2020 ver.

파이썬 프로그래밍 포트폴리오

<제출자>
정보통신과 **20152652** 김주한

목차

- 강의계획서
- 파이썬언어
- 파이썬 기초 다지기
- 문자열과 논리연산
- 조건과 반복
- 리스트와 튜플
- 딕셔너리와 중복을 불허하는 집합
- 소감

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 1학기	전공	컴퓨터정보공학과	학부	컴퓨터공학부
과 목 명	파이썬프로그래밍(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 : 2호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 화요일 13~16

학과 교육목표	
과목 개요	2010년 이후 파이썬의 폭발적인 인기는 제4차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 컴퓨팅 사고력은 누구나가 가져야할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제4차 산업혁명 시대의 기술을 이끌고 있다. 제4차 산업혁명 시대를 주도하는 핵심 기술은 데이터과학과 머신러닝, 딥러닝이며,이러한 분야에 적합한 언어인 파이썬은 매우중요한 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 4차 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	비고
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.

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

1 주차	[개강일(3/16)]
학습주제	교과목 소개 및 강의 계획 1장 파이썬 언어의 개요와 첫 프로그래밍
목표및 내용	• 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 1장, 파이썬 개발환경 설치 파이썬 IDLE
과제,시험,기타	도전 프로그래밍
2 주차	**[2주]**
학습주제	2장 파이썬 프로그래밍을 위한 기초 다지기
목표및 내용	• 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 IDLE을 활용할 수 있다.
미리읽어오기	교재 2장 리터럴과 변수의 이해 아나콘다의 주피터 노트북
과제,시험,기타	도전 프로그래밍
3 주차	**[3주]**
학습주제	3장 일상에서 활용되는 문자열과 논리 연산
목표및 내용	• 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 3장 문자열과 논리연산 파이참(pycharm)
과제,시험,기타	도전 프로그래밍
4 주차	**[4주]**
학습주제	4장 일상생활과 비유되는 조건과 반복
목표및 내용	• 조건에 따라 하나를 결정하는 if문을 구현할 수 있다. • 반복을 수행하는 while문과 for문을 구현할 수 있다. • 임의의 수인 난수를 이해하고 반복을 제어하는 break문과 continue문을 활용할 수 있다. • 파이참(pycharm)을 활용할 수 있다.
미리읽어오기	교재 4장 조건과 반복
과제,시험,기타	도전 프로그래밍

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

5 주차	[5주]
학습주제	5장 항목의 나열인 리스트와 튜플
목표및 내용	• 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 5장 배열과 리스트
과제,시험,기타	도전 프로그래밍
6 주차	[6주]
학습주제	6장 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합
목표및 내용	• 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 6장 집합
과제,시험,기타	도전 프로그래밍
7 주차	[7주]
학습주제	7장 특정 기능을 수행하는 사용자 정의 함수와 내장 함수
목표및 내용	• 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 7장 함수의 정의와 호출
과제,시험,기타	도전 프로그래밍
8 주차	[중간고사]
학습주제	- 직무수행능력평가 1차(중간고사)
목표및 내용	직무수행능력평가, 서술형 평가
미리읽어오기	교재 1장에서 7장까지
과제,시험,기타	
9 주차	[9주]
학습주제	8장 조건과 반복, 리스트와 튜플 기반의 미니 프로젝트 I
목표및 내용	8개의 미니 프로젝트를 스스로 생각하고 프로그래밍해 코딩 능력뿐 아니라 문제 해결 능력을 키울 수 있다.
미리읽어오기	교재 8장
과제,시험,기타	

