

2020년도 파이썬 교과목 포트폴리오

| | |
|----|----------|
| 학부 | 컴퓨터공학부 |
| 학과 | 컴퓨터정보공학과 |
| 학번 | 20161870 |
| 성명 | 모명준 |

목 차

1. 강 의 계 획 서
2. 파이썬 개요
3. 단위별 학습 자료 및 실습 예제
4. 기타 참고 자료
5. 소감 및 향후 계획

| | | | | |
|--------------|------------------------------|----------|----|--------|
| 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 관련성이 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. | | | |



| | |
|----------|---|
| 1 주차 | [개강일(3/16)] |
| 학습주제 | 교과목 소개 및 강의 계획 1장 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 1장, 파이썬 개발환경 설치 파이썬 IDLE |
| 과제,시험,기타 | 도전 프로그래밍 |
| 2 주차 | [2주] |
| 학습주제 | 2장 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 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 | <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 3장 문자열과 논리연산 파이참(pycharm) |
| 과제,시험,기타 | 도전 프로그래밍 |
| 4 주차 | [4주] |
| 학습주제 | 4장 일상생활과 비유되는 조건과 반복 |
| 목표및 내용 | <ul style="list-style-type: none"> 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 |



| | |
|----------|---|
| 5 주차 | [5주] |
| 학습주제 | 5장 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 5장 배열과 리스트 |
| 과제,시험,기타 | 도전 프로그래밍 |
| 6 주차 | [6주] |
| 학습주제 | 6장 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 6장 집합 |
| 과제,시험,기타 | 도전 프로그래밍 |
| 7 주차 | [7주] |
| 학습주제 | 7장 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 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 | |



| | |
|----------|--|
| 10 주차 | [10주] |
| 학습주제 | 9장 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 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 | <ul style="list-style-type: none"> GUI를 이해하고 GUI 표준 모듈인 Tkinter를 사용해 필요한 위젯을 구성하고 윈도우를 생성할 수 있다. 이벤트 처리를 이해하고 Tkinter에서 이벤트 처리를 구현할 수 있다. 써드파티 GUI 모듈인 pygame을 설치해 기본적인 윈도우를 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 | <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 11장 |
| 과제,시험,기타 | 도전 프로그래밍 |
| 13 주차 | [13주] |
| 학습주제 | 12장 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 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 | |



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

2. 파이썬 개요

- 파이썬이란?

1991년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발한 오픈 소스(open source) 프로그래밍 언어이다. 오픈 소스란 누구나 무료로 사용할 수 있음을 뜻한다. 1995년 개발된 자바 언어보다 먼저 만들어졌다.

- 파이썬의 역사

파이썬 창시자인 귀도 반 로섬이 1989년 12월, 취미로 시작한 프로젝트였다. 귀도는 네덜란드 국립 연구소에서 ABC라는 프로그래밍 언어 개발에 참여했는데, 여기서 영감을 얻어 그를 뛰어넘는 기능을 파이썬에 담았다. 파이썬의 사전적 의미는 '비단뱀'으로, 그리스 신화에서 유래했으며, 그로 인해 파이썬 로고 또한 비단뱀이다. 그러나 실제로는 귀도가 애청하던 코미디 프로그램 'Monty Python's Flying Circus'의 주인공인 6인조 코미디 그룹 'Monty Python'에서 따온 것이라고 한다.

- 현재의 파이썬

현재 파이썬은 미국과 우리나라를 포함해 전 세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나다. 특히 비전공자의 컴퓨팅 사고력(computational thinking)을 향상시키기 위해 많이 활용되고 있다. 그 이유는 파이썬이라는 언어 자체의 간결성, 확장성, 유연성이라는 특징 때문이다. 파이썬은 배우기 쉽고 간결하며, 개발 속도가 빠르고 강력하다. 이로 인해 진입장벽이 낮아 프로그래밍 입문자의 컴퓨팅 사고력 증진에 매우 효과적이다. 또 풍부하고 다양한 전문가용 라이브러리 제공으로 데이터 과학자, 머신러닝 등의 전문 엔지니어, 개발자가 더욱 쉽고 빠르게 소프트웨어를 개발하는 데 도움을 준다. 이렇듯 파이썬은 프로그래밍 교육 뿐만 아니라 실무에서도 사용이 급증하고 있다.

- 컴퓨팅 사고력과 프로그래밍

이러한 파이썬의 사용량 증가는 제4차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 2016년 3월, 이세돌 기사와 알파고의 바둑 대국을 계기로 이 시대가 제4차 산업혁명 시대에 진입하고 있음을 알았고, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 현 시대의 기술을 이끌고 있다. 이러한 제4차 산업혁명 시대에 컴퓨팅 사고력은 누구나가 가져야 할 역량이다. 컴퓨팅 사고력이란 '컴퓨터 과학 원리 및 컴퓨팅 시스템을 활용해 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현해 적용할 수 있는 능력'으로 정의할 수 있다. 컴퓨팅 사고력은 컴퓨터 분야의 문제 해결은 물론, 나아가 일상생활에서의 문제까지 효율적으로 해결할 수 있는 방법을 제공한다. 이러한 과정에서 창의성을 높이는 효과도 볼 수 있다. 영국의 컴퓨팅 교육(www.bbc.com/bitesize)에서는 컴퓨팅 사고력을 분해, 패턴 인식, 추상화, 알고리즘 설계를 구성 요소로 제안하고 있다. 파이썬을 활용한 프로그래밍 교육은 컴퓨터 전공자에게는 전문적 지식과 기술을 교육하는 데 적합하고, 비전공자에게는 컴퓨팅 사고력을 증진시키는 데 적합하다.

3. 단 원 별 학 습 자 료 및 실 습 예 제

Chapter 1. 파이썬 언어의 개요와 첫 프로그래밍

- I 파이썬 설치와 파이썬 셸 실행
- II 제 4차 산업혁명 시대, 모두에게 필요한 파이썬

Chapter 2. 파이썬 프로그래밍을 위한 기초 다지기

- I 다양한 자료 : 문자열과 수
- II 변수와 키워드, 대입 연산자
- III 자료의 표준 입력과 자료 변환 함수

Chapter 3. 일상에서 활용되는 문자열과 논리 연산

- I 문자열 다루기
- II 문자열 관련 메소드
- III 논리 자료와 다양한 연산

Chapter 4. 일상생활과 비유되는 조건과 반복

- I 조건에 따른 선택 if ... else
- II 반복을 제어하는 for문과 while문
- III 임의의 수인 난수와 반복을 제어하는 break문, continue문

Chapter 5. 항목의 나열인 리스트와 튜플


- I 여러 자료 값을 편리하게 처리하는 리스트
- II 리스트의 부분 참조와 항목의 삽입과 삭제
- III 항목의 순서나 내용을 수정할 수 없는 튜플

Chapter 6. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

- I 키와 값인 쌍의 나열인 딕셔너리
- II 중복과 순서가 없는 집합
- III 내장함수 zip()과 enumerate(), 시퀀스 간의 변환

Chapter 1. 파이썬 언어의 개요와 첫 프로그래밍

I 파이썬 설치와 파이썬 셸 실행

파이썬 프로그래밍에 앞서 파이썬 개발 도구를 설치해야 한다. 파이썬 홈페이지(<https://www.python.org/>)에 접속하여 Downloads 메뉴를 누르면 나오는 Download for Windows 아래에 있는 Python을 클릭해 설치한다. 만약 Windows 사용자가 아니라면 Mac OS X나 Other Platforms를 눌러서 자신의 운영체제에 맞는 파이썬을 설치하면 된다. 다운로드가 완료되면 자동 설치 파일인 exe파일을 실행시킨다. 파이썬은 기본적으로 윈도우 로그인계정 이름 하부에 설치된다. 파이썬 설치 프로그램 첫 화면에서 Install Now를 누르면 간단하게 설치가 가능하다. 정상적으로 설치되면 Setup was successful이 나타난다. Close를 눌러 설치를 종료한다. 설치가 완료되었으면 실행시켜 본다. 을 눌러서 Python 폴더를 찾고 IDLE(Python 3.8 32-bit(사용자마다 다름))을 실행한다. 실행이 되면 Python 3.8.2 Shell 이라는 창이 표시된다.

첫 코딩을 해보자. 글자 Hello World!를 셸에 출력하는 프로그램을 만들어 볼 것이다. 셸에 `print('Hello World!')`를 입력한다.(1-1) 모든 명령어는 첫 칸부터 입력해야 한다. 공백으로 두고 입력하게 되면 `SyntaxError`가 발생한다. (그림 1-2)

```
>>> print('Hello World')
Hello World
>>>
```

그림 1-1

```
>>> print('Hello World')
SyntaxError: unexpected indent
>>>
```

그림 1-2

II 제 4차 산업혁명 시대, 모두에게 필요한 파이썬

프로그램 언어는 인터프리터 언어와 컴파일 언어가 있다. 인터프리터 언어란 프로그램 코드가 한 줄씩 순서대로 해석되고 실행하는 반면, 컴파일 언어는 여러 문장의 소스 단위로 번역해 기계어 파일의 실행 파일을 만든 후에 실행하는 방식이다. 파이썬은 인터프리터 방식의 언어로 간결해서 배우기 쉽고, 무료이고 생산성이 높으며, 강력한 라이브러리를 제공하고, 프로그램과 호환되는 풀 언어라는 특징을 가지고 있다. 교육과 학술, 실무 등 다양한 분야에 사용되며 제 4차 산업혁명 시대인 현 시대에 인공지능의 구현과 빅데이터 분석 처리 분야에 사용되고 있다.

- Chapter 1 실습 문제

문제 1. 파이썬 IDLE에서 다음을 출력하는 코드를 작성하시오.

안녕, 파이썬!

문제 2. 파이썬 셸에서 다음을 출력하는 프로그램을 지정된 파일에 저장해 실행하시오.

(파일 `hellopython.py`)

파이썬은 재미있는 언어이다.

문제 3. 파이썬 셸에서 자신을소개하는 프로그램을 지정된 파일에 저장해 실행하시오

(파일 `introduce.py`)

이름: 홍길동
대학: 한국대학교
전공: 컴퓨터공학과

20161870_Joon.py - C:/U:

File Edit Format Run O

```
print('안녕, 파이썬!')
```

hellopython.py - C:/Users/suitk/AppData/Lo

File Edit Format Run Options Window

```
print('파이썬은 재미있는 언어이다.')
```

introduce.py - C:/Users/suitk/AppD:

File Edit Format Run Options V

```
print('이름: 모명준')
print('대학: 동양미래대학교')
print('전공: 컴퓨터공학과')
```

Chapter 2. 파이썬 프로그래밍을 위한 기초 다지기

I 다양한 자료 : 문자열과 수

- **문자열** : 문자 하나 또는 문자가 모인 단어나 문장 또는 단락, 즉 ‘일련의 문자 모임’
파이썬은 문자 하나도 문자열로 취급하며, 따옴표(‘, ”)로 둘러싸게 되면 모두 문자열이다. 따옴표는 앞뒤를 동일하게 사용해야 한다. “ + ’ (x) “ + ” (o)
- **연결 연산자 +, 반복 연산자 *** (그림 2-1) : 연결 연산자는 없어도 연결이 가능하며, 반복 연산자를 사용할 때는 반드시 숫자가 있어야 한다. ex) ‘문자열’ * ‘언어’ = X , ‘문자열’ * 3 = 문자열 문자열 문자열
- **삼중 따옴표** : 문자열이 길거나 필요에 의해 여러 줄에 걸쳐 문자열을 처리하기 위해 사용

```
>>> print("원의 원주율 " + '3.141592')
원의 원주율 3.141592
>>> print("python " 'programming ' + 'language')
python programming language
>>> print('파이썬 언어는' + " 강력하다")
파이썬 언어는 강력하다
>>> print('파이썬' + "언어! " + 3 * "방가 ")
파이썬 언어! 방가 방가 방가
>>>
```

그림 2-1

- **주석** (그림 2-2): 코드 내부에 문법과 상관 없는 파일 이름이나 소스 설명, 문장 설명을 위해 사용, 따옴표 안에는 사용 불가. 삼중 따옴표도 주석으로 사용할 수 있다.

