파이썬 포트폴리오

CHAPTER. 00

Check

목차

CHAPTER. 01

강의 계획서

Check

강 의 계 획 서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

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 누구나가 가져야할 역량이며, 인공지능, 빅데이터, 사물인터넷 등의 첨단 정보기술이 제4차 산업혁명 시대의 기술을 이끌고 있다. 제4차 산업혁명 시대를 주도하는 핵심 기술은 데이터과학과 머신러닝, 딥러닝이며,이러한 분야에 적합한 언어인 파이썬은 매우중요한 언어가 되었다. 본 교과목은 파이썬 프로그래밍의 기초적이고 체계적인 학습을 수행한다. 본 교과목을 통하여 데이터 처리 방법에 대한 효율적인 파이썬 프로그래밍 방법을 학습한다.
학습목표 및 성취수준	1. 컴퓨팅 사고력의 중요성을 인지하고 4차 산업혁명에서 파이썬 언어의 필요성을 이해할 수 있다. 2. 기본적인 파이썬 문법을 이해하고 데이터 처리를 위한 자료구조를 이해하여 적용할 수 있다. 3. 문제 해결 방법을 위한 알고리즘을 이해하고 데이터 처리에 적용 할 수 있다. 4. 파이썬 프로그램을 이용하여 실무적인 코딩 작업을 할 수 있다.

	도서명	저자	출판사	비고
주교재	파이썬으로 배우는 누구나 코딩	강환수, 신용현	홍릉과학출판사	
수업시 사용도구	파이썬 기본 도구, 파이참, 아나콘다와 주피터 노트북			
평가방법	중간고사 30%, 기말고사 40%, 과제물 및 퀴즈 10% 출석 20%(학교 규정, 학업성적 처리 지침에 따름)			
수강안내	1. 파이썬의 개발환경을 설치하고 활용할 수 있다. 2. 파이썬의 기본 자료형을 이해하고 조건과 반복 구문을 활용할 수 있다. 3. 파이썬의 주요 자료인 리스트, 튜플, 딕셔너리, 집합을 활용할 수 있다. 4. 파이썬의 표준 라이브러리와 외부 라이브러리를 이해하고 활용할 수 있다. 5. 파이썬으로 객체지향 프로그래밍을 수행할 수 있다.			

강 의 계 획 서

동양미래대학교 DONGYANG MIRAE UNIVERSITY

아시아 직업교육 허브대학

1 주차	[개강일(3/16)]
학습주제	교과목 소개 및 강의 계획 1장 파이썬 언어의 개요와 첫 프로그래밍
목표및 내용	• 파이썬 언어란 무엇인지 이해하고 이 언어가 인기 있는 이유를 설명할 수 있다. • 파이썬 개발 도구를 설치해 프로그램을 구현할 수 있다. • 파이썬의 특징과 활용 분야를 설명할 수 있다.
미리읽어오기	교재 1장, 파이썬 개발환경 설치 파이썬 IDLE
과제,시험,기타	도전 프로그래밍
2 주차	[2주]
학습주제	2장 파이썬 프로그래밍을 위한 기초 다지기
목표및 내용	• 파이썬의 재료인 문자열과 수에 대해 이해하고 코드로 구현할 수 있다. • 변수를 이해하고 다양한 대입 연산자를 활용할 수 있다. • 표준 입력으로 문자열을 입력받은 후 원하는 자료로 변환해 활용할 수 있다. • 파이썬 IDLE를 활용할 수 있다.
미리읽어오기	교재 2장 리터럴과 변수의 이해 아나콘다의 주피터 노트북
과제,시험,기타	도전 프로그래밍
3 주차	[3주]
학습주제	3장 일상에서 활용되는 문자열과 논리 연산
목표및 내용	• 문자열에서 문자나 부분 문자열을 반환하는 여러 방법을 구현할 수 있다. • 문자열 객체에 소속된 다양한 메소드를 이해하고 활용할 수 있다. • 논리 값을 이해하고 다양한 연산을 사용해 실생활에서의 표현에 활용할 수 있다. • 아나콘다의 주피터 노트북을 활용할 수 있다.
미리읽어오기	교재 3장 문자열과 논리연산 파이참(pycharm)
과제,시험,기타	도전 프로그래밍
4 주차	[4주]
학습주제	4장 일상생활과 비유되는 조건과 반복
목표및 내용	• 조건에 따라 하나를 결정하는 if문을 구현할 수 있다. • 반복을 수행하는 while문과 for문을 구현할 수 있다. • 임의의 수인 난수를 이해하고 반복을 제어하는 break문과 continue문을 활용할 수 있다. • 파이참(pycharm)을 활용할 수 있다.
미리읽어오기	교재 4장 조건과 반복
과제,시험,기타	도전 프로그래밍