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

10 주차	[10주]
학습주제	9장 라이브러리 활용을 위한 모듈과 패키지
목표및 내용	• 표준 모듈을 이해하고 사용자 정의 모듈도 직접 구현해 사용할 수 있다. • 표준 모듈인 turtle을 사용해 기본적인 도형을 그릴 수 있다. • 써드파티 모듈 numpy와 matplotlib 등을 설치해 활용할 수 있다.
미리읽어오기	교재 9장
과제,시험,기타	도전 프로그래밍
11 주차	**[11주]**
학습주제	10장 그래픽 사용자 인터페이스 Tkinter와 Pygame
목표및 내용	• GUI를 이해하고 GUI 표준 모듈인 Tkinter를 사용해 필요한 위젯을 구성하고 윈도를 생성할 수 있다. • 이벤트 처리를 이해하고 Tkinter에서 이벤트 처리를 구현할 수 있다. • 써드파티 GUI 모듈인 pygame을 설치해 기본적인 윈도를 구현할 수 있다.
미리읽어오기	교재 10장
과제,시험,기타	도전 프로그래밍
12 주차	**[12주]**
학습주제	11장 실행 오류 및 파일을 다루는 예외 처리와 파일 입출력
목표및 내용	• 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 11장
과제,시험,기타	도전 프로그래밍
13 주차	**[13주]**
학습주제	12장 일상생활의 사물 코딩인 객체지향 프로그래밍
목표및 내용	• 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 12장
과제,시험,기타	도전 프로그래밍
14 주차	**[14주]**
학습주제	13장 GUI 모듈과 객체지향 기반의 미니 프로젝트 II
목표및 내용	학습한 파이썬 문법 구조와 프로그래밍 기법을 활용해 8개의 미니 프로젝트를 스스로 생각하고 프로그래밍해 코딩 능력뿐 아니라 문제 해결 능력을 키울 수 있다.
미리읽어오기	교재 1장
과제,시험,기타	

강의계획서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

15 주차	[기말고사]
학습주제	직무수행능력평가 2차(기말고사)
목표및 내용	직무수행능력평가, 서술형평가
미리읽어오기	8장에서 13장까지
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.

파이썬언어 - 파이썬이란?

- 배우기 쉽고 누구나 무료로 사용할 수 있는 오픈 소스 (Open Source) 프로그래밍 언어이다.
- 1991 년 네덜란드의 귀도 반 로섬 (Guido van Rossum) 이 개발 , 현재는 비영리 단체인 파이썬 소프트웨어 재단 (PSF : Python Software Foundation) 이 관리하고 있다.
- 현재 미국과 우리나라의 대학 등 전 세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나이다.

파이썬언어 - 컴퓨팅 사고력과 파이썬

- 컴퓨팅 사고력이란 컴퓨팅의 기본 개념과 원리를 기반으로 문제를 효율적으로 해결하는 사고 능력이다.
- 파이썬을 활용한 프로그래밍 교육은 컴퓨터 전공자에겐 전문적인 프로그래밍 절차와 기술을 교육하는데 적합하고, 비정공자에겐 컴퓨팅 사고력을 교육하는 데 적합하다.

파이썬언어 - 다양한 종류의 파이썬

- C 파이썬
- 아이파이썬
- 자이썬
- 아이언파이썬
- 파이파이

파이썬언어 - 다양한 종류의 개발환경

- 기본 IDE 에 추가 : 비주얼 스튜디오 파이썬 도구 , PyDev 설치 이클립스
- 파이썬 전용 IDE : PyCharm, Spyder, Jupyter Notebook, Jupyter Lab, Comodo, Wing python IDE, PyScripter , Thonny 등
- 편집기 전문 개발 환경 : Sublime Text, Visual Studio Code, Notepad++, Atom, Vim 등

파이썬언어 - 인터프리트 방식의 언어

- 파이썬은 인터프리터 위에서 실행되는인터프리트 방식의 언어이다.
- 인터프리터 언어는 프로그램의 코드가 한 줄씩 순서대로 해석되고 실행되기를 반복한다. 따라서 코드가 완전히 작성되지 않아도 작성된 부분까지 실행해 볼 수 있는 장점이 있다.
- 인터프리터는 한 줄 한 줄의 해석을 담당하고 컴파일러는 컴파일을 담당하는 개발 도구 소프트웨어다.

  대표적 인터프린터 - 파이썬 쉘

  대표적 컴파일러 - C, Java

파이썬 기초 다지기 - 문자열과 수

- 문자열 - 문자 하나 또는 문자가 모인 단어나 문장 또는 단락 등 '일련의 문자 모임'이라 할 수 있다.
- 작은따옴표나 큰따옴표로 앞뒤를 둘러싸 '문자열' 또는 "문자열"로 표현한다.

```
print('hello world!')
print("Hi, python")
```

```
hello world!
Hi, python
```

- 문자열의 따옴표는 앞뒤를 동일하게 사용
- #을 사용하여 주석처리 가능

```
#print('hello world!')
#print("Hi, python")
```