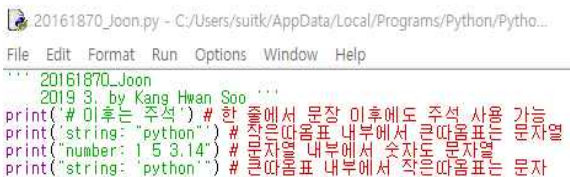


그림 2-2

```
=== RESTART: C:/Use
# 이후는 주석
string: "python"
number: 1 5 3.14
string: 'python'
>>>
```

그림 2-2 실행결과

- 정수와 실수 : 수는 파이썬에서 바로 사용 가능. 실수 표현 시 문자 e, E 사용 가능
- **산술 연산자** : 수를 연산하는데 필요한 연산자. 예제) 2차원 좌표에서 두 점 사이의 거리 계산(그림 2-3)

| 연산자 | 명칭 | 의미 | 우선순위 | 예 |
|-----|-----------|---------------------------------|------|-----------|
| + | 더하기 | 두 피연산자를 더하거나 수의 부호 | 4, 2 | 4 + 5, +3 |
| - | 빼기 | 두 피연산자를 빼거나 수의 부호 | 4, 2 | 5 - 4, -8 |
| * | 곱하기 | 두 피연산자 곱하기 | 3 | 5 * 5 |
| / | 나누기 | 왼쪽을 오른쪽 피연산자로 나누기 | 3 | 8 / 4 |
| % | 나머지 | 왼쪽을 오른쪽 피연산자로 나눈 나머지 | 3 | 12 % 5 |
| // | 몫 나누기 | 왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수 | 3 | 7 // 3 |
| ** | 지수승, 거듭제곱 | 왼쪽을 오른쪽 피연산자로 거듭제곱 | 1 | 3 ** 2 |

```
>>> print(4 ** (1/2))
2.0
>>> print( ((3**2 + 4**2)) ** (1/2) )
5.0
>>> print( ((2-3.1)**2 + (5-4.8)**2) ** (1/2))
1.118033988749895
```

그림 2-3

- 예제) 연산자 //와 %를 사용한 지폐 계산 방법 (그림 2-4)

```
>>> print('계산금액')
계산금액
>>> print(78000)
78000
>>> print('오만 원권')
오만 원권
>>> print(78000 // 50000) ← 몫 연산자로 필요한 5만원권 숫자 계산
1
>>> 78000 % 50000 ← 5만원을 지불 한 후 나머지 금액 계산
28000
>>> print('만 원권')
만 원권
>>> print(_ // 10000) ← _에는 바로 전에 계산된 28000이 저장되어 만원권의 개수 계산
2
>>> _ % 10000 ← 이 식은 ( 78000 % 50000 ) % 10000 과 같음
8000
>>> print('오천원권')
오천원권
>>> print(_ // 5000 + 1) ← 8000원을 5천원 권으로 계산해야 하므로 + 1 해준다.
2
>>> print('잔돈')
잔돈
>>> print(5000 - _ % 5000) ← 잔돈은 마지막으로 계산된 2만8천원을 5000원권으로 나눈 나머지 값 3000원을 5000원에서 빼줌.
2000
>>> print(5000 - (78000 % 50000) % 10000 % 5000) ← 위 식 _ % 5000 은 ( 78000 % 50000 ) % 10000 % 5000 과 같다
2000
```

그림 2-4

- 예제) 예금의 단리와 복리 계산 (그림 2-5)

- 단리 만기 총액 = 원금 x (1 + 이자율 x 기간)

- 복리 총액 = 원금 x ((1 + 이자율))^{기간}

```
>>> 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
```

그림 2-5

- 예제) 한 번에 여러 자료 출력 (그림 2-6) : 출력에 이용되는 함수 print()는 콤마(,)로 구분해 여러 자료를 출력 가능

| | |
|---|---|
| <pre>print(1, 2, -5, 3.14, 2/71828) print('Hi, ', 'Python!') print('23000원은', '5000원 ?개', '1000원 ?개') print('5000원', 23000 // 5000, '개') print('2000원', (23000 % 5000) // 1000, '개')</pre> | <pre>1 2 -5 3.14 2.7844294704015148e-05 Hi, Python! 23000원은 5000원 ?개 1000원 ?개 5000원 4 개 2000원 3 개</pre> |
|---|---|

그림 2-6

그림 2-6 실행결과

II 변수와 키워드, 대입 연산자

- type() 함수 : 자료형을 알아보는 함수, 자료형에는 int, str, float 등이 있다.
- 변수 : 변하는 자료를 저장하는 메모리 공간, 대입 연산자 =를 사용해 값을 저장한다. 변수 이름을 붙일 때는 맨 앞에 숫자 불가, 키워드 사용 불가, 특수문자(_를 제외한) 사용 불가 세 가지 규칙을 지켜야 한다. (그림 2-7)

```
>>> value = 10
>>> Value = 200
>>> value
10
>>> Value
200
>>> total_price = 100000
>>> coffePrice = 4500
>>> _month = 11
>>> 2020year = 2020 # 오류: 변수 이름이 수로 시작
SyntaxError: invalid syntax
>>> sale@ = 20 # 오류: 문자 @ 사용 불가
SyntaxError: invalid syntax
>>> #오류: 공백문자 사용불가
>>> sale price = 16000
SyntaxError: invalid syntax
>>> import = 30 #오류: 키워드 사용 불가
SyntaxError: invalid syntax
...
```

그림 2-7

- 키워드 : 프로그래밍 언어가 이미 정해놓은 단어
- 다양한 대입 연산자

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

- 예제) 섭씨온도를 화씨 온도로 변환 (그림 2-8)

```
celsius = 37
fahrenheit = celsius * 9/5 + 32 # 화씨로 변환
print('섭씨: ', celsius, ', ', '화씨: ', fahrenheit)
celsius += 3 # 3도 증가
fahrenheit = celsius * 9/5 + 32 # 화씨로 변환
print('섭씨: ', celsius, ', ', '화씨: ', fahrenheit)
```

그림 2-8

섭씨: 37 , 화씨: 98.6
섭씨: 40 , 화씨: 104.0

그림 2-8 실행결과

- 예제) 세일을 적용해 가격 계산 및 출력 (그림 2-9)

```
price = 56000
sale = 20
salePrice = price * ( 100 - sale ) / 100
print(salePrice)
```

그림 2-9

44800.0

그림 2-9 실행결과

- 예제) 지구와 달까지의 거리를 만 단위로 출력, 함수 divmod() 사용 (그림 2-10)

```
distance = 384400
unit = 10000
manUnit, remainder = divmod(distance, unit)
print('지구에서 달까지의 거리: ', manUnit, '만', remainder, '킬로미터')
```

그림 2-10

지구에서 달까지의 거리: 38 만 4400 킬로미터

그림 2-10 실행결과

III 다양한 자료 : 문자열과 수

- 표준 입력 : 프로그램 과정에서 셸이나 콘솔에서 사용자의 입력을 받아 처리하는 방식. input() 함수 사용
- 예제) 학교와 이름을 입력받아 출력 (그림 2-11)

```
univ = input('대학은? ')
name = input('이름은? ')
print('대학: ', univ, '이름: ', name)
```

그림 2-11

```
대학은? 모명준 ← 내용 입력 후 Enter키 입력
이름은? 동양미래대
대학: 동양미래대 이름: 모명준
```

그림 2-11 실행결과

- 자료변환 함수 : str(), int(), float()
- 예제) 함수 int()와 float()에서의 변환 주의 (그림 2-12)

```
>>> int('python') # 일반문자
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    int('python') # 일반문자
ValueError: invalid literal for int() with base 10: 'python'
>>> int('6400i') # 정수에 문자 i가 포함
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    int('6400i') # 정수에 문자 i가 포함
ValueError: invalid literal for int() with base 10: '6400i'
>>> float('3.141592f') # 실수에 문자 f가 포함
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    float('3.141592f') # 실수에 문자 f가 포함
ValueError: could not convert string to float: '3.141592f'
>>> int('2.71828') # 정수가 아닌 실수 형태
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    int('2.71828') # 정수가 아닌 실수 형태
ValueError: invalid literal for int() with base 10: '2.71828'
>>> int('0b11') # 10진수 형태만 가능
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    int('0b11') # 10진수 형태만 가능
ValueError: invalid literal for int() with base 10: '0b11'
>>> int(float('2.71828')) # 정실적으로 2 반환 = int(2.71828)을 수행
2
```

그림 2-12

- 예제) 나이를 입력받아 만 나이 출력 (그림 2-13)

```
>>> age = input('나이는? ')
나이는? 20
>>> print('실제 나이는', int(age) + 1) ← 문자열 20을 int자료형으로 변환
실제 나이는 21
```

그림 2-13

- 예제) 행성 지구 반지름을 입력받아 지구 둘레 길이 구하기 (그림 2-14)

```
planet = input('원하는 행성은? ')
strRadius = input(planet + ' 반지름은? ')
radius = int(strRadius)

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

그림 2-14

```
원하는 행성은? 지구
지구 반지름은? 6400
지구 반지름: 6400
지구 둘레길: 40192.0
```

그림 2-14 실행결과

- 진수 상수 표현 : 16진수 (0x), 8진수 (0o), 2진수 (0b).
- 진수 변환 함수 : 16진수 = hex(), 8진수 = oct(), 2진수 = bin().
- 예제) 10진수 정수를 입력받아 2진수, 8진수, 10진수, 16진수 출력 (그림 2-15)

```
data = int(input('정수입력 >> '))
print('2진수: ', bin(data)) # 2진수로 출력
print('8진수: ', oct(data)) # 8진수로 출력
print('10진수: ', data) # 10진수로 출력
print('16진수: ', hex(data)) # 16진수로 출력
```

그림 2-15

```
정수입력 >> 22
2진수: 0b10110
8진수: 0o26
10진수: 22
16진수: 0x16
```

그림 2-15 실행결과

```
정수입력 >> 0xff ← int()함수는 10진수만 변환 가능
Traceback (most recent call last):
  File "C:\Users\jsuik\AppData\Local\Programs\Python\Python3
8-32\20161870_Joon.py", line 1, in <module>
    data = int(input('정수입력 >> '))
ValueError: invalid literal for int() with base 10: '0xff'
그림 2-15 오류코드
```

- 예제) 16진수 정수를 입력받아 2진수, 8진수, 10진수, 16진수 출력

```
invar = input('16진수 정수 입력 >> ')
data = int(invar, 16) #입력 문자열을 16진수로 인지해 변환
# 여러 진수로 출력
print('2진수: ', bin(data))
print('8진수: ', oct(data))
print('10진수: ', data)
print('16진수: ', hex(data))
```

그림 2-16

```
16진수 정수 입력 >> 0x1c
2진수: 0b111100
8진수: 0o34
10진수: 28
16진수: 0x1c
```

그림 2-16 실행결과(1)

```
16진수 정수 입력 >> 32
2진수: 0b110010
8진수: 0o62
10진수: 50
16진수: 0x32
↳ 32도 16진수 32로 인지
```

그림 2-16 실행결과(2)

- Chapter 2 실습 문제

1. 킬로미터 단위로 거리를 입력받아 마일(mile) 단위로 변환해 출력하는 프로그램을 작성하시오. (1마일은 1.61km)

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
length = input('차의 속도를 입력(km) ')
mile = (float(length)) * 100 / 161
print(int(length), '(km)은 ', mile, '마일(miles)이다.')
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
차의 속도를 입력(km) 130
130 (km)은 80.74534161490563 마일(miles)이다.
>>>
```

2. 다음 조건을 참고해 지구를 원이라고 보고 원의 둘레를 계산해 실제와의 차이를 알아보는 프로그램을 작성하시오.

