.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Ptyt...
File Edit Format Run Options Window Help

food = ['짜장면', '짬뽕', '우동', '물면']
print(food)
food.append('항수육')
print(food)
food[1] = '굴짬뽕'
print(food)

food[food.index('우동')] = '물만두'
print(food)

Ln: 11 Col: 0

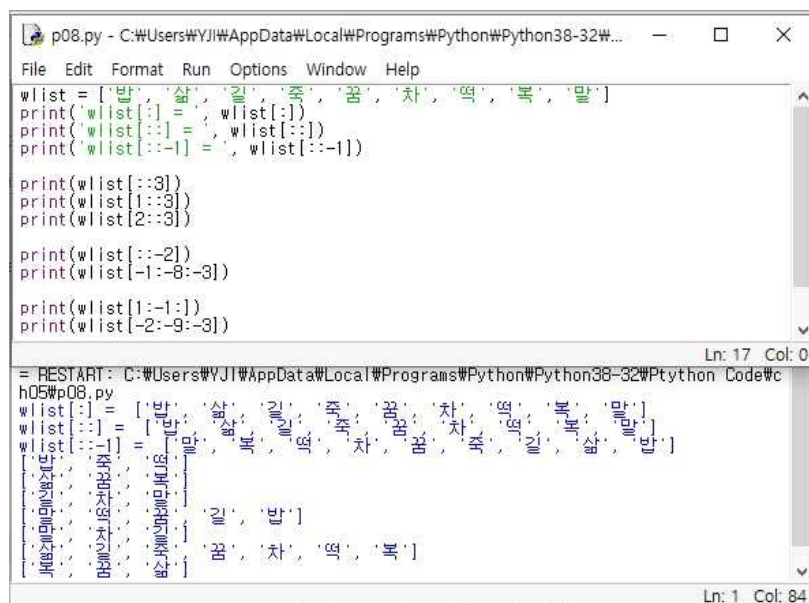
>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05/5-1_4.py
['짜장면', '짬뽕', '우동', '물면']
['짜장면', '짬뽕', '우동', '물면', '항수육']
['짜장면', '굴짬뽕', '우동', '물면']
['짜장면', '굴짬뽕', '물만두', '물면']

Ln: 38 Col: 92
```

4. 리스트의 항목 수정

- index(값)는 인자인 값의 항목이 위치한 첨자를 반환한다.
- 리스트의 첨자를 이용한 항목을 대입 연산자의 오른쪽에 위치시켜 리스트의 항목을 수정할 수 있다.
- 첨자는 유효한 것이어야 하므로 적어도 하나 이상의 항목이 있는 경우에 수정할 수 있다.

리스트 부분 참조와 항목 삽입, 삭제



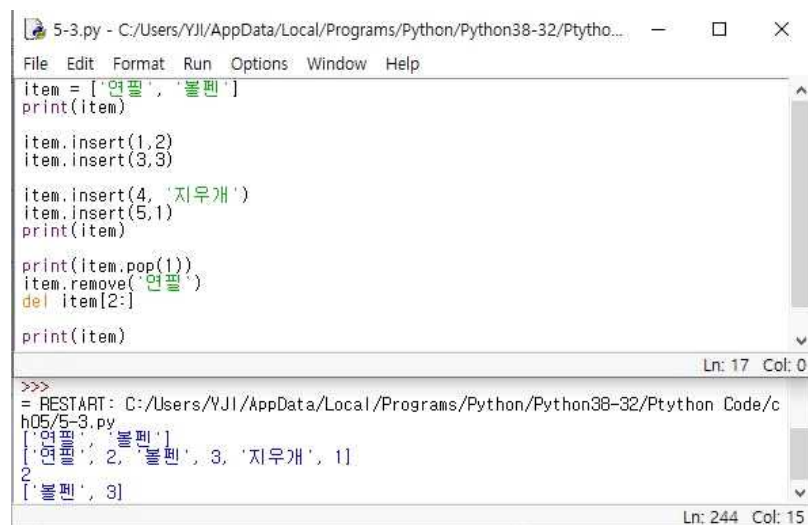
1. 리스트 부분 참조 슬라이싱

슬라이스: 리스트[start: stop: step]

첨자 start에서 첨자 stop-1까지 step 간격으로 부분 리스트 반환

- 0 에서 시작하는 오름차순(수가 차례로 늘어가는 것)
- 마지막 요소 -1에서 시작하는 내림차순의 첨자도 가능하다.
- 다소 복잡하지만 두 순차의 첨자를 함께 사용 가능

리스트 부분 참조와 항목 삽입, 삭제

A screenshot of a Python IDE window titled '5-3.py'. The code defines a list 'item' with elements '연필' and '볼펜'. It then performs several operations: inserting '지우개' at index 2, inserting '1' at index 3, popping the element at index 1, removing the element '연필', and deleting the slice from index 2 onwards. The output shows the list after each operation. The IDE interface includes a menu bar (File, Edit, Format, Run, Options, Window, Help) and a status bar at the bottom showing 'Ln: 244 Col: 15'.