- 삼중따옴표를 사용하여 여러줄 주석처리 가능

파이썬 기초 다지기 - 문자열과 수

- 정수는 print(정수 로 바로 출력가능하고 대화형 모드에서는 숫자만 입력해도 된다.
- 실수는 문자 e 를 사용해 지수승으로 표현할 수 있다.
- 연산자에는 더하기 +, 빼기 -, 곱하기 *, 나누기 /, 몫 //, 나머지 %, 지수승 ** 이 있다
- 함수 eval() 은 실행 가능한 연산식 문자열인 expression을 실행한 결과를 반환

```
a = eval('3 + 15 / 2')
print(a)
b = eval('4 * 3 % 5')
print(b)
```

```
10.5
2
```

파이썬 기초 다지기 - 변수와 키워드, 대입 연산자

- 자료형을 직접 알아보려면 type() 함수 사용한다.
- 변수란 변하는 자료를 저장하는 메모리 공간이다.
- 값을 변수에 저장하기 위해서는 대입 연산자(=)가 필요하다.
- 대입 연산자의 오른쪽에는 값 , 왼쪽에는 반드시 저장공간인 변수가 와야 한다.
- 대입 연산자를 사용하면 여러 변수에 같은 값을 대입할 수 있다.
- 변수의 공간은 하나이므로 마지막에 저장된 값만 기억한다.
- 파이썬 문법에서 사용되는 단어를 키워드 라고 하는데 파이썬에서는 총 33 개이다.

```
a = eval('3 + 15 / 2')
print(a)
b = eval('4 * 3 % 5')
print(b)
```

```
10.5
2
```

```
print(type(a))
print(type(b))
```

```
<class 'float'>
<class 'int'>
```

파이썬 기초 다지기 - 자료의 표준 입력과 자료 변환 함수

- 표준입력 - 프로그램 과정에서 쉘이나 콘솔에서 사용자의 입력을 받아 처리하는 방식이다.
- 함수 input() - 입력되는 표준 입력을 문자열로 읽어 반환하는 함수이고 대입 연산자 를 사용해 변수에 저장한다.
- ()안에 문장을 넣으면 콘솔에 출력되고 이후 표준 입력 문자열을 입력받을 수 있어 편리하다.

```
input('내용입력 >> ')
```

```
내용입력 >>
```

- 함수 str() 은 정수와 실수를 문자열로 변환한다.
- 함수 int() 는 정수 형태의 문자열을 정수한다.
- 함수 float() 는 소수점이 있는 실수 형태의 문자열을 실수로 변환한다.

파이썬 기초 다지기 - 예제

hw(6)_2week_20152652.py - C:/Users/wngks/Desktop/3학년 1학기/파이썬/...

File Edit Format Run Options Window Help

```
a = first_num = int(input('Enter First number: '))
b = second_num = int(input('Enter Second number: '))
div = a / b
mod = a % b
f_div = a // b
exp = a ** b
print('{} / {} ==> {}'.format(a,b,div))
print('{} % {} ==> {}'.format(a,b,mod))
print('{} // {} ==> {}'.format(a,b,f_div))
print('{} ** {} ==> {}'.format(a,b,exp))
```

Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: C:/Users/wngks/Desktop/3학년 1학기/파이썬/2주차/hw(6)_2week_201526
52.py ===
Enter First number: 12
Enter Second number: 5
12 / 5 ==> 2.4
12 % 5 ==> 2
12 // 5 ==> 2
12 ** 5 ==> 248832
>>>
```

Ln: 11 Col: 4

문자열과 논리연산 - 문자열 다루기

- 파이썬에서 문자열은 '문자의 나열'로, 텍스트 시퀀스 라고도 한다.
- 문자열의 자료형은 class str이다.
- 문자열 내부에 작은 따옴표와 큰따옴표를 번갈아 쓰면서 표현 가능하다.

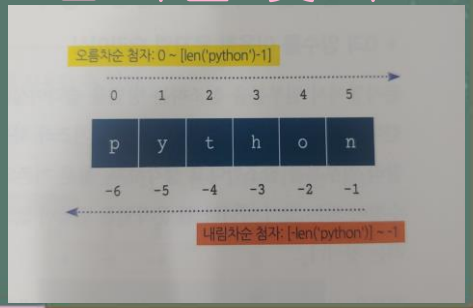
```
>>> print("'작은따옴표, 큰따옴표 번갈아 쓰기'")
'작은따옴표, 큰따옴표 번갈아 쓰기'
>>> print('"작은따옴표, 큰따옴표 번갈아 쓰기"')
"작은따옴표, 큰따옴표 번갈아 쓰기"
>>>
```

- 함수 len()은 문자열의 길이를 알 수 있다