- 지구의 둘레는 4만 120km, 반지름은 6378.1km
- 원둘레 공식: $2 \times 3.141592 \times \text{반지름}$

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
r = 6378.1;
length = 2 * 3.141592 * r;
earth = 40120;

print('알려진 지구 둘레: ', length);
print('지구와 같은 둘레: ', length);
print('차이: ', (earth - length), '(km)');
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
알려진 지구 둘레: 40120
지구와 같은 둘레: 40074.77587040001
차이: 45.22412959999201 (km)
>>>
```

3. 다음 조건을 참고해 섭씨온도를 입력받아 화씨 온도로 변환하는 프로그램을 작성하시오.

- 다음 식을 사용해 정확한 계산을 수행하고
- 약식 계산은 $f = c \times 2 + 30$

$$F = C \cdot \frac{9}{5} + 32 \quad C = (F - 32)$$

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
celsius = int(input('온도 입력 >> '))
f = float(celsius) * 9/5 + 32
if (f - 32) * 5/9:
    lc = celsius * 2 + 30
    lc = (lc - 30) * 1/2
print('정확 계산: 섭씨: ', c, '화씨: ', f)
print('약식 계산: 섭씨: ', lc, '화씨: ', lf)
print('차이: ', f - float(lf))
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
온도 입력 >> 38
정확 계산: 섭씨: 38.0 화씨: 100.4
약식 계산: 섭씨: 38.0 화씨: 106
차이: -5.599999999999994
>>>
```

4. 두 정수를 입력받은 후 산술연산 /, %, //, ** 4개를 수행해 결과를 출력하는 프로그램을 작성하시오.

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
n1 = int(input('Enter First number: '))
n2 = int(input('Enter Second number: '))

print(n1, '/', n2, '=>', n1 / n2)
print(n1, '%', n2, '=>', n1 % n2)
print(n1, '//', n2, '=>', n1 // n2)
print(n1, '**', n2, '=>', n1 ** n2)
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
Enter First number: 12
Enter Second number: 5
12 / 5 => 2.4
12 % 5 => 2
12 // 5 => 2
12 ** 5 => 248832
>>>
```

5. 다음에서 설명하는 함수 float.fromhex(str)를 이해하고 두 16진수 실수를 입력받아 사칙연산을 수행하는 프로그램을 작성하시오.

- 실수 float에 속한 메소드 float.fromhex(str)는 16진수 형태의 문자열 str을 실수로 변환하는 함수
- 즉 float.from('f')는 15, float.from('e.1')은 14.0625

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
n1 = float.fromhex(input('첫번째 16진수 실수 입력: '))
n2 = float.fromhex(input('두번째 16진수 실수 입력: '))

print('실수1: ', n1, '실수2: ', n2)
print('합: ', n1 + n2)
print('차: ', n1 - n2)
print('곱하기: ', n1 * n2)
print('나누기: ', n1 / n2)
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
첫번째 16진수 실수 입력: f
두번째 16진수 실수 입력: e.1
실수1: 15.0 실수2: 14.0625
합: 29.0625
차: 0.9375
곱하기: 210.9375
나누기: 1.0656666666666667
>>>
```

6. 네 자릿수 정수를 입력받은 후 그 정수를 역순으로 출력하는 프로그램을 작성하시오.

20161870_Joon.py - C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
File Edit Format Run Options Window Help

```
num = int(input('네 자릿수 정수 입력: '))

num1 = num % 10;
num2 = (num % 100) // 10;
num3 = (num % 1000) // 100;
num4 = num // 1000

newnum = (num1 * 1000) + (num2 * 100) + (num3 * 10) + num4

print(newnum)
```

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.
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-32\Python.exe
네 자릿수 정수 입력: 5432
2345
>>>
```

Chapter 3. 일상에서 활용되는 문자열과 논리 연산

I 문자열 다루기

- 문자열은 클래스 str의 객체이다.
- len() 함수 : 문자열의 길이 참조
- 문자열의 문자 참조 (그림 3-1) : 'python'[0] = 'p', 'python'[5] = n 문자열을 구성하는 문자는 0부터 시작되는 첨자를 사용한다. -1부터 시작하는 역순의 첨자도 존재한다. 첨자가 유효 범위를 벗어나면 IndexError가 발생한다.

```
str = 'Hello python!'
n = len(str)
print('문자열', str, '길이', n)
print('첫 문자', str[0], str[-n])
print('가운데 문자', str[n//2], str[-n//2])
print('마지막 문자', str[n-1], str[-1])
```

그림 3-1

문자열 Hello python! 길이 13
첫 문자 H H
가운데 문자 p p
마지막 문자 ! !

그림 3-1 실행결과

- 문자열 슬라이싱** : 문자열의 부분 문자열 참조 방식. 0과 양수를 이용한 방식으로는 str[start:end]로 사용하며 start 첨자에서 end-1첨자까지의 문자열 반환. 음수를 이용한 방식도 str[start:end]이다. 만약 start와 end로 구성하는 문자열이 없다면 빈 문자열을 반환한다. start와 end를 비우면 처음부터 끝까지를 의미한다. (그림 3-2 참조)

```
>>> str = 'Monty Python'
>>> len(str)
12
>>> str[0:5], str[6:], str[6:12]
('Monty', 'Python', 'Python')
>>> str[-12:-7], str[-6:], str[-6:0] ← -6첨자부터 0첨자까지의 구성문자가 없으므로 빈 문자열 반환
('Monty', 'Python', '')
```

그림 3-2

- 문자열 슬라이싱에서 step으로 문자사이 간격 조정 가능 : 1 = 모든 문자열, 2 = 한 문자씩 건너뛰기, 3 = 두 문자씩 건너뛰기, -1 = 역순으로 출력, -2 = 역순으로 한 문자씩 건너뛰기 (그림 3-3)

```
str = '일요일 기러기'
print(str[:3], str[4:]) # 양수 이용
print(str[-4], str[-3:]) # 음수 이용
print(str[:], str[:]) # 모든 문자열 참조
print(str[:2]) # 한 문자씩 건너뛰기
print(str[:3]) # 두 문자씩 건너뛰기
print(str[::-2]) # 역순으로 한 문자씩 건너뛰기
print(str[::-1]) # 역순으로 출력
```

그림 3-3

일요일 기러기
일요일 기러기
일요일 기러기 일요일 기러기 일요일 기러기
일요일 기러기
기러기 일요일
기러기 일요일

그림 3-3 실행결과

- 문자함수 ord(), chr() : ord() = 문자를 유니코드 번호로 변환, chr() = 유니코드 번호를 문자로 변환
- 이스케이프 시퀀스 문자 : 역슬래시(\)로 시작하는 조합으로 표현하는 문자.
- 문자함수 max(), min() : max() = 인자의 최댓값 반환, min() = 인자의 최솟값 반환.

II 문자열 관련 메소드

- replace() : 문자열을 바꿔서 반환. str.replace(a,b,c) = 문자열 str에서 a가 나타나는 모든 부분을 b로 c번 바꾼다.
- 예제) 실수의 모든 자릿수 더하기 (그림 3-4)

```
value = input('실수(세 자리, 두 자리로 345.78 처럼)를 하나 입력하세요. >> ')
num = value.replace('.', '')
sum = 0
sum += int(num[0])
sum += int(num[1])
sum += int(num[2])
sum += int(num[3])
sum += int(num[4])
print('입력값:', value)
print('모든 자릿수 합:', sum)
```

그림 3-4

실수(세 자리, 두 자리로 345.78 처럼)를 하나 입력하세요. >> 345.67
입력값: 345.67
모든 자릿수 합: 25

그림 3-4 실행결과

- count() : 문자열에서 문자나 부분 문자열의 출현 횟수 반환
- join() : 문자와 문자 사이에 원하는 문자열 삽입
- find(), index() : 문자열을 찾아 처음 나타나는 위치의 첨자를 반환. index()는 찾는 문자열이 없을 경우 ValueError를 발생, find()는 오류를 발생시키지 않고 -1을 반환. 역순으로 찾는 rfind()와 rindex()도 존재한다.
- 예제) 문자열에 두 단어의 순서 교환과 역순 출력 (그림 3-5)

```
str = input('2개의 단어를 빈 공간으로 구분해 입력하세요. >> ')
pos = str.find(' ')
preWord = str[:pos]
postWord = str[pos+1:]
print(preWord, postWord)
print(preWord[::-1], postWord[::-1])
```

2개의 단어를 빈 공간으로 구분해 입력하세요. >> 사과 복숭아
사과 복숭아
과사 마송복

그림 3-5 실행결과

- split() : 문자열에서 공백을 기준으로 문자열을 나눠주는 메소드. str.split(',')처럼 괄호 안에 인자가 있을 경우 인자를 기준으로 문자열을 나눈다. split을 활용하여 표준 입력에서 여러 값을 받을 수 있다.
- 예제) 4개의 수를 입력받아 합, 평균값, 최댓값, 최솟값을 출력 (그림 3-6)