```
5-3.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
File Edit Format Run Options Window Help
item = ['연필', '볼펜']
print(item)

item.insert(1,2)
item.insert(3,3)

item.insert(4, '지우개')
item.insert(5,1)
print(item)

print(item.pop(1))
item.remove('연필')
del item[2:]

print(item)

Ln: 17 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05/5-3.py
['연필', '볼펜']
2
['연필', 2, '볼펜', 3, '지우개', 1]
2
['볼펜', 3]

Ln: 244 Col: 15
```

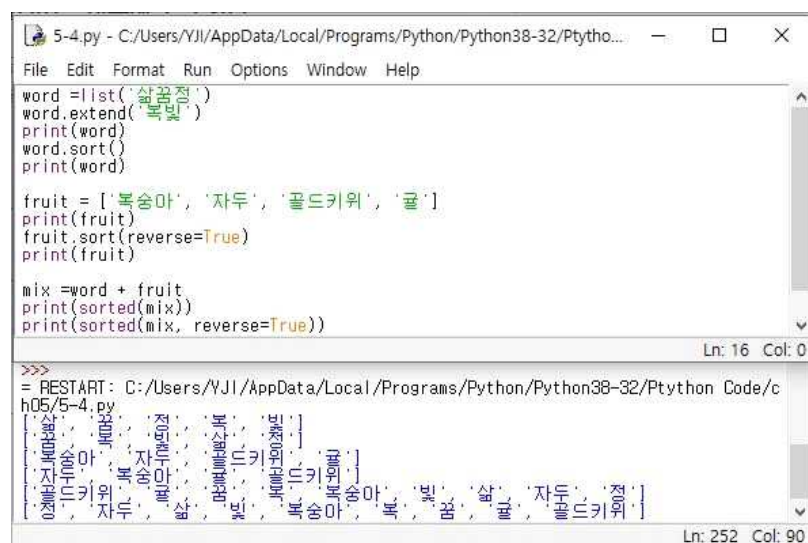
2. 리스트 항목 삽입, 삭제

리스트.insert(첨자, 항목) : 리스트의 첨자 위치에 항목을 삽입

리스트.remove(값) : 리스트에서 지정된 값의 항목을 삭제한다.

리스트.pop(첨자) : 지정된 첨자의 항목을 삭제하고 반환한다.

리스트 부분 참조와 항목 삽입, 삭제



```
5-4.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help

word = list('살곶늻')
word.extend('복날')
print(word)
word.sort()
print(word)

fruit = ['복숭아', '자두', '골드키위', '귤']
print(fruit)
fruit.sort(reverse=True)
print(fruit)

mix = word + fruit
print(sorted(mix))
print(sorted(mix, reverse=True))

Ln: 16 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05/5-4.py
['살곶늻복날']
['곶', '늻', '살']
['복숭아', '자두', '골드키위', '귤']
['복숭아', '자두', '골드키위', '귤']
['곶', '늻', '살', '복숭아', '자두', '골드키위', '귤']
['곶', '늻', '살', '복숭아', '자두', '골드키위', '귤']

Ln: 252 Col: 90
```

3. 리스트 항목의 순서와 정렬

reverse() : 항목 순서를 반대로 뒤집는다.

sort() : 리스트 항목의 순서를 오름차순으로 정렬한다.

sort(reverse = True)로 호출하면 역순인 내림차순으로 정한다.

sorted() : 리스트의 항목 순서를 오름차순으로 정렬한 새로운 리스트를 반환한다.

리스트 부분 참조와 항목 삽입, 삭제

```

5-5.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
File Edit Format Run Options Window Help
a = []
for i in range(10):
    a.append(i)
print(a)

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

s = []
for i in range(10):
    if i % 2 == 1:
        s.append(i**2)
print(s)

squares = [i**2 for i in range(10) if i % 2 == 1]
print(squares)

Ln: 19 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05/5-5.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 9, 25, 49, 81]
[1, 9, 25, 49, 81]

Ln: 258 Col: 4

```

4. 리스트 컴프리헨션

리스트를 만드는 간결한 방법을 제공한다.