```
>>> a = 'class101'
>>> len(a)
8
>>>
```

- 문자열을 구성하는 문자는 0부터 시작되는 첨자를 대괄호 안에 기술해 참초가능하다. -1 부터 시작돼 -2,-3 으로 작아지는 첨자도 역순으로 참조한다. 첨자가 유효 범위를 벗어나면 IndexError

```
>>> a[6]
'0'
>>> a[8]
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    a[8]
IndexError: string index out of range
```



문자열과 논리연산 - 슬라이싱(slicing)

- 문자열에서 일부분을 참조하는 방법을 슬라이싱 이라고 한다.
- 콜론을 사용한 [start:end] 로 부분 문자 환하는데 start 에서 end-1 첨자까지 문자열을 반

```
>>> a[0:7]
'class10'
>>> a[0:8]
'class101'
>>> a[0:len(a)]
'class101'
>>> a[-8:-1]
'class10'
```

- 슬라이스는 음수도 사용할 수 있다.

- start, end를 비우면 '처음부터'와 '끝까지' 를 의미한다.
- 슬라이싱에서 문자사이의 간격을 step 으로 조정한다. [start:end:step] 을 사용하고 step 을 생략하면 1 이다

문자열과 논리연산 - 이스케이프 시퀀스 문자

- 하나의 문자를 역슬래시 □ 로 시작하는 조합으로 표현하는 문자를 이스케이프 시퀀스 문자 라고 한다.
- 내장함수 min(), max() 는 인자의 최솟값과 최대값을 반환한다

이스케이프 시퀀스 문자	설명
\\	역슬래시
\'	작은따옴표
\"	큰따옴표
\a	벨소리(알람)
\b	백스페이스(이전 문자 지우기)
\n	새 줄
\N{name}	유니코드의 이름
\r	동일한 줄의 맨 앞으로 이동 (파이썬 쉘에서는 다음 줄로)
\f	폼피드(form feed) (예전 프린터에서 다음 페이지의 첫 줄로 이동)
\t	수평 탭
\v	수직 탭
\uxxxx	16비트 16진수 코드
\Uxxxxxxxx	32비트 16진수 코드
\ooo	8진수의 코드 문자
\xhh	16진수의 코드 문자

문자열과 논리연산 - 문자열 관련 메소드

- replace() - str.replace(a,b) 로 사용하고, 문자열 str에서 a가 나타나는 모든 부분을 b로 모두 바꾼 문자열을 반환한다.
- count() - str.count(부분 문자열) 로 사용하고, 문자나 부분 문자열의 출현 횟수를 출력한다.
- join() - str.join() 로 사용하고, 문자열 중간중간에 str을 삽입한다.

```
>>> '>'.join('class101')
'c>l>a>s>s>1>0>1'
>>>
```

- find(), index() - 문자열에서 문자열s 가 맨 처음에 위치한 첨자를 반환받으려면 str.find(s), str.index(s) 로 사용하면 된다. 찾는 문자열이 없는 경우 index()함수는 ValueError를 find()함수는 -1을 반환한다.

문자열과 논리연산 - 문자열 관련 메소드

- split() - str.split() 으로 사용하고, 문자열 str에서 공백을 기준으로 문자열을 나눠 준다. 결과는 나눠진 항목들이 리스트라는 형태로 표시된다.
- 영문자 알파벳 변환 메소드 - upper() - 대문자, lower() - 소문자
- center() - 폭을 지정하고 중앙에 문자열 배치하는 메소드
- ljust () - 문자열 폭 지정 후 왼쪽 정렬
- rjust () - 문자열 폭 지정 후 오른쪽 정렬
- strip() - 문자열 앞뒤의 특정 문자들을 제거
- zfill() - 제로 0을 채워 넣는 메소드

문자열과 논리연산 - 문자열 관련 메소드

- format() - 간결한 출결 처리

```
a = first_num = int(input('Enter First number: '))
b = second_num = int(input('Enter Second number: '))
div = a / b
mod = a % b
f_div = a // b
exp = a ** b
print('{} / {} ==> {}'.format(a,b,div))
print('{} % {} ==> {}'.format(a,b,mod))
print('{} // {} ==> {}'.format(a,b,f_div))
print('{} ** {} ==> {}'.format(a,b,exp))
```