강 의 계 획 서

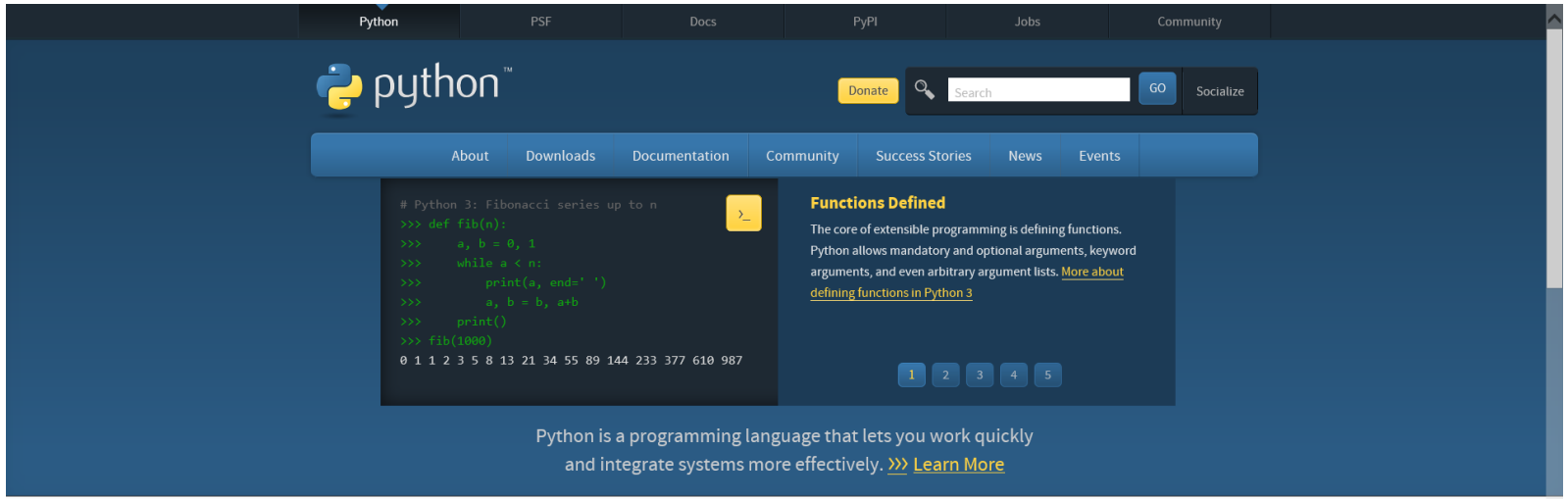
동양미래대학교 DONGYANG MIRAE UNIVERSITY

아시아 직업교육 허브대학

5 주차	[5주]
학습주제	5장 항목의 나열인 리스트와 튜플
목표및 내용	• 다양한 종류의 항목을 쉽게 나열하는 리스트를 구현할 수 있다. • 리스트에서 부분 참조 방법, 이를 이용한 수정, 리스트 연결, 삽입과 삭제 그리고 리스트 컴프리헨션 등을 구현할 수 있다. • 수정할 수 없는 다양한 종류의 항목 나열을 쉽게 처리하는 튜플을 구현할 수 있다.
미리읽어오기	교재 5장 배열과 리스트
과제,시험,기타	도전 프로그래밍
6 주차	[6주]
학습주제	6장 키와 값의 나열인 딕셔너리와 중복을 허허하는 집합
목표및 내용	• 키와 값의 쌍인 항목을 관리하는 딕셔너리를 생성하고 수정하는 방법을 이해하고, 다양한 방법으로 딕셔너리를 구현할 수 있다. • 집합의 특징을 이해하고, 합집합 등과 같은 다양한 집합의 연산을 구현할 수 있다. • 내장 함수 zip( )과 enumerate( ), 시퀀스 강의 변환을 이해하고, 구현할 수 있다.
미리읽어오기	교재 6장 집합
과제,시험,기타	도전 프로그래밍
7 주차	[7주]
학습주제	7장 특정 기능을 수행하는 사용자 정의 함수와 내장 함수
목표및 내용	• 함수의 내용과 필요성을 이해하고 함수를 직접 정의해 호출할 수 있다. • 인자의 기본 이해와 기본값 지정, 가변 인수와 키워드 인수를 활용할 수 있다. • 간편한 람다 함수와 표준 설치된 내장 함수를 사용할 수 있다.
미리읽어오기	교재 7장 함수의 정의와 호출
과제,시험,기타	도전 프로그래밍
8 주차	[중간고사]
학습주제	- 직무수행능력평가 1차(중간고사)
목표및 내용	직무수행능력평가, 서술형 평가
미리읽어오기	교재 1장에서 7장까지
과제,시험,기타	
9 주차	[9주]
학습주제	8장 조건과 반복, 리스트와 튜플 기반의 미니 프로젝트 I
목표및 내용	8개의 미니 프로젝트를 스스로 생각하고 프로그래밍해 코딩 능력뿐 아니라 문제 해결 능력을 키울 수 있다.
미리읽어오기	교재 8장
