

# 파이썬 포트폴리오

20161897 정윤교

# 목차

## 1.파이썬 소개

- 컴퓨팅 사고력이란?

## 2.문자열과 산술연산자 다루기

- 예제 코드를 통한 설명

## 3.문자열과 논리연산 다루기

- 예제 코드를 통한 설명

## 4.조건문과 반복문

- 조건문

- 반복문

- 예제 코드를 통한 설명

## 5.리스트와 튜플

- 리스트

- 튜플

- 예제코드와 설명

## 6.딕셔너리와 집합

- 딕셔너리

- 집합

- 예제코드와 설명

## 1.파이썬 소개

파이썬 언어는 오픈소스 프로그래밍 언어로서, 1991년 네덜란드의 귀도 반 로섬 이 개발한 언어이다. 현재는 비영리 파이썬 소프트웨어 재단에서 관리하고 있다.

파이썬은 비전공자의 **컴퓨팅 사고력**을 키우기 위한 프로그래밍 언어로 많이 활용되고 있다.

C, Java와 비교해서 배우기 쉽고 간결하며, 개발속도가 빠르고 강력한 특징이 있다. 이러한 특징덕분에 다양한 라이브러리와 다양한 개발환경을 제공하고 있어서 쉽고 빠른 소프트웨어 개발에 큰 도움을 주고 있다. 또한 프로그래밍 교육 분야뿐 아니라 실무에서의 사용도 급증하고 있어 스택오버플로 에서 ‘가장 빠르게 성장하는 프로그래밍 언어’로 선정되기도 했다.

파이썬의 특징을 다시 정리하면, 문법이 다른 언어에 비해 간단해 배우기 쉬우며, 다양한 자료구조의 제공으로 개발속도가 빠르고 생산성이 높으며, 다양한 라이브러리 제공을 통한 데이터과학과 머신 러닝 분야에 적용하기 적합하다는 특징이 있다.

이중 데이터과학과 머신 러닝 분야에 적용하기 적합한 특징은 4차 산업혁명의 실무분야에 적합하기 때문에 더욱더 각광받고 있다.

### -컴퓨팅 사고력이란?

4차 산업혁명 시대 인재의 핵심 역량으로 문제 해결능력, 창의·융합 사고능력, 의사소통 능력, 협업능력, 자기주도 학습능력을 요구한다.

컴퓨팅 사고력은 이러한 4차 산업혁명 시대의 핵심 역량을 모두 포함한 개념으로 볼 수 있는데, 컴퓨팅 사고력은 컴퓨터 과학 원리와 개념을 활용해 자신의 영역과 융합할 수 있는 역량을 의미한다. ‘컴퓨터 과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용해 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현해 적용할 수 있는 능력’으로 정의 할 수 있다.

컴퓨팅 사고력의 구성요소로 영국의 컴퓨팅 교육에서 분해, 패턴인식, 추상화, 알고리즘을 제안했다. 최근 초·중·고등학교에서는 프로그래밍 교육이 이루어지고 있는데, 이러한 컴퓨팅 사고력 능력을 키우는 방법 중 하나로 직접 프로그래밍을 하는 코딩 교육이 있기 때문이다.

## 2. 문자열과 산술연산자 다루기

문자열과 산술연산자는 다른 언어에서도 기초적으로 다루는 내용이다. 이번 차례에서는 도전! 프로그래밍의 문제 몇 개를 보고, 어떤 코드가 사용되었는지 설명한다.

### -예제 코드를 통한 설명

(1) 두 문자열을 표준 입력으로 받아 한 줄에 출력하는 프로그램 작성하기

<pre>File Edit Format Run Opt str1=input('문자열 1: ') str2=input('문자열 2: ') print(str1, str2)</pre>	<pre>File Edit Shell Debug O Python 3.8.2 (tags/v3.8. tel)] on win32 Type "help", "copyright" &gt;&gt;&gt; ===== RESTART: 문자열 1: python 문자열 2: 언어 python 언어 &gt;&gt;&gt;  </pre>
---	--

`str1`과 `str2` 라는 변수를 선언해서 `input()` 함수를 통해 표준입력을 받고 있다.

이때, `input()` 함수에는 '문자열 1:' 과 '문자열 2:' 가 존재하는데, 왼쪽과 같이 코드를 구성하면, 오른쪽 결과 창에서, '문자열 1:'을 출력 한 이후 입력을 기다리게 된다. 즉, `input()`은 표준 입력을 받는 함수로 이해할 수 있다.

파이썬에서는 `print()` 함수로 내용의 출력을 다루는데, 왼쪽의 코드에서는 `str1`과 `str2` 변수에 입력받은 문자열을 한 줄에 출력하기 위해 `print(str1, str2)` 로 코드가 구성되었음을 알 수 있다.

(2) 아메리카노를 주문받아 총 가격을 출력하는 프로그램 작성하기

<pre>File Edit Format Run Options Window Help count = int(input('아메리카노 몇 개 주문하세요? ')) price = count * 3500 print('총 가격은', price, '이다.')</pre>	<pre>File Edit Shell Debug Options V Python 3.8.2 (tags/v3.8.2:7b3ab tel)] on win32 Type "help", "copyright", "cred &gt;&gt;&gt; ===== RESTART: E:\Pyth 아메리카노 몇 개 주문하세요? 2 총 가격은 7000 이다. &gt;&gt;&gt;</pre>
---	--

이번 문제에는 아메리카노의 가격이 3500원 이라는 조건이 붙어있다.

이번 문제의 해결을 위해서 산술연산자 \* 가 사용되었다. 총 가격에 해당하는 `price` 변수에 `count * 3500`을 저장하도록 코드가 구성되어 있다.