```
52.py ===
Enter First number: 12
Enter Second number: 5
12 / 5 ==> 2.4
12 % 5 ==> 2
12 // 5 ==> 2
12 ** 5 ==> 248832
>>>
```

  - {n:md} - 정수를 형식 유형으로 출력 처리
  - {n:mf} - 실수를 형식 유형으로 출력 처리
- %d, %f - 전통적인 정형화 방식을 사용한 출력 처리

문자열과 논리연산 - 논리 자료와 다양한 연산

- 논리 값으로 True, False 를 키워드 제공한다.
- True, False 는 int 함수로 각각 1, 0 으로 변환할 수 있다.
- 논리곱 and(&) - 두 항 모두 참이면 True, 하나라도 거짓이면 False
- 논리합 연산자 or(|) - 두 항 모두 거짓이면 False, 하나라도 거짓이면 True
- 배타적 논리합 ^ - 두 항이 다르면 True, 같으면 False
- 연산자 not - 뒤에 위치한 논리값 변환

조건과 반복 - if ... else

- 조건에 따라 해야 할 일 (문장들) 처리해야 하는 경우if 문을 사용한다
- if 문에서 논리 표현식 이후에는 반드시 콜론이 있어야 한다
- 콜론 이후 다음 줄부터 시작되는 블록은 반드시 들여쓰기 (indentation)를 해야 한다 . 그렇지 않으면 오류가 발생한다.

```
height = 152
if 140 <= height:
    print('롤러코스터 T-Express, 즐기세요!!')
```

```
롤러코스터 T-Express, 즐기세요!!
```

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

```
1학기 평균 평점은? 4.4
축하합니다! 장학금 지급 대상자입니다.
당신의 1학기 평균 평점은 4.40이다.
```

조건과 반복 - if ... else

- if 문에서 논리 표현식 결과가 True 이면 논리 표현식 콜론 이후 블록을 실행한다
- 논리 표현식의 결과가 False 이면 else: 이후의 블록을 실행한다

```
from time import localtime
hour = localtime().tm_hour
mnt = localtime().tm_min

if hour < 10:
    print('지금 시각: %d시 %d분, 조조 할인 된다. ' % (hour, mnt))
else:
    print('지금 시각: %d시 %d분, 조조 할인 안 된다. ' % (hour, mnt))
```

```
지금 시각: 14시 44분, 조조 할인 안 된다.
>>>
```

조건과 반복 - if ... elif

- 다중 택일 결정 구조 - 조건의 여러 경로 중 하나를 선택하는 if... elif 문
- elif 는 필요한 만큼 늘릴 수 있으며 , 마지막 else 는 선택적으로 생략할 수 있다.

```
PM = float(input('미세먼지(100마이크로그램)의 농도는 ? '))
if 151 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '매우나쁨'))
elif 81 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '나쁨'))
elif 31 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '보통'))
else:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '좋음'))
```

```
미세먼지(100마이크로그램)의 농도는 ? 150
미세먼지 농도: 150.00, 등급: 나쁨
```

조건과 반복 - for, while

- while 문 - 반복조건 에 따라 반복을 결정한다.
- for 문 - 항목의 나열인 시퀀스 의 구성 요소인 모든 항목이 순서대로 변수에 저장돼 반복을 수행한다.
- 반복 for 문의 시퀀스에 내장함수 range() 활용 - 범위를 설정하는 함수로 안에 숫자를 넣으면 0 부터 해당 숫자까지 범위를 나타낸다.

```
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= ' ')
```

```
10진수의 한 자릿수 입력 >> 5
두 자릿수 정수에서 최소 한 자릿수가 5인 정수 찾기:
====================== 결과 ======================
15 25 35 45 50 51 52 53 54 55 56 57 58 59 65 75 85 95
```

```
MAXNUM = 4
MAXHEIGHT = 130

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

```
키는 ? 150
커서 못들어갑니다.
키는 ? 130
커서 못들어갑니다.
키는 ? 110
들어가세요. 1명
키는 ? 90
들어가세요. 2명
키는 ? 80
들어가세요. 3명
키는 ? 70
들어가세요. 4명
4명 모두 찼습니다. 다음 번에 이용하세요.
```

조건과 반복 - random.randint(a,b)

- random.randint(a,b) - 임의의 수를 발생하는 난수, a~b 중 한 가지 수를 임의로 얻을 수 있다.
- randint 를 사용하기 위해서는 모듈 random 을 import 한 후 사용해야 한다

```
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')
```

```
======= 모의 로또 당첨 번호 ========
(11, 17, 28, 30, 33, 35)

---------- 내 번호 확인 ----------
36 x  39 x  7 x  38 x  44 x  19 x
0 개 맞음
```

