

파이썬프로그래밍 포토폴리오



반	PC
이름	이재원
학번	20161863

파이썬프로그래밍 강의계획서



강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 1학기	전공	컴퓨터정보공학과	학부	컴퓨터공학부
과 목 명	파이썬프로그래밍(2019009-PC)			
강의실 과 강의시간	화:1(3-217),2(3-217),3(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 누구나 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>

Contents

Chapter01

Section 01 파이썬 언어와 컴퓨팅 사고력

1.1 파이썬 언어란?

1.2 컴퓨팅 사고력과 파이썬

Section 02 파이썬 설치와 파이썬 셸 실행

2.1 파이썬 개발 도구 설치와 파이썬 셸의 실행

2.2 파이썬 셸에서 첫 대화형 프로그래밍

2.3 편집기에서 첫 파일 프로그래밍

Section 03 제4차 산업혁명 시대,모두에게 필요한 파이썬

3.1 쉽고 강력한 언어

3.2 빅데이터 처리와 머신 러닝 등

다양한 분야에 적합한 언어

3.3 다양한 종류의 파이썬과 개발 환경

3.4 인터프리트 방식의 언어,파이썬

Chapter02

Section 01 다양한 자료: 문자열과 수

- 1.1 자료의 종류와 문자열 표현
- 1.2 문자열 연산자 $+$, $*$ 와 주석
- 1.3 정수와 실수의 이해
- 1.4 정수와 실수의 다양한 연산

Section 02 변수와 키워드, 대입 연산자

- 2.1 자료형
- 2.2 변수와 대입연산자

Section 03 제4차 산업혁명 시대,모두에게 필요한 파이썬

- 3.1 표준 입력과 다양한 변환 함수
- 3.2 16진수,10진수,8진수,2진수와 활용

Chapter03

Section 01 문자열 다루기

- 1.1 문자열 str 클래스와 부분 문자열 참조 슬라이싱
- 1.2 문자열의 부분 문자열 참조 방식
- 1.3 문자 함수와 이스케이프 시퀀스

Section 02 문자열 관련 메소드

- 2.1 문자열 대체와 부분 문자열 출현 횟수, 문자열 삽입
- 2.2 문자열 찾기
- 2.3 문자열 나누기
- 2.4 다양한 문자열 변환 메소드
- 2.5 출력을 정형화하는 함수 format()
- 2.6 C 언어의 포매팅 스타일인 %d와 %f 등으로 출력

Section 03 논리 자료와 다양한 연산

- 3.1 논리 값과 논리 연산
- 3.2 관계 연산

3.3 멤버십 연산 in

3.4 비트 논리 연산

3.5 비트 이동 연산

Chapter04

Section 01 조건에 따른 선택 if~~else

- 1.1 조건의 논리 값에 따른 선택
- 1.2 조건에 따라 하나를 선택하는 if~else
- 1.3 여러 조건 중에서 하나를 선택하는 구문 if~elif
- 1.4 중첩된 조건

Section 02 반복을 제어하는 for문과 while문

- 2.1 시퀀스의 내부 값으로 반복을 실행하는 for문
- 2.2 횟수를 정하지 않은 반복에 적합한 while 반복
- 2.3 중첩된 반복

Section 03 임의의 수인 난수와 반복을 제어하는 break문 continue문

- 3.1 임의의 수인 난수 발생과 반복에 활용
- 3.2 반복을 제어하는 break문과 continue문

Section 01 여러 자료 값을 편리하게 처리하는 리스트

- 1.1 리스트의 개념과 생성
- 1.2 리스트의 항목 참조
- 1.3 리스트의 항목 수정
- 1.4 리스트 내부에 다시 리스트를 포함하는 중첩 리스트

Section 02 리스트의 부분 참조와 항목의 삽입과 삭제

- 2.1 리스트의 부분 참조인 슬라이싱
- 2.2 리스트의 부분 수정
- 2.3 리스트의 항목 삽입과 삭제
- 2.4 리스트의 추가, 연결과 반복
- 2.5 리스트의 항목의 순서와 정렬
- 2.6 리스트 컴프리헨션
- 2.7 리스트 대입과 복사

Section 03 항목의 순서나 내용을 수정할 수 없는 튜플

- 3.1 괄호로 정의하는 시퀀스 튜플
- 3.2 튜플 연결과 반복, 정렬과 삭제

Section 01 키와 값인 쌍의 나열인 딕셔너리

- 1.1 딕셔너리의 개념과 생성
- 1.2 다양한 인자의 함수 dict()로 생성하는 딕셔너리
- 1.3 딕셔너리 키는 수정 불가능한 객체로 사용
- 1.4 딕셔너리 항목의 순회
- 1.5 딕셔너리 항목의 참조와 삭제
- 1.6 딕셔너리 항목 전체 삭제와 변수 제거
- 1.7 딕셔너리 결합과 키의 멤버십 검사 연산자 in

Section 02 중복과 순서가 없는 집합

- 2.1 수학에서 배운 집합을 처리하는 자료형
- 2.2 내장함수 set()을 활용한 집합 생성
- 2.3 중괄호로 직접 원소를 나열해 집합 생성
- 2.4 집합의 원소 추가와 삭제
- 2.5 집합의 주요 연산인 합집합, 교집합, 차집합, 여집합

2.6 함수 len()과 소속 연산 in

Section 03 내장 함수 zip()과 enumerate(), 시퀀스 간의 변환

3.1 내장 함수 zip()

3.2 내장 함수 enumerate()

3.2 시퀀스 간의 변환

Chapter01

Section01

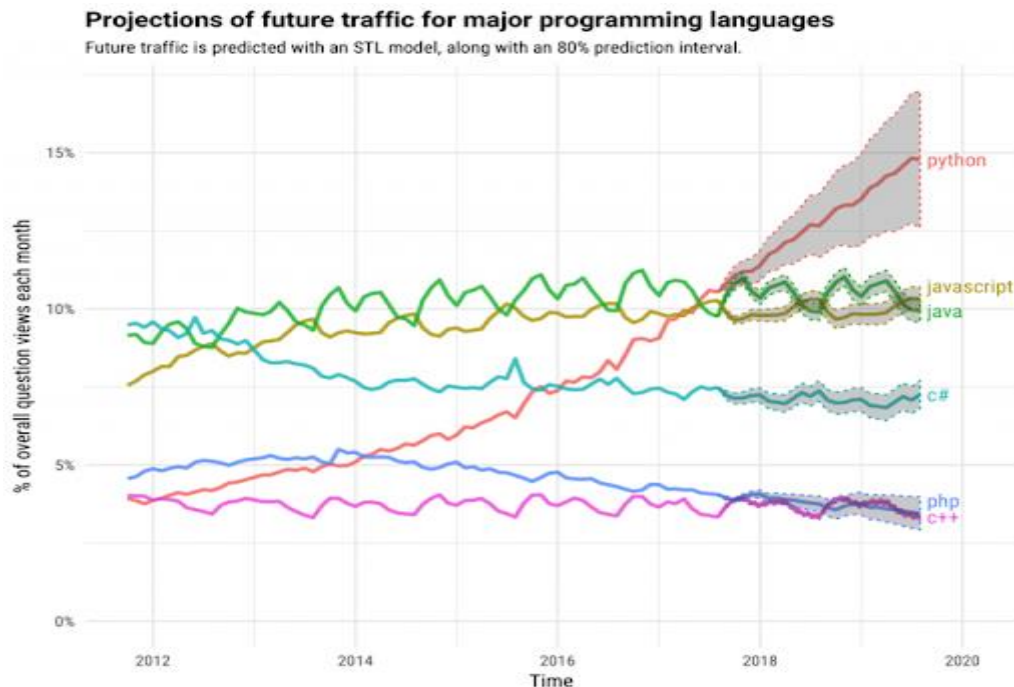
1.1 파이썬 언어란?

-파이썬(python)은 배우기 쉽고 누구나 무료로 사용할 수 있는 오픈 소스(open source) 프로그래밍 언어(programming language)이다.

-파이썬은 1991년 네덜란드의 귀도 반 로섬이 개발했으며,현재는 비영리 단체인 파이썬 소프트웨어 재단(PSF: Python Software Foundation)이 관리하고 있음.

-파이썬의 사전적 의미는 '비단뱀' 으로, 그리스 신화에서 유래했으며, 파이썬 로고가 비단뱀인 것이 바로 이 때문이다.

-파이썬은 현재 미국과 우리나라의 대학 등 전세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나다. 특히 비전공자의 컴퓨팅 사고력(computational thinking)을 키우기 위한 프로그래밍 언어로도 많이 활용되고 있다. 파이썬은 배우기 쉽고 간결하며, 개발 속도가 빠르고 강력하기 때문이다. 또한 파이썬은 라이브러리(library)가 풍부하고 다양한 개발 환경을 제공하고 있어 개발자가 쉽고 빠르게 소프트웨어를 개발하는 데 도움을 준다. 아래는 파이썬의 성장 그래프이다.



프로그래밍 언어의 예상 트래픽

1.2 컴퓨팅 사고력과 파이썬

-제4차 산업혁명 시대는 모든 사물이 연결된 초연결 사회이며, 모든 산업은 컴퓨터 과학,공학 기술을 기반으로 융합된다. 결국 모든 산업 분야에서 컴퓨터 과학 기술을 필요로 하게 된다는 것이다. 그러므로 미래의 인재는 컴퓨터 과학 원리와 개념을 활용해 자신의 영역과 융합할 수 있는 역량을 갖추어야 한다.이러한 능력을 컴퓨팅 사고력(CT:Computational Thinking)이라고하는데, 이는 '컴퓨터 과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용해 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현해 적용할 수 있는 능력'으로 정의할 수 있다.

-컴퓨팅 사고력

컴퓨터 과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용해 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적인 솔루션을 구현하는 능력



컴퓨팅 사고력(Computational Thinking)

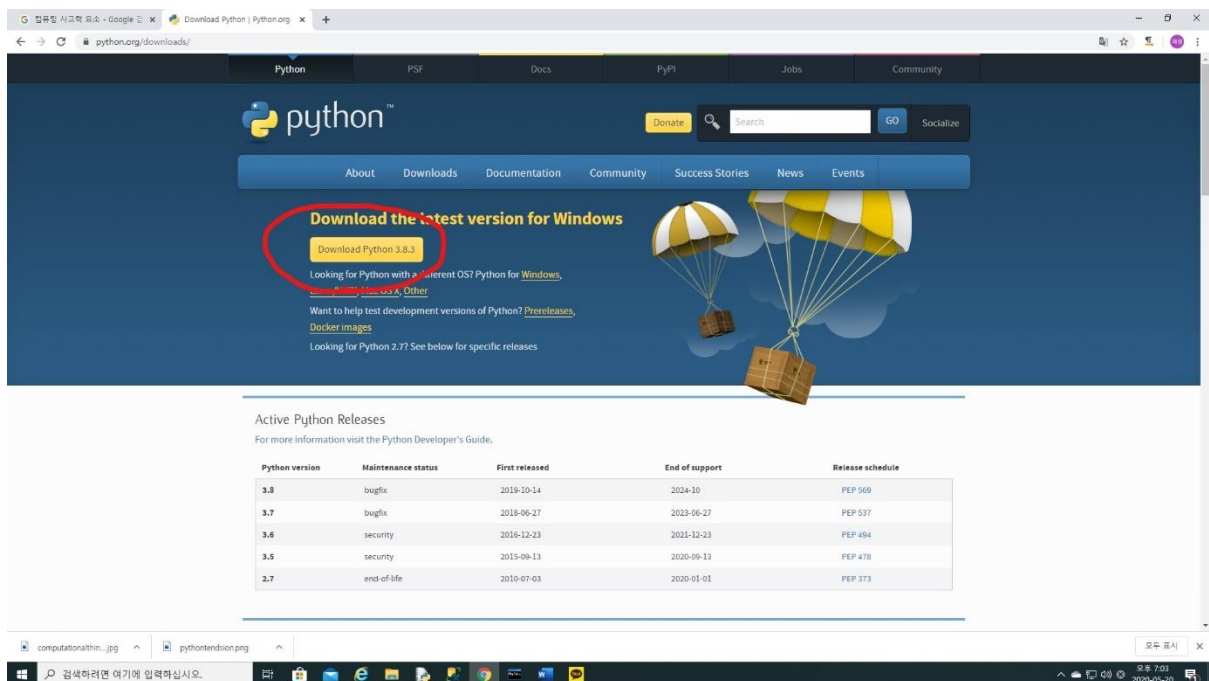
-컴퓨터 사고력 구성 요소(분해,패턴 인식,추상화,알고리즘)

- 분해(decomposition): 데이터, 프로세스 또는 문제를 작고 관리 가능한 부분으로 나눔
- 패턴 인식(pattern recognition): 데이터의 패턴, 추세 및 정규성 관찰
- 추상화(abstraction): 패턴을 생성하는 일반 원칙을 규정
- 알고리즘 설계(algorithm design): 이 문제와 유사한 문제 해결을 위한 단계별 지침을 개발

Section02

2.1 파이썬 개발 도구 설치와 파이썬 쉘의 실행

- 파이썬 다운로드 홈페이지(www.python.org/downloads)로 이동한 후 Download Python 3.x.x를 눌러 설치가능.(시간이 지남에 따라 버전이 달라질 수 있음)
- 실행 버튼을 눌러 바로 실행하거나 저장을 눌러 설치 파일을 저장 후 설치할 수도 있음.



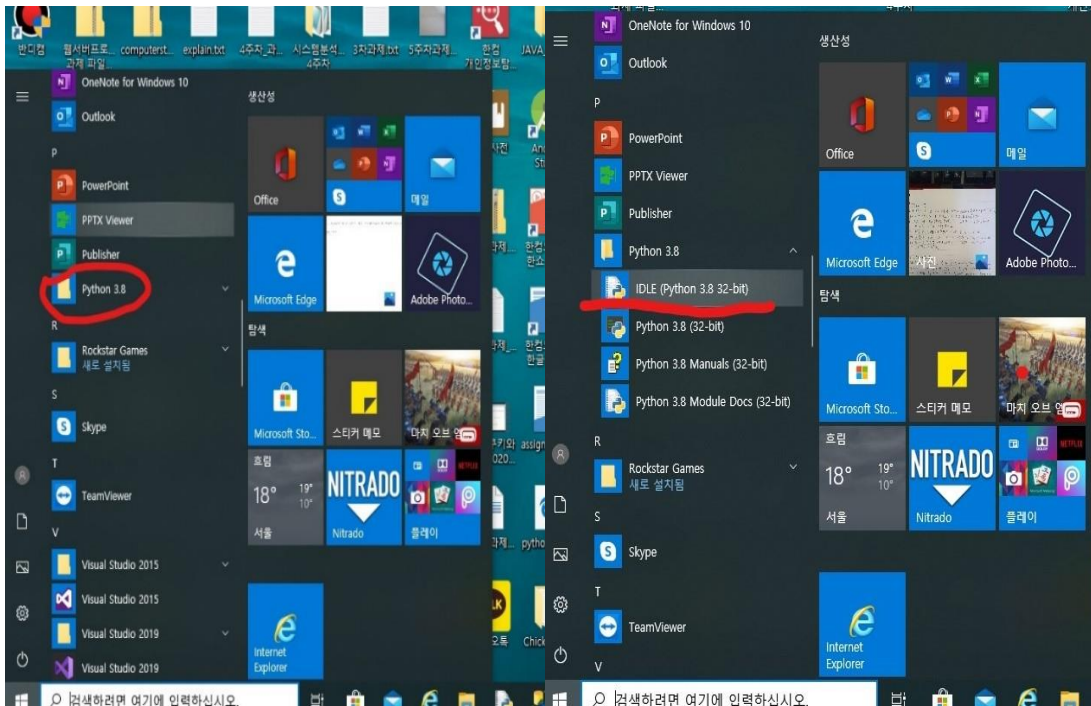
파이썬 다운로드 홈페이지

*빨간색 원의 다운로드 버튼을 눌러 다운로드 할 수 있음

-설치한 파이썬 쉘의 실행

설치된 파이썬 쉘을 실행하려면 [윈도우 화면의 하단에 위치한 윈도 시작 버튼을 클릭]

->[설치된 Python 3.8을 클릭] -> [IDLE (Python 3.8 32-bit)클릭]



윈도우 버튼을 눌러 IDLE 실행하기

***사용하기 편하게 작업표시줄에 설정 추천**

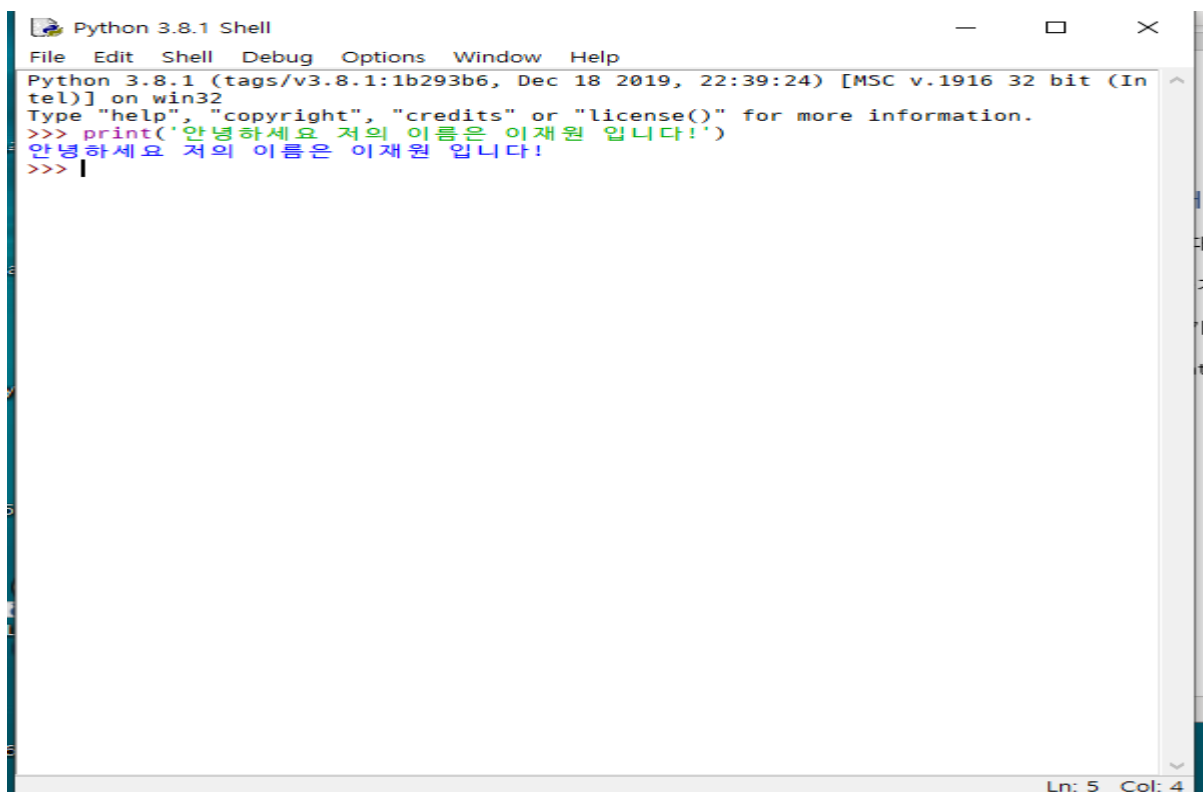
2.2 파이썬 셸에서 첫 대화형 프로그래밍

-파이썬 셸에서 첫 대화형 프로그래밍을 실행해보자

1.셸에 출력을 하기 위해서 print()함수를 사용

2.문자를 출력하기 위해서 작은따옴표('), 큰따옴표(") 중 하나 사용

소스코드: print('안녕하세요 저의 이름은 이재원 입니다!')