```
even = []
```

```
even = [i for i in range(2, 11, 2)]
```

```
for i in range(2, 11, 2):
```

```
    even.append(i)
```

```
print(even)
```

```
print(even)
```

```
[2, 4, 6, 8, 10]
```

```
[2, 4, 6, 8, 10]
```


튜플

```

5-6.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
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[_]))

Ln: 15 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/c
h05/5-6.py
('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
Ln: 266 Col: 12

```

1. 튜플 생성, 참조

튜플은 문자열, 리스트와 같은 항목의 나열인 시퀀스다.

튜플은 리스트와 달리 수정이 불가하다

튜플도 모두 콤마로 구분된 항목들의 리스트로 표현된다.

1) 튜플 생성

```
empty1 = ()
```

```
empty2 = tuple()
```

```
credit = (16, 17)
```

2) 튜플 참조

```
tuple[index[]]
```

```
tuple[start:stop:step]
```

튜플

2. 튜플 연결과 반복, 정렬과 삭제

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> kpop = ('BTS', '블랙핑크')
>>> num = (7, 4)
>>> print(kpop + num)
('BTS', '블랙핑크', 7, 4)
>>> days = ('1학기', '2학기')
>>> print(days * 4)
('1학기', '2학기', '1학기', '2학기', '1학기', '2학기', '1학기', '2학기')
Ln: 315 Col: 18
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> fruit = ('사과', '귤', '복숭아', '파인애플')
>>> tup = sorted(fruit)
>>> print(type(tup), tup)
<class 'list'> ['귤', '복숭아', '사과', '파인애플']
>>> print(sorted(fruit, reverse=True))
['파인애플', '사과', '복숭아', '귤']
>>> print(fruit)
('사과', '귤', '복숭아', '파인애플')
Ln: 346 Col: 4
```

```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> kpop = ('BTS', '장범준', '블랙핑크', '잔나비')
>>> del kpop
>>> print(kpop)
Traceback (most recent call last):
  File "<pyshe11#2B>", line 1, in <module>
    print(kpop)
NameError: name 'kpop' is not defined
Ln: 337 Col: 37
```

1) 튜플 연결 + 연산자와 반복 * 연산자

리스트와 같이 +와 *는 각각 튜플을 연결하고
항목이 횟수만큼 반복된 튜플을 반환한다.

2) 튜플 항목의 순서를 정렬한 리스트를 반환하는 내장 함수 sorted()

튜플 항목의 순서를 오름차순으로 정렬한 새로운 리스트를 반환한다.

sorted(튜플, reverse=True) : 역순이 내림차순으로 정렬되리스트를 반환한다.

3) 문장 del에 의한 튜플 변수 자체의 제거

문장 del에서 변수 kpop을 지정하면 변수 자체가 사라진다.

삭제 이후에는 변수 자체가 메모리에서 제거되므로 참조하면 오류가 발생한다.

06 문자열과 논리 연산 2

6-1 딕셔너리

6-2 집합

6-3 함수 `zip()`, `enumerate()`, 시퀀스 간의 변화

딕셔너리



```

6-1.py - C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/ch...
File Edit Format Run Options Window Help
bts1 = {'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준'}
bts1['소속사'] = '빅히트 엔터테인먼트'
print(bts1)
bts2 = dict(['그룹명', '방탄소년단'], ['인원수', 7])
print(bts2)
bts3 = dict([('리더', '김남준'), ('소속사', '빅히트 엔터테인먼트')])
print(bts3)

bts = dict(그룹명 = '방탄소년단', 인원수 = 7, 리더 = '김남준', 소속사 = '빅히트 엔터테인먼트',
구성원 = ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국'])
print(bts)
print(bts['구성원'])

Ln: 16 Col: 0

>>>
= RESTART: C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/ch06/6-1.py
{ '그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔터테인먼트' }
{ '그룹명': '방탄소년단', '인원수': 7 }
{ '인원수': 7, '그룹명': '방탄소년단' }
{ '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' }

Ln: 15 Col: 4

```

1. 딕셔너리 생성

1) 딕셔너리는 키와 값의 쌍인 항목을 나열한 시퀀스다.