조건과 반복 - break 문과 continue 문

- for 나 while 반복 내에서 문장 break 는 else: 블록을 실행시키지 않고 반복을 무조건 종료한다.
- break 문은 특정한 조건에서 즉시 반복을 종료할 경우 사용한다.
- for 나 while 문 내부에서 continue 문은 이후의 반복 몸체를 을 위해 논리 조건을 수행한다.

```
from random import randint
LUCKY = 7

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

```
1, 7 나올 때까지 계속!
2, 7 나올 때까지 계속!
6, 7 나올 때까지 계속!
6, 7 나올 때까지 계속!
드디어 7, 종료!
```

리스트와 튜플 - 리스트

- 리스트는 여러 자료 값을 편리하게 처리한다.
- 리스트는 대괄호 [] 사이에 항목을 기술한다.
- 빈 대괄호로 빈 리스트를 만들 수 있다.
- list() - 빈 리스트를 생성 가능
- append() - 리스트 맨 뒤에 항목을 추가

```
goods = []
for i in range(3):
    item = input('구입할 품목은 ? ')
    goods.append(item)
    print(goods)
print('길이: %d' % len(goods))
```

```
구입할 품목은 ? 과자
['과자']
구입할 품목은 ? 우유
['과자', '우유']
구입할 품목은 ? 고기
['과자', '우유', '고기']
길이: 3
```

- count() - () 안에 값을 갖는 항목수를 출력
- index() - () 안에 값의 항목이 위치한 첨자를 출력

리스트와 튜플 - 부분 참조, 삽입, 삭제

- 문자열과 동일하게 리스트도 부분 참조인 슬라이싱이 가능하다.

```
wlist = ['밥', '삶', '길', '쭉', '꿈', '차', '떡', '복', '말']
print('wlist[:] = ', wlist[:])
print('wlist[::] = ', wlist[::])
print('wlist[::-1] = ', wlist[::-1])

print(wlist[::3])
print(wlist[1::3])
print(wlist[2::3])

print(wlist[::-2])
print(wlist[-1:-8:-3])

print(wlist[1:-1:])
print(wlist[-2:-9:-3])
```

```
wlist[:] =  ['밥', '삶', '길', '쭉', '꿈', '차', '떡', '복', '말']
wlist[::] =  ['밥', '삶', '길', '쭉', '꿈', '차', '떡', '복', '말']
wlist[::-1] =  ['말', '복', '떡', '차', '꿈', '쭉', '길', '삶', '밥']
['밥', '쭉', '떡']
['삶', '꿈', '복']
['길', '차', '말']
['말', '떡', '꿈', '길', '밥']
['말', '차', '길']
['삶', '길', '쭉', '꿈', '차', '떡', '복']
['복', '꿈', '삶']
```

리스트와 튜플 - 부분 참조, 삽입, 삭제

- list.insert() - 리스트의 첨자 위치에 항목을 삽입 , 항목은 무엇이든 가능하며 빈 리스트도 가능하다.
- list.remove() - 리스트에서 지정된 값의 항목을 삭제한다.
- list.pop() - () 안에 지정된 첨자의 항목을 삭제, 첨자가 없다면 마지막 항목을 삭제하고 출력한다.
- del - 문장 del 은 뒤에 위치한 변수나 항목을 삭제한다.
- list.clear() - 모든 항목을 제거하고 빈 리스트로 만든다.
- list.extend() - ()안에 첨자를 list 가장 뒤에 추가한다.
- 연산자(+) - 리스트와 리스트를 연결할 수 있다.
- list.reverse() - 리스트의 항목 순서를 반대로 뒤집는다.
- list.sort() - 리스트 항목의 순서를 오름차순으로 정렬한다. ()에 reverse = True 를 넣으면 역순인 내림차순으로 정렬한다.

리스트와 튜플 - 리스트 컴프리헨션

- 리스트 컴프리헨션을 사용하면 한 리스트의 모든 항목 각각에 대해 어떤 조건을 적용한 후 그 반환값을 항목으로 갖는 다른 리스트를 쉽게 만들 수 있다.