```
m, n, x, y = input('4개의 수 입력 >> ').split()
a, b, c, d = float(m), float(n), float(x), float(y)
print('입력값: ', a, b, c, d)
sum = a + b + c + d
print('합: ', sum, '평균: ', sum / 4)
print('최대: ', max(a, b, c, d), '최소: ', min(a, b, c, d))
```

4개의 수 입력 >> 3.7 5.8 9.0 2.5
입력값: 3.7 5.8 9.0 2.5
합: 21.0 평균: 5.25
최대: 9.0 최소: 2.5

그림 3-6 실행결과

- 문자열 변환 메소드 : upper() = 대문자로 변환, lower() = 소문자로 변환, capitalize() = 첫 문자만 대문자로 변환, title() = 단어마다 첫 문자를 대문자로 변환, swapcase() = 소문자와 대문자를 서로 변환해 반환 등이 있다.
- center() : 문자열의 폭을 지정하고 중앙에 문자열 배치, 좌우 나머지는 문자로 채운다. ex) '문자열'.center(30, '*')
- ljust(), rjust() : center()와 같지만 문자열을 왼쪽 정렬, 오른쪽 정렬 시킨다.
- strip() : 문자열에서 맨 앞뒤의 공백문자를 제거한다. 괄호 안에 문자들을 넣으면 문자들에 속한 모든 문자를 제거.
- zfill() : 문자열의 지정한 폭 앞 빈부분에 0을 채운다. ex) '234'.zfill(5) = '00234'
- format() : 문자열의 중간중간에 변수나 상수를 함께 출력. 문자열 '{ } + { } = { }' 내부에서 사용할 인자들을 format(3, 4, 3 + 4)와 같이 넣어준다. 중괄호에 0부터 시작하는 숫자를 넣으면 인자가 들어갈 순서를 바꿀 수 있다.
- C언어의 포매팅 스타일 % : C언어에서 사용하는 형식지정자도 파이썬에서 지원한다.

| | | | | | |
|----|------|----|------|----|-----|
| %d | 10진수 | %x | 16진수 | %o | 8진수 |
| %f | 실수 | %c | 문자 | %s | 문자열 |

III 논리 자료와 다양한 연산

- bool과 bool() : bool클래스는 참과 거짓을 의미하는 True와 False를 가지고 있다. 정수의 0, 실수의 0.0, 빈 문자열 '' 등은 False, 그 이외의 값은 True를 반환하는 bool().
- 논리곱 and, 논리합 or : and = 두 항이 모두 참이어야 True, or = 두 항이 모두 거짓이어야 False, 하나라도 참이라면 True.
- 배타적 논리합 연산자 ^, not : ^ = 두 항이 다르면 True, 같으면 False. not = 뒤에 위치한 논리 값을 바꿈.
- 논리 연산 우선순위 : not → and → or 순서
- 관계 연산자 : 수나 문자열 등의 크기 비교에 사용되는 연산자.

| 연산자 | 연산 사용 | 의미 | 문자열 관계 연산 |
|-----|--------|---------|-------------------|
| > | a > b | 크다. | 사전 순서에서 뒤에 |
| >= | a >= b | 크거나 같다. | 사전 순서에서 뒤에 있거나 동일 |
| < | a < b | 작다. | 사전 순서에서 앞에 |
| <= | a <= b | 작거나 같다. | 사전 순서에서 앞에 있거나 동일 |
| == | a == b | 같다. | 사전 순서에서 동일 |
| != | a != b | 다르다. | 사전 순서와 다름 |

• 예제) 문자열과 실수의 관계 연산 (그림 3-7)

```
>>> print('cat' > 'dog')
False
>>> print('tiger' < 'lion')
False
>>> print(3.5 <= 3.56)
True
>>> speed = 60
>>> print(50 > speed)
False
>>> print(80 < speed)
False
>>> print(50 < speed and speed <=80)
True
>>> print(50 < speed <= 80)
True
>>> print(True > False)
True
>>>
```

그림 3-7

• 예제) 키와 몸무게로 비만도 지수 BMI 판정 (그림 3-8)

```
h, w = input('당신의 키(cm)와 몸무게(kg)는? >> ').split()
height = float(h)
weight = float(w)
bmi = weight / (height/100)**2
print('키:%5.1f(cm), 몸무게:%5.1f(kg), BMI:%5.1f' % (height, weight, bmi))
print('{} {}'.format('고도 비만', 40 <= bmi))
print('{} {}'.format('중등도 비만', 35 <= bmi < 40))
print('{} {}'.format('비만', 30 <= bmi < 35))
print('{} {}'.format('과체중', 25 <= bmi < 30))
print('{} {}'.format('정상', 18.5 <= bmi < 25))
print('{} {}'.format('저체중', bmi < 18.5))
```

그림 3-8

```
당신의 키(cm)와 몸무게(kg)는? >> 171.2 67.5
키 171.2(cm), 몸무게: 67.5(kg), BMI: 23.0
고도 비만 False
중등도 비만 False
비만 False
과체중 False
정상 True
저체중 False
```

그림 3-8 실행결과

• 예제) 전기 사용량의 기본 요금 계산 (그림 3-9)

```
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 : %d' % (usage, base))
```

그림 3-9

```
가정의 전기 사용량(kWh)은? >> 180
전기 사용량(kw): 180, 기본 요금 : 730
```

그림 3-9 실행결과(1)

```
가정의 전기 사용량(kWh)은? >> 450
전기 사용량(kw): 450, 기본 요금 : 6060
```

그림 3-9 실행결과(2)

• 멤버십 연산 in : x in s = 문자열 s에 부분 문자열 x가 있는지 검사, 있으면 True, 없으면 False

• 예제) 멤버십 검사 in으로 배운 파이썬 키워드 검사 (그림3-10)

```
inkey = input('배운 파이썬 키워드를 입력하세요 >> ')
# inkey = 'key'
keywords = "False", "True", "and", "in", "is", "not", "or"
print('입력 단어 {}, 키워드인가? {}'.format(inkey, inkey in keywords))
```

그림 3-10

```
배운 파이썬 키워드를 입력하세요 >> and
입력 단어 and, 키워드인가? True
```

그림 3-10 실행결과(1)

```
배운 파이썬 키워드를 입력하세요 >> const
입력 단어 const, 키워드인가? False
```

그림 3-10 실행결과(2)

• 비트 논리 연산자 &, |, ^ : 정수로 저장된 메모리에서 비트와 비트의 연산

| m | n | 논리곱 | 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 |

| 연산식 | 10진수 | 2진수 표현 | | | | | | | | 설명 |
|-------|------|--------|----|----|----|---|---|---|---|--------------------|
| | | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 | |
| a | 23 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 23의 2진수 00010111 |
| b | 57 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 57의 2진수 00111001 |
| a & b | 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 비트가 모두 1이면 1 |
| a b | 63 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 비트가 하나라도 1이면 1 |
| a ^ b | 46 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 두 비트가 다르면 1, 같으면 0 |

• 예제) 비트 연산자 &로 정수의 특정 비트 알아내기 (그림 3-11)

```

a = int(input('정수 하나를 입력하세요 >> '))
mask = 0b1111 # 0x1도 가능
print('정수 {0} 2진수로는 {0:b}'.format(a))
print('가장 오른쪽 4비트 : {0:04b}'.format(a & mask))

```

그림 3-11

```

정수 하나를 입력하세요 >> 195
정수 195 2진수로는 11000011
가장 오른쪽 4비트 : 0011 정수로는 3

```

그림 3-11 실행결과

• 예제) 비트 배타적 논리합 ^으로 ID 암호화 (그림 3-12)

```

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

```

그림 3-12

```

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

```

그림 3-12 실행결과

• 비트 이동 연산자 >>, << : 연산자의 방향인 오른쪽 또는 왼쪽으로, 비트 단위로 지정된 횟수만큼 이동.

• 예제) 표준 입력으로 비트 이동 연산 << 계산 (그림 3-13)

```

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

```

그림 3-13

```

이동 연산 num << cnt를 수행할 정수 num 입력 ? 25
이동 연산 num << cnt를 수행할 정수 cnt 입력 ? 3
0000000000000000000000000000000011001 11001 :num
0000000000000000000000000000000011001000 11001000 :25 << 3
0000000000000000000000000000000011001000 11001000 :25 * 2**3

```

그림 3-13 실행결과

• 연산자 우선순위 : 지수 → 단항 → 산술 → 비트 → 관계 → 논리

| 순위 | 연산자 | 설명 | 부류 |
|----|-------------------------------------|----------------|-------|
| 1 | ** | 지수승 | 지수 연산 |
| 2 | ~ | 비트 보수 | 단항연산 |
| 3 | +x, -x | 부호 | |
| 4 | * / // % | 곱, 나누기, 몫, 나머지 | 산술 연산 |
| 5 | + - | 더하기, 빼기 | |
| 6 | >> << | 비트 이동 | 비트 연산 |
| 7 | & | 비트 and | |
| 8 | ^ | 비트 xor | |
| 9 | | 비트 or | 관계 소속 |
| 10 | < <= > >= == != in not in is is not | 관계, 소속, id 비교 | |
| 11 | = %= /= //=- += *= **= | 대입, 여러 대입 | |
| 12 | not | 논리 not | 논리 연산 |
| 13 | and | 논리 and | |
| 14 | or | 논리 or | |

• 예제) 표준 입력의 연도가 윤년인지 검사해 출력 (그림 3-14)

```

year = int(input('윤년을 검사할 연도 입력 >> '))
print('입력한 연도: %d' % year)
cond1 = year % 4 == 0
cond2 = year % 100 != 0
cond3 = year % 400 == 0
result1 = cond1 and cond2 or cond3
print('개별 검사 {} and {} or {}: {}'.format(cond1, cond2, cond3, result1))
result2 = year % 4 == 0 and year % 100 != 0 or year % 400 == 0
print('통합 검사: %s' % result2)

```

그림 3-14

```

윤년을 검사할 연도 입력 >> 2020
입력한 연도: 2020
개별 검사 True and True or False: True
통합 검사: True

```

그림 3-14 실행결과(1)

```

윤년을 검사할 연도 입력 >> 2021
입력한 연도: 2021
개별 검사 False and True or False: False
통합 검사: False

```

그림 3-14 실행결과(2)

- 19 -

Chapter 4. 일상생활과 비유되는 조건과 반복

I 조건에 따른 선택 if ... else

- **if문** : 조건에 따라 문장들을 처리해야 하는 경우 사용. 조건문의 결과가 True일 경우 문장 실행 False일 경우 실행하지 않는다. 조건문 이후에는 반드시 :이 들어가야 한다. 콜론 이후 다음 줄부터 들여쓰기가 필요하다.
- 예제) 키가 140 이상이면 놀이기구 타기 (그림 4-1)

```
height = 152 # 탑승을 체크할 키를 대입
if 140 <= height:
    print('롤러코스터 T-Express, 즐기세요!')
```

그림 4-1

롤러코스터 T-Express, 즐기세요!

그림 4-1 실행결과

- 예제) 평균 평점 3.8 이상이면 장학금 지급 대상자 (그림 4-2)

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

그림 4-2

1학기 평균 평점은? 3.89
축하합니다! 장학금 지급 대상자입니다.
당신의 1학기 평균 평점은 3.89입니다.

그림 4-2 실행결과(1)

1학기 평균 평점은? 3.5
당신의 1학기 평균 평점은 3.50입니다.

그림 4-2 실행결과(2)

- **if ... else문** : else: 블록은 조건문이 False일 때 실행. if문과 열을 맞춰서 입력해야 한다.
- 예제) 영화 조조 할인 판정 (그림 4-3)

```
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 안된다.' % (hour, mnt))
```

그림 4-3

지금 시각: 17시 53분, 조조 할인 안된다.

그림 4-3 실행결과(1)

지금 시각: 8시 15분, 조조 할인 된다.

그림 4-3 실행결과(2)

- 예제) 정수의 홀수와 짝수 판정 (그림4-4)

```
n = int(input('정수 입력 >> '))
if n%2 == 0:
    print('%d는 짝수다.' % n)
else:
    print('%d는 홀수다.' % n)
```

그림 4-4

정수 입력 >> 10
10는 짝수다.

그림 4-4 실행결과(1)

정수 입력 >> 11
11는 홀수다.

그림 4-4 실행결과(2)

- **다중 택일 결정 구조 if ... elif** : C언어나 Java에서의 else if와 같은 구조. 조건의 여러 경로 중 하나를 선택. 조건1이 True이면 문장1을 실행 후 종료한다. 조건1이 False여야 조건2로 진행한다.
- 예제) 미세먼지 예보 (그림4-5)

```
PM = float(input('미세먼지(100마이크로그램)의 농도는? '))
if 151 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM, '매우 나쁨'))
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'))
```

그림 4-5

미세먼지(100마이크로그램)의 농도는? 140
미세먼지 농도: 140.00, 등급: 나쁨

그림 4-5 실행결과(1)

미세먼지(100마이크로그램)의 농도는? 30
미세먼지 농도: 30.00, 등급: 좋음

그림 4-5 실행결과(2)

- 예제) 커피와 주스 선택 이후 메뉴 선택 (그림 4-6)

