



2020 학년도 1 학기 학생

교과목 포트폴리오

TM

학과	컴퓨터정보공학과
과목	파이썬 프로그래밍
성명	아즈라이
학번	20191753

목차

1. 강의계획서	-----3
2. 머리말	-----8
3. 파이썬 개요	
3.1. 파이썬 언어란?	-----9
3.2. 파이썬 셸의 실행	-----11
3.3. 파이썬 첫 프로그래밍	-----12
4. 파이썬 프로그래밍 기초	
4.1. 문자열과 수	-----13
4.2. 변수와 키워드, 대입 연산자	-----16
4.3. 자료의 표준 입력 및 반환 함수	-----17
5. 문자열과 논리 연산	
5.1. 문자열	-----20
5.2. 문자열 관련 메소드	-----22
5.3. 논리 자료와 다양한 연산	-----27
6. 조건과 반복	
6.1. 조건문	-----30
6.2. 반복문	-----33
6.3. 임의의 수인 난수	-----34
6.4. break 문, continue 문	-----35

7. 리스트와 튜플

7.1.	리스트	-----	36
7.2.	튜플	-----	42

8. 딕셔너리와 집합

8.1.	딕셔너리	-----	44
8.2.	집합	-----	47
8.3.	내장 함수 zip()과 enumerate()	-----	49

9. 맺음말	-----	51
--------	-------	----



2020 학년도 1 학기	전공	컴퓨터정보공학과	학부	컴퓨터공학부
과 목 명	파이썬 프로그래밍(2019009-PD)			

강의실 과 강의시간	수:6(3-217),7(3-217),8(3-217)	학점	3
교과분류	이론/실습	시수	3
담당 교수	강환수 + 연구실 : 2 호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 화요일 13~16		
학과 교육목표			
과목 개요	2010 년 이후 파이썬의 폭발적인 인기는 제 4 차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 컴퓨팅 사고력은 누구나 가져야할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제 4 차 산업혁명 시대의 기술을 이끌고 있다. 제 4 차 산업혁명 시대를 주도하는 핵심 기술은 데이터과학과 머신러닝, 딥러닝이며, 이러한 분야에 적합한 언어인 파이썬은 매우 중요한 언어가 되었다. 본 교과목은 파이썬 프로그래밍의 기초적이고 체계적인 학습을 수행한다. 본 교과목을 통하여 데이터 처리 방법에 대한 효율적인 파이썬 프로그래밍 방법을 학습한다.		
학습목표 및 성취수준	1. 컴퓨팅 사고력의 중요성을 인지하고 4 차 산업혁명에서 파이썬 언어의 필요성을 이해할 수 있다. 2. 기본적인 파이썬 문법을 이해하고 데이터 처리를 위한 자료구조를 이해하여 적용할 수 있다. 3. 문제 해결 방법을 위한 알고리즘을 이해하고 데이터 처리에 적용 할 수 있다. 4. 파이썬 프로그램을 이용하여 실무적인 코딩 작업을 할 수 있다.		
	도서명	저자	출판사
주교재	파이썬으로 배우는 누구나 코딩	강환수, 신용현	홍릉과학출판사
수업시 사용도구	파이썬 기본 도구, 파이참, 아나콘다와 주피터 노트북		
평가방법	중간고사 30%, 기말고사 40%, 과제물 및 퀴즈 10% 출석 20%(학교 규정, 학업성적 처리 지침에 따름)		
수강안내	1. 파이썬의 개발환경을 설치하고 활용할 수 있다. 2. 파이썬의 기본 자료형을 이해하고 조건과 반복 구문을 활용할 수 있다. 3. 파이썬의 주요 자료인 리스트, 튜플, 딕셔너리, 집합을 활용할 수 있다. 4. 파이썬의 표준 라이브러리와 외부 라이브러리를 이해하고 활용할 수 있다. 5. 파이썬으로 객체지향 프로그래밍을 수행할 수 있다.		

1 주차	[개강일(3/16)]
학습주제	교과목 소개 및 강의 계획 1 장 파이썬 언어의 개요와 첫 프로그래밍

목표및 내용	<ul style="list-style-type: none"> 파이썬 언어란 무엇인지 이해하고 이 언어가 인기 있는 이유를 설명할 수 있다. 파이썬 개발 도구를 설치해 프로그램을 구현할 수 있다. 파이썬의 특징과 활용 분야를 설명할 수 있다.
미리읽어오기	교재 1 장, 파이썬 개발환경 설치 파이썬 IDLE
과제,시험,기타	도전 프로그래밍
2 주차	[2 주]
학습주제	2 장 파이썬 프로그래밍을 위한 기초 다지기
목표및 내용	<ul style="list-style-type: none"> 파이썬의 재료인 문자열과 수에 대해 이해하고 코드로 구현할 수 있다. 변수를 이해하고 다양한 대입 연산자를 활용할 수 있다. 표준 입력으로 문자열을 입력받은 후 원하는 자료로 변환해 활용할 수 있다. 파이썬 IDLE 을 활용할 수 있다.
미리읽어오기	교재 2 장 리터럴과 변수의 이해 아나콘다의 주피터 노트북
과제,시험,기타	도전 프로그래밍
3 주차	[3 주]
학습주제	3 장 일상에서 활용되는 문자열과 논리 연산
목표및 내용	<ul style="list-style-type: none"> 문자열에서 문자나 부분 문자열을 반환하는 여러 방법을 구현할 수 있다. 문자열 객체에 소속된 다양한 메소드를 이해하고 활용할 수 있다. 논리 값을 이해하고 다양한 연산을 사용해 실생활에서의 표현에 활용할 수 있다. 아나콘다의 주피터 노트북을 활용할 수 있다.
미리읽어오기	교재 3 장 문자열과 논리연산 파이참(pycharm)
과제,시험,기타	도전 프로그래밍
4 주차	[4 주]
학습주제	4 장 일상생활과 비유되는 조건과 반복
목표및 내용	<ul style="list-style-type: none"> 조건에 따라 하나를 결정하는 if 문을 구현할 수 있다. 반복을 수행하는 while 문과 for 문을 구현할 수 있다. 임의의 수인 난수를 이해하고 반복을 제어하는 break 문과 continue 문을 활용할 수 있다. 파이참(pycharm)을 활용할 수 있다.
미리읽어오기	교재 4 장 조건과 반복
과제,시험,기타	도전 프로그래밍

5 주차	[5 주]
학습주제	5 장 항목의 나열인 리스트와 튜플

목표및 내용	<ul style="list-style-type: none"> • 다양한 종류의 항목을 쉽게 나열하는 리스트를 구현할 수 있다. • 리스트에서 부분 참조 방법, 이를 이용한 수정, 리스트 연결, 삽입과 삭제 그리고 리스트 컴프리헨션 등을 구현할 수 있다. • 수정할 수 없는 다양한 종류의 항목 나열을 쉽게 처리하는 튜플을 구현할 수 있다.
미리읽어오기	교재 5 장 배열과 리스트
과제,시험,기타	도전 프로그래밍
6 주차	[6 주]
학습주제	6 장 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합
목표및 내용	<ul style="list-style-type: none"> • 키와 값의 쌍인 항목을 관리하는 딕셔너리를 생성하고 수정하는 방법을 이해하고, 다양한 방법으로 딕셔너리를 구현할 수 있다. • 집합의 특징을 이해하고, 합집합 등과 같은 다양한 집합의 연산을 구현할 수 있다. • 내장 함수 zip()과 enumerate(), 시퀀스 간의 변환을 이해하고, 구현할 수 있다.
미리읽어오기	교재 6 장 집합
과제,시험,기타	도전 프로그래밍
7 주차	[7 주]
학습주제	7 장 특정 기능을 수행하는 사용자 정의 함수와 내장 함수
목표및 내용	<ul style="list-style-type: none"> • 함수의 내용과 필요성을 이해하고 함수를 직접 정의해 호출할 수 있다. • 인자의 기본 이해와 기본값 지정, 가변 인수와 키워드 인수를 활용할 수 있다. • 간편한 람다 함수와 표준 설치된 내장 함수를 사용할 수 있다.
미리읽어오기	교재 7 장 함수의 정의와 호출
과제,시험,기타	도전 프로그래밍
8 주차	[중간고사]
학습주제	- 직무수행능력평가 1 차(중간고사)
목표및 내용	직무수행능력평가, 서술형 평가
미리읽어오기	교재 1 장에서 7 장까지
과제,시험,기타	
9 주차	[9 주]
학습주제	8 장 조건과 반복, 리스트와 튜플 기반의 미니 프로젝트 I
목표및 내용	8 개의 미니 프로젝트를 스스로 생각하고 프로그래밍해 코딩 능력뿐 아니라 문제 해결 능력을 키울 수 있다.
미리읽어오기	교재 8 장
과제,시험,기타	