```
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)
```

```
[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]
```

리스트와 튜플 - 리스트 대입과 복사

- 대입 연산자 = 는 얕은 복사라고 하고 'x1 = x2' 라고 한다면 x1 과 x2 는 동일한 메모리 공간을 사용하는 리스트를 공유하게 된다.
- 슬라이스[:] 나 copy() 또는 list() 함수를 이용하면 리스트에서 새로운 리스트를 만들어 복사할수 있고 이를 깊은 복사라고 한다.

리스트와 튜플 - 튜플

- 항목의 순서나 내용을 수정할 수 없다.
- 모두 콤마로 구분된 항목들의 리스트로 표현되며, 각각의 항목은 정수, 실수, 문자열, 리스트, 튜플 등 제한이 없다.
- 괄호는 생략할 수 있다.
- 튜플 이름 = () 로 빈 튜플을 생성할 수 있다.
- tuple[start:stop:step] 으로 슬라이스가 가능하다.

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

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

```
('BTS', '봄빨간사춘기', 'BTS', '블랙핑크', '태연')
('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
2
1
0

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

리스트와 튜플 - 튜플 연결과 반복, 정렬과 삭제

- 연산자 +,* - 튜플을 연결하고 항목이 횟수만큼 반복된 튜플을 반환한다.
- sorted(tuple) - 튜플 항목의 순서를 오름차순으로 정렬한 새로운 리스트를 반환한다. sorted(tuple, reverse=True) 로 호출하면 내침차순으로 정렬된 리스트를 반환한다. 튜플 자체는 수정 되지 않는다.
- del tuple - 튜플을 삭제한다.

```
day1 = ('monday','tuesday','wednesday')
day2 = ('thursday','friday', 'saturday')
day3 = ('sunday', )

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

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

```
<class 'tuple'>
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday')
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday',
sunday', 'sunday')
```

딕셔너리와 중복을 불허하는 집합 - 딕셔너리

- 딕셔너리는 키와 값의 쌍인 항목을 나열한 시퀀스이며, 콤마로 구분된 항목들의 리스트로표현된다
- 딕셔너리는 중괄호 사이에 키와 값을 항목을 기술한다
- 딕셔너리의 항목 순서는 의미가 없으며 , 키는 중복될 수 없다
- 키는 수정될 수 없지만 값은 수정될 수 있다
- 값은 키로 참조된다
- 딕셔너리는 빈 중괄호로 만들 수 있으면 내장 함수 dict() 를 사용해 만들 수도 있다.
- 딕셔너리 항목 값으로 리스트나 튜플 등이 가능하다.

딕셔너리와 중복을 불허하는 집합 - 딕셔너리 값 참조

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

```
lect = dict()
lect['강좌명'] = '파이썬 기초';
lect['개설년도'] = [2020,1];
lect['학점시수'] = (3,3);
lect['교수'] = '김민국';
print(lect)
print(len(lect))
print()

print(lect['개설년도'], lect['학점시수'])
print(lect['강좌명'], lect['교수'])
```

```
{'강좌명': '파이썬 기초', '개설년도': [2020, 1], '학점시수': (3, 3), '교수': '김
민국'}
4

[2020, 1] (3, 3)
파이썬 기초 김민국
```

딕셔너리와 중복을 불허하는 집합 - dict()

- 내장함수 dict() - 리스트나 튜플을 사용해 딕셔너리를 만들 수 있다.
- day = dict([])
- day = dict(())
- 리스트나 튜플 내부에서 일련의 키-값 쌍으로 [키,값] 리스트 형식과 (키,값) 튜플 형식을 모두 사용 가능하다.

```
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 엔터테
bts['구성원'] = ['RM','진','슈가','제이홉', '지민', '뷔', '정국']

print(bts)
print(bts['구성원'])
```

```
{'그룹명': '방탄소년단', '인원수': '7', '리더': '김남준', '소속사': '빅히트 엔터
테인먼트'}
{'그룹명': '방탄소년단', '인원수': 7}
{'리더': '김남준', '소속사': '빅히트 엔터테인먼트'}
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔터테
인먼트', '구성원': ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']}
['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']
```

딕셔너리와 중복을 불허하는 집합 - 딕셔너리 키

- 딕셔너리의 키는 수정 불가능한 객체는 모두 가능하다.
- 원주율, 튜플
- 리스트는 키로 사용 불가능하다.

딕셔너리와 중복을 불허하는 집합 - 딕셔너리 메소드

- dict.key() - 키로만 구성된 리스트를 반환한다
- dict.items() - (키, 값) 쌍의 튜플이 들어 있는 리스트를 반환한다. 각 튜플의 첫 번째 항목은 키, 두 번째 항목은 값 이다.
- dict.values() - 값으로 구성된 리스트를 반환한다.
- for 문에서는 시퀀스 위치에 있는 딕셔너리 변수만으로도 모든 키를 순회할 수 있다.