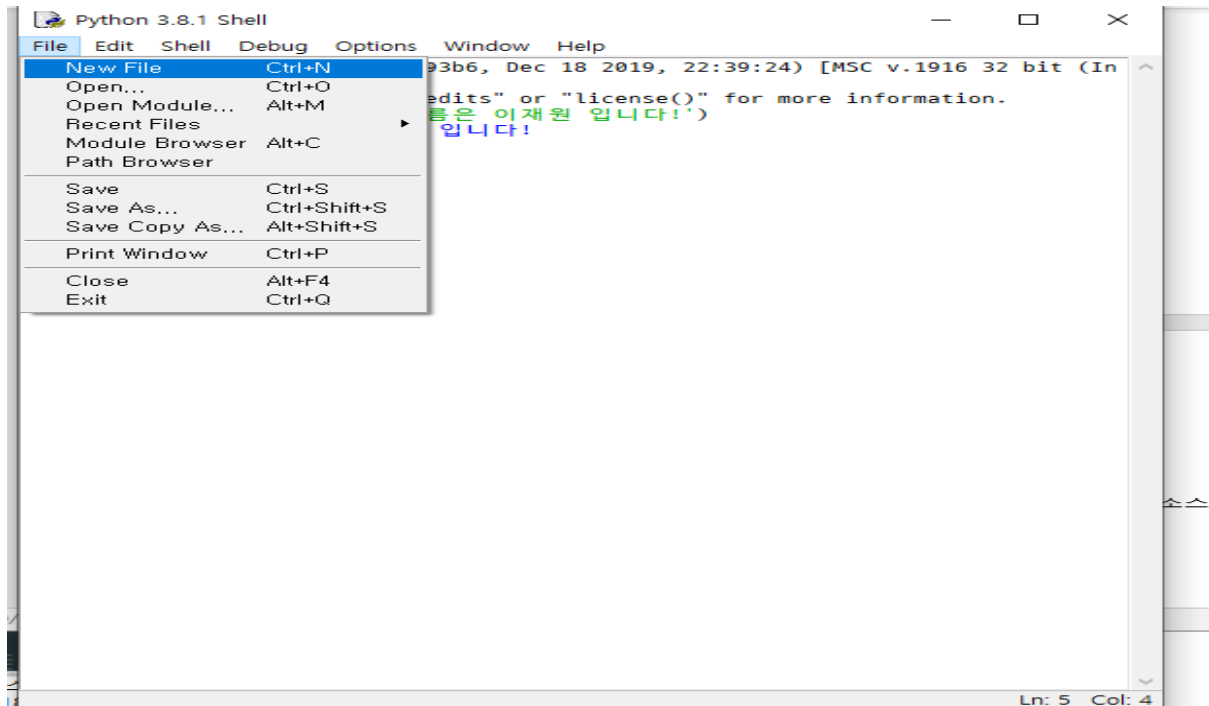
```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('안녕하세요 저의 이름은 이재원 입니다!')
안녕하세요 저의 이름은 이재원 입니다!
>>> |
```

-파이썬 셸의 명령어는 첫 칸부터 입력 해야함 아니면 오류 발생

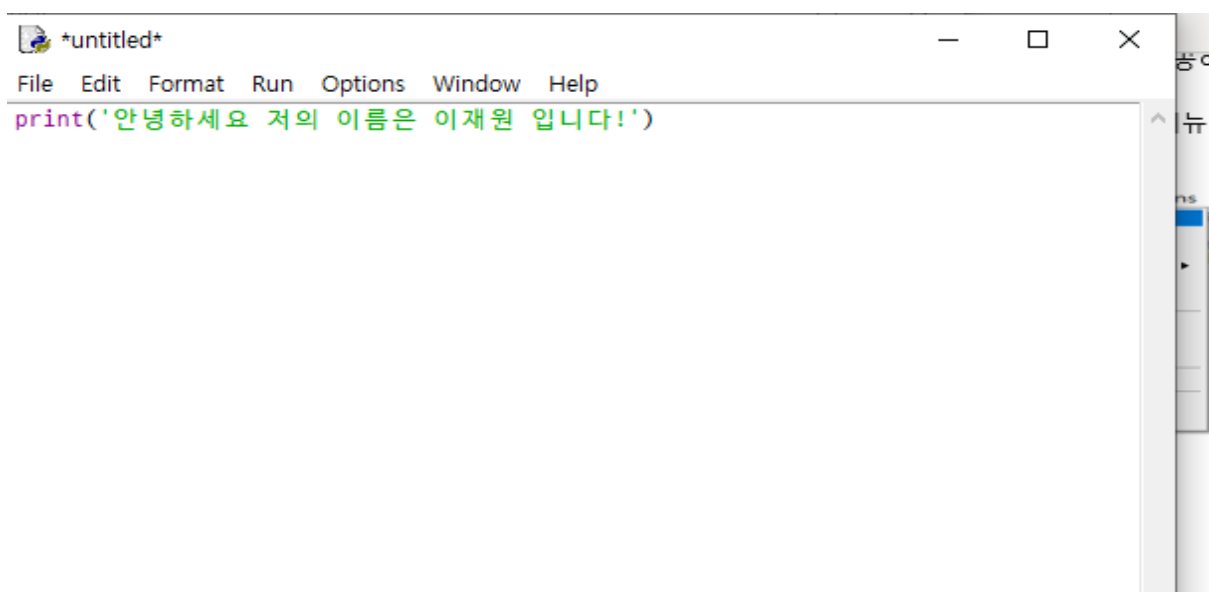
2.2 파이썬 셸에서 첫 대화형 프로그래밍

-파이썬에서는 셸이 제공하는 편집기를 이용하여 소스코드를 저장한후 실행가능

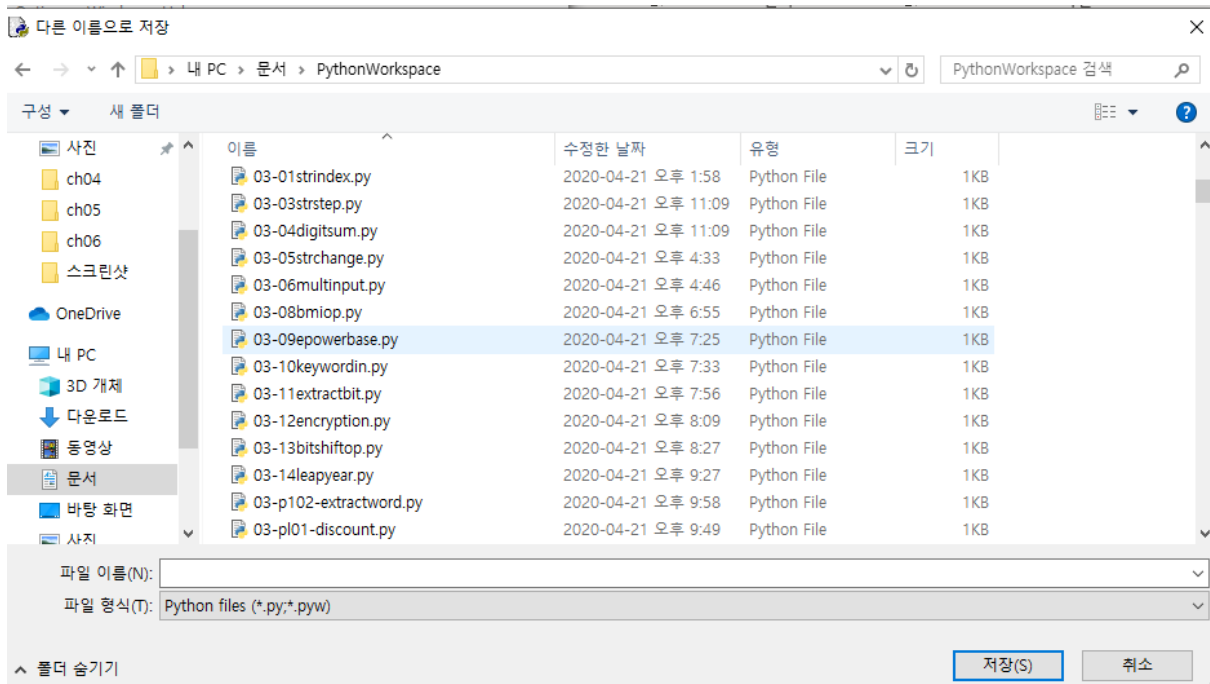
1. 파이썬 셸에서 File메뉴 -> New File 클릭



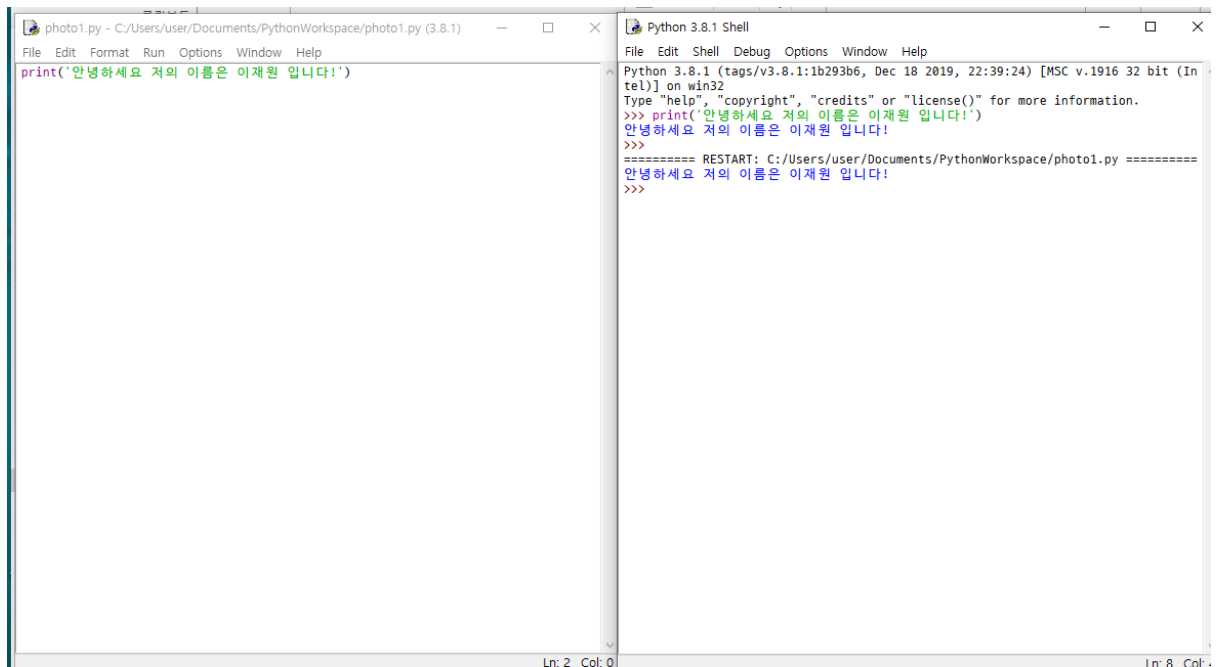
-열린 편집기에 코딩하기



-입력한 소스코드를 실행하기 위해서 자신 만의 저장소에 저장하기



-저장 후 실행(F5)을 눌러 소스코드를 실행



3.1 쉽고 강력한 언어

-파이썬의 특징

- 간결하고, 인간의 사고 체계와 닮아 사용하기 쉽다.
- 무료이며, 다양한 자료 구조의 제공으로 생산성이 높다.
- 라이브러리가 독보적이고 강력하며, 데이터과학과 머신 러닝 등과 같은 분야에 적용가능

-파이썬 활용 분야

- 데이터과학
- 인공 지능의 머신 러닝과 딥 러닝
- 빅데이터 처리를 위한 통계 및 분석

-인터프리트 방식과 컴파일 방식

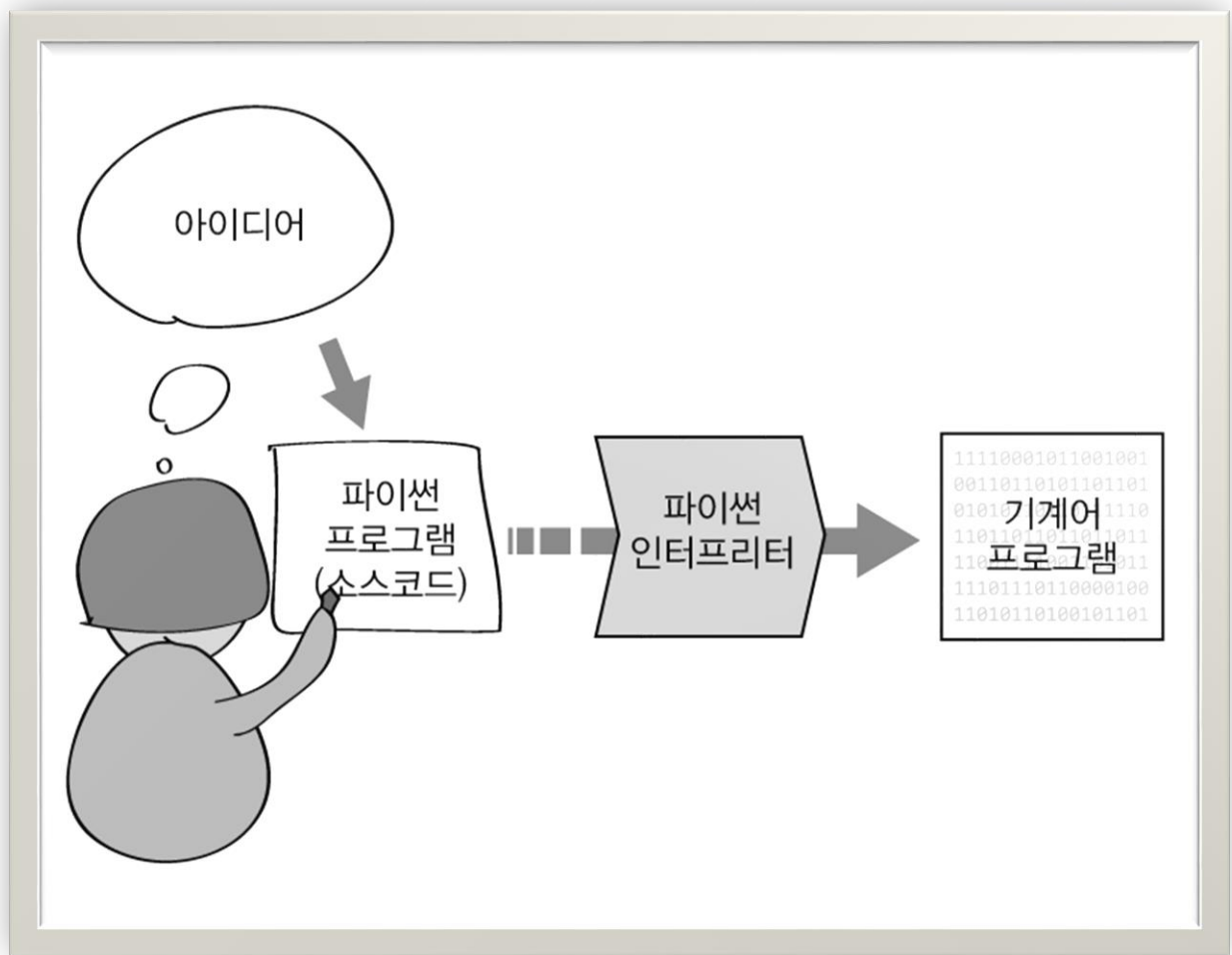
*인터프리터 방식 : 베이직(basic), 파이썬(python)

*컴파일 방식 : 자바(java), C 등

-컴퓨터가 이해하고 실행하는 언어는 기계어(machine language)뿐이다. 그러므로 사람이 작성한 고급언어를 다시 0과1로 이루어진 기계어로 변환해야 한다. 이때 필요한게 인터프리터와 컴파일러이다

*인터프리터 방식은 동시 통역처럼 파이썬의 문자열 한 줄 한 줄마다 즉시 번역해 실행하는 방식이다.