10 주차	[10 주]
학습주제	9 장 라이브러리 활용을 위한 모듈과 패키지
목표및 내용	<ul style="list-style-type: none"> 표준 모듈을 이해하고 사용자 정의 모듈도 직접 구현해 사용할 수 있다. 표준 모듈인 turtle 을 사용해 기본적인 도형을 그릴 수 있다. 써드파티 모듈 numpy 와 matplotlib 등을 설치해 활용할 수 있다.
미리읽어오기	교재 9 장
과제,시험,기타	도전 프로그래밍
11 주차	[11 주]
학습주제	10 장 그래픽 사용자 인터페이스 Tkinter 와 Pygame
목표및 내용	<ul style="list-style-type: none"> GUI 를 이해하고 GUI 표준 모듈인 Tkinter 를 사용해 필요한 위젯을 구성하고 윈도우를 생성할 수 있다. 이벤트 처리를 이해하고 Tkinter 에서 이벤트 처리를 구현할 수 있다. 써드파티 GUI 모듈인 pygame 을 설치해 기본적인 윈도우를 구현할 수 있다.
미리읽어오기	교재 10 장
과제,시험,기타	도전 프로그래밍
12 주차	[12 주]
학습주제	11 장 실행 오류 및 파일을 다루는 예외 처리와 파일 입출력
목표및 내용	<ul style="list-style-type: none"> 예외 처리의 필요성을 이해하고 try except 구문을 사용해 예외를 처리할 수 있다. 프로그램에서 파일을 생성하는 필요성을 이해하고 필요한 파일을 만들 수 있다. 이미 생성된 파일에서 내용을 읽어 처리할 수 있다
미리읽어오기	교재 11 장
과제,시험,기타	도전 프로그래밍
13 주차	[13 주]
학습주제	12 장 일상생활의 사물 코딩인 객체지향 프로그래밍
목표및 내용	<ul style="list-style-type: none"> 객체와 클래스를 이해하고 필요한 클래스를 정의하고 객체를 만들어 활용할 수 있다. 클래스 속성과 인스턴스 속성, 정적 메소드와 클래스 메소드를 이해하고 정의할 수 있다. 상속을 이해하고 부모 클래스와 자식 클래스를 정의할 수 있다. 추상 메소드와 추상 클래스를 이해하고 정의할 수 있다
미리읽어오기	교재 12 장
과제,시험,기타	도전 프로그래밍
14 주차	[14 주]
학습주제	13 장 GUI 모듈과 객체지향 기반의 미니 프로젝트 II
목표및 내용	<p>학습한 파이썬 문법 구조와 프로그래밍 기법을 활용해 8 개의 미니 프로젝트를 스스로 생각하고 프로그래밍해 코딩 능력뿐 아니라 문제 해결 능력을 키울 수 있다.</p>

미리읽어오기	교재 1 장
과제,시험,기타	

15 주차	[기말고사]
학습주제	직무수행능력평가 2 차(기말고사)
목표및 내용	직무수행능력평가, 서술형평가
미리읽어오기	8 장에서 13 장까지
과제,시험,기타	
수업지원 안내	장애학생을 위한 별도의 수강 지원을 받을 수 있습니다. 언어가 문제가 되는 학생은 글로 된 과제 안내, 확대문자 시험지 제공 등의 지원을 드립니다.



머리말

이 포트폴리오를 설계하는 과정에서 많은 생각을 했다. 어떻게 만드는지, 수업 과정처럼 하는지 많은 고민을 해 봤다. 교수님께서 나중에 시험 보거나 취업 준비 때 우리가 더 쉽게 복습할 수 있도록 이 포트폴리오를 자유롭게 만들라고 하셨다. 그래서 내가 나중에 필요할 때 쉽게 참조 할 수 있는 포트폴리오를 만들었다. 이 포트폴리오를 작성하는 과제에서 여러 학습 자료를 참고 했다. 예를 들어, 수업 중 한 실습, 전문가 웹사이트 및 슬라이드 등이 있다. 포트폴리오를 작성할 때 얻을 수 있는 좋은 점은 내가 과제를 완성하는 동시에 복습도 같이 할 수 있는 점이다. 그래서 부담 없이 이 포트폴리오를 완성할 때 좋은 생각으로 잘 만든 것 같다.

외국인 유학생으로서 솔직히 말해서 수업 시간이나 과제를하는 동안 많은 문제가 있었다. 가끔 수업 이해하기가 힘들어서 수업 시간에 제대로 공부하는 것보다 자습 공부를 더 많이 하는 편이다. 그래서 이 포트폴리오를 만드는 것이 내 학습의 효과를 높이는 좋은 기회라고 생각한다. 한국어 실력 향상에도 도움이 된다고 생각한다. 또한 전에 알지 못했던 파이썬에 대해 많은 것을 알게 된다.

이 포트폴리오를 작성했을 때 이점을 받고 진심으로 내 발전에 도움이 된다. 나중에 이 포트폴리오를 읽을 사람에게는 파이썬 학습에 좀 도움이 되면 좋겠다. 앞으로 이 분야에서 성공할 수 있기를 바란다.

컴퓨터정보공학과
아즈라이

파이썬 개요

파이썬 언어란 ?

파이썬 (Python)은 1991 년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 <Monty Python's Flying Circus>에서 따온 것이다.

파이썬은 인터프리터(interpreter) 언어로서, 리눅스, Mac OS X, 윈도우즈 등 다양한 시스템에 널리 사용된다.

컴파일러와 인터프리터 차이점은 아래 표에서 볼 수 있다.

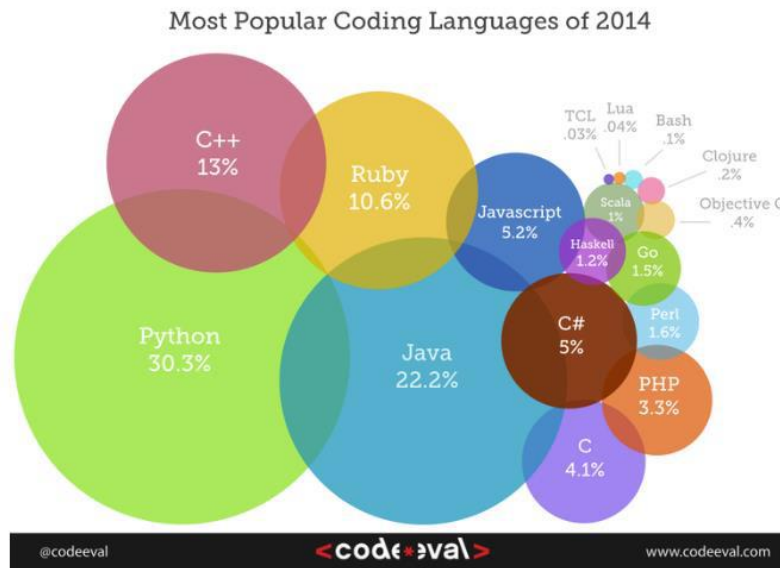
구분	컴파일러	인터프리터
작동 방식	소스코드를 기계어로 먼저 번역하고, 해당 플랫폼에 최적화되어 프로그램을 실행함	별도의 번역 과정 없이 소스코드를 실행 시점에 해석하여 컴퓨터가 처리할 수 있도록 함
장점	실행 속도가 빠름	간단히 작성, 메모리가 적게 필요
단점	한 번에 많은 기억 장소가 필요함	실행 속도가 느림
주요 언어	C, 자바(Java), C++, C#	파이썬, 스칼라

[컴파일러와 인터프리터 비교]