이때, `count` 변수는 `input()` 함수로 입력받은 값을 `int()` 함수를 통해 정수 형태로 저장하도록 한다.

(1)과 마찬가지로 출력을 위해 `print()` 함수가 사용되었다.

(3) 섭씨 온도를 입력받아 화씨 온도로 변환하는 프로그램 작성하기

File Edit Format Run Options Window Help	File Edit Shell Debug Options Wind
<pre>c = int(input('온도 입력 &gt;&gt; ')) f_Ac = c * (9/5) + 32 f_Srt = c * 2 + 30 print('정확 계산: 섭씨:', c, ', 화씨:', f_Ac) print('약식 계산: 섭씨:', c, ', 화씨:', f_Srt) print('차이:', f_Ac - f_Srt)</pre>	<pre>Python 3.8.2 (tags/v3.8.2:7b3ab59, tel)] on win32 Type "help", "copyright", "credits" &gt;&gt;&gt; ===== RESTART: E:\Python_ 온도 입력 &gt;&gt; 38 정확 계산: 섭씨: 38 , 화씨: 100.4 약식 계산: 섭씨: 38 , 화씨: 106 차이: -5.5999999999999994 &gt;&gt;&gt;</pre>

이번 문제는 식을 주며 두 개의 조건을 제시했다.

- 식 : 화씨 = 섭씨\*(9/5)+32, 섭씨 = (화씨-32)\*(5/9)
- 조건1 : 식을 사용해 정확한 계산을 수행
- 조건2 : 화씨 = 섭씨\*2+30 으로 약식계산과 차이 출력

다양한 산술연산자가 나오는데, C언어의 printf() 함수에서처럼 함수 내에서의 계산이 허용되는 것을 f\_Ac - f\_Srt로 확인할 수 있다.

지금까지 설명 한 것과 마찬가지로 c 변수에 input() 함수로 입력받은 내용을 int() 함수로 정수형태 저장할 수 있고, f\_Ac 변수와 f\_Srt 변수에 c 변수를 이용한 계산식을 저장해, 두 가지의 계산결과를 출력하고, 계산결과에 나온 차이를 계산해 보여주는 내용이다.

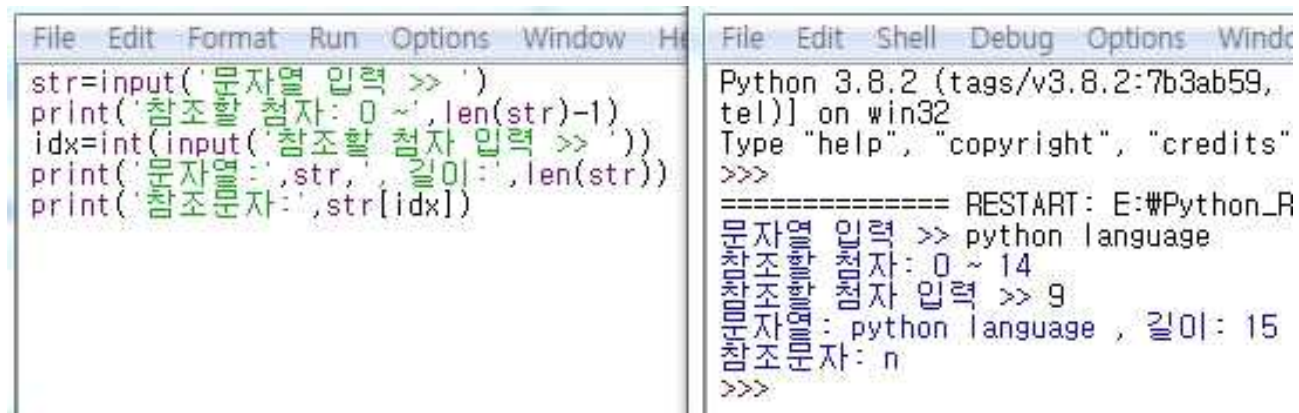
### 3. 문자열과 논리연산 다루기

앞서 설명한 문자열을 다루는 방법에서 더 나가 파이썬의 특징을 통해 다루는 방법과 다른 언어에서도 다루는 논리연산을 다루는 내용이다.

앞선 차례와 마찬가지로 **도전! 프로그래밍**의 문제 몇 개를 보고, 어떤 코드가 사용되었는지 설명한다.

#### -예제 코드를 통한 설명

(1) 한 줄의 문자열을 표준입력으로 받아 문자열의 소속 문자를 참조할 범위를 출력하고, 다시 첨자 하나를 입력받아 문자열의 전체와 길이 그리고 참조문자를 출력하는 프로그램 작성하기