*컴파일 방식은 외국 책을 번역할 때처럼 여러 문장의 소스 단위로 번역해 기계어 파일의 실행 파일을 만든 후에 실행하는 방식이다.



파이썬 소스코드에서 목적코드로의 변환

Chapter02

Section01

1.1 자료의 종류와 문자열 표현

-파이썬에서는 문자 하나 또는 문자가 모인 단어나 문장 또는 단락 등을 문자열(string)이라 한다.

-문자열은 작은따옴표(') 혹은 큰따옴표(") 안에 기술한다.

-파이썬은 문자 하나도 문자열로 취급하며, 따옴표로 둘러싼 숫자 '34','3.14' 도 문자열로 취급함 즉 따옴표로 둘러 싸이면 모두 문자열로 취급

-문자열을 표현할 때 작은따옴표로 시작하면 작은따옴표로 마무리 짓고 큰따옴표로 시작하면 큰따옴표로 마무리 짓는다.

1.2 문자열 연산자+,*와 주석

-문자열과 문자열을 연결할 때는 +(문자열 연결 연산자)를 이용한다

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print('안녕하세요 저의 이름은 이재원 입니다!')
안녕하세요 저의 이름은 이재원 입니다!
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo1.py =====
안녕하세요 저의 이름은 이재원 입니다!
>>> print('안녕'+ '하세요')
안녕하세요
>>>
```


-문자열을 반복하려면 문자열반복연산자(*)을 사용한다 문자열반복연산자(*)의 피연산자는 문자열과 정수이다 ex('hello'*3) 피연산자의 순서는 바뀌어도 무관하다.

```
>>> print('안녕하세요'*3)
안녕하세요안녕하세요안녕하세요
>>> |
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

-주석(comments)은 소스 설명으로, 인터프리터는 주석을 무시한다. 파이썬은 주석을 #으로 시작하고 그 줄의 끝까지 유효하다. 주석 #은 한 줄만 가능하므로 여러 줄에 걸친 주석이 필요하면 줄마다 맨 앞에 #을 넣거나 문자열의 삼중 따옴표를 사용한다.

```
*photo1.py - C:/Users/user/Documents/PythonWorkspace/photo1.py (3.8.1)*
File Edit Format Run Options Window Help
#이것은 한줄 주석입니다. 인터프리터에게 전혀 영향을 받지 않습니다.
...
    이것은 여러줄 주석입니다.
    파이썬은 재미있습니다.
    여러분도 그렇게 생각하나요? ...
|
```

Ln: 7 Col: 0

1.2 정수와 실수의 이해

-숫자는 간단히 정수(integer)와 실수(float)로 나눈다.

-20,156 등은 정수, 소수점이 있는 3.141592, 45.45 등은 실수이다.

-실수는 문자e를 사용해 153.153을 1.53153e2와같이 표현 할 수 있다.

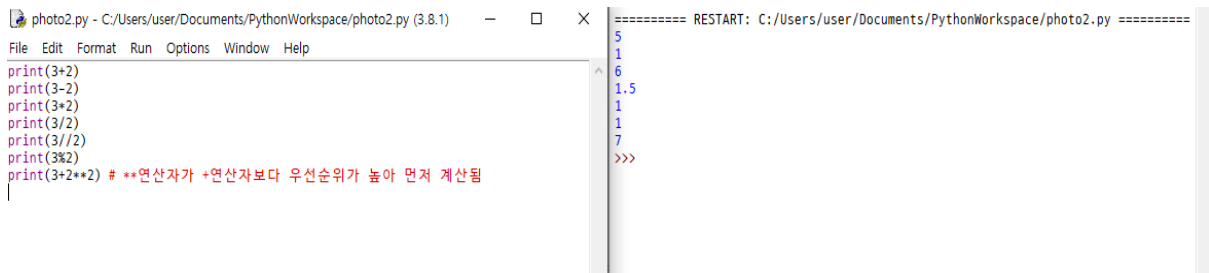
```
>>> 1.53153e2
153.153
>>> |
```

-1.4 정수와 실수의 다양한 연산

-산술연산자 종류

연산자	명칭	의미	우선순위	예
*	곱하기(multiply)	두 피연산자 곱하기	3	3*4
/	나누기(divide)	왼쪽을 오른쪽 피연산자로 나누기	3	20/3
%	나머지(modulus)	왼쪽을 오른쪽 피연산자로 나눈 나머지	3	21%4
//	몫 나누기(floor division)	왼쪽을 오른쪽 피연산자로 나눈 결과에서 작거나 같은 정수	3	10//3
**	거듭제곱,지수승(exponent)	왼쪽을 오른쪽 피연산자로 거듭제곱	1	2**3

-편집기에서 사칙연산자를 이용한 수의 계산



The screenshot shows a Python IDE window titled 'photo2.py - C:/Users/user/Documents/PythonWorkspace/photo2.py (3.8.1)'. The code in the editor is as follows:

```
print(3+2)
print(3-2)
print(3*2)
print(3/2)
print(3//2)
print(3%2)
print(3+2**2) # **연산자가 +연산자보다 우선순위가 높아 먼저 계산됨
```

The output on the right side of the IDE is:

```
5
1
6
1.5
1
1
7
>>>
```

-기본연습문제1

나머지연산자 % 와 정수몫나누기 연산자 // 를 이용하여 56600원의 만원
권,천원권, 오백원, 백원 의 개수와 나머지 잔돈을 구해보자

소스코드:

```
money = 56610
```

```
print('지불한 총 금액: %d원'%(money))
```

```
#만원권 갯수 구하기
```

```
manwon = money/10000
```

```
#총 금액을 만원으로 나눈 나머지를 구함
```

```
money %= 10000
```

```
#천원권 객수 구하기
```

```
cheonwon = money/1000
```

```
#현재 금액을 천원으로 나눈 나머지를 구함
```

```
money %= 1000
```

```
#오백원 갯수 구하기
```

```
ohbackwon = money/500
```

```
#현재 금액을 오백으로 나눈 나머지를 구함
```

```
money %= 500
```

```
print('잔돈 : %d원'%(money))
```

```
photo3.py - C:/Users/user/Documents/PythonWorkspace/photo3.py (3.8.1)
File Edit Format Run Options Window Help

money = 56610
print('지불한 총 금액: %d원'%(money))

#만원권 갯수 구하기
manwon = money/10000
#총 금액을 만원으로 나눈 나머지를 구함
money %= 10000
#천원권 갯수 구하기
cheonwon = money/1000
#현재 금액을 천원으로 나눈 나머지를 구함
money %= 1000
#오백원 갯수 구하기
ohbackwon = money/500
#현재 금액을 오백으로 나눈 나머지를 구함
money %= 500
#백원 갯수 구하기
backwon = money/100
#나머지 잔돈 구하기
money %= 100

print('만원 권 : %d개'%(manwon))
print('천원 권 : %d개'%(cheonwon))
print('오백원 갯수 : %d개'%(ohbackwon))
print('백원 갯수 : %d개'%(backwon))
print('잔돈 : %d원'%(money))

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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=====
RESTART: C:/Users/user/Documents/PythonWorkspace/photo3.py =====
지불한 총 금액: 56610원
만원권 : 5개
천원권 : 6개
오백원 갯수 : 1개
백원 갯수 : 1개
잔돈 : 10원
>>> |
```

-표현식 문자열 실행 함수 `eval('expression')`은 문자열에 연산식이 포함된 표현식이 있으면 표현식의 결과를 문자열로 반환해 준다.

```
>>> eval('3 + 5 * 2')
13
>>> eval('2*2**2')
8
>>> |
```

Section2

2.1 자료형

-자료형에는 기본자료형으로 int,float,str 등이 있다 이는 정수,실수,문자열 순이다.

-자료형을 알아볼때는 내장함수인 type()함수를 사용한다.

```
>>> type(10)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type('안녕')
<class 'str'>
>>>
```

위의 화면과 같이 type()함수에 리터럴을 넣어주면 리터럴에 해당하는 자료형이 반환된다.

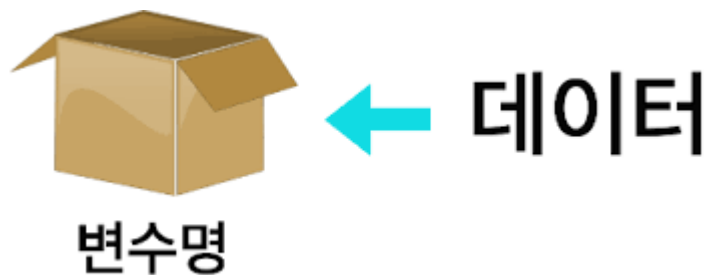
2.2 변수와 대입연산자

-변수는 쉽게 말해 어떤 데이터를 담을 수 있는 공간이다 변수는 메모리에 생성되고

우리는 메모리 주소대신 변수이름으로 변수를 참조한다. 변수에 어떤 데이터를 담을때는

=(대입연산자)를 사용한다

Ex) num = 10 , fnum = 3.14 , str1 = '문자열'



-다양한 대입연산자

다양한 대입 연산자	형태	원래형태	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

-기본연습문제2

+=연산자를 이용하여 1부터 100까지의 덧셈하기

소스코드:

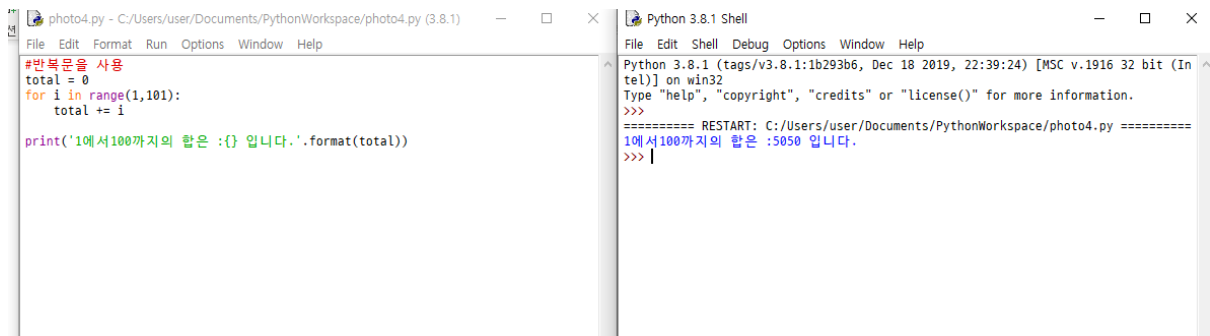
#반복문을 사용

```
total = 0
```

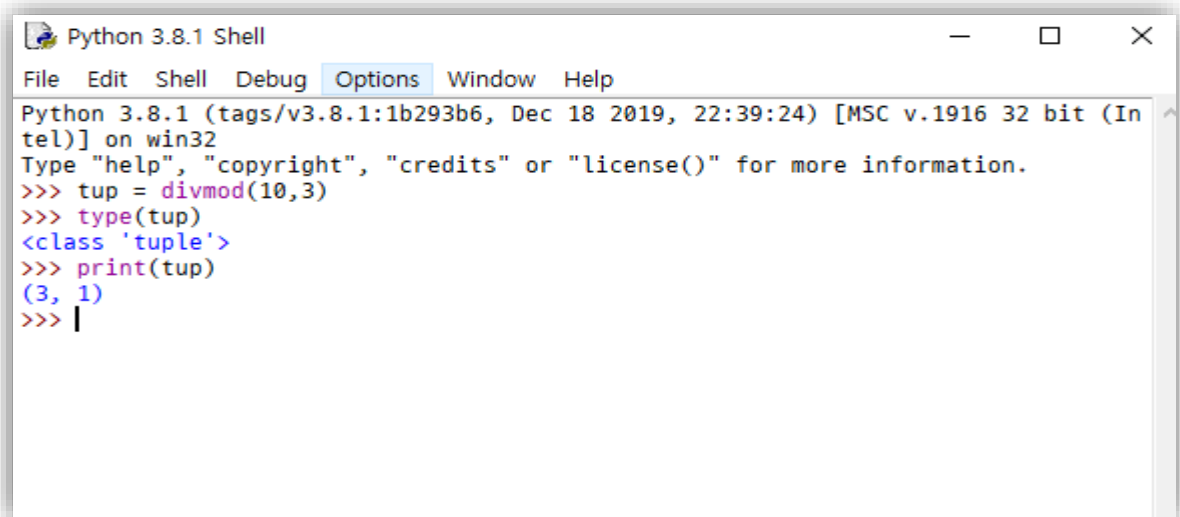
```
for i in range(1,101):
```

```
    total += i
```

```
print('1에서100까지의 합은 :{} 입니다.'.format(total))
```



-divmod(a,b)를 이용하면 몫과 나머지를 손쉽게 얻을 수 있다. 반환값은 a를 b로 나눈 몫과 나머지의 쌍인 튜플이 반환된다.



-파이썬에서는 콤마로 구분된 여러 변수에 순서대로 값을 대입할 수 있다.

Ex) a , b = 5 , 9


```
>>> a,b = 5,9
>>> print(a,b)
5 9
>>> |
```

Section03

3.1 표준 입력과 다양한 변환 함수

-표준 입력은 프로그램 과정에서 셸이나 콘솔에서 사용자의 입력을 받아 처리하는 방식을 표준 입력(standard input)이라 한다.

-함수 input()은 입력되는 표준 입력을 문자열로 읽어 반환하는 함수이고,input()에서 반환되는 입력 문자열을 변수에 저장하려면 대입 연산자(=)을 사용해 변수에 저장해야 한다.

Ex) str1 = input()

-사용자가 입력하는 내용을 설명해주려면 input('문자열') 처럼 인자로 설명문을 문자열로 input함수의 인자로 주면 된다.

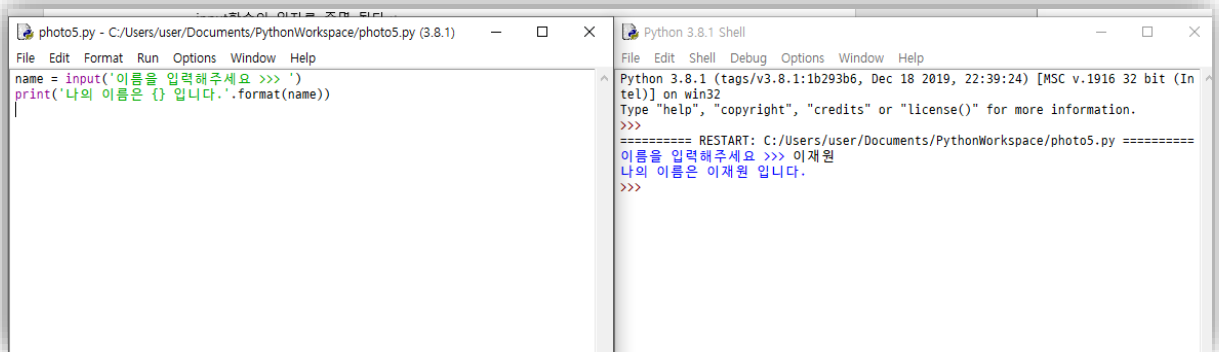
Ex) str1 = input('이름을 입력해 주세요 >>> ')

-기본연습문제3

Input()함수를 이용하여 자신의 학과를 입력 받아 출력하는 프로그램을 만들어보자

소스코드:

```
name = input('이름을 입력해주세요 >>> ')
print('나의 이름은 {} 입니다.'.format(name))
```



-문자열과 정수, 실수 간의 자료 변환 함수 str(),int(),float()

- 함수 str()은 주로 정수와 실수를 문자열로 변환하는 데 사용

```
>>> str(15)
'15'
>>> str(3.14)
'3.14'
>>> |
```

- 함수 int()는 정수 형태의 문자열인 '56', '7' 등을 정수로 float()는 소수점이 있는 실수 형태의 문자열인 '3.141592', '5.15' 등을 실수로 변환한다.

```
>>> str(15)
'15'
>>> str(3.14)
'3.14'
>>> int('56')
56
>>> float('3.141592')
3.141592
>>>
```

- 함수 int()는 실수를 소수점이 제거된 정수, float()는 정수를 실수로 변환한다.

```
>>> int(3.141592)
3
>>> float(15)
15.0
>>> |
```

-문자열을 함수 int()나 float()로 정수나 실수로 변환할 때는 주의해야 할 점이 있다.

문자열이 정수나 실수 형태가 아니면 오류가 발생한다. 즉, 문자열에 수가 아니라 문자가 포함되어 있으면 오류가 발생

```
>>> int('10pi')
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    int('10pi')
ValueError: invalid literal for int() with base 10: '10pi'
>>> float('3.141592f')
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    float('3.141592f')
ValueError: could not convert string to float: '3.141592f'
>>>
```

3.2 16진수, 10진수, 8진수, 2진수와 활용

-수의 상수를 표현하는 방법은 10진법뿐 아니라 16진법, 8진법, 2진법 등이 있다.

16진수는 맨 앞에 0x로 시작하며, 8진수는 0o, 2진수는 0b로 시작한다.

-10진수를 2진수, 8진수, 16진수로 변화해주는 함수는 각각 bin(), oct(), hex() 함수이다.