파이썬의 특징

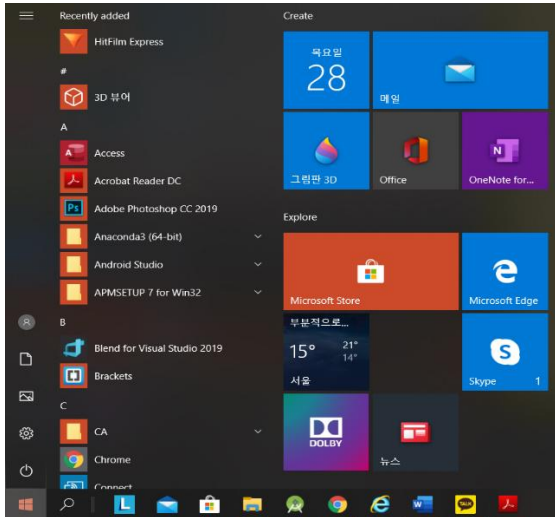
1. 파이썬은 객체지향 / 인터프리터 언어이다.
2. 파이썬은 문법이 쉬워 빠르게 배울 수 있다.
3. 파이썬은 무료이지만 강력하다.
4. 파이썬은 간결하다.
5. 파이썬은 프로그래밍을 즐기게 해준다.
6. 파이썬은 개발 속도가 빠르다.

파이썬은 또한 세계적으로 가장 인기 있는 프로그램 언어가 된다.

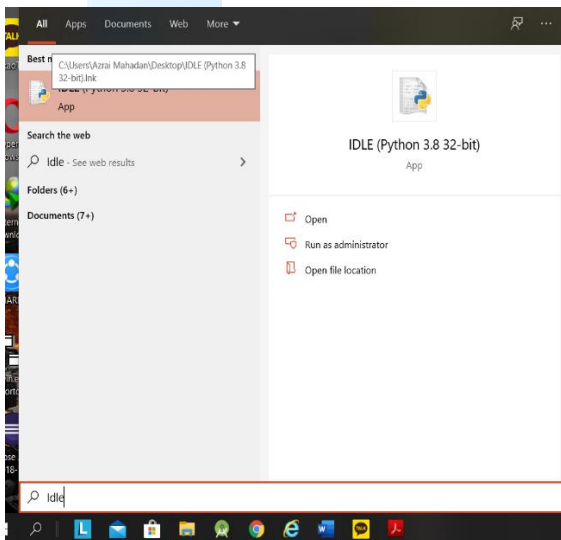


파이썬 쉘의 실행

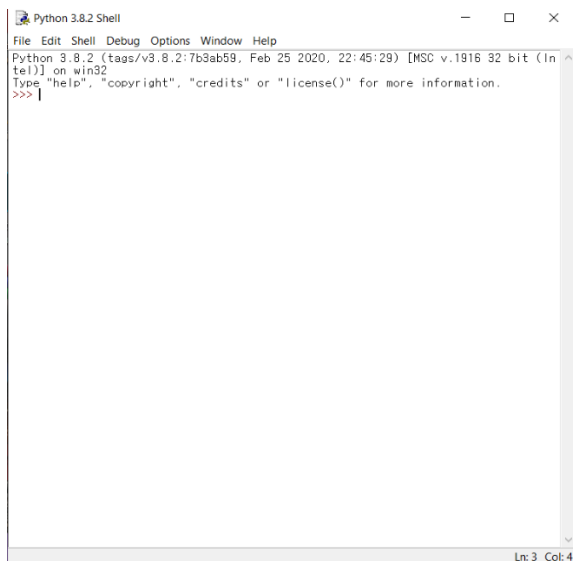
1.



2.

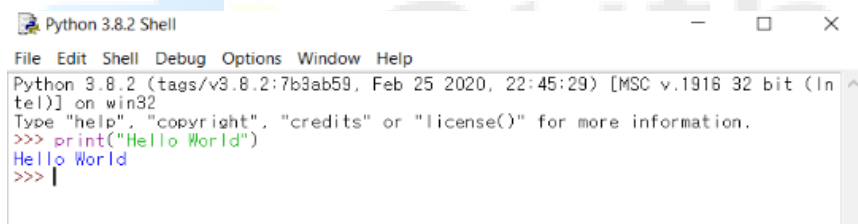


3.



```
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.
>>>
```

파이썬 첫 프로그래밍



```
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
>>>
```

파이썬 프로그래밍 기초

문자열과 수

문자열(String)이란 문자, 단어 등으로 구성된 문자들의 집합을 의미한다. 예를 들어 다음과 같은 것들이 문자열이다.

```
"Life is too short, You need Python"  
"a"  
"123"
```

파이썬에서 문자열을 만드는 방법은 총 4 가지이다.

1. 큰따옴표(")로 양쪽 둘러싸기

```
"Hello World"
```

2. 작은따옴표(')로 양쪽 둘러싸기

```
'Python is fun'
```

3. 큰따옴표 3 개를 연속(""")으로 써서 양쪽 둘러싸기

```
"""Life is too short, You need python"""
```

4. 작은따옴표 3 개를 연속('')으로 써서 양쪽 둘러싸기

```
'''Life is too short, You need python'''
```

문자열 더해서 연결하기(Concatenation)

```
>>> head = "Python"  
>>> tail = " is fun!"  
>>> head + tail  
'Python is fun!'
```

문자열 곱하기

```
>>> a = "python"
```

```
>>> a * 2
'pythonpython'
print("=" * 50)
print("My Program")
print("=" * 50)
=====
My Program
=====
```

정수와 실수

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8 진수	0o34, 0o25
16 진수	0x2A, 0xFF

사칙연산

```
>>> a = 3
>>> b = 4
>>> a + b
7
>>> a * b
12
>>> a / b
0.75
```

x의 y 제곱을 나타내는 ** 연산자

```
>>> a = 3
>>> b = 4
>>> a ** b
81
```

나눗셈 후 나머지를 반환하는 % 연산자

프로그래밍을 처음 접하는 독자라면 % 연산자는 본 적이 없을 것이다. %는 나눗셈의 나머지 값을 돌려주는 연산자이다. 7 을 3 으로 나누면 나머지는 1 이 될 것이고 3 을 7 로 나누면 나머지는 3 이 될 것이다. 다음 예로 확인해 보자.

```
>>> 7 % 3
1
>>> 3 % 7
3
```

나눗셈 후 몫을 반환하는 // 연산자

```
>>> 7 / 4
1.75
>>> 7 // 4
1
변수
```

다음 예와 같은 a, b, c 를 변수라고 한다.

```
>>> a = 1
>>> b = "python"
>>> c = [1,2,3]          #변수 이름 = 변수에 저장할 값
```


키워드 및 대입 연산자

키워드(Keyword)는 파이썬에서 이미 예약되어있는 문자열로서 다른 용도로 사용이 불가능한 문자열 입니다.

[False]	[None]	[True]	[and]	[as]
[assert]	[break]	[class]	[continue]	[def]
[del]	[elif]	[else]	[except]	[finally]
[for]	[from]	[global]	[if]	[import]
[in]	[is]	[lambda]	[nonlocal]	[not]
[or]	[pass]	[raise]	[return]	[try]
[while]	[with]	[yield]				

대입 연산자(Assignment Operations)

연산자	동치
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a %= b	a = a % b
a //= b	a = a // b
a **= b	a = a ** b
a &= b	a = a & b
a = b	a = a b
a ^= b	a = a ^ b
a <=< b	a = a << b
a >>= b	a = a >>b

```

a = 1
print('a =', a)
a = a + 1
print('a =', a)
a += 1
print('a =', a)
a -= 1
print('a =', a)
a *= 2
print('a =', a)
a //= 2
print('a =', a)
a %= 5
print('a =', a)
a <= 3
print('a =', a)
a >= 2
print('a =', a)
a **= 2
print('a =', a)
a &= 0b101010
print('a =', a)
a |= 0b11110000
print('a =', a)
a ^= 0b10101010
print('a =', a)

```

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: C:/Python/test.py =====
a = 1
a = 2
a = 3
a = 2
a = 4
a = 2
a = 2
a = 16
a = 4
a = 16
a = 0
a = 240
a = 90
>>> |

divmod() 함수

divmod(a, b)는 2 개의 숫자를 입력으로 받는다. 그리고 a를 b로 나눈 몫과 나머지를 튜플 형태로 돌려주는 함수이다.

```

>>> divmod(7, 3)

(2, 1)