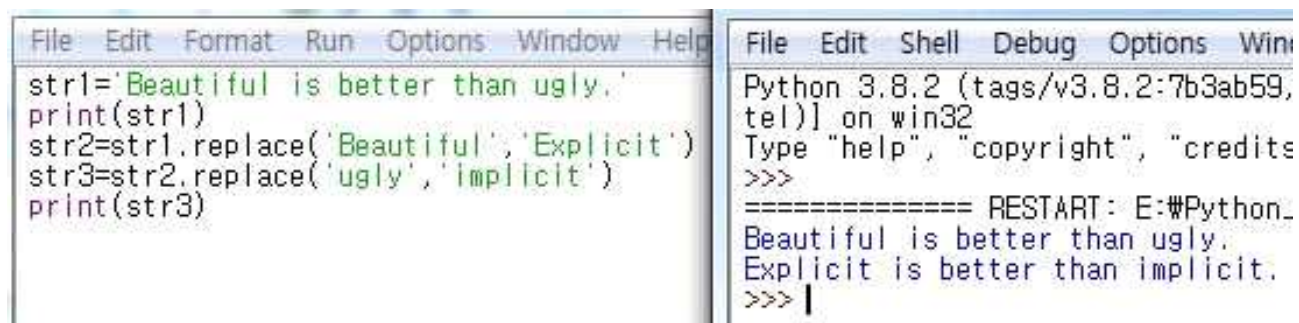


```
File Edit Format Run Options Window Help
str=input('문자열 입력 >> ')
print('참조할 첨자: 0 ~',len(str)-1)
idx=int(input('참조할 첨자 입력 >> '))
print('문자열:',str,' 길이:',len(str))
print('참조문자:',str[idx])

File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59,
tel)] on win32
Type "help", "copyright", "credits"
>>>
===== RESTART: E:\Python_R
문자열 입력 >> python language
참조할 첨자: 0 ~ 14
참조할 첨자 입력 >> 9
문자열: python language , 길이: 15
참조문자: n
>>>
```

새로운 함수 **len()** 함수가 나왔다. **len()** 함수는 문자열의 길이를 참조하는 함수인데, **str** 문자열 변수를 인자로 받아서 길이를 참조하고 있다. 이때 **-1을 한 이유**는 보이는 문자열로 범위제한을 위해서이다. 그리고 **str** 변수에 **[idx]**를 통해 **idx**의 숫자에 해당하는 첨자를 참조하는데, 문자열은 '문자의 나열'로 보기 때문에 문자열에서 해당 첨자에 해당하는 문자를 참조 할 수 있는 것이다.

(2) 파이썬에 철학을 저장한 후, **replace()** 메소드를 이용해 다른 철학으로 다시 저장해 출력하는 프로그램 작성하기



```
File Edit Format Run Options Window Help
str1='Beautiful is better than ugly.'
print(str1)
str2=str1.replace('Beautiful','Explicit')
str3=str2.replace('ugly','implicit')
print(str3)

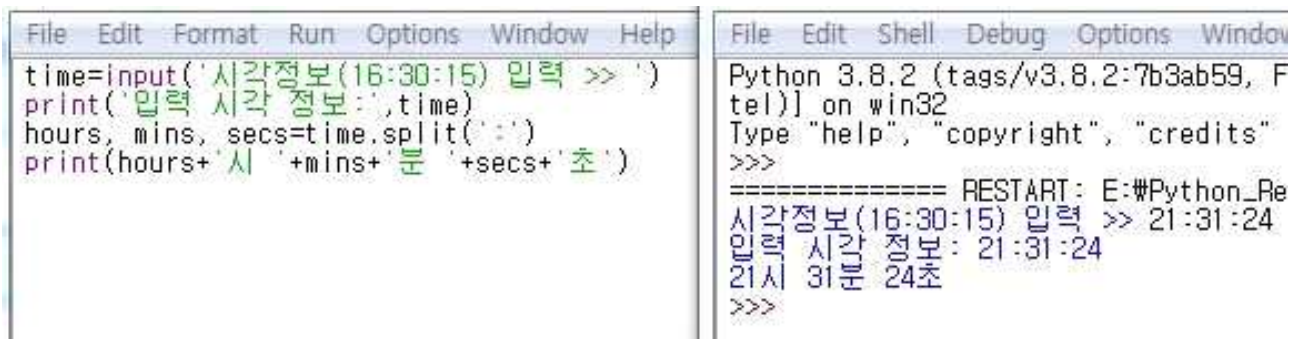
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59,
tel)] on win32
Type "help", "copyright", "credits"
>>>
===== RESTART: E:\Python_
Beautiful is better than ugly.
Explicit is better than implicit.
>>> |
```

**replace()** 메소드를 사용하여 저장된 문자열을 바꾸는 코드이다.

**str1** 변수에 **Beautiful is better than ugly**를 저장했는데, **replace()**를 이용해 일부분을 바꿨다.

**replace()** 메소드는 **.replace(a,b)** 형태로 사용되는데, **a** 에는 바꾸려는 문자열을, **b** 에는 **a** 대신 들어갈 문자열을 입력해 사용 할 수 있다. 두 번의 변경을 통해 **str3**에 내용을 저장하고, 출력하는 내용이다.

(3) 문자열의 `split()` 메소드를 사용해 '14:21:45'와 같은 시각 정보를 표준입력으로 받아 입력된 문자열을 출력하고, 다시 시, 분, 초 형태로 출력하는 프로그램 작성하기



The screenshot shows a Python IDE with two windows. The left window displays the following code:

```
time=input('시각정보(16:30:15) 입력 >> ')
print('입력 시각 정보:',time)
hours, mins, secs=time.split(':')
print(hours+'시 '+mins+'분 '+secs+'초')
```

The right window shows the execution output:

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, F
tel)] on win32
Type "help", "copyright", "credits"
>>>
===== RESTART: E:\Python_Re
시각정보(16:30:15) 입력 >> 21:31:24
입력 시각 정보: 21:31:24
21시 31분 24초
>>>
```

`split()` 메소드는 문자열을 여러 문자열로 나누는데 사용하는 메소드이다.

`split()` 메소드의 인자로 특정 문자열을 주면, 해당 문자열을 기준으로 문자열이 나누어지는데, 입력을 21:31:24 형태로 받아 `split(':')`를 통해 콜론(:)을 기준으로 나누어 각각 `hours`, `mins`, `secs` 변수에 저장한 뒤, `print()` 함수에서 출력하는 모습을 보여주고 있다.

+ 연산자의 경우, 문자열에서 사용되는 경우, 각각의 문자열을 연결해 주는 역할을 수행한다.

#### 4.조건문과 반복문

다른 언어에서 조건문과 반복문을 비중 있게 다룬다. 파이썬에서는 다른 언어와 크게 다르지 않은 방법으로 조건문과 반복문을 사용하는데, 다른 언어에서는 {중괄호}를 통해 구분하지만 파이썬에서는 {중괄호}를 통한 구분 없이 들여쓰기를 통한 구분을 하기 때문에 주의를 요한다.

앞선 차례와 마찬가지로 **도전! 프로그래밍**의 문제 몇 개를 보고, 어떤 코드가 사용되었는지 설명한다.

## -조건문

조건문은 if문으로 시작되는 특정 조건에 해당하면 동작하도록 작성된 코드를 말한다. if문에는 논리표현식이 따르는데, 논리표현식의 결과가 True인 경우에 if문에 작성된 코드가 동작한다.

논리표현식의 결과가 False인 경우에 뒤의 문장이 없는 경우 조건문이 종료되고, if else 의 형태로 조건문이 구성된 경우 if문의 논리표현식의 결과가 False인 경우 else를 수행한다.

만약 조건을 여러 개 가져야 하는 경우에는 if elif 의 형태로 조건문을 구성하게 되는데, elif문에는 else 와 달리 논리표현식이 따른다.

## 마무리

반복문에는 for문과 while문이 존재하는데, 각각 동작방식이 조금 다르다.

for문은 횟수를 제한하는 반복에서 주로 사용되고, while문은 반복조건에 맞다면 무한정 반복된다.

for문은 for i in <sequence> 의 형태로 사용되는데, 여기서 <sequence> 는 다른 언어에서의 반복조건으로 이해하면 편하다. 약간 다른 점으로 <sequence> 에는 문자열이나 튜플이 올 수 있는데, 문자열이 온 경우 문자열의 문자를 하나씩 i 가 받는다.

튜플의 경우 이후에 자세히 다루겠지만 지금은 수의 나열이라 이해하면 편하다. `for i in 1, 2, 3` 형태로 받는 경우인데, 이렇게 작성하면 `i`는 반복을 수행 할 때마다 1, 2, 3의 숫자를 각각 받게 된다.

### -예제 코드를 통한 설명

(1) 조건을 참고해 월(month)을 표준 입력으로 입력받아 계절을 출력하는 프로그램 작성하기

```
File Edit Format Run Options Window
month=int(input('월 입력? '))
if ((0 < month and month < 4) or (10 < month and month < 12)):
    print('%d월 겨울'%(month))
elif (3 < month and month < 6):
    print('%d월 봄'%(month))
elif (5 < month and month < 9):
    print('%d월 여름'%(month))
elif (8 < month and month < 11):
    print('%d월 가을'%(month))
else:
    print('%d월 잘못된 입력'%(month))
```

```
Python 3.8.2 (tags/3.8.2:1001f644f, Oct 30 2019, 16:03:20) on win32
Type "help", "copyright()", "credits()" and "quit()" for more
>>>
===== RE
월 입력? 5
5월 봄
>>>
===== RE
월 입력? 7
7월 여름
>>>
===== RE
월 입력? 13
13월 잘못된 입력
>>>
```

주어진 조건은 다음과 같다.

- 조건 : 봄:4.5월, 여름:6.7.8월, 가을:9.10월, 겨울:11.12.1.2.3월

if elif 형태의 조건문으로 구성된 조건문이다. 주어진 조건에 맞춰 논리표현식이 존재하고, 마지막 else 문으로 예외처리를 했다.



(2) 다음을 참고해 1에서 99까지의 난수인 임의의 정수 3개를 대상으로 가장 큰 정수를 출력하는 프로그램 작성하기

```

File Edit Format Run Options Window Help
from random import randint
r1, r2, r3 = randint(1,99), randint(1,99), randint(1,99)
if (r1 >= r2) and (r1 >= r3):
    print('%d %d %d 중에서 최대: %d' % (r1, r2, r3, r1))
elif (r2 >= r1) and (r2 >= r3):
    print('%d %d %d 중에서 최대: %d' % (r1, r2, r3, r2))
elif (r3 >= r1) and (r3 >= r2):
    print('%d %d %d 중에서 최대: %d' % (r1, r2, r3, r3))

Python 3.8.2 (tags/v3.8.2
tel)] on win32
Type "help", "copyright",
>>>
===== RESTART: E
71 91 37 중에서 최대: 91
>>>
===== RESTART: E
2 89 99 중에서 최대: 99
>>>

```

참고하라고 주어진 내용은 다음과 같다

- `from random import randint`

난수를 발생시키는 `random` 모듈의 `randint()` 함수를 사용하여 조건에 맞게 처리하는 내용의 코드이다. `r1, r2, r3` 변수에 `randint(1,99)` 함수로 1부터 99까지의 수 중에서 임의의 숫자 하나를 저장하여 if elif 조건문에서 크기비교를 수행한다.

오른쪽의 결과는 두 번째 논리표현식과 세 번째 논리표현식의 결과이다.

(3) 다음을 참고해 섭씨온도 20도에서 41도까지 3도씩 증가하면서 화씨로 변환해주는 프로그램 작성하기

```

File Edit Format Run Options Window Help
for i in range(20,42,3):
    F = i * (9 / 5) + 32
    f = i * 2 + 30
    print('섭씨: %d 화씨: %.1f 화씨(약식): %d 차이: %.2f' % (i, F, f, abs(F-f)))

Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, ;
tel)] on win32
Type "help", "copyright", "credits" or "license(
>>>
===== RESTART: E:Python_Result#Subject1
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
>>>

```

참고할 내용은 다음과 같다

- 섭씨온도에서 화씨온도로의 변환 공식은 [화씨 = 섭씨\*(9/5)+32]
- 약식계산 (섭씨\*2+30)으로 계산
- 정식계산과 약식계산의 차이를 내장함수 `abs()`(인자)로 출력

`for`문을 이용한 반복문이다. 20도에서 41도까지의 섭씨온도 표시를 위해 내장함수 `range()`를 `for`문에 이용했다. `range()` 함수에 20,42,3 이 인자로 전달되었는데, 20부터 시작해서 3의 간격으로 41까지로 해석할 수 있다. 두 번째 인자 42는 두 번째 인자 42미만 까지라는 의미를 가지고 있다.

다른 내장함수로 `abs()`가 사용되었는데, `abs()`는 절댓값을 출력하는 내장함수이다.

계산결과인 정식계산-약식계산이 음수 값으로 나오는데, 이를 절댓값으로 표현해주기 위해 사용되었다.

차이에 출력되는 절댓값에 소수점 두 자리가 있는데, `%.2f`의 형태로 받았기 때문에 출력된다.

`%.2f`는 `float`형으로 소수점 두 번째 자리까지 출력한다는 의미를 가지고 있다.

## 5.리스트와 튜플

이번 차례에서는 리스트와 튜플에 대한 내용을 정리한다. 리스트와 튜플은 여러 항목을 하나의 항목으로 묶어서 관리하는 자료형으로, 항목의 나열 시퀀스로 볼 수 있다.

앞선 차례에서는 **도전! 프로그래밍**의 내용을 다루었으나, 리스트와 튜플은 몇 가지 **예제코드**를 통해 설명한다.

### -리스트

리스트는 여러 항목을 [대괄호] 안에 콤마로 구분하여 작성, 출력 하는 시퀀스이다.

각 항목의 자료형이 모두 같은 자료형일 필요는 없으며, 순서에는 의미가 있다. 그리고 값이 중복돼도 상관없다.

앞선 내용에서 문자열을 다루었는데, 문자열은 시퀀스이고, 그 문자열을 첨자를 활용해서 접근해보았다. 리스트도 시퀀스이기 때문에, 첨자를 통한 접근이 가능하다.

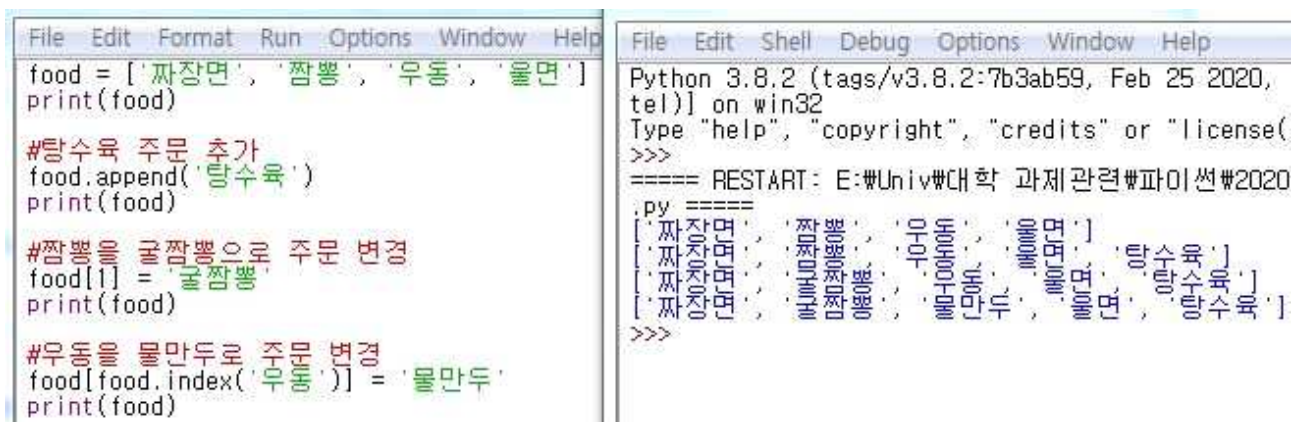
### -튜플

튜플은 리스트와 같은 시퀀스이다. 다른 점은 항목의 순서나 내용의 수정이 불가능하다.

리스트에서와 마찬가지로 각 항목의 자료형이 같을 필요가 없으며, (괄호) 안에 콤마로 구분하여 항목이 작성, 출력된다.

### -예제코드

(1) 중국집에서 음식 주문하기



```
File Edit Format Run Options Window Help
food = ['짜장면', '짬뽕', '우동', '물면']
print(food)

#탕수육 주문 추가
food.append('탕수육')
print(food)

#짬뽕을 굴짬뽕으로 주문 변경
food[1] = '굴짬뽕'
print(food)

