



파이썬

포트폴리오

담당교수	강 환 수 교수님
학 과	컴퓨터정보공학과
학 번	20191790
이 름	김 유 진

목차

강의계획서.....	3
머리말	7
1장. 파이썬 개요	8
2장. 파이썬 기초 다지기.....	9
3장. 문자열과 논리 연산.....	16
4장. 조건과 반복	27
5장. 리스트와 튜플	36
6장. 딕셔너리와 집합.....	47
마무리	58

강의계획서



동양미래대학교
DONGYANG MIRAE UNIVERSITY

강 의 계 획 서

아시아 직업교육 허브대학

2020 학년도 1학기	전공	컴퓨터정보공학과	학부	컴퓨터공학부
과 목 명	파이썬프로그래밍(2019009-PD)			
강의실 과 강의시간	수:6(3-217),7(3-217),8(3-217)		학점	3
교과분류	이론/실습		시수	3
담당 교수	강환수 + 연구실 : 2호관-706 + 전 화 : 02-2610-1941 + E-MAIL : hskang@dongyang.ac.kr + 면담가능기간 : 화요일 13~16			
학과 교육목표				
과목 개요	2010년 이후 파이썬의 폭발적인 인기는 제4차 산업혁명 시대의 도래와도 밀접한 연관성이 있다. 컴퓨팅 사고력은 누구나가 가져야할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제4차 산업혁명 시대의 기술을 이끌고 있다. 제4차 산업혁명 시대를 주도하는 핵심 기술은 데이터과학과 머신러닝, 딥러닝이며, 이러한 분야에 적합한 언어인 파이썬은 매우중요한 언어가 되었다. 본 교과목은 파이썬 프로그래밍의 기초적이고 체계적인 학습을 수행한다. 본 교과목을 통하여 데이터 처리 방법에 대한 효율적인 파이썬 프로그래밍 방법을 학습한다.			
학습목표 및 성취수준	1. 컴퓨팅 사고력의 중요성을 인지하고 4차 산업혁명에서 파이썬 언어의 필요성을 이해할 수 있다. 2. 기본적인 파이썬 문법을 이해하고 데이터 처리를 위한 자료구조를 이해하여 적용할 수 있다. 3. 문제 해결 방법을 위한 알고리즘을 이해하고 데이터 처리에 적용 할 수 있다. 4. 파이썬 프로그램을 이용하여 실무적인 코딩 작업을 할 수 있다.			
	도서명	저자	출판사	비고
주교재	파이썬으로 배우는 누구나 코딩	강환수, 신용현	홍릉과학출판사	
수업시 사용도구	파이썬 기본 도구, 파이참, 아나콘다와 주피터 노트북			
평가방법	중간고사 30%, 기말고사 40%, 과제를 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	장애학생을 위한 별도의 수강 지원을 받을 수 있습니다. 언어가 문제가 되는 학생은 글로 된 과제 안내, 확대문자 시험지 제공 등의 지원을 드립니다.

머리말

프로그래밍 언어를 배우는 이유

- 우리의 머릿속에는 많은 개념들이 정의되어 있다. 이 개념들을 실제로 구현하기 위해 물리적으로 매핑하는 과정을 ‘알고리즘’이라고 하는데, 알고리즘을 통해 프로그래밍 문제를 논리적으로 분석하고 구현하는 능력이 향상된다.
- 4차 산업 혁명으로 IT 발전이 더욱 중요해지면서, 프로그래밍 언어 습득은 거의 필수적이라고 할 수 있을 정도가 되었다. 언어는 시간이 지남에 따라 발전하거나 쇠퇴하므로 한 언어를 영원히 사용할 수 없다. 상황에 따라 필요한 언어도 각각 다양하므로, 언어들의 기본적인 개념을 알고 있다면 다양한 언어를 알고 학습하는데 더 도움이 된다.

프로그래밍 입문에 적합한 파이썬

- 파이썬(Python)은 배우기 쉽고, 사용하기 쉽다. 많은 대학의 교양 과목으로 자리잡을 만큼 프로그래밍 교육용 언어로도 인정받고 있다.
또한 실무에도 사용하기 좋아 다양한 분야에서 활용 지분을 차지하고 있다.
프로그래밍의 기초를 배운 뒤 다른 분야로 나아갈 때에도 계속 활용이 가능하다.

1장. 파이썬 개요

■ 파이썬(Python) 살펴보기

파이썬은 네덜란드 개발자 귀도 반 로섬(Guido van Rossum)이 개발한 언어로, 누구나 무료로 사용할 수 있는 오픈 소스 프로그래밍 언어이다.

‘파이썬(Python)’이란 영어의 의미는 그리스 신화에 나오는 뱀 이름이다.

파이썬 로고에 두 개의 뱀이 서로 마주본 듯한 그림이 있는 이유도 이 때문이다.

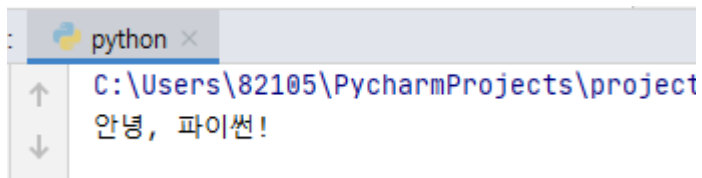
현재 파이썬은 글로벌 기업부터 스타트업까지 다양하게 안정적으로 활용되고 있다.



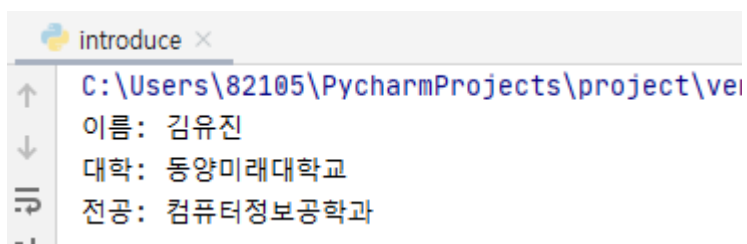
파이썬 로고 <출처 : 파이썬 공식 홈페이지>

■ 프로그래밍 예제

```
1      # 첫 파이썬 프로그램
2      print('안녕, 파이썬!')
```



```
1      # 자신을 소개하는 프로그램
2      print('이름: 김유진')
3      print('대학: 동양미래대학교')
4      print('전공: 컴퓨터정보공학과')
```



2장. 파이썬 기초 다지기

■ 문자열과 수

1-1. 문자열

문자열이란 문자의 모임이라고 할 수 있다. 1장에서 `print('안녕, 파이썬!')`라는 코드를 작성했는데, 여기서 '안녕, 파이썬!'이 바로 문자열이다.

텍스트는 작은따옴표 또는 큰따옴표로 감싸 표현해주면 된다.

문자열에 관해서는 3장에서 더 자세히 다뤄보도록 하자.

1-2. 문자열의 연결과 반복

사칙연산 연산자 중 덧셈 연산자와 곱셈 연산자를 문자열에 적용할 수 있다.

덧셈 연산자는 문자열을 연결하고, 곱셈 연산자는 문자열 반복을 수행한다.

```
>>> '파이썬' + '프로그래밍'
'파이썬프로그래밍'
>>> '안녕' * 3
'안녕안녕안녕'
```

1-3. 정수와 실수의 이해

파이썬에서 수를 나타내는 유형에는 정수, 실수, 복소수 등 여러 가지가 있다.

이들은 모두 수에 속하지만, 표현할 수 있는 수의 종류와 범위가 다르다.

하나씩 살펴보도록 하자.

정수(integer)는 자연수에 음양 부호(+, -)가 붙은 것이다.

비트 하나로 부호를 표현하고, 여러 개의 비트 묶음으로 자연수를 표현한다.

정수는 8진수나 16진수로도 표현이 가능하다.

8진수는 0o로 시작하고, 16진수는 0x로 시작한다.

```
>>> 0o20          # 8진수 20
63
>>> 0xff          # 16진수 ff
255
```

실수(float)는 3.14, -2.5처럼 소수점이 있는 수를 말한다.

파이썬에서는 부동소수점 수(floating point number, float)라는 데이터 유형으로

실수를 다룬다. 부동소수점 수는 ‘소수점이 떠다니는 실수’를 의미한다.

1-4. 수의 연산

파이썬 셸은 일종의 간단한 계산기처럼 사용할 수 있다.

파이썬 코드로 다양한 계산식을 표현하면, 컴퓨터는 이 식을 그대로 계산해준다.

파이썬에는 사칙연산을 비롯해 여러 가지 산술 연산자가 있다.

연산자	의미	우선순위
+	덧셈(add)	4, 2
-	뺄셈(subtract)	4, 2
*	곱셈(multiply)	3
/	나눗셈(divide)	3
%	나머지(modulus)	3
//	몫(floor division)	3
**	거듭제곱(exponent)	1

익숙하지 않은 몫, 거듭제곱을 구하는 연산자가 생소할 것이다.

어떻게 동작하는지 파이썬 셸에서 확인해보자.

```
>>> 2 ** 3          # 2의 3승
8
>>> 100 / 3         # 소수점 아래까지 구해준다
33.33333333333333
>>> 100 // 3        # 소수점 아래는 버린다
33
```

지금 소개한 연산자는 모두 수의 계산의 관련된 산술 연산자이다.

산술 연산자 외에도 변수에 값을 대입하는 대입 연산자, 값을 비교하는

비교 연산자, 조건식을 연결하는 논리 연산자 등이 있다.

이들에 대해서도 곧 배워보도록 하자.

■ 변수와 대입연산자

2-1. 자료형과 데이터 유형 확인하기

지금까지 알아본 정수와 실수, 문자열 등을 자료형이라고 한다.

데이터 유형을 직접 확인해 볼 때는 `type()` 함수를 사용하면 된다.

이 함수는 매개변수로 전달된 데이터의 유형을 반환해 준다.

대화형 모드에서 여러 데이터 유형을 확인해 보자.

```
>>> type(10)           # 정수
<class 'int'>
>>> type(1.0)          # 실수
<class 'float'>
>>> type(5 + 6j)        # 복소수
<class 'complex'>
>>> type('python')     # 문자열
<class 'str'>
```

2-2. 변수와 대입연산자

변수(variable)는 프로그램을 작성할 때 필요한 데이터를 저장할 수 있는 그릇과 같은 개념으로, 담아둔 값을 바꿀 수 있으며 필요할 때마다 가져와 사용할 수 있다.