```

자료의 표준 입력과 자료 변환 함수

Python에서는 표준 입력을 하는 함수로 input 함수를 지원한다.

```

>>> number = input("숫자를 입력하세요: ")
숫자를 입력하세요: 3
>>> print(number)
3

```

정수 입력

```

data = eval(input("정수를 입력하시오 : "))
print(data, type(data), data + 1)

data = int(input("정수를 입력하시오 : "))
print(data, type(data), data + 1)

data = int(input("2 진수를 입력하시오 : "), 2)
print(data, type(data), data + 1)

data = int(input("8 진수를 입력하시오 : "), 8)

```

```

print(data, type(data), data + 1)

data = int(input("10 진수를 입력하시오 : "), 10)
print(data, type(data), data + 1)

data = int(input("16 진수를 입력하시오 : "), 16)
print(data, type(data), data + 1)

```

실행 결과:

```

정수를 입력하시오 : 1
1 <class 'str'>
정수를 입력하시오 : 2
2 <class 'int'> 3
정수를 입력하시오 : 3
3 <class 'int'> 4
2 진수를 입력하시오 : 1010
10 <class 'int'> 11
8 진수를 입력하시오 : 10
8 <class 'int'> 9
10 진수를 입력하시오 : 10
10 <class 'int'> 11
16 진수를 입력하시오 : 1a
26 <class 'int'> 27

```

자료 변환 함수

```

>>> float(3)      #실수형으로 바꿈
3.0
>>> int(3.0)      #정수형으로 바꿈
3
>>> str(3)        #문자열로 바꿈
'3'

```

다른 진수로 바꿀 수도 있다.

```

>>> hex(12)       #16 진수로 바꿈
'0xa'
>>> oct(10)       #8 진수로 바꿈
'0o12'
>>> bin(10)       #2 진수로 바꿈
'0b1010'

```

다시 10 진수로 바꿀 때는 자동으로 바뀐다.

```
>>> a = 0b0010
>>> a
2
```

다른 진수값이 문자열이면 int() 함수를 사용한다

```
>>> int('0xc', 16)
12
>>> int('0o14', 8)
12
>>> int('0b1100', 2)
12
```



문자열과 논리 연산

문자열

문자열 참조 활용

```
>>> a = "Life is too short, You need Python"
>>> a[0]
'L'
>>> a[12]
's'
>>> a[-1]
'n'
>>> a[-0]
'L'
>>> a[-2]
'o'
>>> a[-5]
'y'
```

문자열 슬라이싱

```
>>> a = "Life is too short, You need Python"
>>> a[0:4]
'Life'
>>> a[0:3]
'Lif'
```

```
>>> a[0:5]
'Life '
>>> a[0:2]
'Li'
>>> a[5:7]
'is'
>>> a[12:17]
'short'
>>> a[19:]
'You need Python'
>>> a[:17]
'Life is too short'
>>> a[:]
'Life is too short, You need Python'
>>> a[19:-7]
'You need'
```

```

>>> a = "20010331Rainy"
>>> date = a[:8]
>>> weather = a[8:]
>>> date
'20010331'
>>> weather
'Rainy'

>>> a = "20010331Rainy"
>>> year = a[:4]
>>> day = a[4:8]
>>> weather = a[8:]
>>> year
'2001'
>>> day
'0331'
>>> weather
'Rainy'

```

이스케이프 시퀀스 문자

Escape Sequence Characters	Non-Printing Characters
\'	Single quote, needed for character literals.
\"	Double quote, needed for string literals.
\\	Backslash, needed for string literals.
\0	Unicode character 0.
\a	Alert.
\b	Backspace.
\f	Form feed.
\n	New line.
\r	Carriage return.
\t	Horizontal tab.
\v	Vertical tab.
\xhh	Matches an ASCII character using hexadecimal representation (exactly two digits). For example, \x61 represents the character 'a'.
\uhhhh	Matches a Unicode character using hexadecimal representation (exactly four digits). For example, the character \u0020 represents a space.

문자열 관련 메소드

문자 개수 세기(count)

```
>>> a = "hobby"
>>> a.count('b')
2
```

위치 알려주기 1(find)

```
>>> a = "Python is the best choice"
>>> a.find('b')
14
>>> a.find('k')
-1
```

위치 알려주기 2(index)

```
>>> a = "Life is too short"
>>> a.index('t')
8
>>> a.index('k')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: substring not found
```

문자열 삽입(join)

```
>>> ",".join('abcd')
'a,b,c,d'
>>> ",".join(['a', 'b', 'c', 'd'])
'a,b,c,d'
```

소문자를 대문자로 바꾸기(upper)

```
>>> a = "hi"
>>> a.upper()
'HI'
```

대문자를 소문자로 바꾸기(lower)

```
>>> a = "HI"
>>> a.lower()
'hi'
```

왼쪽 공백 지우기(lstrip)

```
>>> a = " hi "  
>>> a.lstrip()  
'hi '
```

오른쪽 공백 지우기(rstrip)

```
>>> a= " hi "  
>>> a.rstrip()  
' hi'
```

양쪽 공백 지우기(strip)

```
>>> a = " hi "  
>>> a.strip()  
'hi'
```

문자열 바꾸기(replace)

```
>>> a = "Life is too short"  
>>> a.replace("Life", "Your leg")  
'Your leg is too short'
```

문자열 나누기(split)

```
>>> a = "Life is too short"  
>>> a.split()  
['Life', 'is', 'too', 'short']  
>>> b = "a:b:c:d"  
>>> b.split(':')  
['a', 'b', 'c', 'd']
```


형식에 맞추어 출력 예제

```
# 형식에 맞추어 출력하기

# 이전 방식
print('나의 이름은 %s입니다.'%( '한사람' ))
print('나의 이름은 "%s"입니다. 나이는 %d세이고 성별은 %s입니다.'%( '한사람', 33, '남성' ))
print('나이는 %d세이고 성별은 %s입니다. 나의 이름은 "%s"입니다.'%( 33, '남성', '한사람' ))
print('나이는 %03d세이고 신장은 %5.2f입니다. 나의 이름은 "%s"입니다.'%( 33, 56.789, '한사람' ))
print('-' * 40)

# 파이썬(Python) 3 포매팅 방식
print('나의 이름은 %s입니다.'format( '한사람' ))
print('나의 이름은 "%s"입니다. 나이는 %d세이고 성별은 %s입니다.'format( '한사람', 33, '남성' ))
print('나이는 %d세이고 성별은 %s입니다. 나의 이름은 "%s"입니다.'format( '한사람', 33, '남성' ))
print('나이는 (age)세이고 성별은 (gender)입니다. 나의 이름은 (name)입니다.'
      .format(name='한사람', age=33, gender='남성' ))
print('만세삼십 : {0}!!! {0}!!! {0}!!!'.format('만세'))
print('삼십칠 박수 : {0}!!! {0}!!! {1}!!!'.format('짝'*3, '짝'*7))
print('-' * 40)
```

```
===== RESTART: C:/Python/test.py =====
나의 이름은 한사람입니다
나의 이름은 "한사람"입니다. 나이는 33세이고 성별은 남성입니다.
나이는 33세이고 성별은 남성입니다. 나의 이름은 "한사람"입니다.
나이는 033세이고 신장은 56.79입니다. 나의 이름은 "한사람"입니다.

-----
나의 이름은 한사람입니다
나의 이름은 "한사람"입니다. 나이는 33세이고 성별은 남성입니다.
나이는 33세이고 성별은 남성입니다. 나의 이름은 "한사람"입니다.
나이는 33세이고 성별은 남성입니다. 나의 이름은 "한사람"입니다.
만세삼십 : 만세!!! 만세!!! 만세!!!
삼십칠 박수 : 짹짝!!! 짹짝!!! 짹짝짝짝짝!!!

>>> 1
```

```
1 print('Python is [{:15}]'.format('good'))
2 print('Python is [{:<15}]'.format('good'))
3 print('Python is [{:>15}]'.format('good'))
4 print('Python is [{:^15}]'.format('good'))
5 print('당신의 나이는 [{:15}]세'.format(22))
6 print('당신의 나이는 [{:<15}]세'.format(22))
7 print('당신의 나이는 [{:>15}]세'.format(22))
8 print('당신의 나이는 [{:^15}]세'.format(22))
9 print('-' * 40)
10
```

```
Python is [good
Python is [good
Python is [
Python is [
good
]
]
]
]
당신의 나이는 [ 22]세
당신의 나이는 [22
]세
당신의 나이는 [
22]세
당신의 나이는 [22
]세
```

```
File Edit Format Run Options Window Help

1 # 왼쪽 정렬
2 for x in range(1, 4):
3     print('{0:2d} [{1:3d}] [{2:4d}]'.format(x, x*x, x*x*x))
4 print()
5 # 가운데 정렬
6 for x in range(1, 4):
7     print('{0:^2d} [{1:^3d}] [{2:^4d}]'.format(x, x*x, x*x*x))
8 print()
9 # 오른쪽 정렬
10 for x in range(1, 4):
11     print('{0:<2d} [{1:<3d}] [{2:<4d}]'.format(x, x*x, x*x*x))
12 print()
13 # 0으로 채우기
14 for x in range(1, 4):
15     print('{0:02d} [{1:03d}] [{2:04d}]'.format(x, x*x, x*x*x))
16 print()
17 # 부호 출력
18 print('{0:05d} [{1:05d}] [{2:05d}]'.format(1,-2,3)) # 음수만 부호
19 print('{0:+05d} [{1:+05d}] [{2:+05d}]'.format(1,-2,3)) # 부호
20 print('{0:<5d} [{1:<5d}] [{2:<5d}]'.format(1,-2,3)) # 정렬
21 print('-' * 40)
22
```

```

[ 1] [ 1] [ 1]
[ 2] [ 4] [ 8]
[ 3] [ 9] [ 27]

[1 ] [ 1 ] [ 1 ]
[2 ] [ 4 ] [ 8 ]
[3 ] [ 9 ] [ 27 ]

[1 ] [1 ] [1 ]
[2 ] [4 ] [8 ]
[3 ] [9 ] [27 ]

[01] [001] [0001]
[02] [004] [0008]
[03] [009] [0027]

[00001] [-0002] [00003]
[+0001] [-0002] [+0003]
[1 ] [-2 ] [3 ]