- 딕셔너리는 중괄호{ } 사이에 키와 값의 항목을 기술한다.
- 딕셔너리의 항목 순서는 의미가 없으며, 키는 중복될 수 없다.
- 키는 수정될 수 없지만, 값은 수정될 수 있다.

```
lect = {}
```

2) 함수 dict()로 생성하는 딕셔너리

- 내장 함수 dict() 인자로 리스트나 튜플 1개를 사용해 딕셔너리를 만들 수 있다.

```
day = dict([])
```

```
day = dict(['월', 'monday'], ['화', 'tuesday'])
```

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

2. 딕셔너리 참조

딕셔너리 값은 대괄호를 사용한 딕셔너리[키]로 해당 키의 값을 참조할 수 있다.

```
lect['개설년도']
```

딕셔너리

```

6-2.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/
File Edit Format Run Options Window Help
season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
print(season.keys())
print(season.items())
print(season.values())

for key in season.keys():
    print('%s %s' % (key, season[key]))

for item in season.items():
    print('{} {}'.format(item[0], item[1]), end=' ')
print()

for item in season.items():
    print('{} {}'.format(*item), end=' ')
print()

Ln: 14 Col: 27

= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-2.py
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

Ln: 128 Col: 21

```

3. 딕셔너리 항목의 순회

- keys() : 키로만 구성된 리스트를 반환한다.
- items() : (키, 값) 쌍으로 튜플이 들어 있는 리스트를 반환한다.
각 튜플의 첫 번째 항목은 키, 두 번째 항목은 키 값이다.
- values() : 값으로 구성된 리스트를 반환한다.
- for문에서 시퀀스 위치에 있는 딕셔너리 변수만으로도 모든 키를 순회 할 수 있다.
game = dict(일월='소나무', 이월='매화', 삼월='벚꽃')
for key in game:
 print('%s: %s' % (key, game[key]))

딕셔너리

```

6-3.py - C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help
color = dict(검은색='black', 흰색='white', 녹색='green', 파란색='blue')
print(color)

print(color.get('녹색'))
print(color.get('노란색'))
print()

color['노란색'] = 'yellow'
print(color)
print()

c = '흰색'
print('삭제: %s %s' % (c, color.pop('흰색')))
print(color)
c = '빨간색'
print('삭제: %s %s' % (c, color.pop(c, '없어요'))

print('임의 삭제: {}'.format(color.popitem()))
print('임의 삭제 후: {}'.format(color))

c = '검은색'
del color[c]
print('{} 삭제 후: {}'.format(c, color))

color.clear()
print(color)

Ln: 29 Col: 0

>>>
= RESTART: C:/Users/YJL/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-3.py
{'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
{'검은색': 'black', '노란색': 'yellow'}
임의 삭제 후: ('노란색', 'yellow')
{'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
색 삭제 후: {'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
{}

Ln: 182 Col: 33

```

4. 딕셔너리 항목의 참조와 삭제

get(키) : 키의 해당 값을 반환한다.

딕셔너리에 키가 없어도 오류가 발생하지 않고 None을 반환한다.

pop(키) : 키인 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.

삭제할 키가 없다면 두 번째에 지정한 값이 반환된다.

키만 적었을 경우 키가 없다면 오류가 발생한다.

popitem() : 임의 (키, 값)의 튜플을 반환하고 삭제한다.

만일 데이터가 하나도 없다면 오류가 발생한다.

clear() : 기존의 모든 키:값 항목을 삭제한다.

집합

```
>>> s1 = {1, 2, 3, 4, 5}
>>> s2 = {'py', 'java', 'go'}
>>> s3 = {(1,1), (2,4), (3,9)}
```