`x = 4`와 같이, 변수에 값을 저장하기 위해서는 대입 연산자(=)가 필요하다.

대입 연산자를 기준으로 왼쪽에 변수, 오른쪽에 값을 넣어주면 된다.

`x, y, z = 4`와 같이 여러 개의 변수에 같은 값을 대입하는 것도 가능하다.

파이썬에서 변수의 이름을 붙일 때는 몇 가지의 규칙이 있다.

변수의 이름이 기본으로 제공되는 여러 대상(키워드)과 겹치는 것을 방지하기 위해서이다. 지금부터 변수 이름을 붙일 때의 규칙을 알아보자.

1. 숫자, 알파벳, `_` 등으로 구성되며, 대소문자가 구별된다.
2. 연산자 기호와 공백은 사용할 수 없다.
3. 숫자는 맨 앞에 올 수 없다. 이름과 수를 구별하기 위해서이다.
4. `if`, `import`, `True`와 같은 키워드는 사용할 수 없다.

변수에 완전히 새로운 값을 대입하는 것이 아니라,

변수에 현재 대입되어 있는 값을 기준으로 하여 수정할 때도 있다.

예를 들어, 변수 x에 대입된 값을 1만큼 증가시키는 것이다.

이것은 $x = x + 1$ 로 표현할 수 있고 간단히 $x += 1$ 로도 표현이 가능하다.

프로그래밍을 하다 보면 변수의 값을 수정해야 하는 일이 빈번하게 발생하기

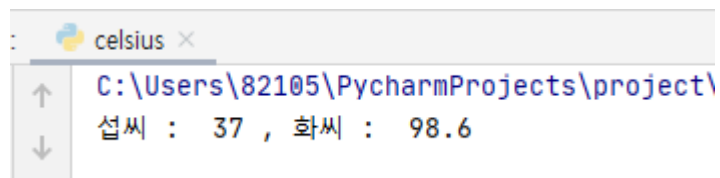
때문에, 복합 대입 연산자라는 편리한 연산자가 제공된다.

연산자	의미
=	오른쪽 값을 왼쪽 변수에 저장
+=	수를 더한 후 다시 대입
-=	수를 뺀 후 다시 대입
*=	수를 곱한 후 다시 대입
/=	수를 나눈 후 다시 대입
%=	수를 나눈 후 나머지 대입
//=	수를 나눈 후 몫 대입
**=	수를 거듭제곱한 후 다시 대입

2-3. 프로그래밍 연습

섭씨(celsius) 온도를 화씨(fahrenheit) 온도로 변환하기

```
1 celsius = 37
2 fahrenheit = celsius * 9 / 5 + 32
3 print('섭씨 : ', celsius, ' , 화씨 : ', fahrenheit)
```



■ 표준 입력과 자료 변환 함수

3-1. 함수 input()

프로그램 작성 과정에서 사용자의 입력을 받아 처리하는 경우가 많다.

프로그램 과정에서 셸이나 콘솔창을 통해 사용자의 입력을 받아 처리하는 방식을 표준 입력(standard input)이라고 한다.

input() 함수를 사용하면 사용자에게서 문자열 데이터를 입력 받을 수 있다.

입력 받은 데이터를 화면에 출력하려면, name = input()과 같이 변수를 사용하여 입력한 값을 저장해 두어야 한다.

input() 함수를 통해 받은 데이터는 문자열 형태이다. 하지만 문자열이 아닌 수를 입력 받기 위해서는 조금 다른 방법이 필요하다. 코드를 통해 알아보자.

```
>>> text = input()          # 문자열 입력
10
>>> text                    # '로 둘러싸여 있다.
'10'
>>> num = int(input())      # 정수 입력
10
>>> num                     # '로 둘러싸여 있지 않다.
10
```

입력 받은 문자열 데이터를 수의 형태로 사용하기 위해서는,

int()함수를 사용하여 정수로 변환해야 한다.

1. 문자열 입력 받기 : 변수 = input()
2. 정수 입력 받기 : 변수 = int(input())
3. 실수 입력 받기 : 변수 = float(input())

■ 프로그래밍 예제

```
1      # 킬로미터 단위 거리를 마일 단위로 변환해 출력하는 프로그램
2      speed = int(input('차의 속도를 입력(km) >> '))
3      print(speed, '(km)은', speed / 1.61, '마일(miles)이다.')
```

inputex ×
C:\Users\82105\PycharmProjects\project\venv\Scripts\python.exe
↑ 차의 속도를 입력(km) >> 135
↓ 135 (km)은 83.85093167701862 마일(miles)이다.
⏏

```
1      # 지구와 원의 차이를 알아보는 프로그램
2      earth = 40120
3      print('알려진 지구 둘레 : ', earth)
4      circle = 2 * 3.141592 * 6378.1
5      print('지구와 같은 원둘레 : ', circle)
6      print('차이 : ', earth - circle, '(km)')
```

earth ×
C:\Users\82105\PycharmProjects\project\venv\Scripts\python.exe
↑ 알려진 지구 둘레 : 40120
↓ 지구와 같은 원둘레 : 40074.77587040001
⏏ 차이 : 45.22412959999201 (km)

```

1      # 연산 수행 후 결과를 출력하는 프로그램
2      num1 = int(input('Enter First number : '))
3      num2 = int(input('Enter Second number : '))
4      print(num1, '/', num2, '==>', num1 / num2)
5      print(num1, '%', num2, '==>', num1 % num2)
6      print(num1, '//', num2, '==>', num1 // num2)
7      print(num1, '**', num2, '==>', num1 ** num2)

```

operation ×

C:\Users\82105\PycharmProjects\project\venv\python.exe

Enter First number : 12

Enter Second number : 5

12 / 5 ==> 2.4

12 % 5 ==> 2

12 // 5 ==> 2

12 ** 5 ==> 248832

```

1      # 네 자릿수 정수를 역순 출력하는 프로그램
2      num = input('네 자릿수 정수 입력 >> ')
3      print(num[::-1])

```

reverse ×

C:\Users\82105\PycharmProjects\project\python.exe

네 자릿수 정수 입력 >> 5432

2345

3장. 문자열과 논리 연산

■ 문자열 클래스와 슬라이싱

1-1. 문자열이란?

파이썬은 텍스트 데이터를 취급하기 위해 문자열(string)이라는 데이터 유형을 제공한다. 문자열이란 문자의 나열(순서 있는 묶음)이라는 뜻이다.

파이썬 코드에서 문자열을 표시하려면 작은따옴표 또는 큰따옴표로 감싸면 된다.

1-2. 이스케이프 문자

이스케이프란 일반적인 방법으로 입력하기 어려운 문자를 입력하는 방법이다.

역슬래시(\) 기호를 쓰고 이스케이프할 문자를 붙인다.

이스케이프를 통해 표현해야 하는 문자로는 다음과 같은 것이 있다.

\\	역슬래시
\'	작은따옴표(작은따옴표 안에서)
\"	큰따옴표(큰따옴표 안에서)
\n	개행 문자
\r	개행 문자
\t	탭 문자

이스케이프 문자를 활용한 코드를 살펴보자.

```
>>> text = 'PythonWnProgramming'
>>> print(text)

Python
Programming
```

1-3. 문자열 길이 세기

문자열의 길이를 알고 싶을 때는 len() 함수를 사용하면 된다.

```
>>> len('python')
6
>>> len('파이썬')
3
```


1-4. 문자열의 최대 최소 구하기

max()와 min() 함수는 인자의 최댓값과 최솟값을 반환해주는 함수이다.

2개 이상의 숫자도 인자가 될 수 있으며 최대와 최소 수를 반환한다.

```
>>> max('python')
'y'
>>> min(1, 2, 3)
1
```

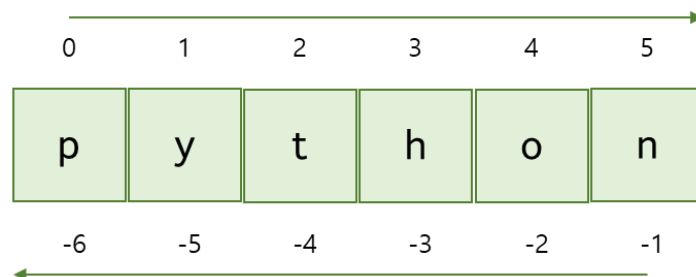
1-5. 문자의 위치 확인하기

문자열이 배열(Array)의 특징을 갖기 때문에, 첨자(index)를 통해 접근 가능하다.

문자열의 순서는 0부터 시작하며 특정 문자를 읽을 때는 인덱스 연산자[]를

사용하여 참조하면 된다. -1부터 시작하는 역순 첨자도 참조 가능하다.

첨자가 유효 범위를 벗어나면 IndexError가 발생하니 주의하자.



```
>>> 'python'[0]
'p'
>>> 'python'[4]
'o'
>>> 'python'[-1]
'n'
>>> 'python'[6]
```

Traceback (most recent call last):

File "<pyshell#4>", line 1, in <module>

'python'[6]

IndexError: string index out of range

1-6. 문자열 슬라이싱

문자열에서 특정 부분을 참조하는 방법을 슬라이싱(slicing)이라고 한다.

이때 기존의 문자열은 절대 수정되지 않고, 참조한 부분이 반환되는 것이다.

[start:end]의 형태로 부분 문자열을 반환하는데, start 첨자부터 end-1 첨자까지의 문자열을 반환한다. 코드를 통해 자세히 알아보자.

```
>>> 'python'[3:5]
'ho'
>>> 'python'[2:4]
'th'
>>> 'python'[0:6]
'python'
```

음수도 사용 가능하며, 양수, 음수를 혼합하여 사용할 수도 있다.

```
>>> 'python'[1:-1]
'ytho'
>>> 'python'[-4:6]
'thon'
>>> 'python'[5:-1]
''
```

마지막처럼 start와 end로 구성하는 문자열이 없다면 아무것도 없는 빈 문자열 ""을 반환한다.

start 첨자를 비우면 처음부터 end-1까지의 첨자 반환을 의미하고,

반대로 end 첨자를 비우면 start 첨자부터 끝까지의 반환을 의미한다.

즉, start와 end 첨자를 모두 비우며 전체 문자열을 반환한다.

슬라이싱에서 문자 사이의 간격을 조정할 수도 있다.