```

File Edit Format Run Options Window Help

```

1 # 채움문자
2 print(' [{0:05d}] [{1:05d}] [{2:05d}]'.format(1,-2,3))
3 print(' [{0:★<5d}] [{1:★<5d}] [{2:★<5d}]'.format(1,-2,3))
4 print(' [{0:★>5d}] [{1:★>5d}] [{2:★>5d}]'.format(1,-2,3))
5 print(' [{0:★^5d}] [{1:★^5d}] [{2:★^5d}]'.format(1,-2,3))
6 print(' [{0:5d}] [{1:5d}] [{2:5d}]'.format(1,-2,3))
7 print(' [{0:=5d}] [{1:=5d}] [{2:=5d}]'.format(1,-2,3))
8 print(' [{0:=05d}] [{1:=05d}] [{2:=05d}]'.format(1,-2,3))
9 # 숫자 표시 형식( , _ )
10 print(' [{0:>5.}] [{1:>5.}] [{2:>5.}]'.format(11544435,-2544254,35454343))
11 print(' [{0:>5_}] [{1:>5_}] [{2:>5_}]'.format(11544435,-2544254,35454343))
12 print(' [{0:>5e}] [{1:>5e}] [{2:>5e}]'.format(11544435,-2544254,35454343))
13 print('-' * 40)
14

```

===== RESTART: C:/Python/test.py =====

```

[00001] [-0002] [00003]
[1★★★★] [-2★★★★] [3★★★★]
[★★★★1] [★★★★-2] [★★★★3]
[★★1★] [★-2★] [★★3★★]
[ 1] [ -2] [ 3]
[ 1] [- 2] [ 3]
[00001] [-0002] [00003]
[11,544,435] [-2,544,254] [35,454,343]
[11_544_435] [-2_544_254] [35_454_343]
[1.154444e+07] [-2.544254e+06] [3.545434e+07]

```

File Edit Format Run Options Window Help

```

1 print(' [{0:5b}] [{1:5b}] [{2:5b}]'.format(11,-22,33)) # 2진수
2 print(' [{0:5o}] [{1:5o}] [{2:5o}]'.format(11,-22,33)) # 8진수
3 print(' [{0:5x}] [{1:5x}] [{2:5x}]'.format(11,-22,33)) # 16진수 소문자
4 print(' [{0:5X}] [{1:5X}] [{2:5X}]'.format(11,-22,33)) # 16진수 대문자
5 print('-' * 40)
6
7 import math
8 print('원주율 [{0:.3f}]'.format(math.pi))
9 print('원주율 [{0:.7f}]'.format(math.pi))
10 print('원주율 [{0:10.3f}]'.format(math.pi))
11 print('원주율 [{0:20.7f}]'.format(math.pi))
12 print('원주율 [{0:0.5f}]'.format(math.pi))
13 print('-' * 40)
14

```

```
===== RESTART: C:/PYTHON/T
[ 1011] [-10110] [100001]
[  13] [  -26] [   41]
[   b] [  -16] [   21]
[   B] [  -16] [   21]
-----
원주율 [3.142]
주주주 [3.1415927]
주주주 [  3.142]
주주주 [  3.1415927]
주주주 [3.14159]
-----
>>> |
```

```
1 # 형식에 맞추어 출력하기
2 print('{0}씨는 상위 {1}%안에 있는 사람입니다.'.format('한사람',10))
3 print('{{0}}씨는 상위 {{1}}%안에 있는 사람입니다.'.format('한사람',10))
4 print('{{{0}}}씨는 상위 {{{1}}}%안에 있는 사람입니다.'.format('한사람',10))
5 print('%s씨는 상위 %d%안에 있는 사람입니다.'%( '한사람',10))
6 print('-' * 40)
7
```

```
===== RESTART: C:/PYTHON/T
한사람씨는 상위 10%안에 있는 사람입니다.
{0}씨는 상위 {1}%안에 있는 사람입니다.
{한사람}씨는 상위 {10}%안에 있는 사람입니다.
한사람씨는 상위 10%안에 있는 사람입니다.
-----
>>> |
```

논리 자료와 다양한 연산

논리 연산자는 and, or, not 이 있다.

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

함수 bool(값)

```
>>> bool(1)
True
>>> bool(0)
False
>>> bool(1.5)
True
>>> bool('False')
True
```

관계 연산자(Relational Operators)

> : 크다
>= : 크거나 같다
< : 작다
<= : 작거나 같다
== : 같다
!= : 같지 않다

```
a = 10
```

```

b = 3

print("10 == 3 =>", a == b)
print("10 != 3 =>", a != b)
print("10 > 3 =>", a > b)
print("10 >= 3 =>", a >= b)
print("10 < 3 =>", a < b)
print("10 <= 3 =>", a <= b)
print()

```

실행 결과

```

10 == 3 => False
10 != 3 => True
10 > 3 => True
10 >= 3 => True
10 < 3 => False
10 <= 3 => False

```

비트 연산자(Bitwise Operators)

& (Binary AND) : bit 단위로 and 연산을 합니다.

| (Binary OR) : bit 단위로 or 연산을 합니다.

^ (Binary XOR) : bit 단위로 xor 연산을 합니다.

~ (Binary NOT) : bit 단위로 not 연산을 합니다.(1 의 보수)

<< (Binary left Shift) : bit 단위로 왼쪽으로 비트단위 밀기 연산을 합니다.