이 함수들은 인자로 정수를 받으며 받아온 정수를 각 함수에 맞는 진수로 변환된 문자열을 반환해준다.

-기본연습문제4

10진정수 하나를 입력 받아 입력 받은 정수를 2진수, 8진수, 10진수, 16진수로 출력해보자

소스코드:

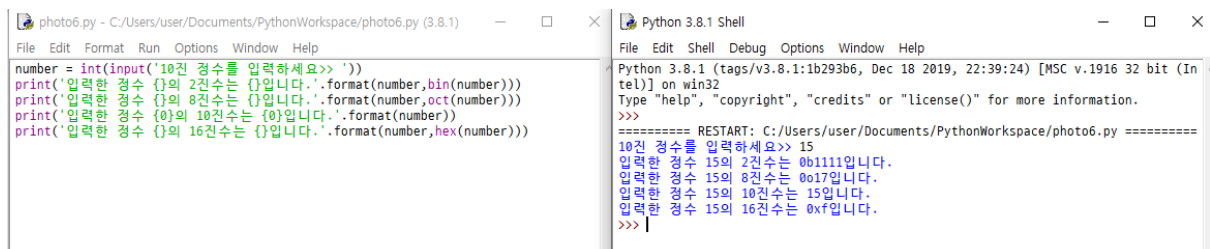
```
number = int(input("10진 정수를 입력하세요>> "))

print('입력한 정수 {}의 2진수는 {}입니다.'.format(number, bin(number)))

print('입력한 정수 {}의 8진수는 {}입니다.'.format(number, oct(number)))

print('입력한 정수 {}의 10진수는 {}입니다.'.format(number))

print('입력한 정수 {}의 16진수는 {}입니다.'.format(number, hex(number)))
```



```
photo6.py - C:/Users/user/Documents/PythonWorkspace/photo6.py (3.8.1) Python 3.8.1 Shell
File Edit Format Run Options Window Help 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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo6.py =====
10진 정수를 입력하세요>> 15
입력한 정수 15의 2진수는 0b1111입니다.
입력한 정수 15의 8진수는 0o17입니다.
입력한 정수 15의 10진수는 15입니다.
입력한 정수 15의 16진수는 0xf입니다.
>>> |
```

-함수 `int('strnum',n)`은 `strnum`을 `n`진수로 인식하고 10진수로 변환해준다.

`int('strnum',0)`은 `strnum`에 붙어있는 접두사에 따라 문자열을 인식한다.

예를들면 `int('0x15',0)` 이면 접두사 `0x`에 따라서 `'0x15'`를 16진수 문자열로 인식하고 `'0x15'`를 10진정수로 변환해준다.

```
>>> int('17')
17
>>> int('17',10)
17
>>> int('11',2)
3
>>> int('10',8)
8
>>> int('1A',16)
26
>>> int('0b11',0)
3
>>> int('0o11',0)
9
>>> int('0x11',0)
17
>>> |
```

내가 만든 문제 1

-두개의 수를 입력 받아 계산하는 계산기 프로그램을 만들어보자

소스코드:

```
while True:

    operand1 = input('첫번째 수를 입력하세요(종료(0)) >>>')

    if '.' in operand1:

        operand1 = float(operand1)

    else:

        operand1 = int(operand1)

    if operand1 == 0:

        break

    operand2 = input('첫번째 수를 입력하세요(종료(0)) >>>')

    if '.' in operand2:

        operand2 = float(operand2)

    else:

        operand2 = int(operand2)

    if operand2 == 0:

        break

    operator = input('+ - * / // ** 중 하나를 선택하세요 >>>')

    if operator == '+':

        print('{} + {} = {}'.format(operand1,operand2,operand1+operand2))

    elif operator == '-':

        print('{} - {} = {}'.format(operand1,operand2,operand1-operand2))
```

```
elif operator == '*':
```

```
print('{} * {} = {}'.format(operand1,operand2,operand1*operand2))
```

```
elif operator == '/':
```

```
print('{} / {} = {}'.format(operand1,operand2,operand1/operand2))
```

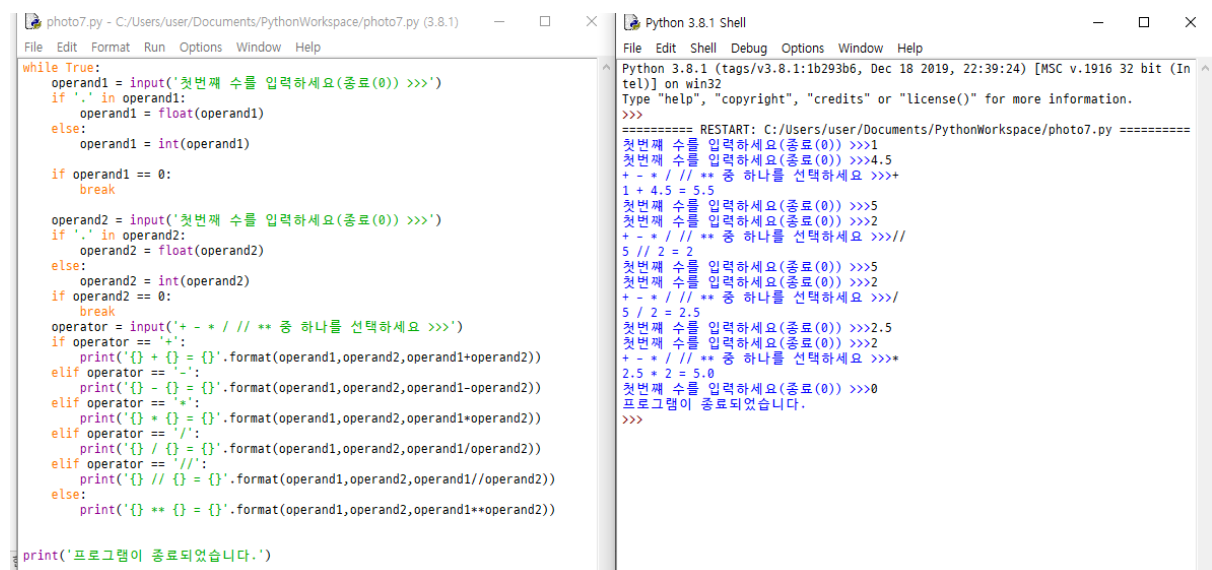
```
elif operator == '//':
```

```
print('{} // {} = {}'.format(operand1,operand2,operand1//operand2))
```

```
else:
```

```
print('{} ** {} = {}'.format(operand1,operand2,operand1**operand2))
```

```
print('프로그램이 종료되었습니다.')
```



```
photo7.py - C:/Users/user/Documents/PythonWorkspace/photo7.py (3.8.1)
File Edit Format Run Options Window Help

while True:
    operand1 = input('첫번째 수를 입력하세요(종료(0)) >>>')
    if '.' in operand1:
        operand1 = float(operand1)
    else:
        operand1 = int(operand1)

    if operand1 == 0:
        break

    operand2 = input('두번째 수를 입력하세요(종료(0)) >>>')
    if '.' in operand2:
        operand2 = float(operand2)
    else:
        operand2 = int(operand2)
    if operand2 == 0:
        break

    operator = input('+ * / // ** 중 하나를 선택하세요 >>>')
    if operator == '+':
        print('{} + {} = {}'.format(operand1,operand2,operand1+operand2))
    elif operator == '-':
        print('{} - {} = {}'.format(operand1,operand2,operand1-operand2))
    elif operator == '*':
        print('{} * {} = {}'.format(operand1,operand2,operand1*operand2))
    elif operator == '/':
        print('{} / {} = {}'.format(operand1,operand2,operand1/operand2))
    elif operator == '//':
        print('{} // {} = {}'.format(operand1,operand2,operand1//operand2))
    else:
        print('{} ** {} = {}'.format(operand1,operand2,operand1**operand2))

print('프로그램이 종료되었습니다.')
```

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo7.py =====
>>>
첫번째 수를 입력하세요(종료(0)) >>>1
두번째 수를 입력하세요(종료(0)) >>>4.5
+ * / // ** 중 하나를 선택하세요 >>>+
1 + 4.5 = 5.5
첫번째 수를 입력하세요(종료(0)) >>>5
두번째 수를 입력하세요(종료(0)) >>>2
+ * / // ** 중 하나를 선택하세요 >>>/
5 / 2 = 2.5
첫번째 수를 입력하세요(종료(0)) >>>5
두번째 수를 입력하세요(종료(0)) >>>2
+ * / // ** 중 하나를 선택하세요 >>>/
5 / 2 = 2.5
첫번째 수를 입력하세요(종료(0)) >>>2.5
두번째 수를 입력하세요(종료(0)) >>>2
+ * / // ** 중 하나를 선택하세요 >>>*
2.5 * 2 = 5.0
첫번째 수를 입력하세요(종료(0)) >>>0
프로그램이 종료되었습니다.
>>>
```

내가 만든 문제2

-다음은 자판기의 기능을 수행하는 코드를 작성해보자 자판기에 0이 입력되기전 까지 주문을 계속 받아 현재의 누적된 가격을 출력하는 프로그램을 작성해보자.

소스코드:

```
message = ""주문하시려는 메뉴와 개수를 선택해주세요
```

```
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)\n>>> ""
```

```
total = 0
```

```
while True:
```

```
    print(message,end="")
```

```
    orderstr = input()
```

```
    if '0' in orderstr :
```

```
        order = orderstr
```

```
    else:
```

```
        order,cnt = orderstr.split()
```

```
        cnt = int(cnt)
```

```
        if order == '0':
```

```
            break
```

```
        else:
```

```
            if order=='1':
```

```
                total += 1000 * cnt
```

```
                print('콜라 {}개의 가격은 {}원 입니다.'.format(cnt,1000*cnt))
```

```
                print('현재 총 지불액은 {}원 입니다'.format(total))
```

```
            elif order=='2':
```

```
                total += 1100 * cnt
```



```
print('사이다 {}개의 가격은 {}원 입니다.'.format(cnt,1100*cnt))
```

```
print('현재 총 지불액은 {}원 입니다'.format(total))
```

```
elif order=='3':
```

```
total += 1500 * cnt
```

```
print('환타 {}개의 가격은 {}원 입니다.'.format(cnt,1500*cnt))
```

```
print('현재 총 지불액은 {}원 입니다'.format(total))
```

```
elif order=='4':
```

```
total += 2000 * cnt
```

```
print('게토레이 {}개의 가격은 {}원 입니다.'.format(cnt,2000*cnt))
```

```
print('현재 총 지불액은 {}원 입니다'.format(total))
```

```
else:
```

```
print('유효하지 않은 메뉴번호를 입력하셨습니다. 다시 주문하세요!')
```

```
print('주문이 종료되었습니다. 총 지불액은 {}원입니다'.format(total))
```

The screenshot displays a Python 3.8.1 IDE with a script for a menu-based ordering system. The script uses a loop to repeatedly prompt the user for an order number (1-4) and the quantity of items. It calculates the total cost based on the selected item and quantity, and prints the current total after each order. The script ends when the user enters 0 or an invalid number, printing the final total.

```
message = '''주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)\n>>>
total = 0

while True:
    print(message,end='')
    orderstr = input()
    if '0' in orderstr :
        order = orderstr
    else:
        order,cnt = orderstr.split()
        cnt = int(cnt)
    if order == '0':
        break
    else:
        if order=='1':
            total += 1000 * cnt
            print('콜라 {}개의 가격은 {}원 입니다.'.format(cnt,1000*cnt))
            print('현재 총 지불액은 {}원 입니다'.format(total))
        elif order=='2':
            total += 1100 * cnt
            print('사이다 {}개의 가격은 {}원 입니다.'.format(cnt,1100*cnt))
            print('현재 총 지불액은 {}원 입니다'.format(total))
        elif order=='3':
            total += 1500 * cnt
            print('환타 {}개의 가격은 {}원 입니다.'.format(cnt,1500*cnt))
            print('현재 총 지불액은 {}원 입니다'.format(total))
        elif order=='4':
            total += 2000 * cnt
            print('게토레이 {}개의 가격은 {}원 입니다.'.format(cnt,2000*cnt))
            print('현재 총 지불액은 {}원 입니다'.format(total))
        else:
            print('유효하지 않은 메뉴번호를 입력하셨습니다. 다시 주문하세요!')
print('주문이 종료되었습니다. 총 지불액은 {}원입니다'.format(total))
```

The Python 3.8.1 Shell window shows the execution of the script. It displays the initial message, the user's input (1 2), the calculated total (2000), and the prompt for the next order. The user enters 2 2, and the total becomes 4200. The user then enters 3 3, and the total becomes 4500. Finally, the user enters 0, and the script prints the final total of 18700.

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo8.py =====
주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)
>>> 1 2
콜라 2개의 가격은 2000원 입니다.
현재 총 지불액은 2000원 입니다
주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)
>>> 2 2
사이다 2개의 가격은 2200원 입니다.
현재 총 지불액은 4200원 입니다
주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)
>>> 3 3
환타 3개의 가격은 4500원 입니다.
현재 총 지불액은 8700원 입니다
주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)
>>> 4 5
게토레이 5개의 가격은 10000원 입니다.
현재 총 지불액은 18700원 입니다
주문하시려는 메뉴와 개수를 선택해주세요
콜라:1000원-(1) 사이다:1100원-(2) 환타:1500-(3) 게토레이:2000-(4) 종료-(0)
>>> 0
주문이 종료되었습니다. 총 지불액은 18700원입니다
>>>
```

Chapter03

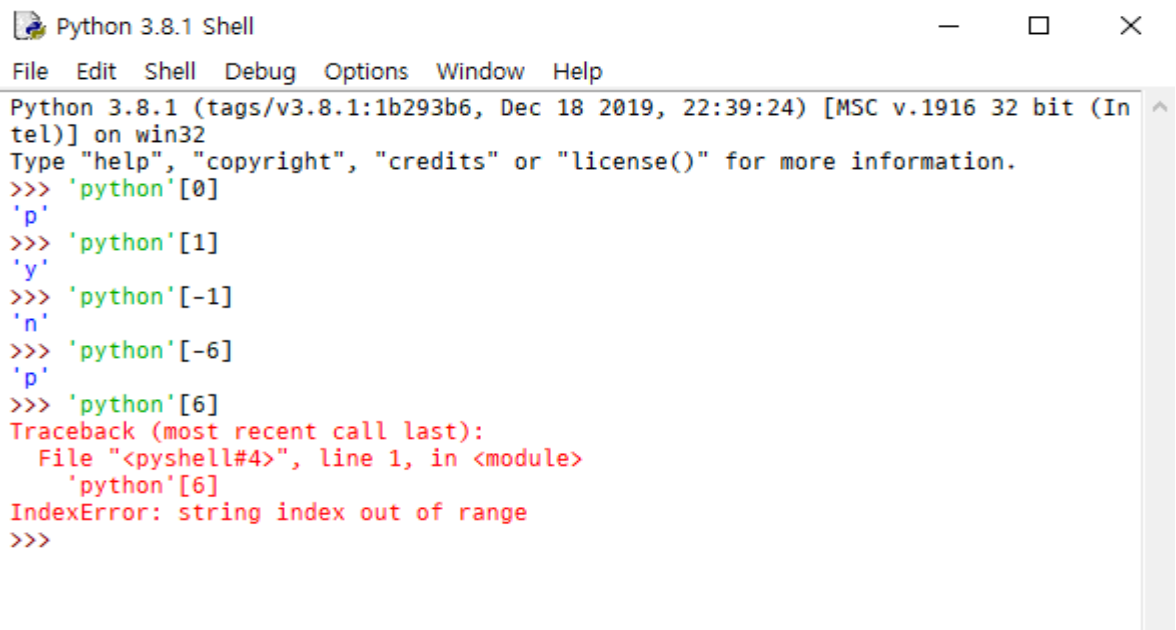
Section01

1.1 문자열 str 클래스와 부분 문자열 참조 슬라이싱

-함수 len(문자열)으로 문자열의 길이를 반환 받을 수 있다

Ex) len("hello") -> 5

-문자열을 구성하는 문자는 0부터 시작되는 첨자(index)를 대괄호 안에 기술해 참조(indexing) 가능하다. -1부터 시작돼 -2,-3으로 작아지는 첨자도 역순으로 참조가능. 첨자가 유효범위를 벗어날 경우 IndexError가 발생



```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'python'[0]
'p'
>>> 'python'[1]
'y'
>>> 'python'[-1]
'n'
>>> 'python'[-6]
'p'
>>> 'python'[6]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    'python'[6]
IndexError: string index out of range
>>>
```

파이썬 문자열 인덱스

0	1	2	3	4	5
p	y	t	h	o	n
-6	-5	-4	-3	-2	-1

1.2 문자열의 부분 문자열 참조 방식

-문자열에서 일부분을 참조하는 방법을 슬라이싱(slicing)이라고 한다.

-슬라이싱을 한다 해서 기존의 문자열이 변경되지 않고 참조하는 부분만 반환됨.