#우동을 물만두로 주문 변경
food[food.index('우동')] = '물만두'
print(food)
```

```
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020,
tel)] on win32
Type "help", "copyright", "credits" or "license(
>>>
===== RESTART: E:\Univ\대학 과제 관련\파이썬\2020
.py =====
[['짜장면', '짬뽕', '우동', '물면'], '탕수육']
[['짜장면', '굴짬뽕', '우동', '물면'], '탕수육']
[['짜장면', '굴짬뽕', '물만두', '물면'], '탕수육']
>>>
```

리스트를 이용한 예제코드이다.

**food** 리스트를 생성하고, **첫 번째 주석** 내용에서 **append()** 메소드를 이용해 **탕수육**을 새로 추가했다.

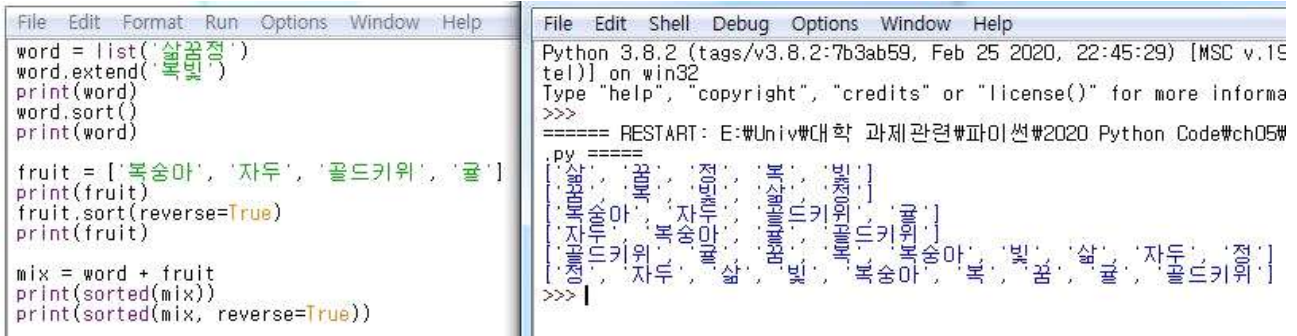
**append()** 메소드는 리스트에 내용을 추가하는 메소드이다.

**두 번째 주석** 내용에서는 리스트의 두 번째 첨자에 있는 내용을 **굴짬뽕**으로 바꾸고 있다.

앞서 리스트는 시퀀스이고, 시퀀스는 첨자를 통해 접근 할 수 있다 설명했는데, 그 과정을 통해 두 번째 첨자의 내용을 수정하고 있다.

**세 번째 주석**의 내용은 **index()** 메소드를 활용해 원하는 내용을 골라서 바꾸는 모습을 보여주고 있다.

## (2) 한 글자 단어와 과일의 정렬



```

File Edit Format Run Options Window Help
word = list('사과', '바나나', '사과', '바나나')
word.extend('사과', '바나나')
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))

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.
>>>
===== RESTART: E:\Univ\대학 과제\관련\파이썬\2020 Python Code\ch05\ch05-13daytuple.py =====
>>>
['사과', '바나나', '사과', '바나나']
['사과', '바나나']
['복숭아', '자두', '골드키위', '귤']
['복숭아', '자두', '골드키위', '귤']
['사과', '바나나', '사과', '바나나', '복숭아', '자두', '골드키위', '귤']
['사과', '바나나', '사과', '바나나', '복숭아', '자두', '골드키위', '귤']
>>>

```

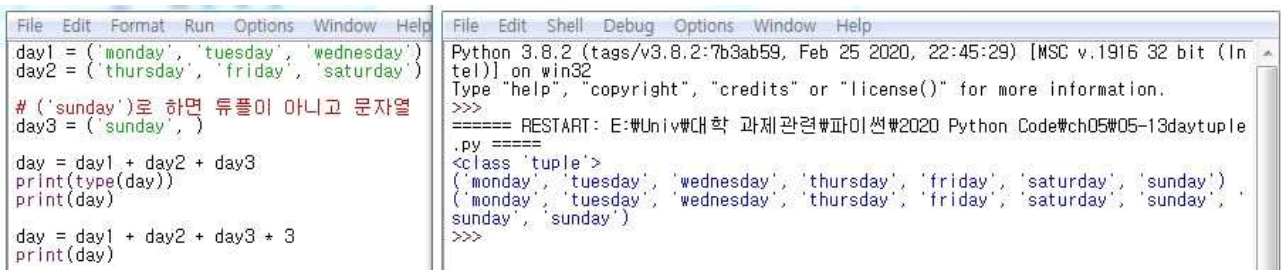
리스트의 다양한 메소드를 활용한 예제이다.

**word** 리스트를 **list()** 함수를 이용한 방법으로 생성하고, **extend()** 메소드를 사용해서 내용을 추가했다. 그렇게 만들어진 **word** 리스트를 **sort()** 메소드를 통해 오름차순으로 정렬했다.

**sort()** 메소드에 인자로 **reverse=True**를 입력하면 내림차순 정렬이 된다.

**mix** 리스트는 **word** 와 **fruit** 리스트를 **+** 연산자로 연결한 리스트인데, 내장함수 **sorted()**를 통해 내용을 정렬해서 보여준다. **sorted()**는 마찬가지로 **reverse=True**를 함께 입력해 내림차순 정렬 할 수 있다.

## (3) 영어 요일 단어로 구성된 튜플 만들기



```

File Edit Format Run Options Window Help
day1 = ('monday', 'tuesday', 'wednesday')
day2 = ('thursday', 'friday', 'saturday')
# ('sunday')로 하면 튜플이 아니고 문자열
day3 = ('sunday',)

day = day1 + day2 + day3
print(type(day))
print(day)

day = day1 + day2 + day3 * 3
print(day)