[start:end:step]의 형태로 사용하면 문자 사이의 간격 조정이 가능하다.

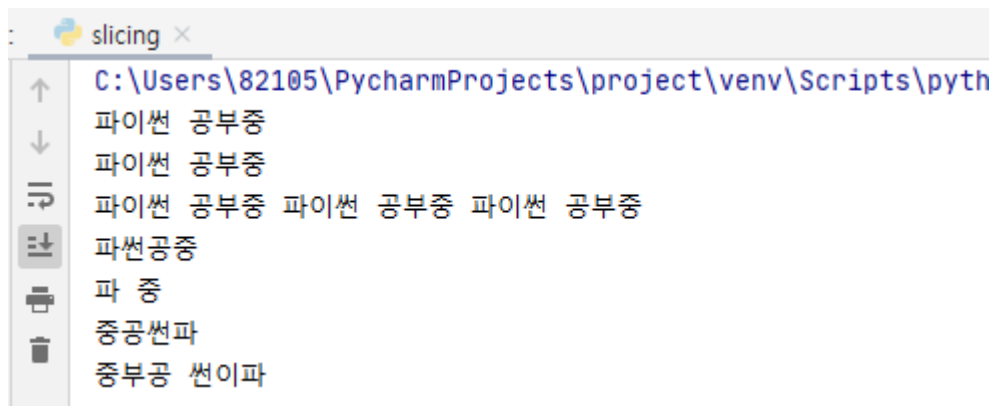
step은 음수도 가능하며, 문자열을 역순으로 반환한다. 코드를 통해 확인해보자.

```
>>> 'python'[::-1]
'nohtyp'
>>> 'python'[::-2]
'pto'
>>> 'python'[::-3]
'ph'
>>> 'python'[::-1]
```

1-7. 프로그래밍 연습

다양한 문자열 슬라이싱하기

```
1 str = '파이썬 공부중'
2 print(str[:3], str[4:])          # 양수
3 print(str[:-4], str[-3:])       # 음수
4 print(str[:], str[:,], str[:,1]) # 모든 문자열 참조
5 print(str[:,2])                 # 한 문자씩 건너뛰기
6 print(str[:,3])                 # 두 문자씩 건너뛰기
7 print(str[:, -2])               # 역순으로 한 문자씩 건너뛰기
8 print(str[:, -1])               # 역순 출력
```



■ 문자열 관련 메소드

2-1. 메소드 설명

클래스에 소속되어 있는 함수를 메소드(method)라고 한다.

메소드의 호출은 '객체 이름.함수 이름(인자)'와 같이 사용한다.

문자열 메소드는 대체된 문자열을 반환하는 것일 뿐 문자열이 수정되지는 않는다.

주로 많이 사용되는 문자열 메소드를 간단히 정리해 보았다.

A. `replace(a, b)` : 문자열에서 a를 b로 치환한 것을 반환한다.

```
>>> str = 'hello, python'
>>> str.replace('python', 'java')
'hello, java'
```

B. `count(str)` : 문자열 안에 나오는 횟수를 알려준다.

```
>>> str = 'hello, python'
>>> str.count('o')
2
```

C. `join(str)` : 문자열의 문자 사이에 원하는 문자열을 삽입한다.

```
>>> str = 'hello, python'
>>> '.'.join(str)
'h.e.l.l.o,...p.y.t.h.o.n'
```

D. `find()`와 `index()` : 문자열이 맨 처음에 위치한 첨자를 반환한다.

하지만 `index()`는 찾는 문자열이 없을 경우, `ValueError`를 발생시키고,

`find()`는 오류를 발생시키지 않고 -1을 반환한다.

`rfind()`와 `rindex()`는 역순으로 문자열을 찾아 첨자를 반환한다.

```
>>> str = 'hello, python'
>>> str.find('python')
7
>>> str.index('python')
7
```

E. `split(str)` : 구분자를 이용하여 문자열을 나눠 준다.

```
>>> str = 'hello, python'
>>> str.split(',')
['hello', ' python']
```

F. `upper()`와 `lower()` : `upper()`는 대문자로 변환하고, `lower()`는 소문자로 변환한다.

```
>>> str = 'Hello, Python'
>>> str.upper()
'HELLO, PYTHON'
>>> str.lower()
'hello, python'
```

G. `center(폭, 채울 문자)` : 폭을 지정하고 문자열을 중앙에 배치한다.

```
>>> ' python '.center(30)
'          python          '
>>> ' python '.center(30, '*')
'***** python *****'
```

H. `strip()` : 문자열 좌우의 공백 문자를 제거하고 반환한다.

```
>>> ' python '.strip()
'python'
>>> ' python '.lstrip()    # 왼쪽 공백 제거
'python '
>>> ' python '.rstrip()    # 오른쪽 공백 제거
' python'
```

I. `format()` : 출력을 정형화한다.

C언어의 `printf()`에서 사용하는 형식 지정자 `%d`와 같은 형태도 지원한다.

```
>>> a, b = 1, 2
>>> print('{ } + { } = {}'.format(a, b, a + b))
1 + 2 = 3
>>> print('%d + %d = %d' % (a, b, a + b))
1 + 2 = 3
```

■ 논리 자료와 다양한 연산

3-1. 논리 연산

논리 연산을 이용하면 여러 개의 명제를 하나로 조합하거나, 부정할 수 있다.

먼저, 논리곱 and와 &연산자는 두 항이 모두 참이어야 True다.

논리합 or과 |연산자는 하나라도 참이면 참이고, 두 항이 모두 거짓이어야 False다.

배타적 논리합 ^연산자는 두 항이 다르면 True, 같으면 False다.

not 연산자는 참을 True를 False로, False를 True로 바꾸는 연산이다.

논리 연산의 우선순위는 not, and, or순이다.

좌변	우변	and 연산	or 연산
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

3-2. 관계 연산

관계 연산자는 크기 비교에 사용되는 연산자로, 결과는 논리 값이다.

문자열의 비교 값은 유니코드 값이며, 사전 순서와 동일하다.

연산자	의미
>	크다.
>=	크거나 같다.
<	작다.
<=	작거나 같다.
==	같다.
!=	다르다.

3-3. 멤버십 연산

파이썬 키워드인 in은 멤버의 소속 여부를 알려주는 연산자로, 결과는 논리값이다.

키워드 in 뒤의 문자열에서 in 앞부분 문자열이 있는지 확인할 수 있다.

반대로 not in문은 부분 문자열이 없으면 True를 반환한다.

3-4. 비트 논리 연산

비트 연산은 정수로 저장된 메모리에서 비트와 비트의 연산을 의미한다.

True를 1, False를 0으로 생각하면 논리 연산과 동일하다.

3-5. 비트 이동 연산

비트 이동 연산자 >>와 <<는 연산자의 방향인 오른쪽 또는 왼쪽으로,

비트 단위로 지정된 횟수만큼 이동시키는 연산자이다.

<<에 의해 생기는 오른쪽 빈 자리는 모두 0으로 채워지며,

>>에 의한 빈자리는 원래의 정수 부호에 따라 0 또는 1로 채워진다.

3-6. 연산자 우선순위

순위	연산자	설명	종류
1	**	거듭제곱	지수 연산
2	~	비트 보수	단항 연산
3	+x, -x	부호	
4	*, /, //, %	곱, 나눗셈, 몫, 나머지	산술 연산
5	+, -	덧셈, 뺄셈	
6	>>, <<	비트 이동	비트 연산
7	&	비트 and	
8	^	비트 xor	
9		비트 or	
10	<, <=, >, >=, ==, !=, in, not in	관계, 소속	관계 소속
11	=, %=, /= ...	대입	대입 연산
12	not	논리 not	논리 연산
13	and	논리 and	
14	or	논리 or	

■ 프로그래밍 예제

```

1      # 다양한 부분 문자열을 출력하는 프로그램
2      str = input('다섯 문자 이상의 문자열 입력 >> ')
3      print('입력 문자열 : ', str)
4      print('첫 문자 : ', str[:1])
5      print('마지막 문자 : ', str[-1:-2:-1])
6      print('첫 문자를 제외한 부분 문자열 : ', str[1:])
7      print('마지막 문자를 제외한 부분 문자열 : ', str[:len(str)-1])
8      print('맨 앞과 뒤의 두 문자씩을 제외한 부분 문자열 : ', str[2:-2])
9      print('문자 하나씩을 건너뛴 부분 문자열 : ', str[::2])
10     print('역문자열 : ', str[::-1])

```

```

C:\Users\82105\PycharmProjects\project\venv\Scripts\python.
다섯 문자 이상의 문자열 입력 >> Python String Slicing
입력 문자열 : Python String Slicing
첫 문자 : P
마지막 문자 : g
첫 문자를 제외한 부분 문자열 : ython String Slicing
마지막 문자를 제외한 부분 문자열 : Python String Slicin
맨 앞과 뒤의 두 문자씩을 제외한 부분 문자열 : thon String Slici
문자 하나씩을 건너뛴 부분 문자열 : Pto tigSiig
역문자열 : gnicilS gnirts nohtyP

```

```

1      # 메소드와 슬라이싱으로 부분 문자열을 출력하는 프로그램
2      url = 'https://docs.python.org/3/tutorial'
3      print(url[url.find('h'):url.find('/')])
4      print(url[url.find('d'):url.find('/3')])
5      print(url[url.rfind('tu'):len(url)])