1. 집합 개념

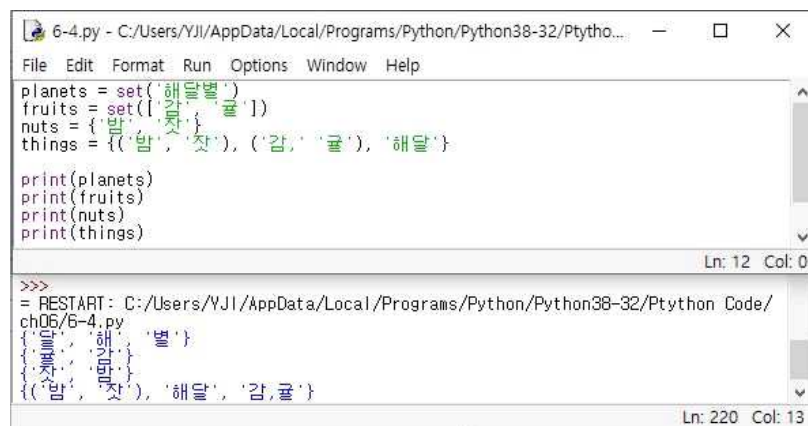
집합은 수학과 같이 원소를 콤마로 구분하며 중괄호로 둘러싸 표현한다.

- 원소는 불변 값으로 중복될 수 없으며 서로 다른 값이어야 한다.
- 원소는 중복을 허용하지 않으며 원소의 순서는 의미가 없다.

2. 내장 함수 set()

set(원소로 구성된 리스트_튜플_문자열)

- 인자가 없으면 빈 집합인 공집합이 생성된다.
- 인자가 있으면 하나이며, 리스트와 튜플, 문자열 등이 올 수 있다.



```
6-4.py - C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Pytho...
File Edit Format Run Options Window Help

planets = set('해달별')
fruits = set(['감', '귤'])
nuts = {'밤', '잰'}
things = {('밤', '잰'), ('감', '귤'), '해달'}

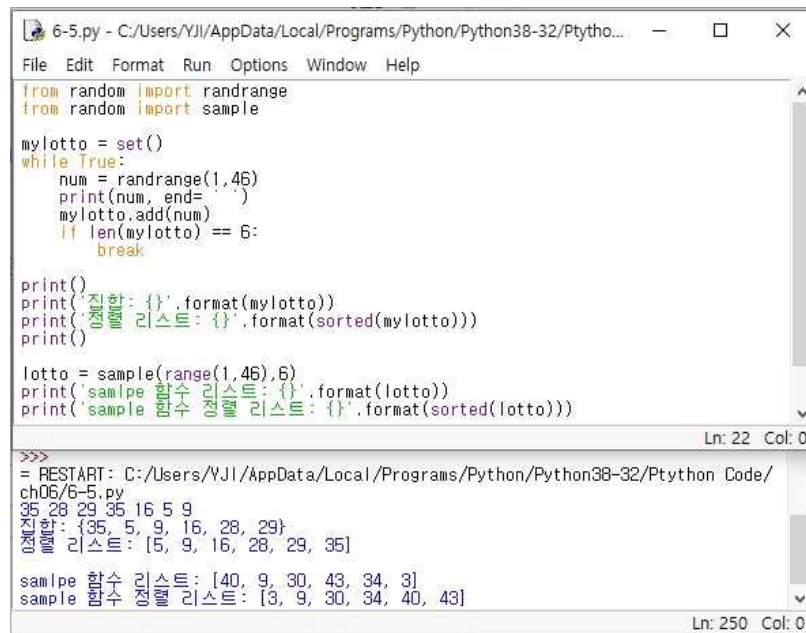
print(planets)
print(fruits)
print(nuts)
print(things)

Ln: 12 Col: 0

>>>
= RESTART: C:/Users/YJ1/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-4.py
{'별'}
{'감', '귤'}
{'밤', '잰'}
{('밤', '잰'), '해달', '감', '귤'}
```

집합

3. 집합의 원소 추가, 삭제



```

6-5.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code
File Edit Format Run Options Window Help

from random import randrange
from random import sample

mylotto = set()
while True:
    num = randrange(1,46)
    print(num, end=' ')
    mylotto.add(num)
    if len(mylotto) == 6:
        break

print()
print('집합: {}'.format(mylotto))
print('정렬 리스트: {}'.format(sorted(mylotto)))
print()

lotto = sample(range(1,46),6)
print('sample 함수 리스트: {}'.format(lotto))
print('sample 함수 정렬 리스트: {}'.format(sorted(lotto)))

Ln: 22 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-5.py
35 28 29 35 16 5 9
집합: {35, 5, 9, 16, 28, 29}
정렬 리스트: [5, 9, 16, 28, 29, 35]

sample 함수 리스트: [40, 9, 30, 43, 34, 3]
sample 함수 정렬 리스트: [3, 9, 30, 34, 40, 43]

Ln: 250 Col: 0