```
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')
```

그림 4-6

원하는 음료는? 1. 커피 2. 주스 1
번호 선택? 1. 아메리카노 2. 카페라떼 3. 카푸치노 2
2. 카페라떼 선택

그림 4-6 실행결과(1)

원하는 음료는? 1. 커피 2. 주스 2
번호 선택? 1. 키위주스 2. 토마토주스 3. 오렌지주스 3
3. 오렌지주스 선택

그림 4-6 실행결과(2)

II 반복을 제어하는 for문과 while문

- 반복 for문 : 여러 자료 값이 순서대로 구성된 시퀀스에서 자료 값의 개수만큼 반복적인 작업을 수행.

for 변수 in <시퀀스>:

반복 몸체인 문장들

- 예제) 수의 나열에서 합과 평균 구하기 (그림 4-7)

```
sum = 0
for i in 1.1, 2.5, 3.6, 4.2, 5.4:
    sum += i
    print(i, sum)
else:
    print('합: %.2f, 평균: %.2f' % (sum, sum/5))
```

그림 4-7

1.1 1.1
2.5 3.6
3.6 7.2
4.2 11.4
5.4 16.8
합: 16.80, 평균: 3.36

그림 4-7 실행결과

- 내장함수 range() : range(5)는 정수 0에서 4까지 5개의 항목인 정수로 구성.

- 예제) 3회에 걸쳐 커피 주문받기 (그림 4-8)

```
for i in range(3):
    coffee = input("주문하세요! [아메리카노] [카페라떼] [카푸치노] >> ")
    if coffee == '아메리카노':
        print('%s 주문' % coffee)
    elif coffee == '카페라떼':
        print('%s 주문' % coffee)
    elif coffee == '카푸치노':
        print('%s 주문' % coffee)
    else:
        print('모르겠어요.')
else:
    print('주문을 마치겠습니다.')
```

그림 4-8

주문하세요! [아메리카노] [카페라떼] [카푸치노] >> 아메리카노
아메리카노 주문
주문하세요! [아메리카노] [카페라떼] [카푸치노] >> 카페라떼
카페라떼 주문
주문하세요! [아메리카노] [카페라떼] [카푸치노] >> 카푸치노
카푸치노 주문
주문을 마치겠습니다.

그림 4-8 실행결과

- 예제) 지정된 최소 한 자릿수가 포함된 두 자리 정수 찾기 (그림 4-9)

```
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 = ' ')
```

그림 4-9

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

그림 4-9 실행결과

- 예제) 어린이를 위한 놀이기구 탑승 검사 (그림 4-10)

```
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 못들어갑니다.')
    if cnt == MAXNUM:
        more = False
else:
    print('%d명 모두 찾습니다. 다음번에 이용하세요.' % cnt)
```

그림 4-10

키는 ? 125
들어가요. 1명
키는 ? 129
들어가요. 2명
키는 ? 131
들어가요. 3명
키는 ? 110
커서 못들어갑니다.
키는 ? 105
들어가요. 4명
4명 모두 찾습니다. 다음번에 이용하세요.

그림 4-10 실행결과

- 예제) 번호로 버거 메뉴 주문받기 (그림 4-11)

```
menu = '''버거 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>

more = True
while more:
    order = int(input(menu))
    if order == 1:
        print('주문 % ' 올인원팩')
    elif order == 2:
        print('주문 % ' 두게더팩')
    elif order == 3:
        print('주문 % ' 트리오팩')
    elif order == 4:
        print('주문 % ' 패밀리팩')
    elif order == 0:
        print('주문 종료'.center(20, '*'))
        more = False
    else:
        print('모르겠어요.')
else:
    print('주문을 마치겠습니다.')
```

그림 4-11

```
버거, 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>
버거, 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>
두게더팩, 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>
트리오팩, 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>
패밀리팩, 콤보 번호로 주문하세요!
0. 주문종료
1. 올인원팩
2. 두게더팩
3. 트리오팩
4. 패밀리팩
>>
주문종료
***** 주문을 마치겠습니다. *****
```

그림 4-11 실행결과

- 예제) 전형적인 구구단 출력 (그림 4-12)

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

그림 4-12

```
2 * 1 = 2 2 * 2 = 4 2 * 3 = 6 2 * 4 = 8 2 * 5 = 10 2 * 6 = 12 2 * 7 = 14 2 * 8 = 16 2 * 9 = 18
3 * 1 = 3 3 * 2 = 6 3 * 3 = 9 3 * 4 = 12 3 * 5 = 15 3 * 6 = 18 3 * 7 = 21 3 * 8 = 24 3 * 9 = 27
4 * 1 = 4 4 * 2 = 8 4 * 3 = 12 4 * 4 = 16 4 * 5 = 20 4 * 6 = 24 4 * 7 = 28 4 * 8 = 32 4 * 9 = 36
5 * 1 = 5 5 * 2 = 10 5 * 3 = 15 5 * 4 = 20 5 * 5 = 25 5 * 6 = 30 5 * 7 = 35 5 * 8 = 40 5 * 9 = 45
6 * 1 = 6 6 * 2 = 12 6 * 3 = 18 6 * 4 = 24 6 * 5 = 30 6 * 6 = 36 6 * 7 = 42 6 * 8 = 48 6 * 9 = 54
7 * 1 = 7 7 * 2 = 14 7 * 3 = 21 7 * 4 = 28 7 * 5 = 35 7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63
8 * 1 = 8 8 * 2 = 16 8 * 3 = 24 8 * 4 = 32 8 * 5 = 40 8 * 6 = 48 8 * 7 = 56 8 * 8 = 64 8 * 9 = 72
9 * 1 = 9 9 * 2 = 18 9 * 3 = 27 9 * 4 = 36 9 * 5 = 45 9 * 6 = 54 9 * 7 = 63 9 * 8 = 72 9 * 9 = 81
```

그림 4-12 실행결과

III 임의의 수인 난수와 반복을 제어하는 break문, continue문

- 난수 발생 함수 random.randint(시작, 끝) : 정수 시작과 끝 수 사이에서 임의의 정수 발생. 시작과 끝 모두 포함.
- 예제) 1에서 45까지의 6개 수를 맞추는 로또 (그림 4-13)

```
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 맞춤')
```

그림 4-13

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

----- 내 번호 확인 -----
40 X 15 X 24 X 44 X 35 O 35 O
2 개 맞춤
```

그림 4-13 실행결과

- 반복 제어문 break : break는 특정한 조건에서 즉시 반복을 종료할 경우 사용.
- 예제) 1을 입력하면 계속하고 0을 입력하면 반복 종료 (그림 4-14)

```
while True:
    menu = input('[0]종료 [1]계속 ? ')
    if menu == '0':
        break
print('종료')
```

그림 4-14

```
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 1
[0]종료 [1]계속 ? 0
종료
```

그림 4-14 실행결과

- 예제) 0에서 9까지의 수 중에서 7이 나오면 반복 종료 (그림 4-15)

```
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.')
그림 4-15
```

```
4, 7 나올 때까지 계속!
5, 7 나올 때까지 계속!
5, 7 나올 때까지 계속!
드디어 7, 종료!
```

그림 4-15 실행결과

- 반복 제어문 **continue** : continue는 이후의 반복 몸체를 실행하지 않고 다음 반복을 위해 조건문 수행.

- 예제) 월, 화, 수 중 영어 철자 하나 검사 (그림 4-16)

```
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, '*'))
그림 4-16
```

```
월, 화, 수 중 하나 영어 단어 입력 >> tuesday
잘못 입력했어요!
월, 화, 수 중 하나 영어 단어 입력 >> tuesday
입력: tuesday, 철자가 맞습니다.
***** 종료 *****
```

그림 4-16 실행결과

- Chapter 4 실습 문제

1. 근로 시급이 12,000원이고, 일주일에 40시간 이상 근무하면 시급의 1.5배의 급여를 준다고 한다. 일주일 근로 시간을 35~50 시간 사이에서 임의의 난수로 정하고, 주급을 계산해 출력하는 프로그램을 작성하시오.

- 반복으로 5회의 근로 시작에 대해 출력

```
import random
for i in range(5):
    n = random.randint(35, 50)
    if n >= 40:
        print('근로시간 %d시간, 주급 %6d' % (n, (40*12000) + (n-40) * (12000*1.5)))
    else:
        print('근로시간 %d시간, 주급 %6d' % (n, n*12000))
```

```
= RESTART: C:\Users\#suitk\AppData\
n.py
근로시간 49시간, 주급 642000
근로시간 38시간, 주급 456000
근로시간 49시간, 주급 642000
근로시간 49시간, 주급 642000
근로시간 36시간, 주급 432000
```

2. 다음을 참고해 인간의 비만도를 측정하는 체질량 지수를 계산해 판정 결과를 출력하는 프로그램을 작성하시오.

- h, w = input('당신의 키(cm)와 몸무게(kg)는? >> ').split()
- 위 코드를 사용해 키와 몸무게 입력
- 키가 tcm, 체중이 wkg일 때, BMI 계산식 $w/(t/100)^2$
- 판정표

| 기준 | 판정 | 관계 연산 표현(BMI) |
|-------------|--------|------------------|
| 40 이상 | 고도 비만 | 40 <= bmi |
| 35 ~ 39.9 | 중등도 비만 | 35 <= bmi < 40 |
| 30 ~ 34.9 | 비만 | 30 <= bmi < 35 |
| 25 ~ 29.9 | 과체중 | 25 <= bmi < 30 |
| 18.5 ~ 24.9 | 정상 | 18.5 <= bmi < 25 |
| 18.5 미만 | 저체중 | bmi < 18.5 |

```
h, w = input('당신의 키(cm)와 몸무게(kg)는? ').split()
H, W = float(h), float(w)
bmi = W / (H/100)**2
print('키: %.1f(cm), 몸무게: %.1f(kg)' % (H, W))
if 40 <= bmi:
    print('BMI: %.1f 고도 비만' % bmi)
elif 35 <= bmi < 40:
    print('BMI: %.1f 중등도 비만' % bmi)
elif 30 <= bmi < 35:
    print('BMI: %.1f 비만' % bmi)
elif 25 <= bmi < 30:
    print('BMI: %.1f 과체중' % bmi)
elif 18.5 <= bmi < 25:
    print('BMI: %.1f 정상' % bmi)
else:
    print('BMI: %.1f 저체중' % bmi)
```

```
= RESTART: C:\Users\#suitk\AppData\Loc
n.py
당신의 키(cm)와 몸무게(kg)는? 171 72
키: 171.0(cm), 몸무게: 72.0(kg)
BMI: 24.6 정상
>>> |
```


3. 다음을 참고해 1에서 99까지 정수인 난수를 2개 생성하고 곱하기의 결과를 출력하는 프로그램을 작성하시오.

- 반복 while True로 계속 난수를 발생해 출력
- 표준 입력으로 y 또는 n을 입력받아 n이면 반복을 종료하고 프로그램을 마침

```
import random
while True:
    n1 = random.randint(1, 99)
    n2 = random.randint(1, 99)
    print('%2d * %2d = %4d' % (n1, n2, n1 * n2))
    more = input('계속 y / n ? ')
    if more == 'n':
        break
print('프로그램을 종료합니다.')
```

```
= RESTART: C:\Users\su
n.py
83 * 88 = 7304
계속 y / n ? y
5 * 4 = 20
계속 y / n ? y
9 * 88 = 792
계속 y / n ? n
프로그램을 종료합니다.
>>> |
```

4. 정수 1개를 표준 입력으로 받아 소수인지를 판별하는 프로그램을 작성하시오.