>> (Binary right Shift) : bit 단위로 오른쪽으로 비트단위 밀기 연산을 합니다.

```

a = 0b10101010
b = 0b01110011

print('a = ', a, ":", bin(a))
print('b = ', b, ":", bin(b))
print('a & b = ', a & b, ":", bin(a & b))
print('a | b = ', a | b, ":", bin(a | b))
print('a ^ b = ', a ^ b, ":", bin(a ^ b))
print('~a = ', ~a, ":", bin(~a))

a = 0b1
print('a = ', a)
a = a << 1 # * 2
print('a = ', a)
a = a << 1 # * 2
print('a = ', a)
a = a << 3 # * 2**3
print('a = ', a)

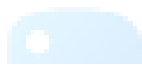
a = a >> 1 # / 2
print('a = ', a)
a = a >> 1 # / 2

```

```
print('a = ', a)
a = a >> 2 # / 2**2
print('a = ', a)
```

실행 결과

```
a = 170 : 0b10101010
b = 115 : 0b1110011
a & b = 34 : 0b100010
a | b = 251 : 0b11111011
a ^ b = 217 : 0b11011001
~a = -171 : -0b10101011
a = 1
a = 2
a = 4
a = 32
a = 16
a = 8
a = 2
```



TM

조건과 반복

조건문

다음은 if 와 else 를 사용한 조건문의 기본 구조이다.

if 조건문:

수행할 문장 1

수행할 문장 2

...

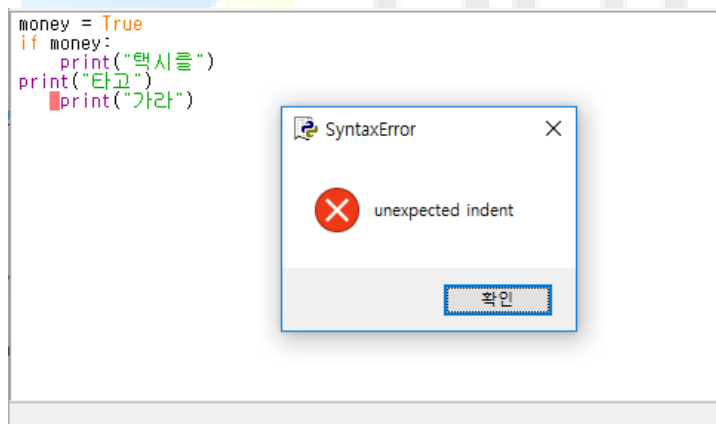
else:

수행할 문장 A

수행할 문장 B

...

if 문을 만들 때는 if 조건문: 바로 아래 문장부터 if 문에 속하는 모든 문장에 들여쓰기(indentation)를 해주어야 한다.



if 조건문에서 "조건문"이란 참과 거짓을 판단하는 문장을 말한다

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
>>>
```

`money >= 3000` 조건문이 거짓이 되기 때문에 `else` 문 다음 문장을 수행하게 된다.

```
>>> money = 2000
>>> card = True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

`money` 는 2000 이지만 `card` 가 `True` 이기 때문에 `money >= 3000 or card` 조건문이 참이 된다. 따라서 `if` 문 다음 '택시를 타고 가라' 문장이 출력된다.

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

`['paper', 'cellphone', 'money']` 리스트 안에 'money'가 있으므로 `'money' in pocket` 은 참이 된다. 따라서 `if` 문 다음 문장이 수행된다.

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket:
...     pass
... else:
...     print("카드를 꺼내라")
...

```

`pocket` 리스트 안에 `money` 문자열이 있기 때문에 `if` 문 다음 문장인 `pass` 가 수행되고 아무 결과값도 보여 주지 않는다.

다양한 조건을 판단하는 elif

```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... elif card:
...     print("택시를 타고가라")
... else:
...     print("걸어가라")
...
택시를 타고가라
```

즉 elif 는 이전 조건문이 거짓일 때 수행된다. if, elif, else 를 모두 사용할 때 기본 구조는 다음과 같다.

```
If <조건문>:
    <수행할 문장 1>
    <수행할 문장 2>
    ...
elif <조건문>:
    <수행할 문장 1>
    <수행할 문장 2>
    ...
elif <조건문>:
    <수행할 문장 1>
    <수행할 문장 2>
    ...
...
else:
    <수행할 문장 1>
    <수행할 문장 2>
    ...
```

위에서 볼 수 있듯이 elif 는 개수에 제한 없이 사용할 수 있다.

반복문

while-문

다음은 while 문의 기본 구조이다.

```
while <조건문>:  
    <수행할 문장 1>  
    <수행할 문장 2>  
    <수행할 문장 3>  
    ...
```

while 문은 조건문이 참인 동안에 while 문 아래의 문장이 반복해서 수행된다.

```
>>> treeHit = 0  
>>> while treeHit < 10:  
...     treeHit = treeHit + 1  
...     print("나무를 %d 번 찍었습니다." % treeHit)  
...     if treeHit == 10:  
...         print("나무 넘어갑니다.")  
...  
나무를 1 번 찍었습니다.  
나무를 2 번 찍었습니다.  
나무를 3 번 찍었습니다.  
나무를 4 번 찍었습니다.  
나무를 5 번 찍었습니다.  
나무를 6 번 찍었습니다.  
나무를 7 번 찍었습니다.  
나무를 8 번 찍었습니다.  
나무를 9 번 찍었습니다.  
나무를 10 번 찍었습니다.  
나무 넘어갑니다.
```

임의의 수인 난수

random 은 난수(규칙이 없는 임의의 수)를 발생시키는 모듈이다.

다음 예는 1 에서 10 사이의 정수 중에서 난수 값을 돌려준다.

```
>>> random.randint(1, 10)
6
```

random 모듈을 사용하여 로또 번호(1~45 사이의 숫자 6 개)를 생성해 보자.

```
import random

result = []
while len(result) < 6:
    num = random.randint(1, 45)    # 1 부터 45 까지의 난수 발생
    if num not in result:
        result.append(num)

print(result)
```



python

break 문, continue 문

```
>>> coffee = 10
>>> money = 300
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee - 1
...     print("남은 커피의 양은 %d 개입니다." % coffee)
...     if coffee == 0:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
... 
```

만약 coffee 가 0 이 되면 `if coffee == 0:` 문장에서 `coffee == 0:` 이 참이 되므로 if 문 다음 문장 "커피가 다 떨어졌습니다. 판매를 중지합니다."가 수행되고 break 문이 호출되어 while 문을 빠져나가게 된다.

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue
...     print(a)
... 
```

1
3
5
7
9

위 예는 1 부터 10 까지의 숫자 중 홀수만 출력하는 예이다. a 가 짝수이면 continue 문장을 수행한다. 이 continue 문은 while 문의 맨 처음(조건문: `a<10`)으로 돌아가게 하는 명령어이다. 따라서 위 예에서 a 가 짝수이면 `print(a)`는 수행되지 않을 것이다.

리스트와 튜플

리스트

리스트를 사용하면 1, 3, 5, 7, 9 숫자 모음을 다음과 같이 간단하게 표현할 수 있다.

```
>>> odd = [1, 3, 5, 7, 9]
```

리스트를 만들 때는 위에서 보는 것과 같이 대괄호([])로 감싸 주고 각 요소값은 쉼표(,)로 구분해 준다.

```
리스트명 = [요소 1, 요소 2, 요소 3, ...]
```

여러 가지 리스트의 생김새를 살펴보면 다음과 같다.

```
>>> a = []
>>> b = [1, 2, 3]
>>> c = ['Life', 'is', 'too', 'short']
>>> d = [1, 2, 'Life', 'is']
>>> e = [1, 2, ['Life', 'is']]
```

리스트의 항목 참조

리스트 역시 문자열처럼 인덱싱을 적용할 수 있다. 먼저 a 변수에 [1, 2, 3] 값을 설정한다.

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
```

a[0]은 리스트 a의 첫 번째 요소값을 말한다.

```
>>> a[0]
1
```

다음 예는 리스트의 첫 번째 요소인 a[0]과 세 번째 요소인 a[2]의 값을 더한 것이다.