```

1) 원소 추가

add(원소) :
 odd = {1, 3, 5}
 odd.add(7)

2) 항목 삭제

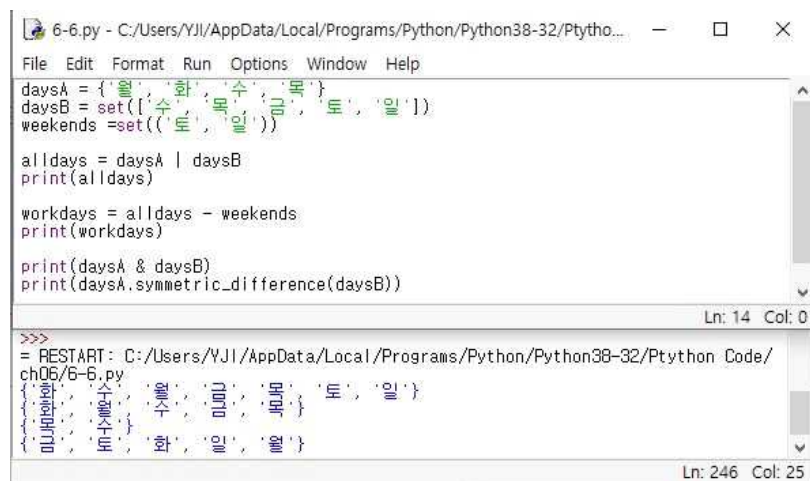
| | | |
|-----------------|-----------------|------------------|
| remove(원소) : | discard(원소) : | pop() : |
| 원소 삭제 | 원소 삭제 | 임의의 원소 삭제 |
| 원소 없을 시 오류 | 원소 없을 시 오류 발생 X | |
| odd = {1, 3, 5} | odd = {1, 3, 5} | odd = {1, 3, 5} |
| odd.remove(3) | odd.discard(3) | print(odd.pop()) |

3) 모든 원소 삭제

clear() :
 집합의 모든 원소를 삭제
 odd = {1, 3, 5}
 odd.clear()

집합

4. 합집합, 교집합, 차집합, 여집합



```

6-6.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help
daysA = {'월', '화', '수', '목'}
daysB = set(['수', '목', '금', '토', '일'])
weekends = set(['토', '일'])

alldays = daysA | daysB
print(alldays)

workdays = alldays - weekends
print(workdays)

print(daysA & daysB)
print(daysA.symmetric_difference(daysB))

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-6.py
{'월', '화', '수', '목', '금', '토', '일'}
{'수', '목'}
{'월', '화', '수', '목', '금', '토', '일'}
{'토', '일'}
  
```

1) 합집합

연산자 |, 메소드 union(), 축약 대입 연산자 |=

2) 교집합

연산자 &, 메소드 intersection(), 축약 대입 연산자 &=

3) 차집합

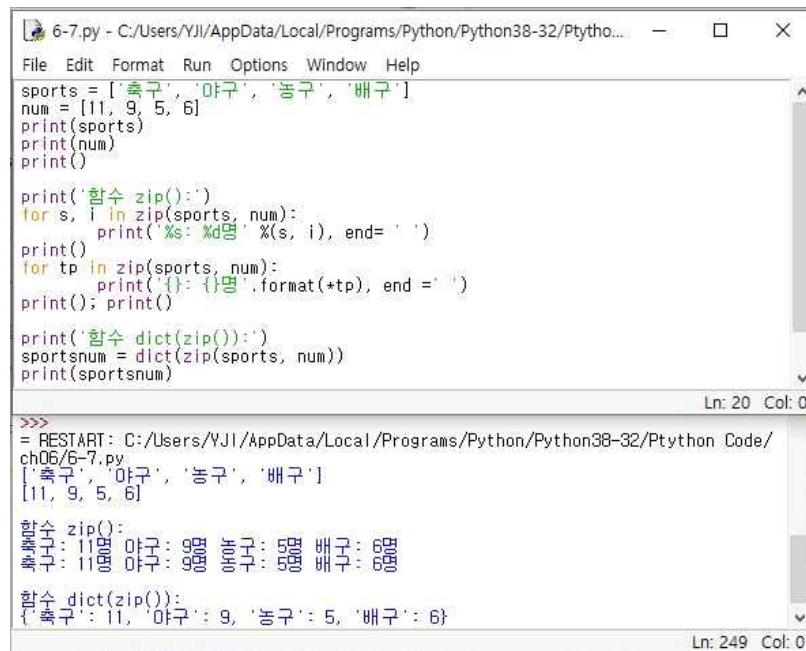
연산자 -, 메소드 difference(), 축약 대입 연산자 -=

4) 여집합

연산자 ^, 메소드 symmetric_difference(), 축약 대입 연산자 ^=

함수 zip(), enumerate()