```
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()
```

```
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
```

딕셔너리와 중복을 불허하는 집합 - 딕셔너리 메소드

- dict.get() - ()안에 키의 해당되는 값을 출력한다.
- dict.pop() - ()안에 키인 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.
- dict.popitem() - 임의의 (키, 값)의 튜플을 반환하고 삭제한다. 만일 데이터가 하나도 없다면 오류가 발생한다.
- del dict - 딕셔너리의 변수나 딕셔너리를 삭제할 수 있다.
- dict.clear() - 기존의 모든 키:값 항목을 삭제한다.
- dict.update() - ()안에 다른 딕셔너리를 합병한다.

딕셔너리와 중복을 불허하는 집합 - 중복을 불허하는 집합

- 집합은 중복되는 요소가 없으며, 순서도 없는 원소의 모임이다.
- 원소를 콤마로 구분하여 중괄호로 둘러싸 표현한다.
- 원소는 불변 값으로 중복될 수 없으며 서로 다른 값이어야 한다.
- 원소는 중복을 허용하지 않으며 원소의 순서는 의미가 없다.
- 집합의 원소는 정수, 실수, 문자열, 튜플 등 수정이 불가능한 것이여야 한다. 리스트나 딕셔너리처럼 가변적인 것은 허용되지 않는다.
- 집합은 내장 함수 set() 으로 생성 할 수 있다. set(원소로 구성된 리스트 or 튜플 or 문자열)
- 인자가 없으면 빈 집합인 공집합이 생성된다.
- 인자가 있으면 하나이며, 리스트와 튜플, 문자열 등이 올 수 있다.

딕셔너리와 중복을 불허하는 집합 - 중복을 불허하는 집합 예제

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

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

```
{'해', '별', '달'}
{'감', '귤'}
{'밤', '잣'}
{('감', '귤'), '해달', ('밤', '잣')}
```

딕셔너리와 중복을 불허하는 집합 - 중복을 불허하는 집합 메소드

- odd.add(원소) - 만들어진 집합에 원소를 추가한다.
- odd.remove(원소) - 집합의 원소를 삭제한다. 삭제하려는 원소가 없으면 KeyError 가 발생한다.
- odd.discard(원소) - 집합의 원소를 삭제한다. 삭제하려는 원소가 없어도 에러가 발생하지 않는다.
- odd.pop() - 임의의 원소를 삭제한다.
- odd.clear() - 집합의 모든 원소를 삭제한다.
- 연산자 a | b - a,b 양쪽 모든 원소를 합하는 합집합을 만든다.
- a.union(b) - a,b 합집합을 반환하며 a 자체는 수정되지않는다.
- 교집합 연산자 & 와 intersection() - 양쪽 집합의 교집합을 반환한다.
- 차집합 연산자 - 와 difference() - 양쪽 집합의 차집합을 반환한다.
- 여집합 연산자 ^ 와 symmetric_difference() - 양쪽 집합의 여집합을 반환한다.

딕셔너리와 중복을 불허하는 집합 - 중복을 불허하는 집합

예제

```
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))
```

```
06₩06-09.py =====
{'일', '목', '금', '월', '화', '토', '수'}
{'금', '목', '월', '화', '수'}
{'수', '목'}
{'월', '일', '금', '토', '화'}
>>>
```

딕셔너리와 중복을 불허하는 집합 - 내장 함수 zip()

- zip() 을 이용하면 몇 개의 리스트나 튜플의 항목으로 조합된 튜플을 만들 수 있다.
- 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어주는 역할을 하는 함수다.
- class 는 'zip' 이다.
- 인자의 수는 2개 이상 올 수 있다.
- 딕셔너리를 간단히 만들 수 있다.

딕셔너리와 중복을 불허하는 집합 - 내장 함수 enumerate()

- 0부터 시작하는 첨자와 항목 값의 튜플 리스트를 생성한다.
- enumerate(시퀀스, start = 1)로 호출하면 시작 첨자를 1로 지정할 수 있다. start는 생략 가능하다.

```
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('{}: {}명'.format(*tp), end=' ')
print(); print()

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

```
['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}
>>>
```

소감

- 지금까지 배웠던 걸 복습하는데 많은 도움이 되었다.
- 머리속에 애매한 개념들이 확실해졌다.
- 다른 언어를 공부할 때 많은 도움이 될 것 같다.
- 확실히 매력있는 언어이다.

감사합니다.