- 2 이상의 정수를 입력해 입력한 정수의 판별

```
num = int(input('소수(prime number)인지를 판별한 2 이상의 정수 입력 >> '))
cnt = 0
for i in range(2, num):
    if num % i == 0:
        cnt += 1

if cnt > 0:
    print('정수 %d는 소수가 아니다' % num)
else:
    print('정수 %d는 소수이다' % num)
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29,
tel)) on win32
Type "help", "copyright", "credits" or "license()" for m
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\
n.py
소수(prime number)인지를 판별한 2 이상의 정수 입력 >> 8
정수 8는 소수가 아니다
>>>
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\
n.py
소수(prime number)인지를 판별한 2 이상의 정수 입력 >> 13
정수 13는 소수이다
```

5. 난수로 발생시킨 1에서 100 사이의 첫 번째 피연산자와 사용자가 표준 입력한 산술 연산자 문자 그리고 표준 입력한 두 번째 피연산자를 계산해 출력하는 프로그램을 작성하시오.

- 반복 While True:로 계속 산술 연산 결과를 출력
- 산술 연산 종류인 연산자는 표준 입력으로 '+-*/%' 중 하나를 입력
- 입력한 연산자가 위 연산자가 아니면 반복을 종료하고 프로그램을 마침

```
import random
y = {'+', '-', '*', '/', '%'}
while True:
    n1 = random.randint(1, 100)
    print('첫 값은 %d이다' % n1)
    s = input('산술연산의 종류를 입력하세요 >> ')
    if s in y:
        n2 = int(input('두번째 피연산자를 입력하세요 >> '))
        if s == '+':
            print('%d %c %d = %d' % (n1, s, n2, n1 + n2))
        elif s == '-':
            print('%d %c %d = %d' % (n1, s, n2, n1 - n2))
        elif s == '*':
            print('%d %c %d = %d' % (n1, s, n2, n1 * n2))
        elif s == '/':
            print('%d %c %d = %d' % (n1, s, n2, n1 / n2))
        elif s == '%':
            print('%d %c %d = %d' % (n1, s, n2, n1 % n2))
    else:
        break
print('종료'.center(30, '*'))
```

```
= RESTART: C:\Users\suitk\AppData\L
n.py
첫 값은 380이다
산술연산의 종류를 입력하세요 >> +
두번째 피연산자를 입력하세요 >> 12
38 + 12 = 50
첫 값은 730이다
산술연산의 종류를 입력하세요 >> -
두번째 피연산자를 입력하세요 >> 23
73 - 23 = 50
첫 값은 160이다
산술연산의 종류를 입력하세요 >> *
두번째 피연산자를 입력하세요 >> 3
16 * 3 = 48
첫 값은 90이다
산술연산의 종류를 입력하세요 >> /
두번째 피연산자를 입력하세요 >> 3
9 / 3 = 3
첫 값은 230이다
산술연산의 종류를 입력하세요 >> %
두번째 피연산자를 입력하세요 >> 5
23 % 5 = 3
첫 값은 270이다
산술연산의 종류를 입력하세요 >> &
*****종료*****
```

Chapter 5. 항목의 나열인 리스트와 튜플

I 여러 자료 값을 편리하게 처리하는 리스트

- 리스트 : 항목의 나열인 시퀀스. [항목 1, 항목 2, 항목3, ...] 모든 자료형이 항목이 될 수 있으며 중복도 가능하다.

- 예제) 다양한 커피 종류가 저장된 리스트 (그림 5-1)

```
coffee = ['에스프레소', '아메리카노', '카페라떼', '카페모카']
print(coffee)
print(type(coffee))

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

그림 5-1

```
['에스프레소', '아메리카노', '카페라떼', '카페모카']
<class 'list'>
1. 에스프레소
2. 아메리카노
3. 카페라떼
4. 카페모카
```

그림 5-1 실행결과

- 예제) 간단한 커피 메뉴 만들기 1 (그림 5-2)

```
menu = ['COFFEE', 'BEVERAGE', 'ADE']
coffee = ['에스프레소', '아메리카노', '카페라떼', '카페모카']

print('=' * 45)
for category in menu:
    print('=' * 15).format(category), end = ' '
print()
print('=' * 45)
for ckind in coffee:
    print('{:10s}'.format(ckind))
```

그림 5-2

```
=====
COFFEE      BEVERAGE      ADE
=====
에스프레소
아메리카노
카페라떼
카페모카
```

그림 5-2 실행결과

- 리스트 관련 함수 : append() = 빈 리스트에 항목 추가. len() = 리스트의 항목 수 반환.

- 예제) 리스트로 편의점에서 구입할 품목 만들기 (그림 5-3)

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

그림 5-3

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

그림 5-3 실행결과

- 리스트의 항목 참조 : 문자열 시퀀스와 동일하게 첨자로 항목을 참조.

- 예제) 프로그래밍 언어 리스트에서 첨자로 항목 참조 (그림 5-4)

```
pl = ['C', 'C++', 'Python', 'Java']
print(pl[0])
print(pl[2])
print()

for i in range(len(pl)):
    print(pl[i])
```

그림 5-4

```
C
Python
C
C++
Python
Java
```

그림 5-4

실행결과

- 예제) 가위바위보 리스트 항목 참조 (그림 5-5)

```
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))
```

그림 5-5

```
가위 바위 보
컴퓨터의 가위 바위 보 5개
가위
보
가위
보
바위
```

그림 5-5 실행결과

- 항목 수정 메소드 : count() = 값을 갖는 항목의 수를 반환. index() = 인자인 값의 항목이 위치한 첨자를 반환.

list1[1] = '문자열' list1[3] = '문자열'

- 예제) 중국집에서 음식 주문하기 (그림 5-6)

```
food = ['짜장면', '짬뽕', '우동', '울면']
print(food)
# 탕수육 주문 추가
food.append('탕수육')
print(food)
# 짬뽕을 골짜뽕으로 변경
food[1] = '골짜뽕'
print(food)
# 우동을 물만두로 주문 변경
food[food.index('우동')] = '물만두'
print(food)
```

그림 5-6

```
[['짜장면', '짬뽕', '우동', '울면'], ['짜장면', '골짜뽕', '우동', '울면'], ['짜장면', '골짜뽕', '물만두', '울면'], ['짜장면', '골짜뽕', '물만두', '탕수육']]
```

그림 5-6 실행결과

- 중첩 리스트 : animal = [['사자', '코끼리', '호랑이'], '조류', '어류']. '코끼리'를 참조하려면 animal[0][1] 사용.

- 예제) 동물의 분류를 리스트로 처리 (그림 5-7)

```
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()
```

그림 5-7

```
[['사자', '코끼리', '호랑이'], '조류', '어류']
['사자', '코끼리', '호랑이']
조류
어류
[['사자', '코끼리', '호랑이'], ['독수리', '참새', '까치'], ['갈치', '붕어', '고등어']]
사자 코끼리 호랑이
독수리 참새 까치
갈치 붕어 고등어
```

그림 5-7 실행결과

II 리스트의 부분 참조와 항목의 삽입과 삭제

- 리스트의 부분참조 슬라이싱 : 문자열 슬라이싱과 같은 구조. 3가지 첨자 리스트[시작:끝:간격]

- 예제) 한 글자의 한국 단어로 이해하는 리스트 슬라이싱 (그림 5-8)

```
wlist = ['반', '삼', '길', '죽', '꿈', '차', '억', '복', '말']
print(wlist[:]) = wlist[:]
print(wlist[:]) = wlist[:]
print(wlist[:1]) = wlist[:1]
# 오른쪽에서 왼쪽으로 슬라이싱
print(wlist[::3])
print(wlist[1::3])
print(wlist[2::3])
# 왼쪽에서 오른쪽으로 슬라이싱
print(wlist[::-2])
print(wlist[-1:-8:-3])
# 오른쪽과 왼쪽에서의 슬라이싱
print(wlist[1:-1])
print(wlist[-2:-9:-3])
```

그림 5-8

```
wlist[:] = ['반', '삼', '길', '죽', '꿈', '차', '억', '복', '말']
wlist[:1] = ['반']
wlist[-1] = '말'
wlist[::3] = ['반', '죽', '차', '억', '말']
wlist[1::3] = ['삼', '길', '꿈', '복']
wlist[2::3] = ['길', '밥']
wlist[::-2] = ['반', '죽', '차', '억', '복']
wlist[-1:-8:-3] = ['꿈', '차', '억', '복']
wlist[1:-1] = ['반', '삼', '길', '죽', '꿈', '차', '억', '복']
wlist[-2:-9:-3] = ['꿈', '차', '억', '복']
```

그림 5-8 실행결과

- 리스트의 부분 수정 : 리스트의 일부분 변경

ex) sports[0:3] = ['축구'] ('축구' 대입), sports[0:3] = '축구'('축','구' 대입)

- 리스트 메소드 insert() : 리스트.insert(첨자, 항목) 형식으로 사용하고, 빈 리스트에도 사용 가능.
- 리스트 메소드 remove(), pop() : 리스트.remove(항목) = 지정된 항목을 삭제. 리스트.pop(첨자) = 첨자의 해당되는 항목을 삭제하고 삭제한 항목을 반환.
- 문장 del에 의한 삭제 : 코드 del kpop[0] = kpop리스트의 첫 항목이 삭제. del kpop = kpop리스트 삭제
- 빈 리스트로 만드는 메소드 clear() : kpop.clear() = kpop리스트의 모든 항목 삭제. kpop리스트는 빈 리스트.
- 예제) 학용품 리스트의 항목 삽입과 삭제 (그림 5-9)

```
item = ['연필', '볼펜']
# 현재 학용품 품목 출력
print(item)
# 연필 2개와 볼펜 세 자루 삽입
item.insert(1, 2)
item.insert(3, 3)
# 맨 뒤에 지우개, 1개 삽입
item.insert(4, '지우개')
item.insert(5, 1)
# 현재 학용품 품목 출력
print(item)
# 연필 두 자루 삭제
print(item.pop(1)) # 첨자 1 항목 삭제
item.remove('연필') # 항목 연필 삭제
del item[2:] # 지우개와 수 품목 삭제
# 현재 학용품 품목 출력
print(item)
```

그림 5-9

```
[['연필', '볼펜']]
['연필', 2, '볼펜', 3, '지우개', 1]
2
['볼펜', 3]
```

그림 5-9 실행결과

- 리스트의 추가, 연결, 반복 : 리스트.extend(list) = 리스트에 인자인 list를 가장 마지막에 추가.
- food = korean + chinese = 리스트와 리스트를 연결. 리스트와 정수를 * 연산하면 지정된 정수만큼 반복.
- 리스트 항목의 순서와 정렬 : 리스트.reverse() = 리스트의 항목 순서를 반대로 뒤집는다.
- 리스트.sort() = 리스트 항목의 순서를 오름차순으로 정렬. 인자에 reverse=True를 넣으면 역순인 내림차순으로 정렬.
- sorted(리스트) = 리스트의 항목 순서를 오름차순으로 정렬 후 반환한다. sorted(리스트, reverse=True) = 역순.
- 예제) 한 글자 단어와 과일의 정렬 (그림 5-10)

```
word = list('살곶절')
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))
```

그림 5-10

```
[ '살', '곶', '절', '복', '숭', '아', '자', '두', '골', '드', '키', '위', '귤' ]
[ '자두', '복숭아', '살', '곶', '절', '복', '숭', '아', '골드키위', '귤' ]
```

그림 5-10 실행결과

- 리스트 컴프리헨션 : 조건을 만족하는 항목으로 리스트를 간결히 생성.
- 예제) 간단한 리스트 컴프리헨션 (그림 5-11)