1. 함수 zip(), dict()



```

6-7.py - C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python...
File Edit Format Run Options Window Help

sports = ['축구', '야구', '농구', '배구']
num = [11, 9, 5, 6]
print(sports)
print(num)
print()

print('함수 zip():')
for s, i in zip(sports, num):
    print('%s: %d명' % (s, i), end=' ')
print()
for tp in zip(sports, num):
    print('{}: {}'.format(*tp), end=' ')
print(); print()

print('함수 dict(zip()):')
sportsnum = dict(zip(sports, num))
print(sportsnum)

Ln: 20 Col: 0

>>>
= RESTART: C:/Users/YJI/AppData/Local/Programs/Python/Python38-32/Python Code/
ch06/6-7.py
['축구', '야구', '농구', '배구']
[11, 9, 5, 6]

함수 zip():
축구: 11명 야구: 9명 농구: 5명 배구: 6명
{'축구': 11, '야구': 9, '농구': 5, '배구': 6}

함수 dict(zip()):
{'축구': 11, '야구': 9, '농구': 5, '배구': 6}

Ln: 249 Col: 0
  
```

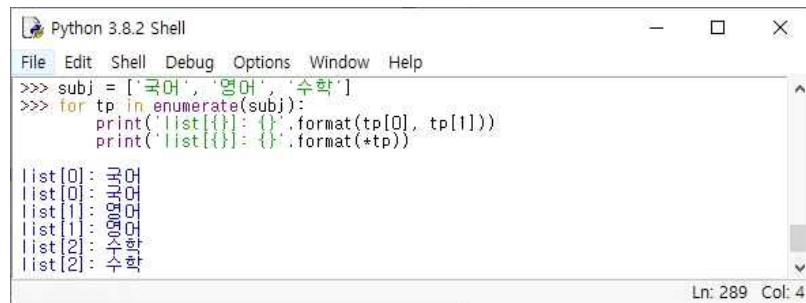
1) 함수 zip()

zip() : 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를
 각각의 리스트로 묶어 주는 역할을 하는 함수다.
 zip()의 결과는 자료형 zip이다.
 자료형 zip은 리스트나 튜플로 변환할 수 있다
 list(zip(a, b)).

1) 함수 dict()

dict() : zip()의 인자 2개를 사용하면,
 앞은 키, 뒤는 값의 조합이 구성돼 딕셔너리가 생성된다.
 dict(zip(a, b))
 zip() 인자들의 길이가 같지 않더라도 짧은 쪽의 인자에 맞는
 키:값의 항목을 구성하고 다른 쪽의 더 긴 부분은 무시한다.
 dict(zip('abcd', 'XY'))
 딕셔너리의 키:값의 항목이므로 zip(키, 값)처럼 인자가 2개여야 한다.

함수 zip(), enumerate()



```
Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
>>> subj = ['국어', '영어', '수학']
>>> for tp in enumerate(subj):
    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
Ln: 289 Col: 4
```

2. 함수 enumerate()

enumerate() : for 반복의 시퀀스에 사용하는 것이 좋다.

변수 tp로 지정하면 tp[0]은 첨자, tp[1]은 값이 된다.

format() 메소드에서 *tp로 지정하면 자동으로 나뉘어
앞부분의 {}에는 첨자, 뒤에{}은 값이 출력된다.

후기

이번 포트폴리오 과제를 만들면서 제가 직접 설명하는 입장이 되어
또 다른 방법으로 공부를 해 볼 수 있어 매우 좋은 시간을 보낼 수 있었습니다.
또한 지금 까지 대면수업에 익숙해져 있어 사이버 강의에 적응이 되지 않아
집중하기 힘들고 들어도 잘 이해가 가지 않는 부분들을
제가 직접 하나 하나 코드를 실습 해보고 설명하여
각 개념들을 이해하는데 많은 도움이 되었습니다.
시간이 오래 걸렸고 어려운 부분들도 있었지만 뜻 깊은 시간이였습니다.
감사합니다.

감사합니다.