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.
>>>
===== RESTART: E:\Univ\대학 과제\관련\파이썬\2020 Python Code\ch05\ch05-13daytuple.py =====
>>>
<class 'tuple'>
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday')
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'sunday', 'sunday')
>>>

```

튜플의 생성, 연결, 반복을 보여주는 예제코드이다.

**day1**과 **day2**는 평범하게 생성된 튜플이다. **day3**는 생성할 때 콤마를 찍도록 되어있는데, 주석의 내용대로 **'sunday'** 만 넣어서 작성하면 **day3**는 튜플이 아닌 **sunday** 라는 문자열을 저장하는 문자열 변수가 된다.

두 번째 예제와 마찬가지로 **day**에 **+** 연산자를 이용해 각 튜플을 연결하고, **type()**을 통해서 **day**의 자료형을 확인했다.

다시 **day**에 튜플을 저장할 때 **day3\*3**의 형태로 **day3**를 세 번 반복시켜 저장했다. 오른쪽 결과에서 **day3**가 세 번 반복된 것을 알 수 있다.

## 6. 딕셔너리와 집합

딕셔너리와 집합에 대한 내용을 정리한다. 앞서 정리한 리스트와 튜플을 이용하는 내용이 많은 시퀀스이며, 딕셔너리와 집합의 내용의 차이가 있기 때문에 다소 어려울 수 있다.

앞선 차례에서처럼 딕셔너리와 집합에 대한 설명을 우선 하고, 몇 가지 예제코드를 통해 설명한다.

### - 딕셔너리

딕셔너리는 키와 값의 쌍으로 이루어진 항목의 나열 시퀀스로 볼 수 있다. 키와 값의 쌍은 키:값 형태로 되어있고, 딕셔너리의 항목들은 키:값 형태로 구성되어있다.

딕셔너리는 {중괄호} 안에 작성, 출력되며, 값은 중복될 수 있어도 키는 중복될 수 없다. 첨자를 통한 접근이 아닌, 키를 통한 접근을 하며, 딕셔너리의 키로 수정 불가능한 객체가 사용된다.

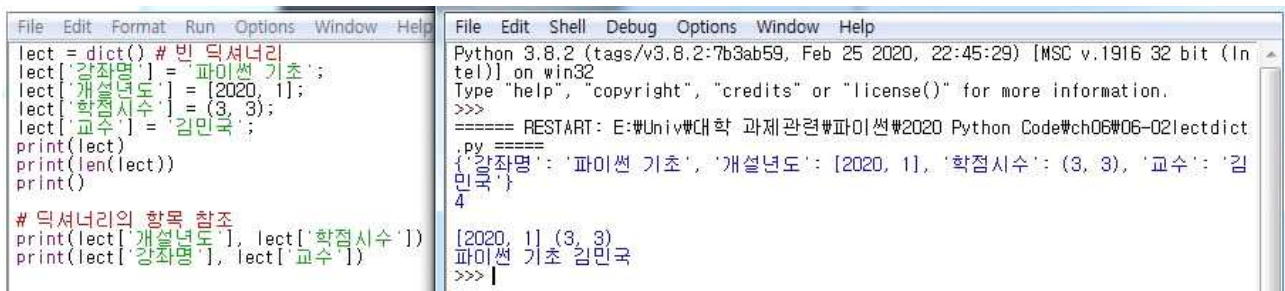
### - 집합

집합은 중복이 없고, 순서가 없는 원소의 모임이다. 중학교 수학의 집합을 생각하면 이해하기 쉽다.

집합은 {중괄호} 안에 작성, 출력되며 항목으로 수정 불가능한 값만이 허용되며, 서로 다른 값으로 구성되어야 한다.

### - 예제코드

(1) 강좌 정보로 구성된 딕셔너리 생성과 참조



```
File Edit Format Run Options Window Help | File Edit Shell Debug Options Window Help
lect = dict() # 빈 딕셔너리
lect['강좌명'] = '파이썬 기초';
lect['개설년도'] = [2020, 1];
lect['학점시수'] = (3, 3);
lect['교수'] = '김민국';
print(lect)
print(len(lect))
print()

# 딕셔너리의 항목 참조
print(lect['개설년도'], lect['학점시수'])
print(lect['강좌명'], lect['교수'])

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.
>>>
===== RESTART: E:\Univ\대학 과제\관련\파이썬\2020 Python Code\ch06\06-02\lectdict.py =====
{'강좌명': '파이썬 기초', '개설년도': [2020, 1], '학점시수': (3, 3), '교수': '김민국'}
4
[2020, 1] (3, 3)
파이썬 기초 김민국
>>>
```

딕셔너리를 생성하고 키 값을 통해 참조하는 내용이다.

lect 라는 이름으로 dict()를 이용해 내용이 없는 빈 딕셔너리를 생성한다.

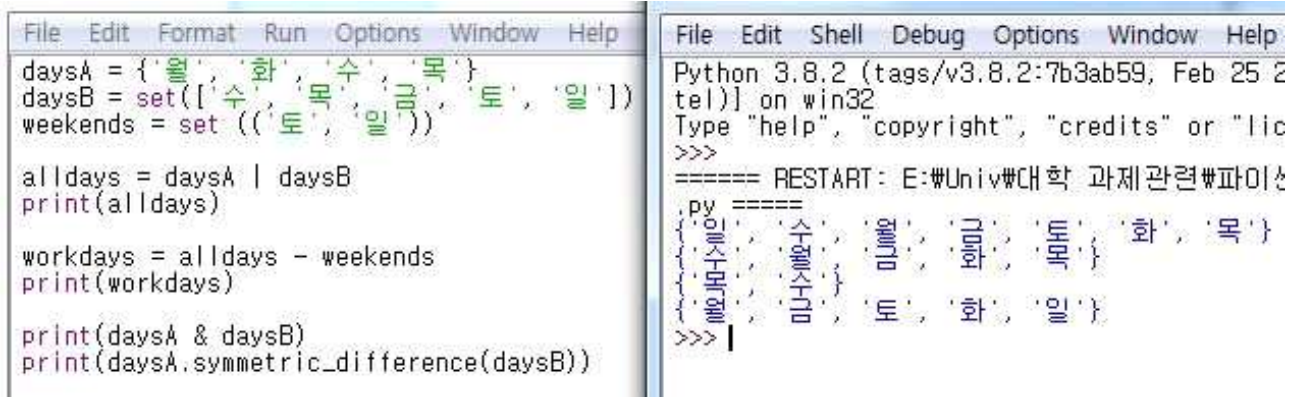
딕셔너리[키]=값 의 형태로 딕셔너리에 항목을 추가할 수 있는데, lect['강좌명'] = '파이썬 기초'; 형태로 입력 딕셔너리에 항목을 추가한다. 또한, 딕셔너리의 값으로 리스트나 튜플이 올 수 있는데, 두 번째와 세 번째 항목삽입이 리스트나 튜플을 값으로 하는 항목의 삽입을 보여주고 있다.

len() 함수를 통해 딕셔너리의 길이를 확인 할 수 있는데, 항목의 개수를 반환해준다.

주석 이후의 내용은 키를 통한 값의 참조를 보여주고 있다.



(2) 요일 문자열 원소로 구성된 집합 연산 수행



The image shows two side-by-side screenshots of a Python IDE. The left screenshot displays the source code for set operations on days of the week. The right screenshot shows the output of the code, including the Python version and the results of the set operations.