```
# for문으로 리스트 생성
a = []
for i in range(10):
    a.append(i)
print(a)

# 컴프리헨션으로 리스트 생성
seq = [i for i in range(10)]
print(seq)

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

# 컴프리헨션으로 리스트 생성
squares = [i**2 for i in range(10) if i%2 == 1]
print(squares)
```

그림 5-11

```
[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]
```

그림 5-11 실행결과

- 리스트 대입, 복사 : 리스트 대입에는 주의가 필요. 리스트에서의 대입 연산자 = 은 대입되는 변수가 동일한 시퀀스를 가리킨다. f2 = f1에서 f2와 f1은 하나의 같은 리스트가 된다. 리스트에 대입을 하기 위해서는 복사 후 대입이 필요하다. 이러한 복사를 깊은 복사라고 한다. f2 = f1[:], f3 = f1.copy(), f4 = list(f1)
- 변수의 동일 객체 여부 검사 is : 피연산자인 변수 2개가 동일한 메모리를 공유하는지 검사. 같으면 True, 다르면 False. print(f1 is f2)

III 항목의 순서나 내용을 수정할 수 없는 튜플

- 튜플 : 문자열, 리스트와 같은 항목의 나열인 시퀀스. 그러나 이들과 다르게 항목의 순서나 내용의 수정이 불가능.
- 튜플은 () 괄호 사이에 항목을 기술한다. 괄호는 생략할 수 있다.
- 튜플 항목 참조와 출력 : 튜플도 리스트와 동일하게 첨자참조와 슬라이스가 가능. 그러나 수정은 불가능.\
- 예제) K-pop 가수와 곡으로 구성된 튜플의 참조 (그림 5-12)

```
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[_]))
```

그림 5-12

```
( '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
```

그림 5-12 실행결과

• 튜플의 연결, 반복, 정렬, 삭제 : 정수 연산과 같이 +, * 사용 가능. 정렬 함수 sorted(튜플, reverse=True)로 호출 시 역순인 내림차순으로 정렬된 리스트를 반환. 절대 튜플 자체가 수정되지 않는다. 문장 del에서 kpop을 지정 시 변수 자체가 사라진다.

• 예제) 영어 요일 단어로 구성된 튜플 만들기 (그림 5-13)

```
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)
```

그림 5-13

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

그림 5-13 실행결과

- Chapter 5 실습 문제

1. 영한 사전과 같이 한글과 영어에 대응되는 튜플 korean과 english를 만든 후, 표준 입력으로 한글을 입력받아 영어를 출력하는 프로그램을 작성하시오.

```
- korean = ('정렬', '초보자', '내포', '사전')
- english = ('sorting', 'novice', 'comprehension', 'dictionary')
- 단어가 사전에 없으면 적절한 메시지를 출력
korean = ('정렬', '초보자', '내포', '사전')
english = ('sorting', 'novice', 'comprehension', 'dictionary')
word = input('찾을 단어 입력 ? ')
n = korean.count(word)
if n == 0:
    print('단어가 없습니다.')
else:
    print(english[korean.index(word)])
```

```
찾을 단어 입력 ? 이름
>>>
= RESTART: C:\Users\#suit
on.py
찾을 단어 입력 ? 사전
dictionary
>>>
= RESTART: C:\Users\#suit
on.py
찾을 단어 입력 ? 초보자
novice
```

2. 다음 중첩된 리스트 data에서 각 행의 합과 열의 합을 리스트 rsum과 csum에 저장해 출력하는 프로그램을 완성하시오. (data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]])

```
data = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]
data_rsum = 0
rsum = []
for j in range(3):
    for i in range(3):
        data_rsum += i
    rsum.append(data_rsum)
    data_rsum = 0
print('각 행의 합: ', rsum)

data_csum = 0
csum = []
for j in range(3):
    for i in range(3):
        data_csum += data[i][j]
    csum.append(data_csum)
    data_csum = 0
print('각 열의 합: ', csum)
```

```
= RESTART: C:\Users\#suitkt
on.py
각 행의 합: [6, 15, 24]
각 열의 합: [12, 15, 18]
>>>
```

3. 다음 리스트 sports와 num을 활용해 스포츠 종목과 팀원 수가 번갈아 나오는 리스트를 만든 후 다음과 같이 출력하는 프로그램을 작성하시오. (sports = ['축구', '야구', '농구', '배구'] num = [11, 9, 5, 6])

```
- 리스트 sports에 insert() 메소드로 직접 팀원 수를 삽입
- 위 결과 리스트에서 슬라이스 sports[1::2]에 num을 대입
- 리스트 sports의 홀수 참조에 빈 문자 ''를 insert() 메소드로 삽입

sports = ['축구', '야구', '농구', '배구']
num = [11, 9, 5, 6]
sports.insert(1, 11)
sports.insert(3, 9)
sports.insert(5, 5)
sports.insert(7, 6)
print('메소드 insert()로 팀원 수 삽입')
print(sports)

for i in range(1,8,2):
    sports[i] = ''
print('메소드 insert()로 '' 삽입')
print(sports)

sports[1::2] = num.copy()
print('슬라이스 sports[1::2]에 num을 대입')
print(sports)
```

```
on.py
메소드 insert()로 팀원 수 삽입
['축구', 11, '야구', 9, '농구', 5, '배구', 6]
메소드 insert()로 '' 삽입
['축구', '', '야구', '', '농구', '', '배구', '']
슬라이스 sports[1::2]에 num을 대입
['축구', 11, '야구', 9, '농구', 5, '배구', 6]
>>>
```

4. 다음 중첩된 리스트를 for문으로 행과 열을 맞춰 항목을 출력한 후 다시 행과 열이 바뀐 형태를 for문으로 출력하는 프로그램을 작성하시오. (m = [[1, 2], [3, 4], [5, 6], [7, 8]])

- 수학의 행렬에서 행과 열이 바뀐 행렬을 전치(transpose) 행렬이라 함.

m = [[1, 2], [3, 4], [5, 6], [7, 8]]

```
print('원 행렬(m) 출력')
for i in range(len(m)):
    for j in range(2):
        print(m[i][j], end = ' ')
    print()

print('전치 행렬 출력')
for i in range(2):
    for j in range(len(m)):
        print(m[j][i], end = ' ')
    print()
```

```
원 행렬(m) 출력
1 2
3 4
5 6
7 8
전치 행렬 출력
1 3 5 7
2 4 6 8
>>>
```

5. 위와 동일한 중첩된 리스트에서 리스트 컴프리헨션을 활용해 행과 열이 바뀐 형태의 리스트를 새로 만들고, 이 변환된 리스트를 다음과 같이 출력하는 프로그램을 작성하시오. (m = [[1, 2], [3, 4], [5, 6], [7, 8]])

- 다음 리스트 컴프리헨션을 활용

transpose = [[row[i] for row in m] for i in range(len(m[0]))]

m = [[1, 2], [3, 4], [5, 6], [7, 8]]

```
transpose = [[row[i] for row in m] for i in range(len(m[0]))]
print('트랜스포즈를 컴프리헨션으로 만들어 그대로 출력')
print(transpose)

print('트랜스포즈를 for문으로 출력')

for i in range(len(m[0])):
    for row in m:
        print(row[i], end = ' ')
    print()
```

```
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-64\python.exe
트랜스포즈를 컴프리헨션으로 만들어 그대로 출력
[[1, 3, 5, 7], [2, 4, 6, 8]]
트랜스포즈를 for문으로 출력
1 3 5 7
2 4 6 8
>>>
```

6. 1에서 99까지의 난수 10개로 리스트를 만든 후 다시 이 리스트를 튜플로 변환하고, 다음과 같이 정렬된 리스트와 합, 항목 수, 최대, 최소, 평균을 출력하는 프로그램을 작성하시오.

- 함수 tuple(리스트)은 리스트를 튜플로 변환해 반환

- 함수 min(튜플)은 튜플에서 최소인 항목을 반환

```
import random
L = []

for i in range(10):
    L.append(random.randint(1,99))
```

```
print(L)
```

```
t = tuple(L)
```

```
print(t)
```

```
s_L = list(sorted(t))
```

```
print(s_L)
```

```
length = len(s_L)
```

```
sum = 0
```

```
for i in range(10):
```

```
    sum += t[i]
```

```
maxL = max(t)
```

```
minL = min(t)
```

```
avg = sum / length
```

```
print('합: %d, 항목수: %d' % (sum, length))
```

```
print('최대: {}, 최소: {}, 평균: {:.2f}'.format(maxL, minL, avg))
```

- 함수 sorted(튜플)은 튜플의 항목을 정렬해 다시 리스트로 반환

- 함수 max(튜플), sum(튜플) 등도 사용

```
= RESTART: C:\Users\suitk\AppData\Local\Programs\Python\Python38-64\python.exe
[98, 5, 48, 5, 91, 77, 24, 49, 80, 68]
(98, 5, 48, 5, 91, 77, 24, 49, 80, 68)
[5, 5, 24, 48, 49, 68, 77, 80, 91, 98]
합: 545, 항목수: 10
최대: 98, 최소: 5, 평균: 54.50
>>>
```

Chapter 6. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

I 키와 값인 쌍의 나열인 딕셔너리

- **딕셔너리**의 개념 : 키와 값의 쌍인 항목을 나열한 시퀀스. dct = { <key>: <value>, <key>: <value>, ... , <key>: <value> }
- 예제) K-pop 그룹의 인원수 (그림 6-1)

```
groupnumber = {'잔나비': 5, '트와이스': 9, '블랙핑크': 4, '방탄소년단': 7}
print(groupnumber)
print(type(groupnumber))
```

```
{'잔나비': 5, '트와이스': 9, '블랙핑크': 4, '방탄소년단': 7}
<class 'dict'>
```

그림 6-1 실행결과

- 예제) 강좌 정보로 구성된 딕셔너리 생성과 참조 (그림 6-2)

```
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'])
```

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

그림 6-2 실행결과

- 다양한 인자의 함수 **dict()**로 생성하는 딕셔너리 : 함수 dict()의 리스트나 튜플 내부에서 키-값, [키, 값] 리스트형식, (키, 값) 튜플형식 모두 사용 가능. 키가 단순문자열일 때, 키=값으로도 지정 가능.
- 예제) 방탄소년단 정보를 저장하는 다양한 딕셔너리 생성과 참조 (그림 6-3)

```
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', 인원=7, 리더='김남준', 소속사='빅히트 엔터테인먼트')
# 구성원 추가
bts['구성원'] = ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']

print(bts) # 전체 출력
print(bts['구성원']) # 구성원 출력
```

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

그림 6-3 실행결과

- 수정 불가능한 객체 딕셔너리 : 딕셔너리의 키는 수정 불가능한 객체 모두 사용 가능. 정수는 물론 실수도 가능. 튜플은 키로 사용 가능하지만, 리스트는 사용 불가능.
- 예제) 월 영어 단어 구성과 검색 (그림 6-4)

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

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

```
1: January, 2: February, 3: March, 4: April, 5: May, 6: June, 7: July, 8: August, 9: September
6: June
4: April
5: May
3: March
7: July
```

그림 6-4 실행결과

- 딕셔너리 항목의 순회 : **key()** = 키로만 구성된 리스트 반환. **items()** = (키, 값) 쌍의 튜플이 들어있는 리스트 반환. **values()** = 값으로 구성된 리스트 반환.

- 예제) 사계절의 영어 사전 생성과 항목 순회 (그림 6-5)

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

# 메소드 keys
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()
```

그림 6-5

```
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 여름 summer 가을 autumn 겨울 winter
봄 여름 summer 가을 autumn 겨울 winter
```

그림 6-5 실행결과

- 딕셔너리 항목의 참조와 삭제 : **get(키)** = 키의 해당 값 반환. **pop(키)** = 키인 항목 삭제, 삭제되는 키의 해당 값 반환. **popitem()** = 임의의 (키, 값)의 튜플을 반환하고 삭제. **del** = 키로 지정하면 해당 항목 삭제.
- 딕셔너리 항목 전체 삭제와 변수 제거 : **clear()** = 기존의 모든 키:값 항목 삭제. **del** = 딕셔너리 변수 자체 제거.
- 예제) 색상 사전의 조회와 삭제 (그림 6-6)

```
color = dict(검은색='black', 흰색='white', 녹색='green', 파란색='blue')
print(color)

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

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

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

# 모든 항목 삭제
color.clear()
print(color)
```

그림 6-6

```
{'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
임의의 삭제: ('노란색', 'yellow')
임의의 삭제 후: {'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
검은색 삭제 후: {'녹색': 'green', '파란색': 'blue'}
{}
.
```

그림 6-6 실행결과

- 딕셔너리 결합과 키의 멤버십 검사 연산자 **in** : **update(다른 딕셔너리)** = 인자인 다른 딕셔너리 합병. **in** = 키가 존재하는지 간단히 검사. **not in**도 사용 가능. 값의 존재 여부는 확인 불가.