과제,시험,기타	

CHAPTER. 02

2. 파이썬 언어의 개요와 프로그래밍



파이썬 언어란?

파이썬은 배우기 쉽고 누구나 무료로 사용할 수 있는 오픈 소스 프로그래밍 언어이다.
파이썬은 현재 전세계적으로 가장 많이 가르치는 프로그래밍 언어 중 하나다.
파이썬은 배우기 쉽고 간결하며, 개발 속도가 빠르고 강력하기 때문이다.
또한 라이브러리가 풍부하고 다양한 개발 환경을 제공하고 있어 개발자가 쉽고 빠르게 소프트웨어를 개발하는 데 도움을 준다.

2. 파이썬 언어의 개요와 프로그래밍

컴퓨팅 사고력

지금은 지능, 정보화 혁명 시대인 제4차 산업혁명 시대이다. 제4차 산업혁명 시대 인재의 핵심역량은 문제 해결 능력과 창의, 융합 사고능력이다.
이러한 능력을 컴퓨팅 사고력이라고 하는데, 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 사고력은 분해, 패턴 인식, 추상화, 알고리즘 설계를 구성 요소로 정의할 수 있다.

코딩 절차에는 이해 -> 설계 -> 구현 -> 공유 순서인데 이해, 설계, 구현 절차에는 컴퓨팅 사고력 요소를 활용하고 공유 절차에는 협업과 평가를 활용한다.

CHAPTER. 02

2. 파이썬 언어의 개요와 프로그래밍

1. 쉽고 강력한 언어

파이썬은 배우기 쉽고 간결하며, 무료이고 생산성이 높다. 또한 강력한 라이브러리를 제공하고 호환되는 풀 언어이다.
컴퓨팅 사고력은 분해, 패턴 인식, 추상화, 알고리즘 설계를 구성 요소로 정의할 수 있다.

2. 빅테이터 처리와 머신 러닝 등 다양한 분야에 적합한 언어

-교육과 학술, 실무 등 다양한 분야에 사용
-인공 지능의 구현과 빅데이터 분석 처리 분야에 사용
-속도가 좀 느리다는 단점

3. 다양한 종류의 파이썬과 개발 환경

시각화, 빅데이터 처리 등의 데이터과학 분야에서는 데이터 처리 라이브러리인 넘파이, 판다스, 다스크 등이 있다.
다양한 종류의 개발 환경
-기본 IDE에 추가 : 비주얼 스튜디오 파이썬 도구 등등..
-파이썬 전용 : PyCharm 등등..
-편집기 전문 개발 환경 : Sublime Text 등등..