```

```

C:\Users\82105\PycharmProjects\project\venv\Scripts\
https
docs.python.org
tutorial

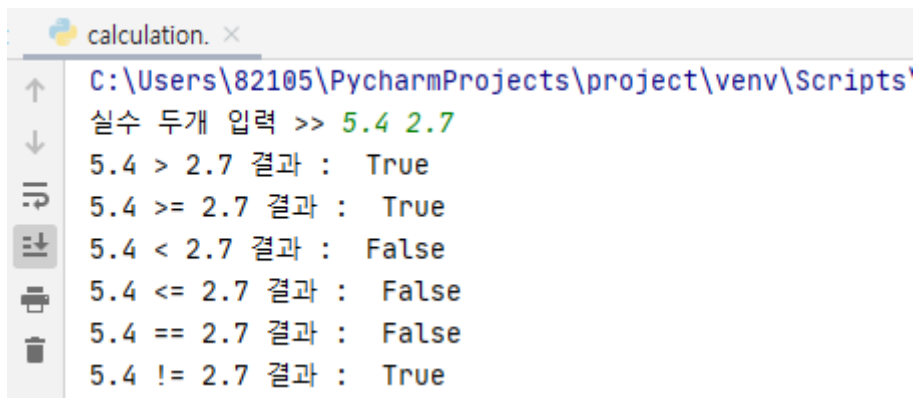
```



```

1      # 관계 연산의 결과를 출력하는 프로그램
2      a, b = input('실수 두개 입력 >> ').split()
3      num1 = float(a)
4      num2 = float(b)
5      print('{} > {}'.format(num1, num2), '결과 : ', num1 > num2)
6      print('{} >= {}'.format(num1, num2), '결과 : ', num1 >= num2)
7      print('{} < {}'.format(num1, num2), '결과 : ', num1 < num2)
8      print('{} <= {}'.format(num1, num2), '결과 : ', num1 <= num2)
9      print('{} == {}'.format(num1, num2), '결과 : ', num1 == num2)
10     print('{} != {}'.format(num1, num2), '결과 : ', num1 != num2)

```



```

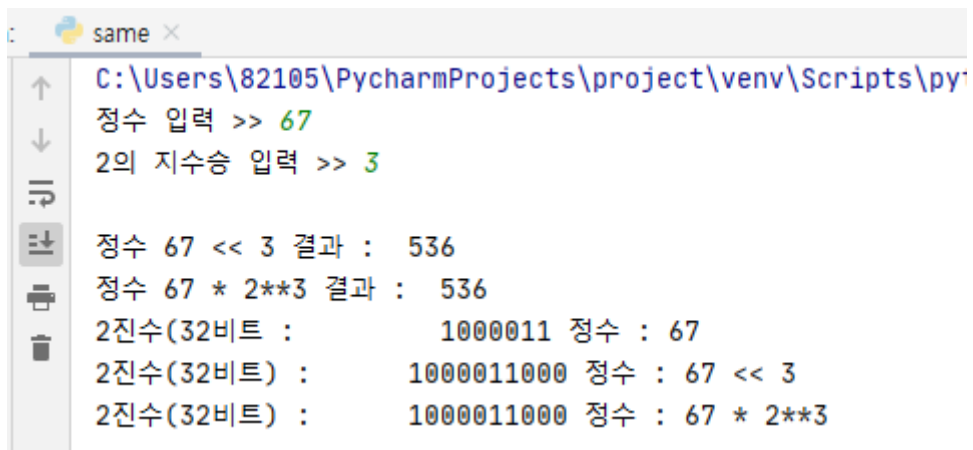
calculation. x
C:\Users\82105\PycharmProjects\project\venv\Scripts
실수 두개 입력 >> 5.4 2.7
5.4 > 2.7 결과 : True
5.4 >= 2.7 결과 : True
5.4 < 2.7 결과 : False
5.4 <= 2.7 결과 : False
5.4 == 2.7 결과 : False
5.4 != 2.7 결과 : True

```

```

1      # 계산한 결과가 같음을 보이는 프로그램
2      a = int(input('정수 입력 >> '))
3      b = int(input('2의 지수승 입력 >> '))
4      print('')
5      print('정수 %d << %d' % (a, b), '결과 : ', a << b)
6      print('정수 %d * %d**%d' % (a, 2, b), '결과 : ', a * 2**b)
7      print('2진수(32비트 : {0:15b} 정수 : {0:d}'.format(a))
8
9      print('2진수(32비트) : {0:15b}'.format(a << b),
            '정수 : {0} << {1}'.format(a, b))
10
11     print('2진수(32비트) : {0:15b}'.format(a * 2**b),
            '정수 : {0} * {1}**{2}'.format(a, 2, b))

```



```

C:\Users\82105\PycharmProjects\project\venv\Scripts\python.exe
정수 입력 >> 67
2의 지수승 입력 >> 3

정수 67 << 3 결과 : 536
정수 67 * 2**3 결과 : 536
2진수(32비트 : 1000011 정수 : 67
2진수(32비트) : 1000011000 정수 : 67 << 3
2진수(32비트) : 1000011000 정수 : 67 * 2**3

```

4장. 조건과 반복

■ if ... else

1-1. if문

파이썬에서는 조건과 선택을 처리할 때 if문을 사용한다.

조건을 검사하여 결과가 True이면 hem를 실행하고, False이면 실행하지 않는다.

if문을 작성하는 형태는 다음과 같다.

if 조건: 문장	1. 조건 이후에 반드시 콜론이 있어야 한다. 2. True일 때 실행할 문장은 들여쓰기 한다.
--------------	--

미세먼지 농도가 81을 넘으면 '나쁨' 수준으로, 마스크를 써야 한다.

이 내용에 관한 예제를 통해 if문을 알아보자.

1	PM = 90
2	if 81 < PM:
3	print('마스크를 착용하세요!')

1-2. if ... else문

if문은 조건이 True일 때만 본문의 내용을 실행하고, 조건이 False라면

실행하지 않고 다음 코드로 넘어가게 된다. 조건이 False일 때 별도의 코드를

실행하게 할 수도 있다. if문에 else절을 추가하면 된다.

if ... else문을 작성하는 형태는 다음과 같다.

if 조건: 문장 1 else: 문장 2	1. 문장 1은 조건이 True일 때 실행된다 2. 문장 2는 조건이 False일 때 실행된다. 3. else 이후에도 반드시 콜론이 있어야 한다.
---------------------------------	--

위 if문에 나왔던 예제를 이용하여 if ... else문을 연습해보자.

```

1    PM = 90
2    if 81 < PM:
3        print('마스크를 착용하세요!')
4    else:
5        print('날씨가 화창해요!')

```

1-3. if ... elif문

선택지가 두 개보다 더 많을 때도 있다. if와 else절 사이에 elif절을 추가할 수 있다.

elif를 사용하면 조건과 선택지를 원하는 만큼 추가할 수 있다.

if ... elif문을 작성하는 형태는 다음과 같다.

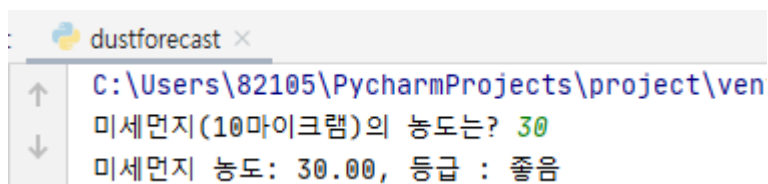
<pre> if 조건 1: 문장 1 elif 조건 2: 문장 2 ... else: 문장 3 </pre>	<ol style="list-style-type: none"> 1. elif는 else if를 의미한다. 2. 모든 조건이 거짓일 때 문장 3을 출력한다.
---	--

미세먼지의 농도를 입력 받아 등급을 출력하는 프로그램을 작성해 보자.

```

1    PM = float(input('미세먼지(10마이크램)의 농도는? '))
2    if 151 <= PM:
3        print('미세먼지 농도: {:.2f}, 등급 : {}'.format(PM, '매우 나쁨'))
4    elif 81 <= PM:
5        print('미세먼지 농도: {:.2f}, 등급 : {}'.format(PM, '나쁨'))
6    elif 31 <= PM:
7        print('미세먼지 농도: {:.2f}, 등급 : {}'.format(PM, '보통'))
8    else:
9        print('미세먼지 농도: {:.2f}, 등급 : {}'.format(PM, '좋음'))

```



■ for, while

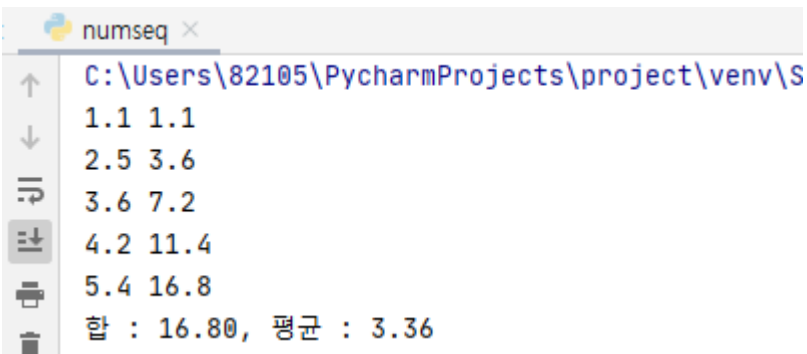
2-1. for문

반복문을 사용하면 중복을 줄여 프로그램이 간결해지고, 프로그램 절차를 효과적으로 수행할 수 있다. for문은 순차적 처리와 컬렉션 순회에 특화된 반복이다. for문을 작성하는 형태는 다음과 같다.

<pre>for 변수 in 시퀀스: 문장 1 else: 문장 2</pre>	<ol style="list-style-type: none">1. 문자열과 같은 시퀀스가 위치한다.2. 반드시 콜론이 있어야 한다.3. 문장 1을 반복 실행한다.4. 문장 2는 반복이 종료된 마지막에 실행된다.
---	--

합과 평균을 구하는 예제를 통해 for문에 대해 알아보자.

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



```
numseq x
C:\Users\82105\PycharmProjects\project\venv\S
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
```

for 문의 시퀀스에 range()함수를 사용하면 매우 간결하다.

```
>>> for i in range(10)
      print(i, end = ' ')

0 1 2 3 4 5 6 7 8 9
```

2-2. while 문

while 문은 지정한 조건이 유지되는 동안 코드를 계속 반복한다.

기능이 단순하고, 특별한 제약 없이 다양한 반복을 처리할 수 있다.

while 문을 작성하는 형태는 다음과 같다.