II 중복과 순서가 없는 집합

- 수학에서 배운 집합을 처리하는 자료형 : 집합은 중복되는 요소가 없으며, 순서도 없는 원소의 모임. 집합의 원소는 중복을 허용하지 않으므로 멤버십 검사와 중복 제거에 주로 사용.
- 내장 함수 `set()`을 활용한 집합 생성 : `set(원소로 구성된 리스트 or 튜플 or 문자열)` = 집합 생성. 인자가 없으면 공집합 생성.
- 중괄호로 직접 원소를 나열해 집합 생성 : { 원소1, 원소2, ... }와 같이 원소를 콤마로 구분해 나열. 리스트나 딕셔너리는 원소로 사용 불가.
- 예제) 한 글자 단어의 집합 만들기 (그림 6-7)

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

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

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

그림 6-7

그림 6-7 실행결과

- 집합의 원소 추가와 삭제 : `add(원소)` = 원소 추가. `remove(원소)`, `discard(원소)` = 원소 삭제. `pop()` = 임의의 원소 삭제. `clear()` = 집합의 모든 원소 삭제.
- 예제) 로또 번호를 `randrange()`와 `sample()` 함수로 생성 (그림 6-8)

```
from random import randrange
from random import sample

# randrange() 함수와 집합을 이용, 중복 제거
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()
```

```
16 45 40 37 21 5
집합: {37, 5, 40, 45, 16, 21}
정렬 리스트: [5, 16, 21, 37, 40, 45]
```

```
# sample() 함수를 이용하면 매우 간편
lotto = sample(range(1, 46), 6)
print('sample 함수 리스트: {}'.format(lotto))
print('sample 함수 정렬 리스트: {}'.format(sorted(lotto)))
```

```
sample 함수 리스트: [8, 9, 26, 45, 25, 19]
sample 함수 정렬 리스트: [8, 9, 19, 25, 26, 45]
```

그림 6-8

그림 6-8 실행결과

- 집합의 주요 연산인 합집합, 교집합, 차집합, 여집합 : 양쪽 모든 원소를 합하는 합집합 = 연산자 `|` 와 메소드 `union()` 사용. 집합 자체가 수정되지 않음. 메소드 `a.update(b)`도 합집합과 같은 효과.
- 양쪽 모든 집합에 속하는 원소로 구성되는 교집합 = 연산자 `&` 와 메소드 `intersection()` 사용. `a.intersection(b)`도 사용.
- 피연산자인 왼쪽 집합에는 있지만 오른쪽에는 없는 원소로 구성되는 차집합 = 연산자 `-` 와 메소드 `difference()` 사용. 교환법칙은 성립하지 않는다.
- 한쪽 집합에는 소속되지만 교집합이 아닌 원소로 구성되는 여집합 = 연산자 `^` 와 메소드 `symmetric_difference()` 사용.
- 집합 연산의 축약 대입 연산자 `|=`, `&=`, `-=`, `^=` : 각 연산으로 생성된 집합의 결과가 왼쪽 피연산자에 대입. 메소드 `update()`의 결과와 동일.
- 예제) 요일 문자열 원소로 구성된 집합 연산 수행 (그림 6-9)

```
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))
```

```
{'토요일', '일요일'}
{'월요일', '수요일', '목요일', '금요일', '화요일'}
{'월', '화', '일'}
{'토', '수', '목'}
{'토', '수', '목', '화', '일'}
```

그림 6-9

그림 6-9 실행결과

- 함수 len()과 소속 연산 in : len() = 집합에 들어 있는 원소의 개수 확인. in = 특정 원소가 집합에 있는지 확인.

III 내장 함수 zip()과 enumerate(), 시퀀스 간의 변환

- 내장함수 zip() : zip() = 몇 개의 리스트나 튜플의 항목으로 조합된 튜플 생성. 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어주는 역할. list(zip(a, b)) = 항목이 튜플인 리스트 생성. tuple(zip(a, b)) = 항목이 튜플인 튜플 생성.
- 2개의 리스트나 튜플로 키-값 항목인 딕셔너리 생성 가능. dict(zip(a, b)) a리스트는 키, b튜플은 값의 조합으로 구성된 딕셔너리 생성.
- 내장 함수 enumerate() : enumerate(시퀀스) = 0부터 시작하는 첨자와 항목 값의 튜플 리스트 생성. for 반복의 시퀀스에 사용하는 것이 좋다. format() 메소드에서 *tp로 지정 시 자동으로 나뉘어 앞 { }에는 첨자, 뒤 { }에는 값이 출력.
- 시퀀스 간의 변환 : list()와 tuple(), set(), dict() 등을 사용해 간편하게 변환 가능. 딕셔너리를 다른 시퀀스로 변환하면 항목이 키로만 구성.
- 예제) 구기 종목과 팀원 수의 리스트에서 딕셔너리 구성 (그림 6-10)

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

# dict()와 zip() 함수로 종목의 이름을 키, 인원수를 값으로 저장
print('함수 dict(zip()):')
sportsnum = dict(zip(sports, num))
print(sportsnum)
```

그림 6-10 실행결과

```
['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}
```

그림 6-10

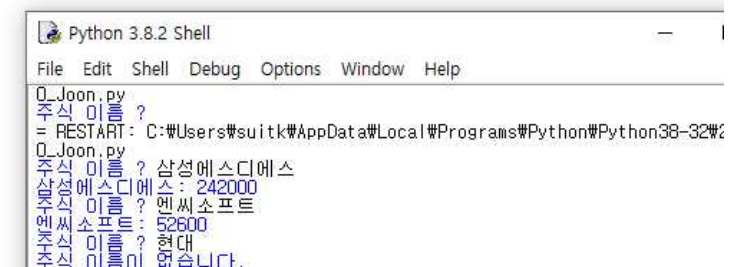
- Chapter 6 실습 문제

1. 다음 회사 6개의 주식 가격을 딕셔너리로 만든 후 다음과 같이 표준 입력으로 검색해 가격을 출력하는 프로그램을 작성하시오.

- 검색을 계속하면서, 주식 이름이 없으면 종료

```
comp = {'삼성에스디에스': 242000, '삼성전자': 47000, '엔씨소프트': 52600, '현대소프트': 5120, '골프존': 215000, '기아': 56300}

while True:
    c = input('주식 이름 ? ')
    if c in comp:
        print('{}: {}'.format(c, comp[c]))
    else:
        break
print('주식 이름이 없습니다.')
```



5. 1에서 20까지의 난수 5개를 두 번 얻어 각각 집합인 변수 A, B에 저장한다. 집합 A와 B의 합집합과 교집합, 차집합, 여집합을 구해 출력하는 프로그램을 작성하시오.

- 모듈 random의 sample() 함수를 사용한 다음 문장을 집합 변수 A에 저장

- A = set(sample(list(range(1, 21)), 5))

```
from random import sample
A = set(sample(list(range(1, 21)), 5))
B = set(sample(list(range(1, 21)), 5))
```

```
print('A = ', A)
print('B = ', B)
```

```
print()
```

```
print('A | B = ', A | B)
print('A & B = ', A & B)
print('A - B = ', A - B)
print('A ^ B = ', A ^ B)
```

```
A = {2, 3, 6, 7, 10}
B = {2, 5, 7, 11, 19}

A | B = {2, 3, 5, 6, 7, 10, 11, 19}
A & B = {2, 7}
A - B = {10, 3, 6}
A ^ B = {19, 3, 5, 6, 10, 11}
>>>
```

4. 기 타 참 고 자 료

| 제목 | 저자 | 발행연도 |
|---------------------------------|------------------|------|
| 절대 JAVA : 자바프로그래밍의 기초부터 안드로이드까지 | 강환수, 조진형 | 2015 |
| Perfect C | 강환수, 강환일, 이동규 | 2015 |
| 자연의 원리에 귀를 기울이다. | 네이버 블로그 | |

- 절대 JAVA : 1학년 자바언어 강의 시간에 사용했던 교재로, 파이썬 초기 강의에서 자바언어와 C언어, 파이썬 간에 비슷한 문법 구조가 보여서 복습과 예습을 겸해 참고한 교재이다.
- Perfect C : 현재 배우고 있는 C언어 강의 시간에 사용하는 교재로, 마찬가지로 복습과 예습을 겸해 참고한 교재이다.
- 자연의 원리에 귀를 기울이다. : 네이버 블로그인데, 본 교재에 나와있는 실습 문제들을 해결하다가 막히는 부분이 있으면 구글링을 하는 편이었다. 구글링 도중 나온 블로그 중 설명이 깔끔하고 예제들도 다양하게 있어서 많이 참고했다.

5. 소 감 및 향 후 계 획

먼저, 해당 과목을 배움에 있어서 1학년 강의 시간에 들던 Java와 C언어와 굉장히 문법적으로 비슷하다는 것을 느꼈다. 하지만 비슷한 것은 초반 문법 몇 가지 정도였고, 중반부로 갈수록 조금 차이가 있는걸 느꼈다. 해당 과목을 학습하면서 개요에서 작성했던 파이썬의 특징들을 확실하게 느낄 수 있었다. 코드가 굉장히 간결하고 빠르며, 여러 라이브러리들을 쉽게 사용할 수 있었다. 문법 구조가 간결해서 배우는 데 큰 어려움이 없었고, 교재에 있는 문제 및 실습 코드들도 쉽게 해결할 수 있었다. 현 포트폴리오에는 교재에 나와있는 예제 코드들과, 과제로 풀었던 짝수번호 실습 문제들에 더해서 응용단계 문제들을 첨부했다. 특히 딕셔너리와 튜플의 참조 부분이 제일 어려웠고, 아직도 헷갈리는 정도다. 현재까지 6장밖에 배우지 않았지만, 앞으로 더 배운다면 다른 언어들과 병행하여 사용할 수 있다고 생각한다. 졸업작품을 어떻게 할지 아직 정하지 못했지만, 파이썬이 필요할 경우를 대비하여 강의에 열심히 참여하고, 실습하며 원활하게 프로그래밍할 수 있을 정도로 숙달시킬 의향이다.