```
>>> a[0] + a[2]
4
>>> a[-1]
3
```

이번에는 다음 예처럼 리스트 a를 숫자 1, 2, 3과 또 다른 리스트인 ['a', 'b', 'c']를 포함하도록 만들어 보자.

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
>>> a[0]
1
>>> a[-1]
['a', 'b', 'c']
>>> a[3]
['a', 'b', 'c']
>>> a[-1][0]
'a'
>>> a[-1][1]
'b'
>>> a[-1][2]
'c'
```

삼중 리스트에서 참조하기

```
>>> a = [1, 2, ['a', 'b', ['Life', 'is']]]
>>> a[2][2][0]
'Life'
```

가위바위보 리스트 항목 참조

```
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))

Python 3.8.2 (tags/v3.8.2:7b
tel)] on win32
Type "help", "copyright", "c
>>>
==== RESTART: C:\Python\2020
가위 바위 보
컴퓨터의 가위 바위 보 5개
보
가위
가위
바위
가위
>>>
```

리스트 길이구하기

리스트 길이를 구하기 위해서는 다음처럼 len 함수를 사용해야 한다.

```
>>> a = [1, 2, 3]
>>> len(a)
3
```

리스트에 요소 추가(append)

append 를 사전에서 검색해 보면 "덧붙이다, 첨부하다"라는 뜻이 있다. 이 뜻을 안다면 다음 예가 바로 이해될 것이다. append(x)는 리스트의 맨 마지막에 x 를 추가하는 함수이다.

```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```

리스트 안에는 어떤 자료형도 추가할 수 있다.

```
>>> a.append([5,6])
>>> a
[1, 2, 3, 4, [5, 6]]
```

리스트에서 값 수정하기

```
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]
```

del 함수 사용해 리스트 요소 삭제하기

```
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
>>> b = [1, 2, 3, 4, 5]
>>> del b[2:]
>>> b
[1, 2]
```

리스트에 포함된 요소 x 의 개수 세기(count)

count(x)는 리스트 안에 x 가 몇 개 있는지 조사하여 그 개수를 돌려주는 함수이다.

```
>>> a = [1,2,3,1]
>>> a.count(1)
2
```

위치 반환(index)

index(x) 함수는 리스트에 x 값이 있으면 x 의 위치 값을 돌려준다.

```
>>> a = [1,2,3]
>>> a.index(3)
2
```

```
>>> a.index(1)
0
```

리스트 더하기(+)

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

리스트 반복하기(*)

```
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

리스트 정렬(sort)

```
>>> a = [1, 4, 3, 2]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

문자 역시 알파벳 순서로 정렬할 수 있다.

```
>>> a = ['a', 'c', 'b']
>>> a.sort()
>>> a
['a', 'b', 'c']
```

리스트 뒤집기(reverse)

```
>>> a = ['a', 'c', 'b']
>>> a.reverse()
>>> a
['b', 'c', 'a']
```

리스트에 요소 삽입(insert)

```
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a
[4, 1, 2, 3]
```

위 예는 0 번째 자리, 즉 첫 번째 요소(a[0]) 위치에 값 4를 삽입하라는 뜻이다.


```
>>> a.insert(3, 5)
>>> a
[4, 1, 2, 5, 3]
```

위 예는 리스트 a 의 a[3], 즉 네 번째 요소 위치에 값 5 를 삽입하라는 뜻이다.

리스트 요소 제거(remove)

remove(x)는 리스트에서 첫 번째로 나오는 x 를 삭제하는 함수이다.

```
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]
```

a 가 3 이라는 값을 2 개 가지고 있을 경우 첫 번째 3 만 제거되는 것을 알 수 있다.

```
>>> a.remove(3)
>>> a
[1, 2, 1, 2]
```

리스트 요소 끄집어내기(pop)

pop()은 리스트의 맨 마지막 요소를 돌려주고 그 요소는 삭제한다.

```
>>> a = [1,2,3]
>>> a.pop()
3
>>> a
[1, 2]
>>> b = [1,2,3]
>>> b.pop(1)
2
>>> b
[1, 3]
```

다른 예제

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

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

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

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

```
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: C:\Python\Python2020\Python\lecture\ch5\05-07nestedanimallist.py
[['사자', '코끼리', '호랑이'], '조류', '어류']
조류
어류
[['사자', '코끼리', '호랑이'], ['독수리', '참새', '까치'], ['갈치', '붕어', '고등어']]
사자 코끼리 호랑이
독수리 참새 까치
갈치 붕어 고등어
>>> |
```

```

word = list('함꽃정')
word.extend('복숭아')
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: C:\Python\Python200 Python\Python200-Python-Lecture\Wch5\W05-11listcomprehension.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]
>>>
```

튜플

튜플은 리스트처럼 요소를 일렬로 저장하지만, 안에 저장된 요소를 변경, 추가, 삭제를 할 수 없다.

변수에 값을 저장할 때 `()`(괄호)로 묶어주면 튜플이 되며 각 값은 `,`(кома)로 구분해줍니다. 또는, 괄호로 묶지 않고 값만 콤마로 구분해도 튜플이 됩니다.

- 튜플 = (값, 값, 값)
- 튜플 = 값, 값, 값

```
>>> a = (38, 21, 53, 62, 19)
>>> a
(38, 21, 53, 62, 19)
>>> b = 38, 21, 53, 62, 19
>>> a
(38, 21, 53, 62, 19)
```

튜플도 리스트처럼 여러 자료형을 섞어서 저장해도 됩니다.

```
>>> person = ('james', 17, 175.3, True)
>>> person
('james', 17, 175.3, True)
```

range를 사용하여 튜플 만들기

- 튜플 = tuple(range(횟수))

```
>>> a = tuple(range(10))
>>> a
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

- 튜플 = tuple(range(시작, 끝))

```
>>> b = tuple(range(5, 12))
>>> b
(5, 6, 7, 8, 9, 10, 11)
```

- 튜플 = tuple(range(시작, 끝, 증가폭))

```
>>> c = tuple(range(-4, 10, 2))
>>> c
(-4, -2, 0, 2, 4, 6, 8)
```

튜플을 리스트로 만들고 리스트를 튜플로 만들기

```
>>> a = [1, 2, 3]
>>> tuple(a)
(1, 2, 3)
```

반대로 list 안에 튜플을 넣으면 새 리스트가 생성됩니다.

```
>>> b = (4, 5, 6)
>>> list(b)
[4, 5, 6]
```

다른 예제

<pre>singer = ('BTS', '불발간사춘기', '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('불발간사춘기')) print(singer.index('BTS')) print() for _ in range(len(singer)): print('%s : %s' % (singer[_], song[_]))</pre>	<pre>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: C:\Python\Python2020\Python\Python-lecture\ch5\05-12koptuple.py ==== ('BTS', '불발간사춘기', 'BTS', '블랙핑크', '태연') ('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계') 2 1 0 BTS : 작은 것들을 위한 시 불발간사춘기 : 나만, 봄 BTS : 소우주 블랙핑크 : Kill This Love 태연 : 사계 >>> </pre>
--	---

<pre>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)</pre>	<pre>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:\Python\Python2020\Python\Python-lecture\ch5\05-13daytuple.py ==== <class 'tuple'> ('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday') ('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', ' sunday', 'sunday') >>> </pre>
--	---



python

딕셔너리와 집합

딕셔너리

Key 와 Value 의 쌍 여러 개가 { }로 둘러싸여 있다. 각각의 요소는 Key : Value 형태로 이루어져 있고 쉼표(,)로 구분되어 있다.

```
{Key1:Value1, Key2:Value2, Key3:Value3, ...}
```

다음 예는 Key 로 정수 값 1, Value 로 문자열 'hi'를 사용한 예이다.

```
>>> a = {1: 'hi'}
```

또한 다음 예처럼 Value 에 리스트도 넣을 수 있다.

```
>>> a = { 'a': [1,2,3]}
```

딕셔너리 쌍 추가하기

```
>>> a = {1: 'a'}  
>>> a[2] = 'b'  
>>> a  
{1: 'a', 2: 'b'}
```

{1: 'a'} 딕셔너리에 a[2] = 'b'와 같이 입력하면 딕셔너리 a 에 Key 와 Value 가 각각 2 와 'b'인 2 : 'b'라는 딕셔너리 쌍이 추가된다.

```
>>> a['name'] = 'pey'  
>>> a  
{1: 'a', 2: 'b', 'name': 'pey'}
```

딕셔너리 a 에 'name': 'pey'라는 쌍이 추가되었다.

```
>>> a[3] = [1,2,3]  
>>> a  
{1: 'a', 2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

딕셔너리에서 Key 사용해 Value 얻기

```
>>> a = {1: 'a', 2: 'b'}  
>>> a[1]  
'a'  
>>> a[2]
```

```
'b'
>>> dic = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> dic['name']
'pey'
>>> dic['phone']
'0119993323'
>>> dic['birth']
'1118'
```

빈 딕셔너리 만들기

- 딕셔너리 = {}
- 딕셔너리 = dict()