<pre>while 조건: 문장 1 else: 문장 2</pre>	<ol style="list-style-type: none">1. 반드시 콜론이 있어야 한다.2. 조건이 True면 문장 1을 반복 실행한다.3. 문장 3은 반복이 종료된 마지막에 실행된다.
--	--

n 이 6 이상이 되면 반복문을 종료하는 예제를 통해 while 문을 알아보자.

```
>>> n = 1
>>> while n <= 5:
        print(n, end = ' ')
        n += 1
else:
        print('\n반복 while 종료: n => %d' % n)

1 2 3 4 5
반복 while 종료: n => 6
```

2-3. 프로그래밍 연습

중첩된 반복 for 문을 이용하여 2 단부터 5 단까지 구구단 출력하기

```
1 for i in range(2, 6):
2     for j in range(1, 10):
3         print('%d * %d = %2d' % (i, j, i * j), end=' ')
4     print()
```

```
multiplicationtable x
C:\Users\82105\PycharmProjects\project\venv\Scripts\python.exe C:/Users/82105/PycharmProjects/proje
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
```

■ break, continue

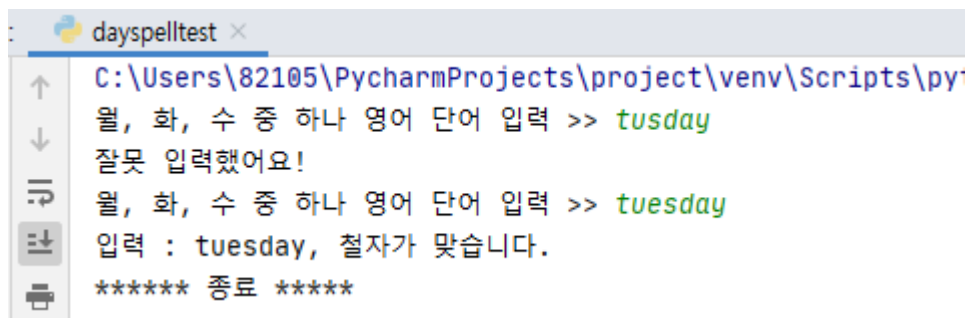
3-1. break 문

while 문의 조건이 True 라면 무한히 반복된다. 이것을 무한 반복이라고 하는데, for 나 while 반복 내부에서 break 는 else:를 실행시키지 않고 무조건 종료한다. 즉, break 문은 특정한 조건에서 즉시 반복을 종료할 때 사용한다.

3-2. continue 문

continue 문은 for 와 while 반복 내에서 한 주기의 실행을 중지하고 다음 주기로 넘어가도록 처리하는 명령이다. 월, 화, 수요일의 영어 단어를 테스트하는 예제를 통해 break 문과 continue 문에 대해 알아보자.

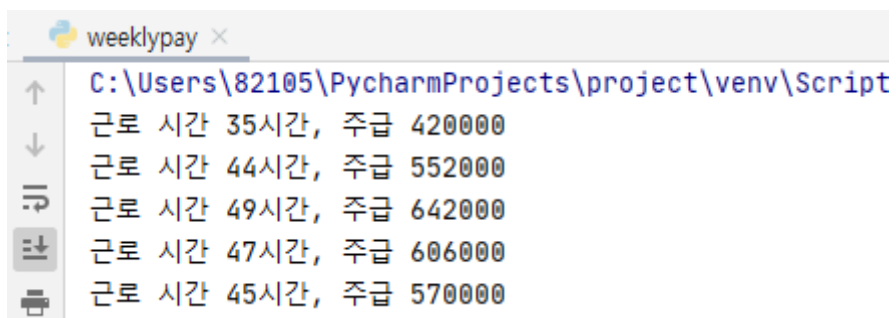
```
1    days = ['monday', 'tuesday', 'wednesday']
2
3    while True:
4        user = input('월, 화, 수 중 하나 영어 단어 입력 >> ')
5        if user not in days:
6            print('잘못 입력했어요!')
7            continue
8        print('입력 : %s, 철자가 맞습니다.' % user)
9        break
10
11    print(' 종료 '.center(15, '*'))
```



```
dayspelltest x
C:\Users\82105\PycharmProjects\project\venv\Scripts\py
월, 화, 수 중 하나 영어 단어 입력 >> tusday
잘못 입력했어요!
월, 화, 수 중 하나 영어 단어 입력 >> tuesday
입력 : tuesday, 철자가 맞습니다.
***** 종료 *****
```

■ 프로그래밍 예제

```
1      # 난수로 근로 시간을 정하고, 주급을 계산해 출력하는 프로그램
2      from random import randint
3      hourly = 12000
4
5      for i in range(5):
6          time = randint(35, 50)
7          if time >= 40:
8              weekly = 40 * hourly + (time - 40) * hourly * 1.5
9              print('근로 시간 %d시간, 주급 %d' % (time, weekly))
10         else:
11             weekly = time * hourly
12             print('근로 시간 %d시간, 주급 %d' % (time, weekly))
```



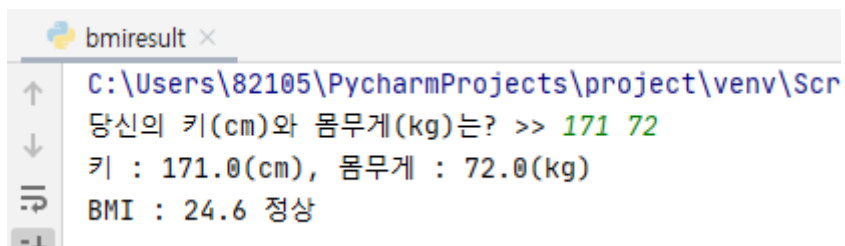
```
↑ C:\Users\82105\PycharmProjects\project\venv\Script
↓ 근로 시간 35시간, 주급 420000
↕ 근로 시간 44시간, 주급 552000
↕ 근로 시간 49시간, 주급 642000
↕ 근로 시간 47시간, 주급 606000
⏏ 근로 시간 45시간, 주급 570000
```



```

1      # 체질량 지수를 계산하여 결과를 출력하는 프로그램
2      h, w = input('당신의 키(cm)와 몸무게(kg)는? >> ').split()
3      height = float(h)
4      weight = float(w)
5      print('키 : %.1f(cm), 몸무게 : %.1f(kg)' % (height, weight))
6      bmi = weight / (height / 100) ** 2
7
8      if bmi >= 40:
9          print('BMI : {:.1f} {}'.format(bmi, '고도 비만'))
10     elif bmi >= 35:
11         print('BMI : {:.1f} {}'.format(bmi, '중등도 비만'))
12     elif bmi >= 30:
13         print('BMI : {:.1f} {}'.format(bmi, '비만'))
14     elif bmi >= 25:
15         print('BMI : {:.1f} {}'.format(bmi, '과체중'))
16     elif bmi >= 18.5:
17         print('BMI : {:.1f} {}'.format(bmi, '정상'))
18     else:
19         print('BMI : {:.1f} {}'.format(bmi, '저체중'))

```



```

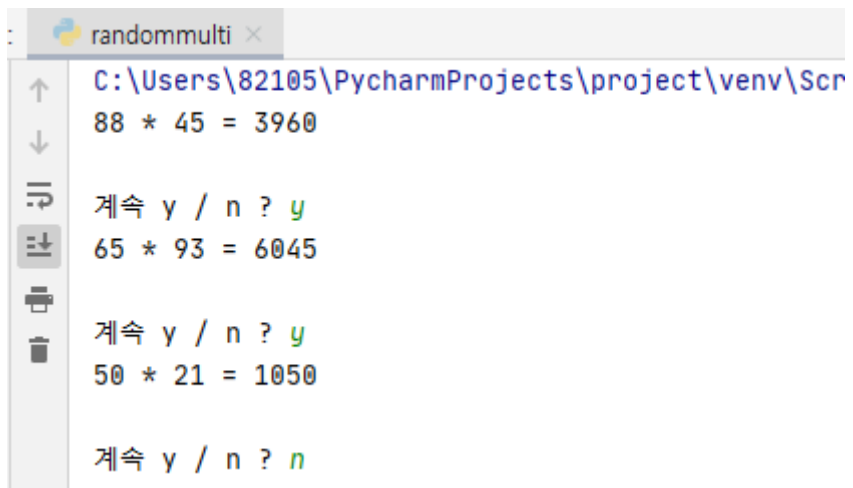
bmireult x
C:\Users\82105\PycharmProjects\project\venv\Scr
당신의 키(cm)와 몸무게(kg)는? >> 171 72
키 : 171.0(cm), 몸무게 : 72.0(kg)
BMI : 24.6 정상

```

```

1      # 난수를 생성하고 곱셈의 결과를 출력하는 프로그램
2      from random import randint
3
4      while True:
5          num1 = randint(1, 99)
6          num2 = randint(1, 99)
7          print('%d * %d = %d\n' % (num1, num2, num1 * num2))
8          option = input('계속 y / n ? ')
9          if option == 'n':
10             break

```



```

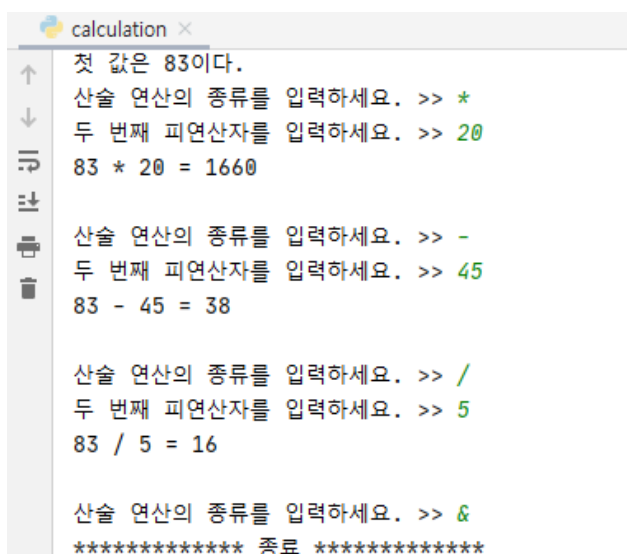
randommulti x
C:\Users\82105\PycharmProjects\project\venv\Scr
88 * 45 = 3960
계속 y / n ? y
65 * 93 = 6045
계속 y / n ? y
50 * 21 = 1050
계속 y / n ? n

```

```

1      # 난수와 반복으로 산술 연산 후 출력하는 프로그램
2      from random import randint
3      num1 = randint(1, 100)
4      calculation = ['+', '-', '*', '/', '%']
5
6      print('첫 값은 %d이다.' % num1)
7      while True:
8          option = input('산술 연산의 종류를 입력하세요. >> ')
9          if option not in calculation:
10             break
11         num2 = int(input('두 번째 피연산자를 입력하세요. >> '))
12         if option == '+':
13             print('%d + %d = %dWn' % (num1, num2, num1 + num2))
14         elif option == '-':
15             print('%d - %d = %dWn' % (num1, num2, num1 - num2))
16         elif option == '*':
17             print('%d * %d = %dWn' % (num1, num2, num1 * num2))
18         elif option == '/':
19             print('%d / %d = %dWn' % (num1, num2, num1 / num2))
20         elif option == '%':
21             print('%d % %d = %dWn' % (num1, num2, num1 % num2))
22
23     print(' 종료 '.center(30, '*'))

```



```

calculation ×
첫 값은 83이다.
산술 연산의 종류를 입력하세요. >> *
두 번째 피연산자를 입력하세요. >> 20
83 * 20 = 1660

산술 연산의 종류를 입력하세요. >> -
두 번째 피연산자를 입력하세요. >> 45
83 - 45 = 38

산술 연산의 종류를 입력하세요. >> /
두 번째 피연산자를 입력하세요. >> 5
83 / 5 = 16

산술 연산의 종류를 입력하세요. >> &
***** 종료 *****

```

5장. 리스트와 튜플

■ 리스트를 통한 자료 값 처리

1-1. 리스트

리스트(list)는 목록이라는 뜻으로, 다양한 데이터를 담을 수 있고 변경할 수 있는 시퀀스다. 파이썬의 시퀀스 중에서 가장 많이 사용되며 가장 대표적이다.