4. 인터프리트 방식의 언어

인터프리트 방식이란, 파이썬의 문자을 한 줄 한 줄마다 즉시 번역해 실행하는 방식이다.

3. 파이썬 기초

문자열 : 글자의 모임 => ''으로 둘러싸면 모두 문자열이다.

함수 print(출력될 자료)는 문자열이나 숫자 등의 자료를 콘솔에 출력하는 일을 수행한다.

문자열의 따옴표는 앞뒤를 동일하게 사용해야 한다.

+ 기호는 문자열에서 문자열을 연결하는 역할을 한다.

```
>>> print("python " + "program")
python program
```

3. 파이썬 기초

* 기호는 반복 연산자이다 따라서 반복 횟수인 정수가 있어야 한다.

```
>>> print('파이썬 ' + 3*"방가")
파이썬 방가방가방가
```

''' ''' 삼중 따옴표 : 문자열이 길거나 여러 줄에 걸쳐 문자열을 처리함

주석처리 : 1. ''' ''' 삼중 따옴표
2. print('# 주석문')

3. 파이썬 기초

자료형을 직접 알아보려면 type()함수를 사용한다

```
>>> type(3)
<class 'int'>
>>> type('nana')
<class 'str'>
>>>
```

변수란?
변하는 자료를 저장하는 메모리 공간이다.

Ex) unit = 3

변수 이름 규칙
문자는 대소문자의 영문자, 숫자, _로 구성되며 대소문자는 구별되고 숫자는 맨 앞에 올 수 없다.
키워드는 사용할 수 없다.

3. 파이썬 기초

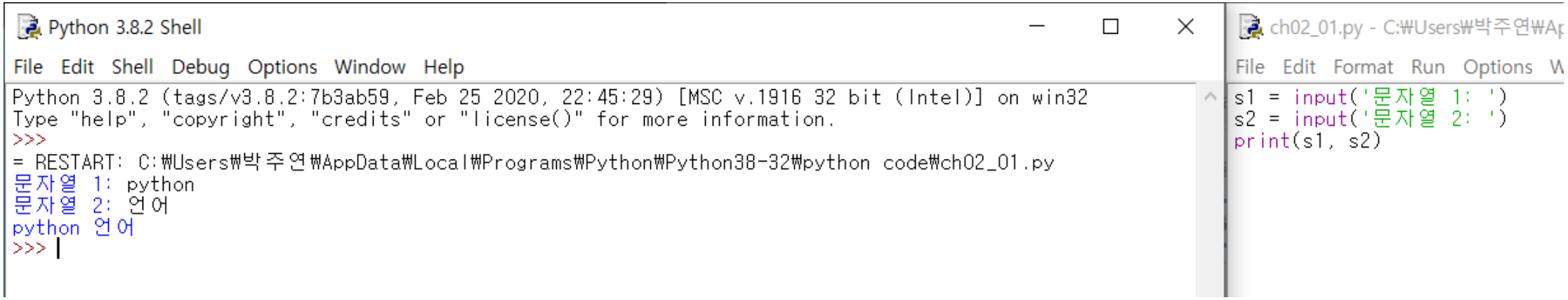
표준 입력이란?
프로그램 과정에서 쉘이나 콘솔에서 사용자의 입력을 받아 처리하는 방식을 말한다.

함수 input()으로 문자열 표준 입력

```
>>> input()
java
'java'
>>> pl = input()
python
>>> print(pl)
python
```

3. 파이썬 기초

함수 input()으로 값을 입력받아 출력 – (과제 2-1)



함수 str() : 주로 정수와 실수를 문자열로 변환하는데 사용
함수 int() : 정수 형태의 문자열
함수 float() : 소수점이 있는 실수 형태의 문자열

3. 파이썬 기초

숫자 형태의 문자열을 정수나 실수로 변환 – (과제 2-2, 2-3)

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/박주연/AppData/Local/Programs/Python/Python38-32/python code/ch02_02.py
아메리카노 몇 개 주문하세요? 7
총 가격은 24500 이다.
>>>
```

ch02_02.py - C:/Users/박주연/AppData/Local/Programs/Py

File Edit Format Run Options Window Help