```
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))
```

```
File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020) on win32
Type "help", "copyright", "credits" or "license()"
>>>
===== RESTART: E:\Univ\대학 과제관련\파이썬
py =====
{'월', '수', '목', '금', '토', '화', '목'}
{'수', '목'}
{'월', '수', '목'}
{'월', '수', '목', '토', '화', '일'}
>>> |
```

집합의 생성과 합집합, 차집합, 교집합, 여집합 출력을 보여주고 있다.

두 가지 방법의 집합 생성을 보여주고 있다. **daysA**는 일반적인 집합 생성 방법이고, **daysB**와 **weekends**는 **set()** 함수를 이용해서 집합을 생성하고 있다.

**set()** 함수를 이용하면 리스트나 튜플을 이용한 집합 생성이 가능하데, 생성된 집합은 중복이 삭제된 항목 구성을 가진다.

**alldays**는 **daysA** 와 **daysB**의 합집합을 보여주고 있다. 연산자 |를 사용했으며, '수', '목' 두 항목이 중복이 되지만 집합의 특징인 중복은 없다 에 의해 '수', '목' 이 두 번이 아닌 한 번만 출력된다.

**workdays**는 **alldays** 와 **weekend**의 차집합을 보여주고 있다. 연산자 -를 사용하며, 월요일부터 일요일 까지 모든 요일이 있는 **alldays**에서 '토', '일' 이 항목인 **weekend**를 제외한 집합을 출력한다.

**&** 연산자는 교집합 연산자로, **daysA** 와 **daysB** 의 교집합인 '수', '목'을 출력한다.

**symmetric\_difference()** 메소드를 이용해 차집합을 보여주고 있다. **daysA** 와 **daysB** 두 집합에서 교집합이 아닌 내용을 출력한다.

(3) 구기종목과 팀원 수의 리스트에서 딕셔너리 구성

```
File Edit Format Run Options Window Help
# 구기 종목 리스트
sports = ['축구', '야구', '농구', '배구']

# 위 종목에 대응하는 팀원 수를 항목으로 구성
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('{}: {}명'.format(*tp), end = ' ')
print(); print()

# dict()와 zip() 함수로 종목의 이름을 키, 인원-
print('함수 dict(zip()):')
sportsnum = dict(zip(sports, num))
print(sportsnum)

File Edit Shell Debug Options Window Help
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020) on win32
Type "help", "copyright", "credits" or "license()"
>>>
==== RESTART: E:\Univ\대학 과제관련\파이썬\2020\
t.py ====
['축구', '야구', '농구', '배구']
[11, 9, 5, 6]

함수 zip():
축구: 11명 야구: 9명 농구: 5명 배구: 6명
축구: 11명 야구: 9명 농구: 5명 배구: 6명

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

내장함수 `zip()`을 이용해 딕셔너리를 구성하는 내용이다.

내장함수 `zip()`을 이용하면 2개의 리스트나 튜플로 딕셔너리를 생성할 수 있다. 다른 내장함수로 첨자를 자동 생성하여 튜플을 만들어주는 `enumerate()` 함수가 있다.

예제 코드에서는 `sports` 리스트와 `num` 리스트를 생성해서 단순 `zip()` 함수를 이용한 방법과 `dict()` 함수에 내장함수 `zip()`을 이용한 `sportsnum` 딕셔너리 생성방법을 보여주고 있다.

`zip()` 함수를 이용한 두 번째 방법에서 `.format(*tp)`를 이용한 출력을 보여주고 있는데, 코드에서처럼

`for tp in zip(sports, num):` 으로 `tp`에 값을 저장하면 `tp[0]`, `tp[2]` ... 에 `sports`의 항목이 저장되고, `tp[1]`, `tp[3]` ... 에 `num`의 항목이 저장된다.

`format()` 메소드에 `*tp`를 인자로 주면 앞선 문자열의 `{}`의 개수에 맞춰 첨자가 자동으로 나뉘어져서 들어가게 된다.

`dict()` 함수에 `zip()` 함수를 넣어서 `sports`와 `num`을 키와 값으로 가지는 딕셔너리를 생성할 수 있다.