리스트는 대괄호 []를 사용하여 표현할 수 있다. 예제를 통해 확인해보자.

```
>>> menu = ['COFFEE', 'BEVERAGE', 'ADE']
>>> menu
['COFFEE', 'BEVERAGE', 'ADE']
>>> print(menu)
['COFFEE', 'BEVERAGE', 'ADE']
```

빈 대괄호를 사용하면 빈 리스트를 만들 수 있다. 출력하면 []로 표시된다.

인자가 없는 내장 함수 list()로도 빈 리스트를 생성할 수 있다.

리스트의 맨 뒤에 항목을 추가하려면 append(삽입할 항목)를 사용한다.

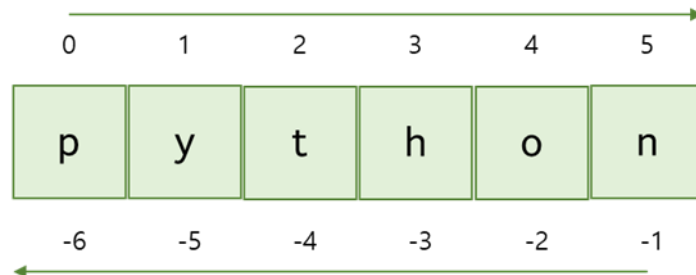
```
>>> pl = list()
>>> pl.append('C++')
>>> pl.append('Java')
>>> print(pl)
['C++', 'Java']
```

리스트의 항목 수를 알고 싶을 때는 len() 함수를 사용하면 된다.

```
>>> py = list('python')
>>> py
['p', 'y', 't', 'h', 'o', 'n']
>>> len(py)
6
```

1-2. 항목 참조

리스트에서 첨자(index)를 사용해 항목을 참조할 수 있는데, 이는 문자열에서 배운 첨자 방식과 동일하다.



1-3. 항목 수정

리스트의 메소드 `count(값)`는 값을 갖는 항목의 개수, `index(값)`는 인자인 값의 항목이 위치한 첨자를 반환한다. 동일한 값이 여러 개이면 첫 번째 위치의 첨자다. 리스트의 첨자를 이용한 항목을 사용하여 수정이 가능하다.

```
>>> num = [1, 3, 3]
>>> num[1] = 2
>>> print(num)
[1, 2, 3]
```

1-4. 중첩 리스트

리스트 내부에 또 리스트가 항목으로 올 수 있다. 예제를 통해 확인해보자.

```
>>> color = [['빨강', '초록', '파랑'], '흰색', '검정']
>>> print(color)
[['빨강', '초록', '파랑'], '흰색', '검정']
>>> print(color[0])
['빨강', '초록', '파랑']
```

■ 리스트 부분 참조와 삽입, 삭제

2-1. 슬라이싱

문자열에서 배웠듯 리스트도 콜론을 사용해 전체나 일부분을 참조할 수 있다.

리스트 슬라이싱의 형태는 다음과 같다.

리스트[start:end:step]	<ol style="list-style-type: none">1. stop-1까지의 부분 리스트를 반환한다.2. 오름차순과 내림차순 혼합 사용이 가능하다.
---------------------	---

2-2. 부분 수정

리스트의 일부분을 다른 리스트로 수정하려면 슬라이싱을 사용해야 한다.

간단한 예제를 통해 알아보자.

```
>>> sports = ['풋살', '축구', '야구', '농구', '배구']
>>> others = ['피구', '골프']
>>> sports[1:3] = others[:2]
>>> print(sports)
['풋살', '피구', '골프', '농구', '배구']
```

2-3. 항목 삽입, 삭제

항목 삽입 : insert(첨자, 항목)

첨자 위치에 항목을 삽입하려면, 리스트.insert()를 이용한다.

삽입되는 항목은 무엇이든 가능하며, 빈 리스트에도 삽입할 수 있다.

```
>>> num = [10, 20]
>>> num.insert(2, 30)
>>> num
[10, 20, 30]
```

항목 삭제 : remove(항목), pop(첨자), pop()

remove(항목)는 리스트에서 지정된 값의 항목을 삭제한다.

pop(첨자)은 지정된 첨자의 항목을 삭제하고 반환한다.

pop()은 마지막 항목을 삭제하고 삭제된 값을 반환한다.

```
>>> num = [10, 20, 30, 40]
>>> num.remove(40)
>>> print(num)
[10, 20, 30]
>>> print(num.pop(1))
20
>>> print(num.pop())
30
>>> print(num)
[10]
```

항목 또는 변수 삭제 : del

문장 del 은 뒤에 위치한 변수나 항목을 삭제한다.

변수 자체를 메모리에서 제거하므로 삭제 이후 참조하면 오류가 발생한다.

```
>>> num = [10, 20, 30]
>>> del num[0]
>>> print(num)
[20, 30]
```

모든 항목 삭제 : clear()

clear()를 사용하면 모든 항목이 삭제된다. 이후 리스트는 빈 리스트가 된다.

```
>>> num = [10, 20, 30]
>>> num.clear()
>>> print(num)
[]
```

2-4. 프로그래밍 연습

```
1 item = ['연필', '볼펜']
2 print(item)
3
4 item.insert(1, 2)
5 item.insert(3, 3)
6 item.insert(4, '지우개')
7 item.insert(5, 1)
8 print(item)
9
10 print(item.pop(1))
11 item.remove('연필')
12 del item[2:]
13
14 print(item)
```

```
iteminsertremove x
C:\Users\82105\PycharmProjects\project\venv\Script
['연필', '볼펜']
['연필', 2, '볼펜', 3, '지우개', 1]
2
['볼펜', 3]
```

2-5. 추가, 연결, 반복

리스트.extend(list)는 리스트에 인자인 list 를 가장 뒤에 추가한다.

```
>>> day1 = ['월', '화', '수']
>>> day2 = ['목', '금', '토', '일']
>>> day1.extend(day2)
>>> print(day1)
['월', '화', '수', '목', '금', '토', '일']
```

문자열에서 알아봤던 덧셈 연산자와 곱셈 연산자를 통한 연결과 반복을 리스트에도 적용할 수 있다.

2-6. 순서, 정렬

항목 순서 뒤집기 : `reverse()`

`reverse()`는 항목의 순서를 반대로 뒤집는다.

```
>>> num = [10, 20, 30]
>>> num.reverse()
>>> print(num)
[30, 20, 10]
```

항목 순서 정렬 : `sort()`

정렬(sorting)의 방식에는 오름차순과 내림차순이 있다.

`sort()`는 리스트 항목의 순서를 오름차순으로 정렬하고, `sort(reverse=True)`로 호출하면 역순인 내림차순으로 정렬한다.

```
>>> num = [20, 10, 30]
>>> num.sort()
>>> print(num)
[10, 20, 30]
>>> num.sort(reverse=True)
>>> print(num)
[30, 20, 10]
```

2-7. 리스트 컴프리헨션

리스트 컴프리헨션은 리스트를 만드는 간결한 방법을 제공한다.

컴프리헨션은 함축, 축약, 내포, 내장 등으로도 불린다.

<pre>리스트 = [] for 항목 in 시퀀스: if 조건: 리스트.append(항목)</pre>	<pre>리스트 = [항목 for 항목 in 시퀀스 if 조건식]</pre>
--	--

일반 방식과 리스트 컴프리헨션 방식으로 각각 출력하는 프로그램을 살펴보자.

```
1    # for문으로 리스트 생성
2    s = []
3    for i in range(10):
4        if i%2 == 1:
5            s.append(i**2)
6    print(s)
7
8    # 컴프리헨션으로 리스트 생성
9    squares = [i**2 for i in range(10) if i%2 == 1]
10   print(squares)
```

```
listcomprehension x
C:\Users\82105\PycharmProjects\projec
[1, 9, 25, 49, 81]
[1, 9, 25, 49, 81]
```

2-8. 대입, 복사

리스트에서 대입 연산자 `=`은 얇은 복사(shallow copy)로, 대입되는 변수가 동일한 시퀀스를 가리킨다.

결국 얇은 복사로 생성된 두 변수는 하나의 같은 리스트이므로 두 변수 중 하나의 항목을 삭제하면 두 변수 모두의 리스트에 반영된다.

완전히 새로운 리스트를 만들어 복사하려면 슬라이스나 `copy()` 또는

`list()` 함수를 이용해야 하는데, 이를 깊은 복사(deep copy)라고 한다.

깊은 복사에 의해 만들어진 두 리스트는 항목은 같지만 전혀 다른 리스트이다.

■ 튜플

3-1. 튜플

튜플은 문자열, 리스트와 같은 항목의 나열인 시퀀스이다.

리스트와 비슷하지만 한 번 담은 항목은 수정이 불가능하다는 점에서 차이가 있다.

튜플은 괄호 ()를 사용하여 표현할 수 있다. 예제를 통해 확인해보자.

```
>>> country = ('대한민국', '서울'), ('영국', '런던')
>>> space = '밤', '낮', '해', '달'
>>> tupa = 1,
```

괄호는 생략 가능하며, tupa 처럼 항목이 하나인 튜플을 표현할 때는 마지막에 반드시 콤마를 붙여야 한다.