-슬라이싱은 콜론(:)을 사용해 [start:end]로 사용할 수 있다 이때 start에서 end-1까지의 인덱스

가 지정된다.

```
>>> 'python'[1:6]
'ython'
>>> 'python'[0:6]
'python'
>>> 'python'[2:4]
'th'
>>>
```

-슬라이스 [start:end]에서는 음수도 사용할 수 있다.

```
>>> 'python'[-6:-1]
'pytho'
>>> 'python'[-6:-2]
'pyth'
>>> 'python'[-5:-2]
'yth'
>>> |
```

-양수인덱스와 음수인덱스를 혼용하여 사용하여 슬라이싱을 할 수 있다.

```
>>> 'python'[0:-1]
'pytho'
>>> 'python'[1:-1]
'ytho'
>>> 'python'[0:-2]
'pyth'
>>> |
```

-슬라이싱에서 start와 end를 비울 수 있으며, 비우면 각각 '처음부터'와 '끝까지'를 의미한다 앞뒤를 모두 비우면 처음부터 끝까지 의미

```
>>> 'python'[0:]
'python'
>>> 'python'[:4]
'pyth'
>>> 'python'[:]
'python'
>>> |
```

-부분 반환 문자열에서 문자 사이의 간격을 step으로 조정하려면 str[start:end:step]을 사용하면 된다 step을 생략하면 기본적으로 1로 설정됨. 또한 간격인 step은 음수도 가능하며, 이 경우 start는 end보다 작아야 한다.

```
>>> 'python'[::-1]
'nohtyp'
>>> 'python'[1:5:2]
'yh'
>>> 'python'[1:5:3]
'yo'
>>> 'python'[::-1]
'nohtyp'
>>> 'python'[-1:-7:-1]
'nohtyp'
>>>
```

1.3 문자 함수와 이스케이프 시퀀스

-문자 함수 ord()와 chr()

ord()	문자의 유니코드 반환
chr()	유니코드 문자를 반환

```
>>> ord('A')
65
>>> chr(65)
'A'
>>> |
```

-자주 사용하는 이스케이프 시퀀스 문자

이스케이프 시퀀스 문자	설명
ww	역슬래시
w'	작은따옴표
w''	큰따옴표
wn	줄 개행
w\t	수평 탭

-내장 함수 max()와 min()으로 인자의 최대 최솟값을 구할 수 있다 인자가 문자열이면 유니코드 값이 가장 큰 문자가 반환되고 숫자이면 제일 큰 숫자가 반환된다

```
>>> max('abcd')
'd'
>>> min('abcd')
'a'
>>> max('12345')
'5'
>>> min('153', '10')
'10'
>>> |
```

Section02

2.1, 2.2, 2.3(통합)

-문자열 관련 메소드들

replace(a,b, count)	인자 a가 나타내는 부분을 모두 b로 바꿈 옵션인 count가 없으면 모두 바꾸고 있으면 앞에서부터 지정한 횟수만큼 바꿈
count('부분 문자열')	문자열에서 인자로 지정한 부분 문자열의 출현 횟수를 알려준다
join('문자열')	문자와 문자 사이에 원하는 문자열을 삽입
find('문자열')	문자열에서 인자로 지정한 문자열이 처음 등장하는 인 덱스를 반환 지정한 인자가 문자열에 없을 때 -1을 반 환
rfind('문자열')	인자로 지정한 문자열을 원래 문자열의 뒤에서부터 찾 아 인덱스를 반환
index('문자열')	문자열에서 인자로 지정한 문자열이 처음 등장하는 인 덱스를 반환 지정한 인자가 문자열에 없을 때 에러 발 생
rindex('문자열')	인자로 지정된 문자열을 원래 문자열의 뒤에서부터 찾 아 인덱스를 반환
split('구분자')	문자열을 구분자를 기준으로 여러 문자열로 나눔 인자 가 지정되지 않을 시 공백을 기준으로 문자열을 나눔
upper() ,lower()	upper()는 모든 문자를 대문자로,lower() 모든 문자를 소 문자로 반환
center(폭,'채움문자')	폭을 지정하고 중앙 정렬하며 나머지 공간에 두번째 인 자인 문자를 채운다
ljust(폭,'채움문자'),rjust(폭,'채움문자')	ljust()는 폭을 지정하고 문자열을 왼쪽에 정렬하며 나머 지 공간은 두번째 인자로 채운다, rjust()는 문자열을 오른쪽 정렬하고 나머지는 ljust()와 같다
strip()	문자열의 앞뒤 공백을 제거
zfill(폭)	문자열의 지정한 폭 앞 빈 부분에 0을 채워 넣음

-다양한 문자열 메소드를 이용해보기

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 'hello hi hello'.replace('hello','hi')
'hi hi hi'
>>> 'hello hi hello'.replace('hello','hi',1)
'hi hi hello'
>>> 'hello hi hello'.count('hello')
2
>>> 'abcde'.join('->')
'-abcde>'
>>> '->'.join('abcd')
'a->b->c->d'
>>> 'hello hi hello'.find('hi')
6
>>> 'hello hi hello'.find('hello')
0
>>> 'hello hi hello'.rfind('hello')
9
>>> 'ab'.find('c')
-1
>>> 'my name is jeawon'.index('name')
3
>>> 'good by'.rindex('b')
5
>>> 'nice to meet you'.split()
['nice', 'to', 'meet', 'you']
>>> '2020-05-21'.split('-')
['2020', '05', '21']
>>> 'abcd'.upper()
'ABCD'
>>> 'ABCD'.lower()
'abcd'
>>> '결과'.center(15,'*')
'*****결과*****'
>>> '결과'.ljust(15,'*')
'결과*****'
>>> '결과'.rjust(15,'*')
'*****결과'
>>> ' hello '.strip()
'hello'
>>> '5'.zfill(10)
'0000000005'
>>> |
```

(width, /) # '/' marks pre
Pad a numeric string with z

2.5 출력을 정형화하는 함수 format()

-파이썬은 문자열 메소드 str.format(인자들)을 사용해 문자열 str 중간중간에 변수나 상수를 함께 출력할 수 있다.

- 문자열 '{0}+{1}={2}' 내부에서 중괄호(curly brace){}가 위치한 부분에 format(3,4,3+4)의 인자인 3,4,7이 순서대로 출력됨
- 문자열에서 {}을 제외한 나머지는 '+ ='은 쓰여 있는 그대로 출력된다.

-{0}, {1}처럼 인자의 순서를 나타내는 정수를 0부터 삽입하면 인자의 순서와 출력 순서를 조정할 수 있다.

Ex) '{0} + {1} = {2}'.format(3,4,3+4) *인자의 순서에 따라 {n}에 대응

-{1:5d}처럼 중괄호의 내부에서 인자의 순번 다음 콜론 이후에 5d,10f등의 출력 폭과 출력 형식(d,f)등을 지정할 수 있다 . d로지정하면 정수 출력이고 f로 지정하면 실수 출력이다.

2.6 C언어의 포매팅 스타일인 %d와 %f등으로 출력

-C언어의 printf()에서 사용하는 형식 지정자 %d와 같은 형식도 지원함. 문자열 중간에 변수나 상수를 출력하는 부분을 %d,%x,%o,%f,%c,%s 등을 사용해 %로 이어지는 뒤의 상수나 변수를 순서대로 10진수,16진수,8진수,소수점,문자,문자열로 출력한다.

Ex) print('%d %x %o %f %c %s' % (10, 10, 10, 3.14, 'C', 'hello'))

Section03

3.1 논리 값과 논리 연산

- 파이썬은 논리 값으로 참과 거짓을 의미하는 True와False 키워드를 제공
- 논리 자료형은 <class 'bool'>로 제공한다.
- 참(True)는 1, 거짓(False)는 0으로 변환됨
- 정수 0, 실수 0.0, None, 빈문자열은 False이고 뭔가가 있는 문자열은 True이다
- 논리 연산자

and	피연산자 둘 다 참이어야 결과가 참 아니면 거짓
or	피연산자중 하나만 참이면 결과가 참 둘 다 거짓이면 결과는 거짓
not	논리값을 반전시킴 ex) not True는 False
^	피연산자의 논리값이 서로 다르면 참 같으면 거짓

- 관계 연산자는 수나 문자열 등의 크기 비교에 사용된다. 문자열을 비교할 때는 유니코드 값으로 비교

종 류	의 미
==	좌항과 우항의 값이 같다
!=	좌항과 우항의 값이 다르다
>, <	좌항 또는 우항이 크거나 작다
>=, <=	좌항 또는 우항이 크거나 같고 작거나 같다

-기본연습문제5

내가 소문자를 입력하면 대문자로 반환해 출력하고 대문자로 입력하면 소문자로 반환해 출력하는 소스코드를 작성해보자

소스코드:

```
str1 = input('대문자나 소문자로 영단어를 입력하세요>>> ')
```

```
print('변환결과'.center(30,'='))
```

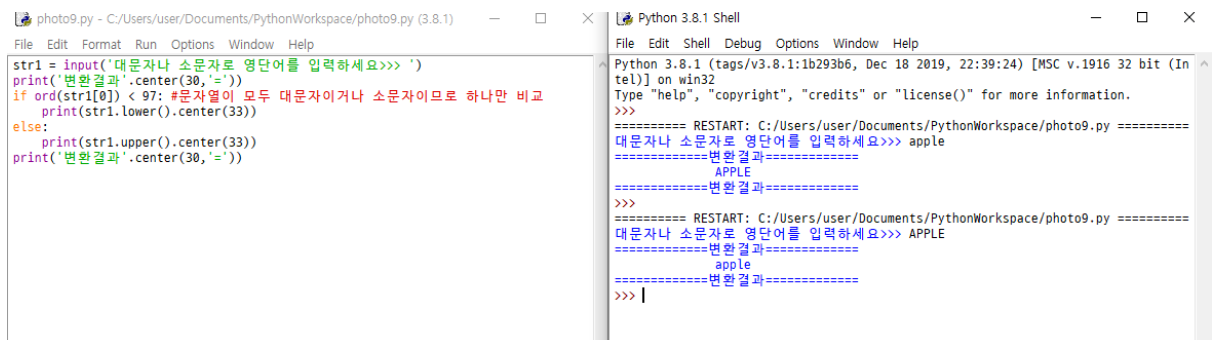
```
if ord(str1[0]) < 97: #문자열이 모두 대문자이거나 소문자이므로 하나만 비교
```

```
    print(str1.lower().center(33))
```

```
else:
```

```
    print(str1.upper().center(33))
```

```
print('변환결과'.center(30,'='))
```



The screenshot shows a Python IDE with two windows. The left window displays the source code for a program that takes a word as input and prints it in either lowercase or uppercase, centered. The right window shows the execution output, where the word 'apple' is entered, and the output shows 'apple' in lowercase, centered, with decorative lines.

```
photo9.py - C:/Users/user/Documents/PythonWorkspace/photo9.py (3.8.1)
File Edit Format Run Options Window Help
str1 = input('대문자나 소문자로 영단어를 입력하세요>>> ')
print('변환결과'.center(30,'='))
if ord(str1[0]) < 97: #문자열이 모두 대문자이거나 소문자이므로 하나만 비교
    print(str1.lower().center(33))
else:
    print(str1.upper().center(33))
print('변환결과'.center(30,'='))

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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo9.py =====
대문자나 소문자로 영단어를 입력하세요>>> apple
=====변환결과=====
apple
=====변환결과=====
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo9.py =====
대문자나 소문자로 영단어를 입력하세요>>> APPLE
=====변환결과=====
apple
=====변환결과=====
>>> |
```

-기본연습문제6

점수에 따라 학점을 부여하는 소스코드를 작성해보자 학점의 기준은 100점 만점에 90점이상이면 A, 80점이상이면 B, 70점이상이면 C, 60점 이상이면 D, 60점 미만은 F학점이다.

소스코드:

```
score = int(input('당신의 점수를 입력해주세요>>> '))
```

```
if score>=90:
```

```
    grade = 'A'
```

```
elif score>=80:
```

```
    grade = 'B'
```

```
elif score>=70:
```

```
    grade = 'C'
```

```
elif score>=60:
```

```
    grade = 'D'
```

```
else:
```

```
    grade = 'F'
```

```
print('당신의 학점은 {}입니다.'.format(grade))
```

The image shows a Python IDE window titled 'photo10.py' and a 'Python 3.8.1 Shell' window. The script in the IDE is as follows:

```

score = int(input('당신의 점수를 입력해주세요>>> '))

if score>=90:
    grade = 'A'
elif score>=80:
    grade = 'B'
elif score>=70:
    grade = 'C'
elif score>=60:
    grade = 'D'
else:
    grade = 'F'

print('당신의 학점은 {}입니다.'.format(grade))

```

The shell window shows the execution of the script with the following output:

```

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo10.py =====
당신의 점수를 입력해주세요>>> 95
당신의 학점은 A입니다.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo10.py =====
당신의 점수를 입력해주세요>>> 82
당신의 학점은 B입니다.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo10.py =====
당신의 점수를 입력해주세요>>> 64
당신의 학점은 D입니다.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo10.py =====
당신의 점수를 입력해주세요>>> 54
당신의 학점은 F입니다.
>>>

```

-비트 연산자에는 비트 논리곱 &, 비트 논리합 |, 비트 배타적 논리합 ^ 이 있다.

비트연산은 같은 비트의 자릿수끼리 이루어지며 논리곱 &은 두비트가 모두 1일때 1

논리합은 두비트 중 하나만 1이여도 1이며, 배타적 논리합 ^은 두비트가 서로 다를 때 1이다.

x (1 비트)	y (1 비트)	x & y (비트 AND)	x y (비트 OR)	x ^ y (비트 XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

-비트 쉬프트 연산자에는 <<,>>가 있으며 1<<2이면 1의 이진수 0001을 왼쪽으로 2칸 이동시키고 1>>2이면 1의 이진수 0001을 오른쪽으로 2칸이동시킨다.

$$1 \ll 2 == 0001(2) \ll 2 == 0100(2) == 4(10)$$

$$1 \gg 2 == 0001(2) \gg 2 == 0000(2) == 0(10)$$

-내가 만든 문제3

컴퓨터가 얻어온 1~100까지의 임의의 수를 맞추는 문제이다.

표준입력을 통해 수를 입력하고 입력한수가 임의의 수보다 작으면

보다 큰수를 입력하라는 메시지가 나오고 더 크면 더 큰수를 입력하라는

힌트 메시지가 출력된다. 맞추면 프로그램이 종료된다.

소스코드:

```
from random import randint
```

```
comnumber = randint(1,100)
```

```
while True:
```

```
    mynumber = int(input('정수를 입력해주세요>>> '))
```

```
    if mynumber == comnumber:
```

```
        print('정답입니다! 컴퓨터가 뽑은 수는 {}였습니다.'.format(comnumber))
```

```
        break
```

```
    elif mynumber < comnumber:
```

```
        print('더 큰수를 입력해주세요!')
```

```
    else:
```

```
        print('더 작은수를 입력해 주세요!')
```

```
print()
```

```
photo11.py - C:/Users/user/Documents/PythonWorkspace/photo11.py (3.8.1)
File Edit Format Run Options Window Help
from random import randint
comnumber = randint(1,100)

while True:
    mynumber = int(input('정수를 입력해주세요>>> '))
    if mynumber == comnumber:
        print('정답입니다! 컴퓨터가 뽑은 수는 {}였습니다.'.format(comnumber))
        break
    elif mynumber < comnumber:
        print('더 큰수를 입력해주세요!')
    else:
        print('더 작은수를 입력해 주세요!')
    print()

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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo11.py =====
정수를 입력해주세요>>> 50
더 큰수를 입력해주세요!

정수를 입력해주세요>>> 60
더 큰수를 입력해주세요!

정수를 입력해주세요>>> 70
더 큰수를 입력해주세요!

정수를 입력해주세요>>> 80
더 작은수를 입력해 주세요!

정수를 입력해주세요>>> 75
더 작은수를 입력해 주세요!

정수를 입력해주세요>>> 72
더 큰수를 입력해주세요!

정수를 입력해주세요>>> 73
정답입니다! 컴퓨터가 뽑은 수는 73였습니다.
>>>
```

Chapter04

Section01

1.1 조건의 논리 값에 따른 선택 if문

-조건에 따라 코드의 흐름을 분기할 때는 if문을 사용한다.

-if문 사용시 논리 표현식 이후에 반드시 콜론이 있어야함.

-콜론 이후 다음 줄부터 시작되는 블록은 반드시 들여쓰기(indentation)를 해야 함.

-기본연습문제6

놀이 기구를 탈 때 키를 검사하는 소스코드를 작성해보자.

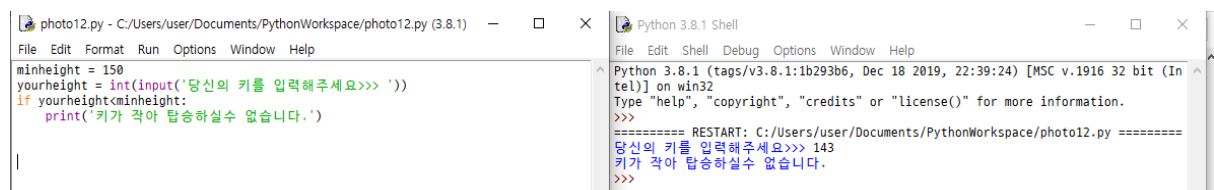
소스코드:

```
minheight = 150
```

```
yourheight = int(input('당신의 키를 입력해주세요>>> '))
```

```
if yourheight<minheight:
```

```
    print('키가 작아 탑승하실수 없습니다.')
```



1.2 조건에 따라 하나를 선택하는 if~else

-조건이 참과 거짓에 따라 사용하는 것이 if~else 문이다.

-else 뒤에는 콜론(:)을 붙이고 조건식은 적지 않는다.

-기본연습문제7

조건에 따라 홀수와 짝수를 판별해 보자.

소스코드:

```
n = int(input('정수 입력 >> '))
```

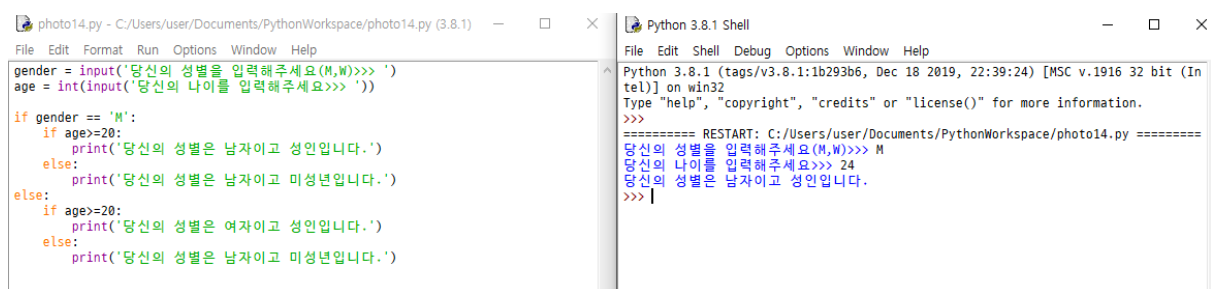
```
if n%2 == 0:
```

```
    print('{}은 짝수입니다.'.format(n))
```

```
else:
```

```
    print('{}은 홀수입니다.'.format(n))
```

-중첩 if문은 if문안에 if문이 또 들어가 있는 것이다. 어떠한 한 조건을 검사하고 또 어떠한 조건을 검사해야 할 때 사용한다.