```
>>> x = {}
>>> x
{}
>>> y = dict()
>>> y
{}

```

dict 로 딕셔너리 만들기

```
lux1 = dict(health=490, mana=334, melee=550, armor=18.72)
# 키=값 형식으로 딕셔너리를 만들

>>> lux1
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux2 = dict(zip(['health', 'mana', 'melee', 'armor'], [490, 334, 550,
18.72]))
# zip 함수로
# 키 리스트와 값
리스트를 묶음
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux3 = dict([('health', 490), ('mana', 334), ('melee', 550), ('armor',
18.72)])
# (키, 값) 형식의 튜플로
딕셔너리를 만들
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
>>> lux4 = dict({'health': 490, 'mana': 334, 'melee': 550, 'armor':
18.72})
# dict 안에서
# 중괄호로
딕셔너리를 만들
{'health': 490, 'mana': 334, 'melee': 550, 'armor': 18.72}
```

Key 리스트 만들기(keys)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> a.keys()
dict_keys(['name', 'phone', 'birth'])
>>> list(a.keys())
['name', 'phone', 'birth']
```

Value 리스트 만들기(values)

```
>>> a.values()
dict_values(['pey', '0119993323', '1118'])
```

Key, Value 쌍 얻기(items)

```
>>> a.items()
dict_items([('name', 'pey'), ('phone', '0119993323'), ('birth', '1118')])
```

Key 로 Value 얻기(get)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> a.get('name')
'pey'
>>> a.get('phone')
'0119993323'
```

해당 Key 가 딕셔너리 안에 있는지 조사하기(in)

```
>>> a = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
>>> 'name' in a
True
>>> 'email' in a
False
```

예제

```
File Edit Format Run Options Window Help
color = dict(검은색='black', 흰색='white', 녹색='green', 파란색='blue')
print(color)

#항목 조회
print(color.get('녹색'))
print(color.get('노란색'))

#항목 추가
color['노란색'] = 'yellow'
print(color)
print()

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

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

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

#모든 항목 삭제
color.clear()
print(color)

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: C:\Python\Python2020\Python\lecture\ch6\06-color\dict.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
원의 삭제: ('노란색', 'yellow')
원의 삭제 후: {'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
검은색 삭제 후: {'녹색': 'green', '파란색': 'blue'}
{}
>>> |
```

```

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()

```

```

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: C:\Python\Python2020\Python\Python2020-Python-lecture\ch6\06-05seasondict.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
>>>

```

집합

집합(set)의 특징

- set 자료형은 중복을 허용하지 않습니다.
- set 은 입력된 순서는 중요하지 않습니다.
- set() 생성자 함수를 이용해서 만들 수 있습니다.
- {}를 이용해서 만들 수 있습니다.
- add() 메서드를 이용하여 추가가 가능합니다.
- update() 메서드를 이용하여 동시에 여러개 추가가 가능합니다.
- remove() 메서드를 이용하여 삭제가 가능합니다.
- copy() 메서드를 이용하여 복사가 가능합니다.
- clear() 메서드를 이용하여 모든 요소 삭제가 가능합니다.

```

>>> s1 = set([1,2,3])
>>> s1
{1, 2, 3}

```

위와 같이 set()의 괄호 안에 리스트를 입력하여 만들거나 다음과 같이 문자열을 입력하여 만들 수도 있다.

```

>>> s2 = set("Hello")
>>> s2
{'e', 'H', 'l', 'o'}

```

교집합, 합집합, 차집합 구하기

1. 교집합

s1 과 s2 의 교집합을 구해 보자.

```

>>> s1 & s2
{4, 5, 6}

```

"&" 기호를 이용하면 교집합을 간단히 구할 수 있다.

```

>>> s1.intersection(s2)

```



```
{4, 5, 6}
```

s2.intersection(s1)을 사용해도 결과는 같다.

2. 합집합

```
>>> s1 | s2  
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

"|" 기호를 사용한 방법이다.

```
>>> s1.union(s2)  
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

또는 union 함수를 사용하면 된다. 교집합에서 사용한 intersection 함수와 마찬가지로 s2.union(s1)을 사용해도 동일한 결과를 돌려준다.

3. 차집합

```
>>> s1 - s2  
{1, 2, 3}  
>>> s2 - s1  
{8, 9, 7}
```

빼기(-) 기호를 사용한 방법이다.

```
>>> s1.difference(s2)  
{1, 2, 3}  
>>> s2.difference(s1)  
{8, 9, 7}
```

difference 함수를 사용해도 차집합을 구할 수 있다.

집합 자료형 관련 함수들

값 1 개 추가하기(add)

```
>>> s1 = set([1, 2, 3])  
>>> s1.add(4)  
>>> s1  
{1, 2, 3, 4}
```

값 여러 개 추가하기(update)

여러 개의 값을 한꺼번에 추가(update)할 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 2, 3])  
>>> s1.update([4, 5, 6])
```

```
>>> s1
{1, 2, 3, 4, 5, 6}
```

특정 값 제거하기(remove)

특정 값을 제거하고 싶을 때는 다음과 같이 하면 된다.

```
>>> s1 = set([1, 2, 3])
>>> s1.remove(2)
>>> s1
{1, 3}
```

내장 함수 zip()과 enumerate()

zip

zip(*iterable)은 동일한 개수로 이루어진 자료형을 묶어 주는 역할을 하는 함수이다.
※ 여기서 사용한 *iterable 은 반복 가능(iterable)한 자료형 여러 개를 입력할 수 있다는 의미이다.

```
>>> list(zip([1, 2, 3], [4, 5, 6]))
[(1, 4), (2, 5), (3, 6)]
>>> list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9]))
[(1, 4, 7), (2, 5, 8), (3, 6, 9)]
>>> list(zip("abc", "def"))
[('a', 'd'), ('b', 'e'), ('c', 'f')]
```

enumerate

enumerate 는 "열거하다"라는 뜻이다. 이 함수는 순서가 있는 자료형(리스트, 튜플, 문자열)을 입력으로 받아 인덱스 값을 포함하는 enumerate 객체를 돌려준다.

※ 보통 enumerate 함수는 다음 예제처럼 for 문과 함께 자주 사용한다.

```
>>> for i, name in enumerate(['body', 'foo', 'bar']):
...     print(i, name)
...
0 body
1 foo
2 bar
```

예제

```
#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('한수 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, 인원수를 값으로 저장
print('한수 dict(zip()):')
sportsnum = dict(zip(sports, num))
print(sportsnum)
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (I
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Python\Python2020\Python\Python-lecture\ch6\06-10teamnumdict.py =
['축구', '야구', '농구', '배구']
[11, 9, 5, 6]

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

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



python™

맺음말

본 포트폴리오를 끝내며 더 열심히 준비할 걸 하는 아쉬운 마음이 들었다. 강의를 더 집중해서 주차별로 노트를 정리하고 넣으면 공부에 더 좋았을 것이라 생각한다. 수강 소감을 적어 정리 하지 못한 점이 아쉽다. 중학교 2년 처음으로 프로그램을 접한 나는 2학년 들어서는 파이썬, C 언어, HTML 등 배워 내 자신이 아주 자랑스럽다. 앞으로 갈 길이 아직 먼 걸 잘 아는데 가끔 내가 이 분야를 전공하는 것을 아주 맞는 선택이라 생각이 들어 더 열심히 하고 싶은 마음이 들었다.

파이썬 프로그래밍 강의에 여러 가지 새로운 것을 배울 수 있어 잘 가르치신 교수님께 정말 고마웠다는 말을 하고 싶다. 교수님 덕분에 내가 그 동안 몰랐던 것을 알게 되고 역시 직장에서도 활용할 수 있다. 그 다음에 응원해주신 가족과 친구들이 없었으면 대학교 생활이 많이 어려웠을 것이며 덕분에 지금까지 잘 버텼고 앞으로도 그랬으면 좋겠다.

처음으로 프로그래밍에 전혀 관심 없는 나는 이 분야에 졸업하고 좋은 인재를 되고 싶다. 그러게 되려면 무슨 문제든 절대 포기되지 않고 더 좋은 사람이 되기 위하여 앞으로 내 자신에게 계속 도전 할 것이다.