3-2. 연결과 반복, 정렬과 삭제

리스트와 같이 덧셈 연산자와 곱셈 연산자를 사용하여 튜플을 연결하고 반복할 수 있다.

```
>>> country = ('대한민국', '영국')
>>> capital = ('서울', '런던')
>>> print(country + capital)
('대한민국', '영국', '서울', '런던')
>>> py = ('hello', 'python')
>>> print(py * 2)
('hello', 'python', 'hello', 'python')
```

내장 함수 sorted(튜플)은 항목의 순서를 오름차순으로 정렬한 리스트를 반환한다.

절대 튜플 자체가 수정되지는 않는다.

문장 del 에서 변수를 지정하면 변수 자체가 삭제되므로 삭제 이후에 변수를 참조하려고 하면 오류가 발생한다.

■ 프로그래밍 예제

```
1      # 튜플 생성 후 한글을 입력 받아 영어를 출력하는 프로그램
2      korean = ('정렬', '초보자', '내포', '사전')
3      english = ('sorting', 'novice', 'comprehension', 'dictionary')
4
5      ko = input('찾을 단어 입력 ? ')
6
7      if ko in korean:
8          i = korean.index(ko)
9          print(english[i])
10
11     else:
12         print('찾으시는 단어가 사전에 없습니다.')
```

tuple ×

↑ C:\Users\82105\PycharmProjects\project\venv\Script
↓ 찾을 단어 입력 ? 초보자
= novice

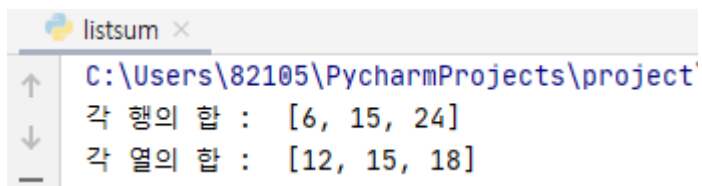
tuple ×

↑ C:\Users\82105\PycharmProjects\project\venv\Script
↓ 찾을 단어 입력 ? 튜플
= 찾으시는 단어가 사전에 없습니다.

```

1      # 중첩된 리스트에서 각 행과 열의 합을 출력하는 프로그램
2      data = [    [1, 2, 3],
3                  [4, 5, 6],
4                  [7, 8, 9]    ]
5      row = []
6      col = []
7
8      for i in range(0, len(data)):
9          c = 0
10         r = 0
11         for j in range(0, len(data[i])):
12             r += data[i][j]
13             c += data[j][i]
14         row.append(r)
15         col.append(c)
16
17     print('각 행의 합 : ', row)
18     print('각 열의 합 : ', col)

```



The screenshot shows a terminal window titled 'listsum' with the following output:

```

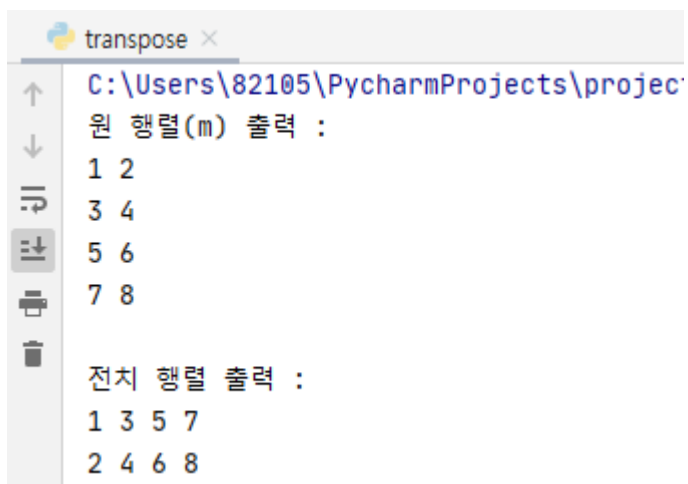
C:\Users\82105\PycharmProjects\project
각 행의 합 : [6, 15, 24]
각 열의 합 : [12, 15, 18]

```

```

1      # 행렬을 for문으로 출력하는 프로그램
2      m = [[1, 2], [3, 4], [5, 6], [7, 8]]
3
4      print('원 행렬(m) 출력 : ')
5      for r in m:
6          print(*r)
7      print()
8
9      print('전치 행렬 출력 : ')
10     for c in range(len(m[0])):
11         for r in range(len(m)):
12             print(m[r][c], end = ' ')
13     print()

```



```

C:\Users\82105\PycharmProjects\projec
원 행렬(m) 출력 :
1 2
3 4
5 6
7 8

전치 행렬 출력 :
1 3 5 7
2 4 6 8

```

6장. 딕셔너리와 집합

■ 딕셔너리

1-1. 개념과 생성

딕셔너리는 말 그대로 ‘사전’을 의미하는데, 사전은 단어의 의미를 설명한다.

여기서 단어를 키(key), 해설을 값(value)이라고 생각하면 된다.

딕셔너리는 키와 값의 항목을 나열한 시퀀스다. 각각의 항목은 키:값과 같이 키와 값을 콜론으로 구분하고, 전체는 중괄호 {}로 묶는다.

항목의 순서는 의미가 없으며, 키는 중복될 수 없고 수정할 수 없다.

```
>>> cafe = {'아메리카노': 2500, '카페라테': 3000, '딸기주스': 3500}
>>> print(cafe)
{'아메리카노': 2500, '카페라테': 3000, '딸기주스': 3500}
>>> type(cafe)
<class 'dict'>
```

인자가 없는 내장 함수 dict()로도 빈 딕셔너리를 생성할 수 있다.

딕셔너리 항목 값으로 리스트나 튜플 등도 들어갈 수 있다.

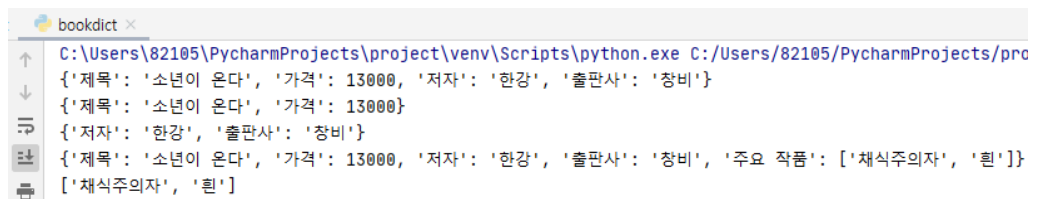
```
>>> lect = dict()
>>> lect['강좌명'] = '파이썬'
>>> lect['개설년도'] = [2020, 1]
>>> lect['학점시수'] = (3, 3)
>>> lect['교수'] = '김민국'
>>> print(lect)
{'강좌명': '파이썬', '개설년도': [2020, 1], '학점시수': (3, 3), '교수': '김민국'}
```

1-2. 함수 dict()로 딕셔너리 생성

내장 함수 dict()에서 리스트나 튜플 1 개를 이용해 딕셔너리 생성이 가능하다.

리스트나 튜플 내부에서 일련의 키-값 쌍으로 [키, 값] 또는 (키, 값) 형식을 모두 사용할 수 있다. 키가 단순 문자열이면 간단히 월='Monday'처럼 키=값 항목 나열로도 지정할 수 있다.

```
1 book1 = {'제목': '소년이 온다', '가격': 13000, '저자': '한강'}
2 book1['출판사'] = '창비'
3 print(book1)
4 book2 = dict(['제목', '소년이 온다'], ['가격', 13000])
5 print(book2)
6 book3 = dict((( '저자', '한강'), ('출판사', '창비'))))
7 print(book3)
8
9 book = dict(제목='소년이 온다', 가격=13000, 저자='한강', 출판사='창비')
10 # 주요 작품 추가
11 book['주요 작품'] = ['채식주의자', '흰']
12
13 print(book) # 전체 출력
14 print(book['주요 작품']) # 주요 작품 출력
```



```
bookdict x
C:\Users\82105\PycharmProjects\project\venv\Scripts\python.exe C:/Users/82105/PycharmProjects/prc
{'제목': '소년이 온다', '가격': 13000, '저자': '한강', '출판사': '창비'}
{'제목': '소년이 온다', '가격': 13000}
{'저자': '한강', '출판사': '창비'}
{'제목': '소년이 온다', '가격': 13000, '저자': '한강', '출판사': '창비', '주요 작품': ['채식주의자', '흰']}
```


1-3. 키는 수정 불가능한 객체로 사용

딕셔너리의 키는 정수와 실수는 물론 튜플도 가능하다.

수정 가능한 리스트는 키로 사용할 수 없으며, `TypeError`가 발생한다.

1-4. 항목 순회

메소드 `keys()`는 키로만 구성된 리스트를 반환한다.

```
>>> day = dict(월='monday', 화='tuesday', 수='wednesday')
>>> print(day)
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
>>> print(day.keys())
dict_keys(['월', '화', '수'])
>>> for key in day.keys():
    print('%s요일 %s' % (key, day[key]))

월요일 monday
화요일 tuesday
수요일 wednesday
```

메소드 `items()`는 (키, 값) 쌍의 튜플이 들어있는 리스트를 반환한다.

```
>>> day = dict(월='monday', 화='tuesday', 수='wednesday')
>>> print(day.items())
dict_items([('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')])
>>> for key, value in day.items():
    print('%s요일 %s' % (key, value))

월요일 monday
화요일 tuesday
수요일 wednesday
```

메소드 `values()`는 값으로 구성된 리스트를 반환한다.

```
>>> day = dict(월='monday', 화='tuesday', 수='wednesday')
>>> print(day.values())
dict_values(['monday', 'tuesday', 'wednesday'])
```

1-5. 항목의 참조, 삭제

메소드 `get(키)`은 키의 해당 값을 반환한다.

딕셔너리에 키가 없어도 오류가 발생하지 않고 `None`을 반환한다.

만일 키 뒤에 다른 인자를 넣으면 키가 없을 때 인자 값을 반환한다.

```
>>> country = {'대한민국': '서울', '영국': '런던', '프랑스': '파리'}
>>> country.get('대한민국')
'서울'
```

메소드 `pop(키)`은 키 항목을 삭제하고, 삭제되는 키의 값을 반환한다.

삭제할 키가 없다면 `KeyError`가 발생하므로 주의해야 한다.

```
>>> country = {'대한민국': '서울', '영국': '런던', '프랑스': '파리'}
>>> print(country.pop('영국'))
런던
>>> country
{'대한민국': '서울', '프랑스': '파리'}
```

메소드 `popitem()`은 임의의 (키, 값)의 튜플을 반환하고 삭제한다.