```
photo14.py - C:/Users/user/Documents/PythonWorkspace/photo14.py (3.8.1)
File Edit Format Run Options Window Help
gender = input('당신의 성별을 입력해주세요(M,W)>>> ')
age = int(input('당신의 나이를 입력해주세요>>> '))

if gender == 'M':
    if age>=20:
        print('당신의 성별은 남자이고 성인입니다.')
    else:
        print('당신의 성별은 남자이고 미성년입니다.')
else:
    if age>=20:
        print('당신의 성별은 여자이고 성인입니다.')
    else:
        print('당신의 성별은 여자이고 미성년입니다.')

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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo14.py =====
당신의 성별을 입력해주세요(M,W)>>> M
당신의 나이를 입력해주세요>>> 24
당신의 성별은 남자이고 성인입니다.
>>>
```


Section02

-2.1 시퀀스 내부 값으로 반복을 실행하는 for문

-for i in 시퀀스: 형태로 사용하며 i에는 시퀀스의 항목이 순차대로 대입됨

-for문 이후에 있는 else문은 반복을 다 끝내고 실행되는 문장임

-기본연습문제8

반복문을 이용하여 1부터 100까지의 합을 구해보자

소스코드:

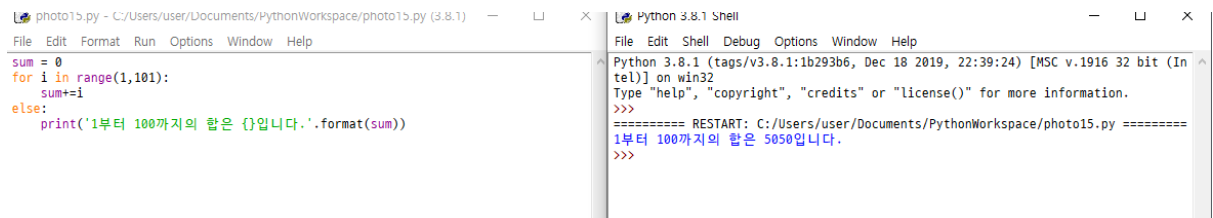
```
sum = 0

for i in range(1,101):

    sum+=i

else:

    print('1부터 100까지의 합은 {}입니다.'.format(sum))
```



-range(start,end,step)은 start~end-1까지의 수의 시퀀스를 구해주고 step은 간격을 설정

2.2 횟수를 정하지 않은 반복에 적합한 while 반복

-while문은 반복횟수가 정해지지 않았을 때 사용하기 적합하다.

-while문 다음의 else문은 while문이 종료되었을 때 실행된다.

-while 조건식: 과 같은 형태로 사용

-기본연습문제8

놀이공원에 4인승 놀이기구가 있고 이 놀이기구는 키 160이상만 탑승가능하다. 놀이기구의 인원이 다 차면 종료하는 코드를 구현해보자.

소스코드:

```
more = True
minheight = 160
cnt = 0
while more:
    height = int(input('당신의 키를 입력해주세요>>> '))
    if height >= minheight:
        cnt += 1
        print('탑승 가능합니다. 탑승해주세요~')
        print('현재 탑승인원 {}명, 잔여석 {}석'.format(cnt, 4-cnt))
    else:
        print('죄송합니다. 탑승하실수 없습니다.')

    if cnt == 4:
        more = False
    print()
print('만석입니다. 다음번에 이용해 주세요')
```

```
photo16.py - C:/Users/user/Documents/PythonWorkspace/photo16.py (3.8.1)
File Edit Format Run Options Window Help

more = True
minheight = 160
cnt = 0
while more:
    height = int(input('당신의 키를 입력해주세요>>> '))
    if height>=minheight:
        cnt+=1
        print('합승 가능합니다. 탑승해주세요~')
        print('현재 탑승인원 {}명, 잔여석 {}'.format(cnt,4-cnt))
    else:
        print('죄송합니다. 탑승하실수 없습니다.')

    if cnt==4:
        more = False
        print()
print('만석입니다. 다음번에 이용해 주세요')
```

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo16.py =====
>>>
당신의 키를 입력해주세요>>> 172
합승 가능합니다. 탑승해주세요~
현재 탑승인원 1명, 잔여석 3석

당신의 키를 입력해주세요>>> 156
죄송합니다. 탑승하실수 없습니다.

당신의 키를 입력해주세요>>> 180
합승 가능합니다. 탑승해주세요~
현재 탑승인원 2명, 잔여석 2석

당신의 키를 입력해주세요>>> 135
죄송합니다. 탑승하실수 없습니다.

당신의 키를 입력해주세요>>> 187
합승 가능합니다. 탑승해주세요~
현재 탑승인원 3명, 잔여석 1석

당신의 키를 입력해주세요>>> 173
합승 가능합니다. 탑승해주세요~
현재 탑승인원 4명, 잔여석 0석

만석입니다. 다음번에 이용해 주세요
>>>
```

-기본연습문제9 :중첩된 반복문을 이용하여 구구단 출력하기

소스코드:

```
for i in range(2,10):
```

```
    for j in range(1,10):
```

```
        print('%d * %d = %2d' % (i,j,i*j), end=' ')
```

```
    print()
```

```
photo17.py - C:/Users/user/Documents/PythonWorkspace/photo17.py (3.8.1)
File Edit Format Run Options Window Help

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

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo17.py =====
>>>
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
>>>
```

Section03

3.1 임의의 수인 난수 발생과 반복에 활용

- 모듈 random의 randint(시작,끝)을 이용하여 난수를 발생시킬수있다.
- 로또 번호나 임의의 수를 이용하여 반복문에 사용하는데 적합.
- 모듈을 사용하기 위해 from random import randint 기술

3.2 반복을 제어하는 break문 continue문

- 반복문이 무한 루프라면 특정한 조건을 만족하면 break문을 통해 빠져나와야 한다.
즉 break문은 반복문을 빠져나올 때 사용한다
- break문에 의해 반복문을 빠져나오면 else문은 실행되지 않는다.
- continue문은 반복문에서 특정한 조건을 만족하면 continue문 아래의 문장들은 실행하지 않고 다시 반복조건으로 돌아간다.

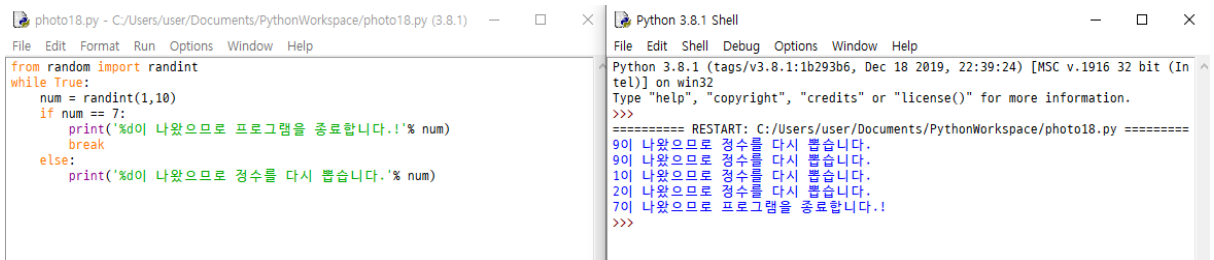
-기본연습문제10

모듈 random을 사용하여 1~10사이에서 정수 7이 나올 때까지 계속 반복하기

소스코드:

```
from random import randint

while True:
    num = randint(1,10)
    if num == 7:
        print('%d이 나왔으므로 프로그램을 종료합니다.!' % num)
        break
    else:
        print('%d이 나왔으므로 정수를 다시 뽑습니다.' % num)
```

The screenshot shows a Python IDE with two windows. The left window, titled 'photo18.py', contains the following code:

```
from random import randint
while True:
    num = randint(1,10)
    if num == 7:
        print('%d이 나왔으므로 프로그램을 종료합니다.' % num)
        break
    else:
        print('%d이 나왔으므로 경수를 다시 뽑습니다.' % num)
```

The right window, titled 'Python 3.8.1 Shell', shows the execution output:

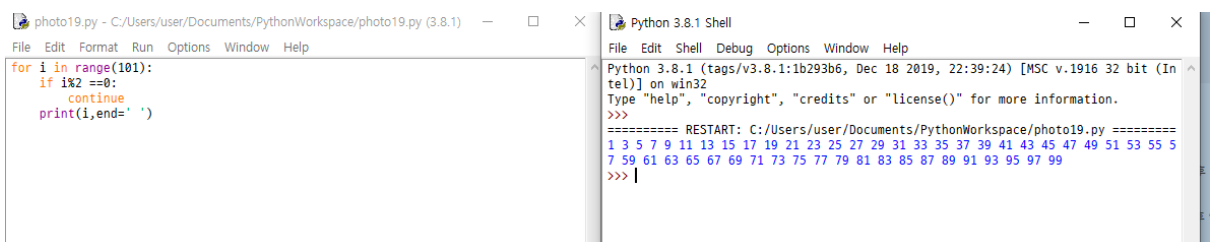
```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo18.py =====
9이 나왔으므로 경수를 다시 뽑습니다.
9이 나왔으므로 경수를 다시 뽑습니다.
10이 나왔으므로 경수를 다시 뽑습니다.
20이 나왔으므로 경수를 다시 뽑습니다.
70이 나왔으므로 프로그램을 종료합니다.!
>>>
```

-기본연습문제11

1부터 100까지의 수중에 홀수만 출력하는 소스코드를 작성해보자

소스코드:

```
for i in range(101):
    if i%2 ==0:
        continue
    print(i,end=' ')
```



The screenshot shows a Python IDE with two windows. The left window, titled 'photo19.py', contains the following code:

```
for i in range(101):
    if i%2 ==0:
        continue
    print(i,end=' ')
```

The right window, titled 'Python 3.8.1 Shell', shows the execution output:

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo19.py =====
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 5
7 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
>>> |
```

-내가 만든 문제4

이번에는 3,6,9게임을 해보자 1~40까지의 수중에서 3,6,9가 한번 들어가면

‘박수짝!’, 2번들어가면 ‘박수짝짝!’ 을 출력해보자.

소스코드:

```
for i in range(1,41):
```

```
    cnt = 0
```

```
    if (i%10 == 3 or i%10 == 6 or i%10==9) and i != 0:
```

```
        cnt+=1
```

```
    if (i//10 == 3 or i//10 == 6 or i//10 == 9) and i != 0:
```

```
        cnt+=1
```

```
    if cnt == 1:
```

```
        print('{} 박수짝!'.format(i))
```

```
    elif cnt == 2:
```

```
        print('{} 박수짝짝!'.format(i))
```

```
    else:
```

```
        print(i)
```

```
photo20.py - C:\Users\user\Documents\PythonWorkspace\photo20.py (...)
```

```
File Edit Format Run Options Window Help
```

```
for i in range(1,41):
    cnt = 0

    if (i%10 == 3 or i%10 == 6 or i%10==9) and i != 0:
        cnt+=1
    if (i//10 == 3 or i//10 == 6 or i//10 == 9) and i != 0:
        cnt+=1

    if cnt == 1:
        print('{} 박수짹!'.format(i))
    elif cnt == 2:
        print('{} 박수짹짹!'.format(i))
    else:
        print(i)
```

```
>>>
===== RESTART: C:\Users\user\Documents\PythonWorkspace\photo20.py =====
1
2
3 박수짹!
4
5
6 박수짹!
7
8
9 박수짹!
10
11
12
13 박수짹!
14
15
16 박수짹!
17
18
19 박수짹!
20
21
22
23 박수짹!
24
25
26 박수짹!
27
28
29 박수짹!
30 박수짹!
31 박수짹!
32 박수짹!
33 박수짹짹!
34 박수짹!
35 박수짹!
36 박수짹짹!
37 박수짹!
38 박수짹!
39 박수짹짹!
40
>>>
```


Chapter05

Section01

1.1 리스트의 개념과 생성

-리스트란 여러 항목(item)을 하나의 단위로 묶어 손쉽게 사용하는 복합(compound) 자료형을 여러 개 제공하는 자료형이다.

-리스트는 대괄호([],),내장함수 list()를 사용하여 생성 가능

Ex) list_1 = [1,2,3,4,5] list_1 = list([1,2,3,4,5])

-빈리스트를 생성할 때는 [],list()로 빈리스트를 생성 가능

1.2 리스트의 항목 참조

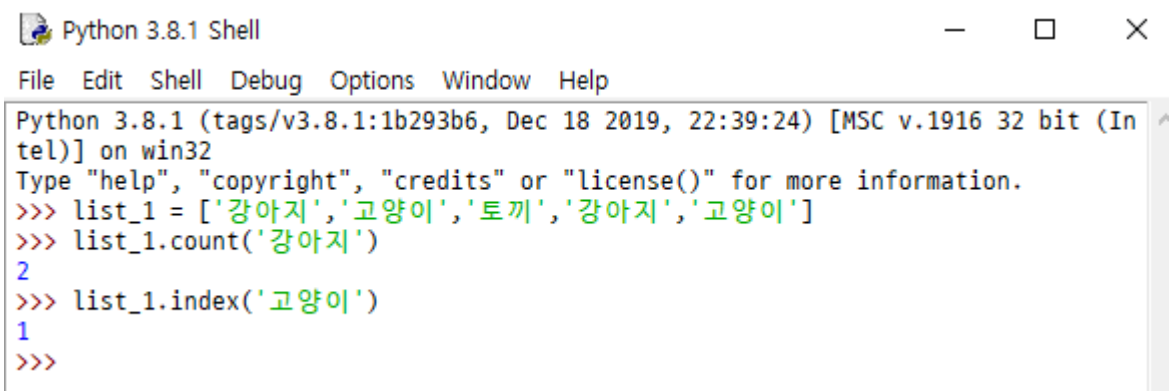
-리스트는 문자열과 같이 항목을 참조할 때 첨자(index)를 사용해 항목을 참조할 수 있음

-첫 요소는 0부터 시작하며 순차적으로 1씩 증가함 반대로 역순으로 참조할 때는 -1부터 시작하여 1씩 감소함

1.3 리스트의 항목 수정

-리스트는 메소드 count()와 index()를 제공

- count(값) : 값을 갖는 항목의 수를 반환
- index(값) : 값의 항목이 위치한 첨자를 반환 동일한 값이 있을 경우 첫 번째로 나오는 첨자를 반환

A screenshot of a Python 3.8.1 Shell window. The window title is "Python 3.8.1 Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The shell prompt shows the Python version and build information: "Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32". It also displays a message: "Type 'help', 'copyright', 'credits' or 'license()' for more information." The user has entered the following commands and received the corresponding output:

```
>>> list_1 = ['강아지', '고양이', '토끼', '강아지', '고양이']
>>> list_1.count('강아지')
2
>>> list_1.index('고양이')
1
>>>
```


-리스트의 첨자를 이용한 항목을 대입 연산자의 오른쪽에 위치시켜 리스트의 항목을 수정할 수 있다.

```
>>> print(list_1)
['강아지', '고양이', '토끼', '곰', '호랑이']
>>> list_1[0] = '다람쥐'
>>> list_1[4] = '코끼리'
>>> print(list_1)
['다람쥐', '고양이', '토끼', '곰', '코끼리']
>>> |
```

1.4 리스트 내부에 다시 리스트를 포함하는 중첩 리스트

-리스트의 항목으로 리스트가 올 수 있다.

Ex) list_1 = [['강아지', '고양이', '호랑이'], '까치', '개구리']

-리스트의 내부의 항목이 리스트를 포함할 경우 이중 for문으로 항목에 접근가능

```
>>> list_1 = [['사자', '코끼리'], ['독수리', '참새'], ['강아지', '호랑이']]
>>> print(list_1)
[['사자', '코끼리'], ['독수리', '참새'], ['강아지', '호랑이']]
>>> for i in list_1:
    for j in i:
        print(j)
    print()
```

사자
코끼리

독수리
참새

강아지
호랑이

>>>

2.1 리스트의 부분 참조인 슬라이싱

-리스트는 문자열처럼 부분 참조인 슬라이싱을 지원한다.

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> hangle = list('가나다라마바사')
>>> print(hangle)
['가', '나', '다', '라', '마', '바', '사']
>>> hangle[0:8]
['가', '나', '다', '라', '마', '바', '사']
>>> hangle[2:5]
['다', '라', '마']
>>> hangle[::-1]
['사', '바', '마', '라', '다', '나', '가']
>>> hangle[3:]
['라', '마', '바', '사']
>>>
```

2.2 리스트의 부분 수정

-리스트의 일부분을 다른 리스트로 수정하려면 슬라이스 방식에 대입 해야함.

-리스트의 일부분을 대체할 때는 리스트 형식으로 대입해야 한다.

```
>>> list_1 = ['호랑이', '곰', '토끼', '고양이', '강아지']
>>> print(list_1)
['호랑이', '곰', '토끼', '고양이', '강아지']
>>> list_1[0:3] = ['사자', '호랑이']
>>> print(list_1)
['사자', '호랑이', '고양이', '강아지']
>>> |
```

2.3 리스트의 항목 삽입과 삭제

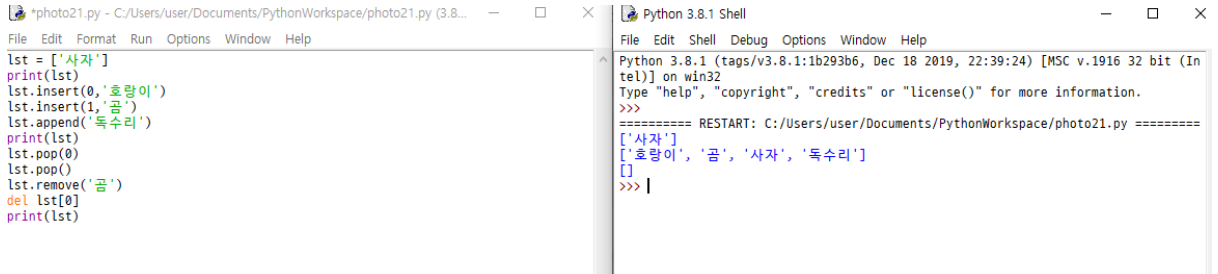
-리스트의 첨자 위치에 항목을 삽입하려면, 리스트 insert(항목,첨자)를 이용한다.

-리스트에서 하나의 항목을 삭제하려면 remove(값) 또는 pop(첨자),pop() 사용

-pop()과 같이 인자가 없을 경우에는 마지막 항목이 삭제되고 삭제된 항목 반환.

-del 문장과 리스트의 첨자를 이용하여 항목을 삭제할 수 있다.

-del 리스트명 과 같이 리스트명만 기술하면 리스트 자체가 메모리에서 삭제된다.



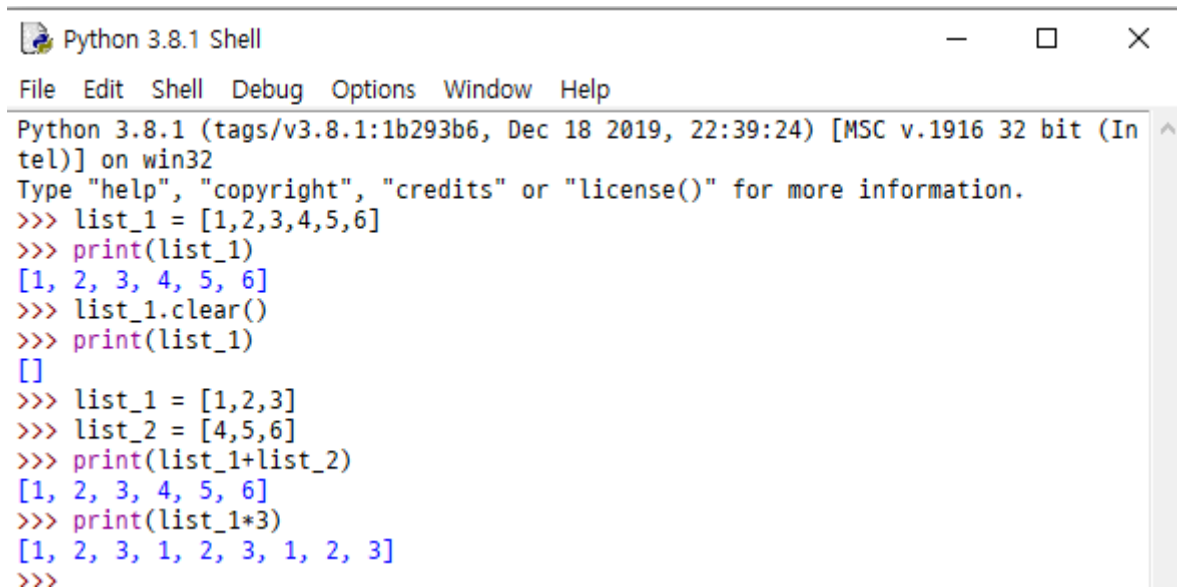
```
File Edit Format Run Options Window Help
lst = ['사자']
print(lst)
lst.insert(0, '호랑이')
lst.insert(1, '곰')
lst.append('독수리')
print(lst)
lst.pop(0)
print(lst)
lst.remove('곰')
del lst[0]
print(lst)
```

-리스트의 모든 항목을 삭제할 때는 clear()메소드를 이용

-리스트와 리스트를 연결할 때는 extend(리스트)를 이용(리스트 원본이 변한다)

-리스트와 리스트를 연결할 때는 +로도 가능하다(리스트 원본이 변하지 않음)

-리스트와 정수와 * 연산을 하면 항목이 지정된 정수만큼 반복됨



```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> list_1 = [1,2,3,4,5,6]
>>> print(list_1)
[1, 2, 3, 4, 5, 6]
>>> list_1.clear()
>>> print(list_1)
[]
>>> list_1 = [1,2,3]
>>> list_2 = [4,5,6]
>>> print(list_1+list_2)
[1, 2, 3, 4, 5, 6]
>>> print(list_1*3)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>>
```

2.5 리스트 항목의 순서와 정렬

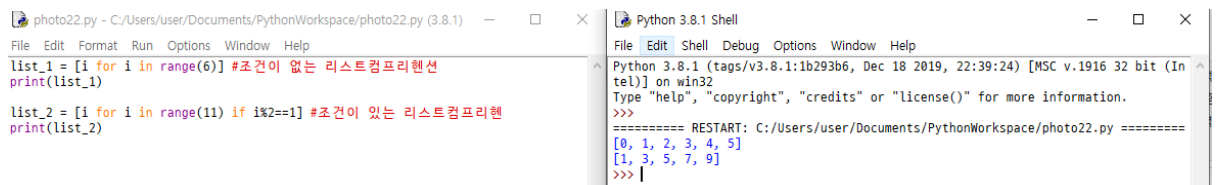
- 리스트의 항목을 정렬할 때는 내장함수 `sorted()`나 리스트의 메소드 `sort()`를 이용
- `sorted(리스트, reverse=True)`, `sort(reverse=True)`일 경우 내림차순으로 정렬
- 리스트를 역순으로 출력할 때는 리스트의 메소드 `reverse()`를 이용한다.

```
>>> list_1 = [1,2,3,4,5,6,7,8,9,10]
>>> list_1.reverse()
>>> print(list_1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> list_1.sort()
>>> print(list_1)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list_1.sort(reverse=True)
>>> print(list_1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>>
```

```
>>> list_1 = list('가다나마라사바')
>>> print(sorted(list_1))
['가', '나', '다', '라', '마', '바', '사']
>>> print(sorted(list_1,reverse=True))
['사', '바', '마', '라', '다', '나', '가']
>>> |
```

2.6 리스트 컴프리헨션

- 리스트 컴프리헨션은 리스트를 만드는 간결한 방법을 제공한다.리스트 컴프리헨션은 리스트를 만들므로 대괄호([])로 감싸고, 구성 항목인 `i`를 가장 앞에 배치하며, `for`문이 오는 마지막에 콜론(:)을 생략한다.



2.7 리스트 대입과 복사

-리스트에서 대입 연산자(=)은 얇은 복사(shallow copy)라고 해서 대입되는 변수가 동일한 시퀀스를 가리킨다.

Visualize Python, Java, JavaScript x Live Programming Mode - Python x +

python tutor.com/live.html#mode=edit

This mode is experimental. Use the [regular Python Tutor](#) to get live help and use more features.

Write code in Python 3.6 (drag lower right corner to resize code editor)

```
1 list1 = [1,2,3,4,5]
2 list2 = list1
```

Frames

Global frame

- list1
- list2

Objects

list

0	1	2	3	4	5
1	2	3	4	5	

Done running (2 steps)

-리스트에서 깊은 복사(deep copy)를 하려면 슬라이스[:]나 copy() 또는 list()함수를 이용

Visualize Python, Java, JavaScript x Live Programming Mode - Python x +

python tutor.com/live.html#mode=edit

This mode is experimental. Use the [regular Python Tutor](#) to get live help and use more features.

Write code in Python 3.6 (drag lower right corner to resize code editor)

```
1 list1 = [1,2,3,4,5]
2 list2 = list1[:]
3 list3 = list1.copy()
4 list4 = list(list1)
```

Frames

Global frame

- list1
- list2
- list3
- list4

Objects

list

0	1	2	3	4	5
1	2	3	4	5	

list

0	1	2	3	4	5
1	2	3	4	5	

list

0	1	2	3	4	5
1	2	3	4	5	

list

0	1	2	3	4	5
1	2	3	4	5	

Done running (4 steps)

-문장 is 는 피연산자인 변수 2 개가 같은 메모리를 공유하는지 검사한다.

같으면 True 다르면 False를 반환

Section 03

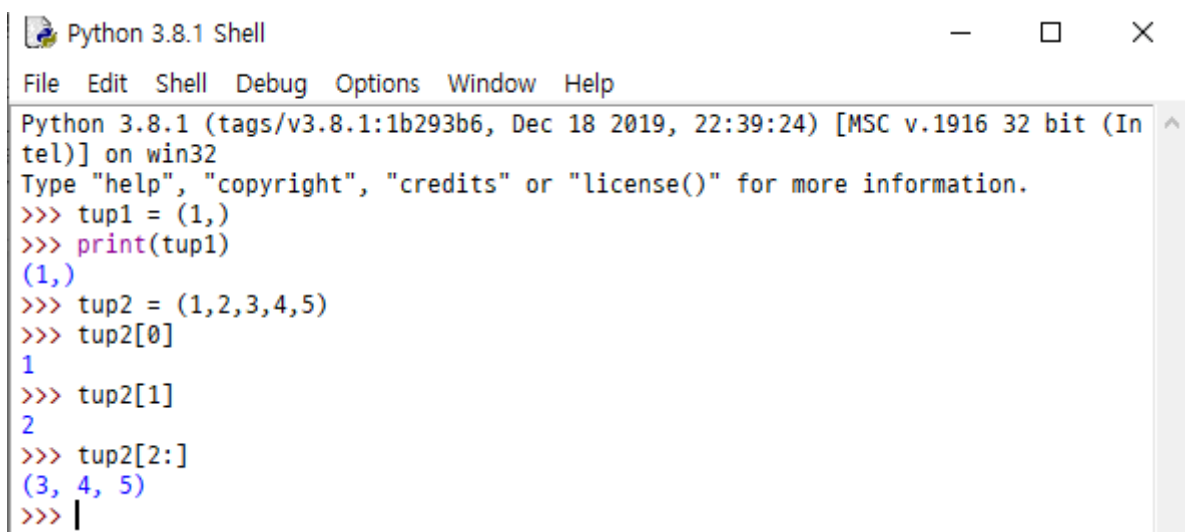
3.1 괄호로 정의하는 시퀀스 튜플

-튜플을 정의할 때는 ()에 항목들을 나열하여 생성 항목이 1개일 경우 항목뒤에 콤마(,)를 꼭 기술해야 한다.

-튜플은 리스트와 달리 수정하거나 변경할 수 없다.

-튜플은 리스트와 같이 첨자로 참조할 수 있다.

-튜플은 수정 불가능하므로 첨자와 슬라이스로 수정할 수 없다.



```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> tup1 = (1,)
>>> print(tup1)
(1,)
>>> tup2 = (1,2,3,4,5)
>>> tup2[0]
1
>>> tup2[1]
2
>>> tup2[2:]
(3, 4, 5)
>>> |
```

-튜플도 리스트와 같이 +,* 연산자를 동일한 기능으로 사용가능

-sorted()를 이용하여 리스트와 같이 정렬이 가능

-del 튜플이름 을 이용하여 튜플 자체를 메모리에서 삭제할 수 있음

-내가 만든 문제5

리스트에 1부터100까지 5개의 임의의 난수를 저장하고 홀수인 항목만 저장하는 프로그램을 만들어 보자.

소스코드:

```
from random import randint
```

```
list_1 = list()
```

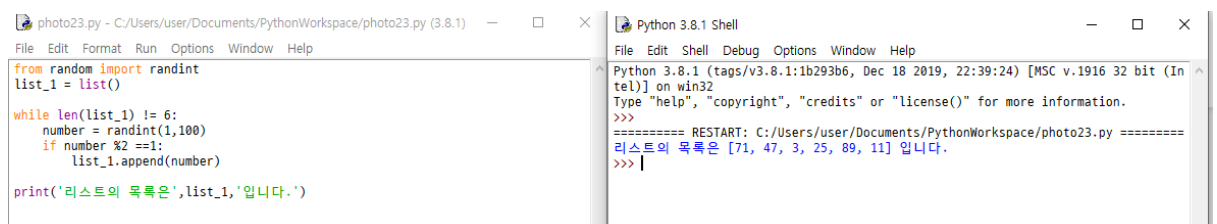
```
while len(list_1) != 6:
```

```
    number = randint(1,100)
```

```
    if number %2 ==1:
```

```
        list_1.append(number)
```

```
print('리스트의 목록은',list_1,'입니다.')
```



```
photo23.py - C:/Users/user/Documents/PythonWorkspace/photo23.py (3.8.1)
File Edit Format Run Options Window Help
from random import randint
list_1 = list()

while len(list_1) != 6:
    number = randint(1,100)
    if number %2 ==1:
        list_1.append(number)

print('리스트의 목록은',list_1,'입니다.')
```

```
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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo23.py =====
리스트의 목록은 [71, 47, 3, 25, 89, 11] 입니다.
>>> |
```

-내가 만든 문제6

국어,수학,영어,과학의 성적을 입력하고 그에 해당하는 등급을 출력해보자

소스코드:

```
subject = ('국어','수학','영어','과학')
grade = list()
for i in subject:
    score = int(input(i+' 성적을 입력해 주세요>>> '))
    if score>=90:
        grade.append('A')
    elif score>=80:
        grade.append('B')
    elif score>=70:
        grade.append('C')
    elif score>=60:
        grade.append('D')
    else:
        grade.append('F')
cnt = 0
for i in subject:
    print('{} : {}'.format(i,grade[cnt]))
    cnt +=1
```

```
photo24.py - C:/Users/user/Documents/PythonWorkspace/photo24.py (3.8.1)
File Edit Format Run Options Window Help
subject = ('국어', '수학', '영어', '과학')
grade = list()
for i in subject:
    score = int(input(i+' 성적을 입력해 주세요>>> '))
    if score>=90:
        grade.append('A')
    elif score>=80:
        grade.append('B')
    elif score>=70:
        grade.append('C')
    elif score>=60:
        grade.append('D')
    else:
        grade.append('F')
cnt = 0
for i in subject:
    print('{ } : {}'.format(i, grade[cnt]))
    cnt +=1

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 (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo24.py =====
국어 성적을 입력해 주세요>>> 99
수학 성적을 입력해 주세요>>> 85
영어 성적을 입력해 주세요>>> 65
과학 성적을 입력해 주세요>>> 75
국어 : A
수학 : B
영어 : D
과학 : C
>>>
```

Chapter06

Section01

1.1 딕셔너리의 개념과 생성

- 딕셔너리는 키와 값의 쌍인 항목을 나열한 시퀀스이다.
- 딕셔너리를 생성할 때는 dict_1 = {키:값} 으로 생성
- lect = {}, lect = dict()로 빈 딕셔너리를 생성할 수 있다.
- 딕셔너리의 항목 값으로 리스트나 튜플 등이 가능하다.
- 딕셔너리를 참조할 때는 lect[키]로 참조할 수 있다.

1.2 다양한 인자의 함수 dict()로 생성하는 딕셔너리

- 내장 함수 dict()함수에서 인자로 리스트[키,값]나 튜플로(키,값)로 딕셔너리 생성 가능.

```
>>> lect = dict(['봄', 'spring'], ['여름', 'summer'], ['가을', 'fall'], ['겨울', 'winter'])
>>> print(lect)
{'봄': 'spring', '여름': 'summer', '가을': 'fall', '겨울': 'winter'}
>>> |
```

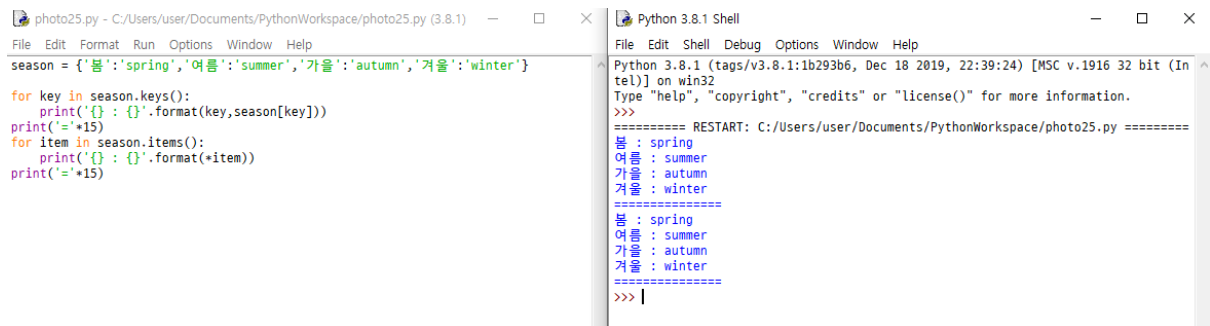
- 키가 단순 문자열이면 dict(봄='spring')와 같이 딕셔너리 생성가능

1.3 딕셔너리 키는 수정 불가능한 객체로 사용

- 딕셔너리의 키는 수정 불가능한 객체는 모두 가능하고 정수나 실수도 가능.
- 튜플은 딕셔너리의 키로 사용될 수 있으나 수정 가능한 리스트는 키로 사용할 수 없음.