만일 삭제할 데이터가 하나도 없다면 오류가 발생한다.

```
>>> country = {'대한민국': '서울', '프랑스': '파리'}
>>> print(country.popitem())
('프랑스', '파리')
>>> print(country.popitem())
('대한민국', '서울')
>>> country
{}
```

문장 `del`로 딕셔너리 항목을 삭제할 수 있다.

```
>>> country = {'대한민국': '서울', '프랑스': '파리'}
>>> del country['프랑스']
>>> country
{'대한민국': '서울'}
```

1-6. 항목 전체 삭제, 변수 제거

메소드 `clear()`는 기존의 모든 키: 값 항목을 삭제한다.

```
>>> country = {'대한민국': '서울', '프랑스': '파리'}
>>> country.clear()
>>> country
{}

```

문장 `del`로 딕셔너리를 키로 지정하면 변수 자체가 메모리에서 제거된다.

따라서 제거 이후에 변수를 참조하는 경우에는 오류가 발생한다.

1-7. 딕셔너리 결합, 연산자 `in`

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

원 딕셔너리와 동일한 키가 있다면 인자 딕셔너리의 값으로 대체된다.

```
>>> kostock = {'Samsung Elec.': 40000, 'Daum KAKAO': 80000}
>>> usstock = {'MS': 150, 'Apple': 180}
>>> kostock.update(usstock)
>>> kostock
{'Samsung Elec.': 40000, 'Daum KAKAO': 80000, 'MS': 150, 'Apple': 180}
>>> usstock.update({'MS': 200})
>>> usstock
{'MS': 200, 'Apple': 180}

```

문장 `in`으로 딕셔너리에 키가 존재하는지 검사할 수 있다.

값의 존재 여부는 확인할 수 없으므로 값으로 조회하면 항상 `False`이다.

```
>>> country = {'대한민국': '서울', '프랑스': '파리'}
>>> '대한민국' in country
True
>>> '영국' in country
False

```

■ 집합

2-1. 수학에서 배운 집합

집합은 중복되는 요소가 없으며, 순서도 없는 원소(collections)의 모임이다.

파이썬의 집합은 수학의 집합에서 따온 것이기 때문에, 거의 비슷한 특징을 가진다.

집합의 특징을 알아보자.

{원소 1, 원소 2, ... 원소 n}	<ol style="list-style-type: none">1. 원소의 순서는 의미가 없다.2. 원소는 불변 값으로 중복될 수 없다.3. 합집합, 교집합 같은 연산이 가능하다.4. 멤버십 검사와 중복 제거에 사용된다.
------------------------	---

2-2. 집합 생성

내장 함수 `set()`으로 생성할 수 있다. 인자가 없으면 공집합이 생성된다.

인자는 항목이 리스트나 딕셔너리처럼 가변적인 것은 허용되지 않는다.

```
>>> s = set()
>>> type(s)
<class 'set'>
```

인자로 리스트 또는 튜플 자체를 사용할 수 있으며, 결과는 시퀀스 항목에서 중복을 제거한 원소로 구성된다. 인자로 문자열이 사용되면 각각의 문자가 원소인 집합이 생성된다. 단, 집합이므로 순서는 의미가 없다.

```
>>> set([1, 2, 3])
{1, 2, 3}
>>> set((1, 2, 2))
{1, 2}
>>> set('abc')
{'b', 'c', 'a'}
```

집합을 생성하는 다른 방법은 중괄호 `{}`안에 직접 원소를 나열하는 방법이다.

집합의 원소는 문자, 문자열, 숫자, 튜플 등과 같이 변할 수 없는 것이어야 한다.

2-3. 원소 추가, 삭제

메소드 `add(원소)`를 통해 집합에 원소를 추가할 수 있다.

```
>>> odd = {1, 3, 5}
>>> odd.add(7)
>>> print(odd)
{1, 3, 5, 7}
```

메소드 `remove(원소)`를 통해 원소를 삭제한다. 삭제하려는 원소가 없으면 오류가 발생하므로 주의해야 한다.

```
>>> odd = {1, 3, 5}
>>> odd.remove(3)
>>> print(odd)
{1, 5}
```

메소드 `discard(원소)`로도 원소를 삭제할 수 있으며, 원소가 없어도 오류가 발생하지 않는다.

메소드 `pop()`은 임의의 원소를 삭제한다.

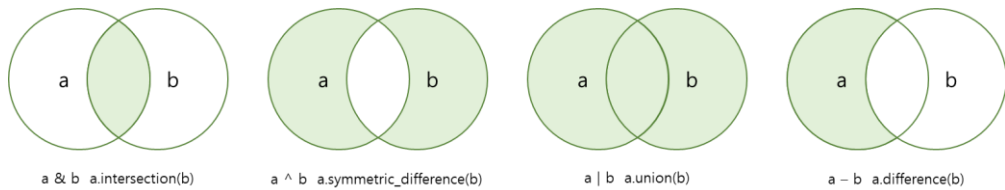
```
>>> odd = {1, 3, 5}
>>> print(odd.pop())
1
>>> print(odd)
{3, 5}
```

메소드 `clear()`를 사용하면 집합의 모든 원소가 삭제된다.

```
>>> odd = {1, 3, 5}
>>> odd.clear()
>>> print(odd)
set()
```

2-4. 합집합, 교집합, 차집합, 여집합

집합 연산을 쉽게 할 수 있다. 수학 시간에 배웠던 벤 다이어그램을 생각해보자.



A. 합집합 연산자 |와 메소드 union(), update()

메소드 union()은 합집합을 반환하며 a 자체가 수정되지는 않는다.

메소드 a.update(b)도 합집합과 같은 효과가 있지만 결과가 a에 반영된다.

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a | b
{3, 4, 6, 8, 9, 10, 12}
>>> a.union(b)
{3, 4, 6, 8, 9, 10, 12}
>>> a.update(b)
>>> a
{3, 4, 6, 8, 9, 10, 12}
```

B. 교집합 연산자 &와 메소드 intersection()

메소드 a.intersection(b)는 집합 a, b 모두에 영향을 미치지 않지만,

메소드 a.intersection_update(b)는 a를 교집합으로 수정한다.

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a & b
{12, 6}
>>> a.intersection(b)
{12, 6}
>>> a.intersection_update(b)
>>> a
{12, 6}
```

C. 차집합 연산자 -와 메소드 difference()

피연산자의 순서에 따라 결과가 달라지므로 교환 법칙은 성립하지 않는다.

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a - b
{8, 10, 4}
>>> a.difference(b)
{8, 10, 4}
>>> b - a
{9, 3}
```

D. 여집합 연산자 ^와 메소드 symmetric_difference()

여집합은 대칭 차집합이라고 부르기도 한다.

```
>>> a = {4, 6, 8, 10, 12}
>>> b = {3, 6, 9, 12}
>>> a ^ b
{3, 4, 8, 9, 10}
>>> a.symmetric_difference(b)
{3, 4, 8, 9, 10}
```

2-5. 함수 len()과 소속 연산 in

함수 len()으로 집합에 들어있는 원소의 개수를 알아볼 수 있다.

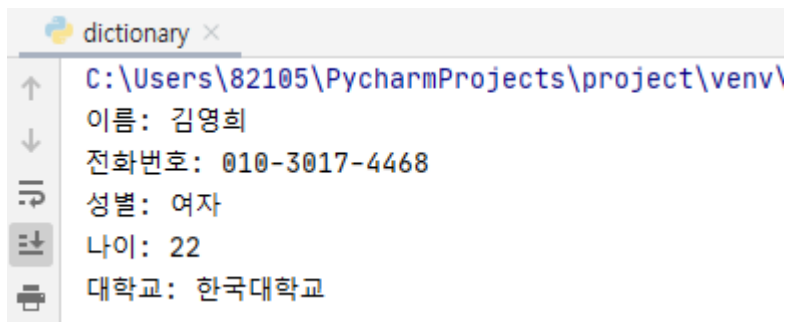
```
>>> s = {'python', 'kotlin'}
>>> len(s)
2
```

소속 연산자 in 으로 특정 원소가 집합에 있는지 확인해볼 수 있다.

```
>>> s = {'python', 'kotlin'}
>>> 'python' in s
True
>>> 'java' in s
False
```

■ 프로그래밍 예제

```
1      # 개인 정보를 딕셔너리에 저장하고 출력하는 프로그램
2      info = {'이름': '김영희', '전화번호': '010-3017-4468'}
3      info['성별'] = '여자'
4      info['나이'] = 22
5      info['대학교'] = '한국대학교'
6
7      for key in info:
8          print('%s: %s' % (key, info[key]))
```




```

1      # 주식 가격을 검색해 가격을 출력하는 프로그램
2      stock = {'삼성에스디에스': 242000, '삼성전자': 47000,
3              '엔씨소프트': 52600, '한디소프트': 5120,
4              '골프존': 215000, '기아': 56300}
5
6      while True:
7          name = input('주식 이름 ? ')
8          if name in stock:
9              key = name
10             print('%s: %dWn' % (name, stock[name]))
11         else:
12             print('주식 이름이 없습니다.')
             break

```

```

dictprice ×
C:\Users\82105\PycharmProjects\project\venv'
주식 이름 ? 엔씨소프트
엔씨소프트: 52600

주식 이름 ? 골프존
골프존: 215000

주식 이름 ? 현대
주식 이름이 없습니다.

```

마무리

- 포트폴리오를 작성하면서 지금까지 배웠던 개념을 다시 한번 정리해보며 보다 세세하게 이해하고 공부할 수 있었다. 예제를 만들면서 미리 출력 값을 예상해보기도 하고, 매 단원마다 책에 나와있는 프로그래밍 예제를 풀어보며 이론 뿐만 아니라 실습하는 능력도 향상시킬 수 있었다.
- 지금까지 배운 내용을 응용하여 실생활에서 접하는 문제를 프로그래밍하여 해결할 수 있을 거란 생각이 든다. 앞으로 계속해서 공부하게 될 문제들을 알고리즘을 통해 분석해보고, 흐름이 어떻게 이루어질지 생각해보고, 논리적으로 따져보는 연습을 많이 해보는 노력이 필요하다고 생각했다.