1.4 딕셔너리의 항목 순회

- 딕셔너리의 메소드 keys()는 키로만 구성된 리스트를 반환.
- 딕셔너리의 메소드 items()는 (키,값) 쌍의 튜플이 들어있는 리스트를 반환
- 딕셔너리의 메소드 values()는 값으로 구성된 리스트를 반환.



The image shows a Python IDE with two windows. The left window, titled 'photo25.py', contains the following code:

```
season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}

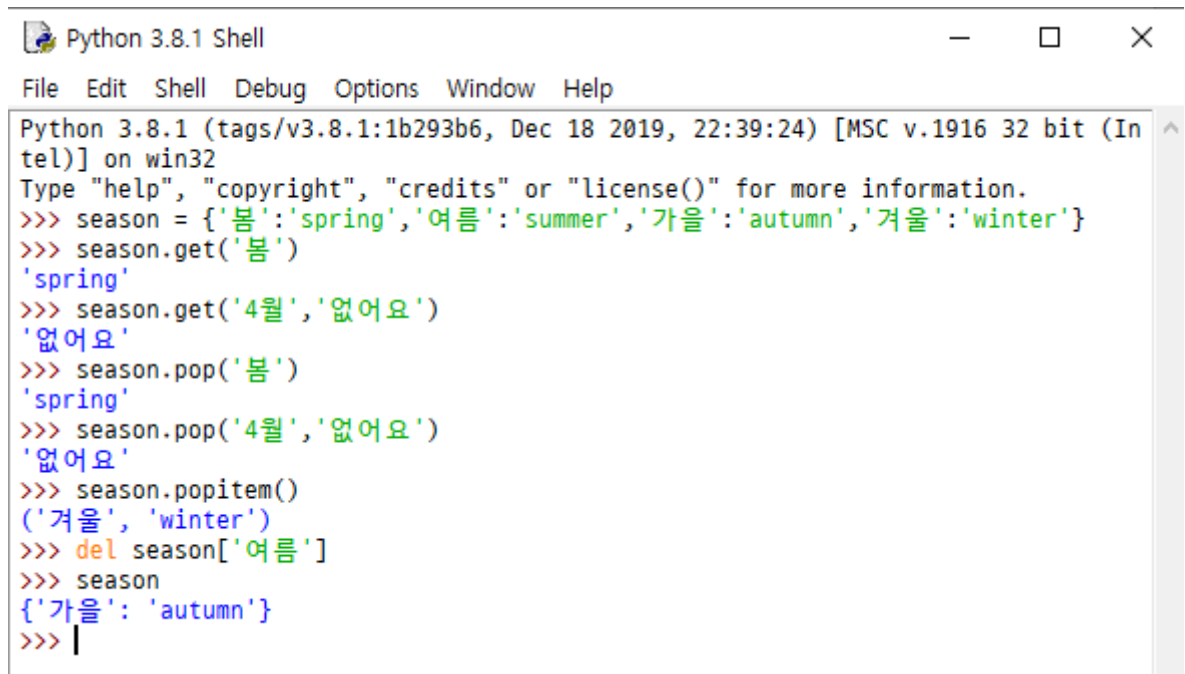
for key in season.keys():
    print('{} : {}'.format(key, season[key]))
print('='*15)
for item in season.items():
    print('{} : {}'.format(*item))
print('='*15)
```

The right window, titled 'Python 3.8.1 Shell', shows the output of the script:

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo25.py =====
봄 : spring
여름 : summer
가을 : autumn
겨울 : winter
=====
봄 : spring
여름 : summer
가을 : autumn
겨울 : winter
=====
>>> |
```

1.5 딕셔너리 항목의 참조와 삭제

- 딕셔너리의 메소드 `get(키)`는 키의 해당 값을 반환한다.
- `get(키, 문자열)`은 해당하는 키가 없을 경우 문자열을 반환
- `pop(키, 문자열)`는 키인 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.
- 만일 해당하는 키가 없을 경우 문자열이 반환된다.
- `popitem()`은 임의의 (키, 값)의 튜플을 반환하고 삭제한다.
- 문장 `del`에 이어 키로 지정하면 해당 항목이 삭제된다.



The image shows a 'Python 3.8.1 Shell' window with the following interactive session:

```
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
>>> season.get('봄')
'spring'
>>> season.get('4월', '없어요')
'없어요'
>>> season.pop('봄')
'spring'
>>> season.pop('4월', '없어요')
'없어요'
>>> season.popitem()
('겨울', 'winter')
>>> del season['여름']
>>> season
{'가을': 'autumn'}
>>> |
```

1.6 딕셔너리 항목 전체 삭제와 변수 제거

-딕셔너리 메소드 `clear()`는 기존의 모든 키:값 항목을 삭제한다.

-`del` 문장 후에 딕셔너리 변수 자체를 기술하면 변수 자체가 메모리에서 제거된다.

```
>>> season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
>>> print(season)
{'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
>>> season.clear()
>>> print(season)
{}
>>> del season
>>> print(season)
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    print(season)
NameError: name 'season' is not defined
>>> |
```

1.7 딕셔너리 결합과 키의 멤버십 검사 연산자 `in`

-딕셔너리의 메소드 `update(딕셔너리)`는 인자인 다른 딕셔너리를 합병한다.

-인자 딕셔너리에 원 딕셔너리와 동일한 키가 있다면 인자 딕셔너리의 값으로 대체된다.

-문장 `in`으로 딕셔너리의 키가 존재하는지 확인할 수 있다. 키가 있으면 `True` 없으면

`False`를 반환한다. `not in` 도 가능하다.

```
>>> season = {'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter'}
>>> season.update({'1월': 'January', '2월': 'February'})
>>> print(season)
{'봄': 'spring', '여름': 'summer', '가을': 'autumn', '겨울': 'winter', '1월': 'January', '2월': 'February'}
>>> '봄' in season
True
>>> |
```

Section2

2.1 수학에서 배운 집합을 처리하는 자료형

-집합은 중복되는 요소가 없으며, 순서도 없는 원소의 모임이다.

-파이썬에서 집합은 수학과 같이 원소를 콤마로 구분하며 중괄호로 둘러싸 표현한다.

2.2 내장 함수 set()을 활용한 집합 생성

-집합은 내장 함수 set()으로 생성할 수 있다.

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

-함수 set()에서 인자는 리스트와 튜플, 문자열처럼 반복적이면 가능함.

그러나 리스트나 튜플의 항목은 변할 수 없는 것이어야 하며, 항목이

리스트나 딕셔너리처럼 가변적인 것은 허용되지 않는다.

```
>>> set1 = set()
>>> print(set1)
set()
>>> set2 = set((1,2,3,4,5))
>>> print(set2)
{1, 2, 3, 4, 5}
>>> set3 = set([1,2,3,4,5])
>>> print(set3)
{1, 2, 3, 4, 5}
>>> |
```

2.3 중괄호로 직접 원소를 나열해 집합 생성

-집합을 생성하는 다른 방법은 중괄호 {} 안에 직접 원소를 콤마로 구분해 나열하는 방법이다. 집합의 원소는 문자,문자열,숫자,튜플 등과 같이 변할 수 없는 것이어야 한다.

```
>>> set1 = {1,2,3,'a',(1,2)}
>>> print(set1)
{1, 2, 3, (1, 2), 'a'}
>>> |
```

2.4 집합의 원소 추가와 삭제

-원소를 추가할 때는 add(원소)를 사용한다.

-원소를 삭제할 때는 remove(원소), discard('원소')를 사용

-임의의 원소를 삭제할 때는 pop()을 사용

-모든 원소를 삭제할 때는 clear()메소드를 사용

```
>>> set1 = set()
>>> set1.add(1)
>>> set1.add(2)
>>> set1.add(3)
>>> set1.add(4)
>>> print(set1)
{1, 2, 3, 4}
>>> set1.remove(2)
>>> print(set1)
{1, 3, 4}
>>> set1.discard(4)
>>> print(set1)
{1, 3}
>>> set1.pop()
1
>>> print(set1)
{3}
>>> set1.clear()
>>> print(set1)
set()
>>>
```


2.5 집합의 주요 연산인 합집합,교집합,차집합,여집합

■ 합집합 연산자 |와 메소드 union,update()

-합집합은 연산자 |와 메소드 union()을 사용.메소드 union()은 합집합을 반환하며 원본 자체가 수정되지 않는다.

- update()메소드는 원본을 변경한다.

```
>>> set1 = {1,2,3,4}
>>> set2 = {5,6,7,8}
>>> print(set1 | set2)
{1, 2, 3, 4, 5, 6, 7, 8}
>>> set1.union(set2)
{1, 2, 3, 4, 5, 6, 7, 8}
>>> set1.update(set2)
>>> print(set1)
{1, 2, 3, 4, 5, 6, 7, 8}
>>>
```

■ 교집합 연산자 &와 메소드 intersection()

-교집합은 연산자 &와 메소드 intersection()을 사용한다.

-메소드 intersection()은 원본을 변경하지 않지만 intersection_update()는 원본을 수정한다.

```
>>> set1 = {1,2,3,4}
>>> set2 = {5,6,2,1}
>>> set1 & set2
{1, 2}
>>> set1.intersection(set2)
{1, 2}
>>> set1.intersection_update(set2)
>>> print(set1)
{1, 2}
>>>
```

■ 차집합 연산자 -와 메소드 difference()

-차집합은 연산자 -와 메소드 difference()를 사용한다. 피연산자의 순서에 따라 결과가 달라지므로 교환 법칙이 성립하지 않는다.

```
>>> set1 = {1,2,3,4}
>>> set2 = {3,4,5,6}
>>> set1-set2
{1, 2}
>>> set1.difference(set2)
{1, 2}
>>> set1.difference_update(set2)
>>> print(set1)
{1, 2}
>>> |
```

- 여집합 연산자 ^와 메소드 symmetric_difference()

-여집합은 연산자 ^와 메소드 symmetric_difference() 를 사용한다.

```
>>> set1={1,2,3,4,5}
>>> set2={4,5,6,7,8}
>>> set1 ^ set2
{1, 2, 3, 6, 7, 8}
>>> set1.symmetric_difference(set2)
{1, 2, 3, 6, 7, 8}
>>> set1.symmetric_difference_update(set2)
>>> print(set1)
{1, 2, 3, 6, 7, 8}
>>> |
```

- 집합연산자는 복합대입연산자와 같이 |=,&=,-=,^= 같은 연산자로 사용가능

2.6 함수 len()과 소속 연산 in

-len() 함수로 집합의 원소 개수를 알 수 있다

-소속 연산자 in으로 집합의 원소를 확인할 수 있다.

```
>>> set1={1,2,3,4,5}
>>> len(set1)
5
>>> 3 in set1
True
>>> 6 in set1
False
>>>
```

Section03

3.1 내장 함수 zip()

-내장 함수 zip()을 이용하면 몇 개의 리스트나 튜플의 항목으로 조합된 튜플을 만들 수 있다. zip()은 동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어 주는 역할을 하는 함수이다.

-zip()의 결과를 내장 함수 list()의 인자로 사용하면 항목이 튜플인 리스트가 만들어 진다.

-zip()의 결과를 내장 함수 tuple()의 인자로 사용하면 항목이 튜플인 튜플이 만들어 진다.

-zip()의 인자는 2개 이상 올 수 있다. zip()의 인자가 3개이면 항목이 3개인 튜플이 구성 된다.

-zip()의 인자들의 길이가 같지 않더라도 짧은 쪽의 인자에 대응시키고 나머지는 버린다.

```
>>> a = ['FTP', 'telnet', 'SMTP', 'DNS']
>>> b = (20, 23, 25, 33)
>>> list(zip(a, b))
[('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 33)]
>>> tuple(zip(a, b))
(('FTP', 20), ('telnet', 23), ('SMTP', 25), ('DNS', 33))
>>>
```

-내장 함수 dict()에서 zip()의 인자 2개를 사용하면 앞은 키, 뒤는 값의 조합이 구성돼

딕셔너리를 생성할 수 있다.

```
>>> a = ['FTP', 'telnet', 'SMTP', 'DNS']
>>> b = (20, 23, 25, 33)
>>> dict(zip(a, b))
{'FTP': 20, 'telnet': 23, 'SMTP': 25, 'DNS': 33}
>>> dict(zip('abcd', '123'))
{'a': '1', 'b': '2', 'c': '3'}
>>> |
```

3.2 내장 함수 enumerate()

-내장 함수 enumerate(시퀀스)는 0부터 시작하는 첨자와 항목 값의 튜플 리스트를 생성한다.

-키워드 인자 start를 사용해 enumerate(시퀀스,start = 1)로 호출하면 시작 첨자를 1로 지정할 수 있다. 키워드 start는 생략할 수 있으면 생략하면 기본값은 0이다.

```
>>> list(enumerate('abcd'))
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
>>> list(enumerate('abcd',1))
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
>>> |
```

Ln: 142 Col: 4

3.3 시퀀스 간의 변환

-내장 함수 list(),tuple(),set(),dict() 등을 사용하면 문자열을 비롯한 시퀀스를 간편하게 변환할 수 있다.

-딕셔너리를 변환시 키만 항목의 값으로 갖는다.

```
>>> day = tuple('월화수목금토일')
>>> print(day)
('월', '화', '수', '목', '금', '토', '일')
>>> print(list(day))
['월', '화', '수', '목', '금', '토', '일']
>>> day = list('월화수목금토일')
>>> print(day)
['월', '화', '수', '목', '금', '토', '일']
>>> print(tuple(day))
('월', '화', '수', '목', '금', '토', '일')
>>> day = {'월요일':'mon','화요일':'tue','수요일':'wed','목요일':'thur','금요일':'fri'}
>>> lday = list(day)
>>> print(lday)
['월요일', '화요일', '수요일', '목요일', '금요일']
```

-내가 만든 문제7

일주일의 요일과 그에 대응하는 영어단어를 딕셔너리에 저장한 후
검색을 해서 영어단어를 알 수 있는 사전을 만들어 보자.

소스코드:

```
day_dict = {'월요일':'Monday','화요일':'Tuesday','수요일':'Wednesday','목요일':  
'Thursday','금요일':'Friday','토요일':'Saturday','일요일':'Sunday'}
```

```
while True:
```

```
    day = input('궁금한 요일을 입력해주세요(종료:0) >>>')
```

```
    if day == '0':
```

```
        break
```

```
    elif day in day_dict:
```

```
        print('{}은 영어로 {}입니다.'.format(day,day_dict[day]))
```

```
    else:
```

```
        print('죄송합니다. 입력하신 단어는 사전에 등록되어 있지 않습니  
다.')
```

```
print('사전검색 기능을 종료합니다.')
```

```
===== RESTART: C:/Users/user/Documents/PythonWorkspace/photo26.py =====  
궁금한 요일을 입력해주세요(종료:0) >>>월요일  
월요일은 영어로 Monday입니다.  
궁금한 요일을 입력해주세요(종료:0) >>>수요일  
수요일은 영어로 Wednesday입니다.  
궁금한 요일을 입력해주세요(종료:0) >>>화  
죄송합니다. 입력하신 단어는 사전에 등록되어 있지 않습니다.  
궁금한 요일을 입력해주세요(종료:0) >>>금요일  
금요일은 영어로 Friday입니다.  
궁금한 요일을 입력해주세요(종료:0) >>>일  
죄송합니다. 입력하신 단어는 사전에 등록되어 있지 않습니다.  
궁금한 요일을 입력해주세요(종료:0) >>>일요일  
일요일은 영어로 Sunday입니다.  
궁금한 요일을 입력해주세요(종료:0) >>>0  
사전검색 기능을 종료합니다.  
>>>
```

레포트를 마치면서

중간고사 대체 리포트를 쓰면서 지금까지 배운 내용들을 한 번 더 복습할 수 있는 정말 좋은 기회가 되었다. 어떠한 일이 모두 그렇듯이 기억에 잘 남는 건 여러 번 보고 여러번 반복하는 것만큼 기억에 잘 남는 건 없는 것 같다 중간고사 대신에 이 리포트를 쓰지만 이전의 내용들을 상기해보고 다시 내용을 정리하니 전보다 더 선명히 머릿속에 기억되는 것 같다

파이썬에서는 특히 메서드들이 많아 금방금방 까먹고 헛갈리는 것도 있었는데 이번 리포트를 작성하면서 다시 상기시키는 좋은 기회가 되었다. 솔직히 정리할 양이 많아 리포트를 쓰는 데는 좀 힘이 들었지만 결국에는 내 것이 된다는 생각에 뿌듯한 마음이 든다. 살면서 리포트를 쓴 경험이 그렇게 많지 않은데 이번 기회에 경험을 할 수 있어서 더욱 좋은 것 같다 다만 아쉬운 점은 워드를 작성하는 법에 서툴러 디자인 측면에서는 많이 서투른 것 같다 또한 리포트를 쓴 경험이 많지 않아 과연 남이 봐도 보기 좋은 리포트인지 걱정이 든다. 하지만 처음부터 잘 하고 싶다는 마음은 욕심인 것 같다. 이것도 경험이니 다음번에는 지금보다 더 나은 리포트를 쓸 수 있을 것이다. 지금까지 리포트를 작성했지만 나에게 너무 좋은 경험을 할 수 있었고 또한 파이썬에 대하여 좀 더 익숙해지고 좀 더 관심을 갖게 된 계기도 된 거 같다