```
a = input('아메리카노 몇 개 주문하세요? ')
print('총 가격은', 3500*int(a) , '이다.')
```

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/박주연/AppData/Local/Programs/Python/Python38-32/python code/ch02_03.py
온도입력 >> 38
정확 계산: 섭씨: 38 , 화씨: 100.4
약식 계산: 섭씨: 38 , 화씨: 106
차이: -5.599999999999994
>>>
```

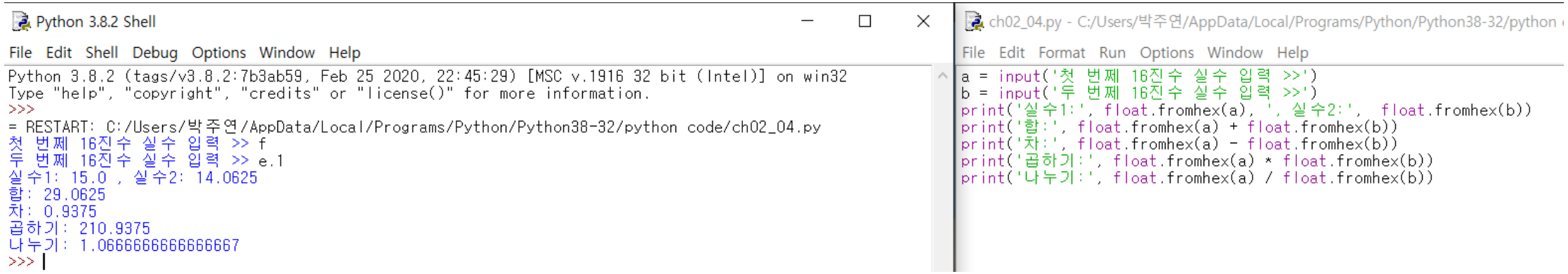
ch02_03.py - C:/Users/박주연/AppData/Local/Programs/Python/Python38-32/python code/ch(

File Edit Format Run Options Window Help

```
d = input('온도입력 >> ')
print('정확 계산: 섭씨:', int(d), ', 화씨:', float((int(d)*9/5)+32))
print('약식 계산: 섭씨:', int(d), ', 화씨:', int((int(d)*2)+30))
print('차이:', float((int(d)*9/5)+32) - int((int(d)*2)+30))
```

3. 파이썬 기초

16진수, 10진수, 8진수, 2진수 상수표현, 변환– (과제 2-4)



두 16진수를 입력받아 함수 float.fromhex()를 이용하여 사칙연산을 수행하는 프로그램 작성

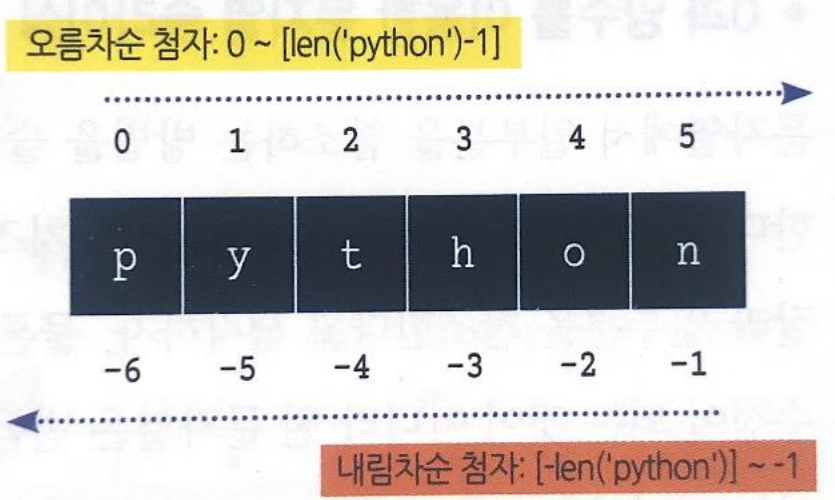
4. 문자열과 논리 연산

함수 len() 문자열의 길이 참조

```
>>> a = 'nanana'
>>> len(a)
6
```

문자열의 문자 참조

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



4. 문자열과 논리 연산

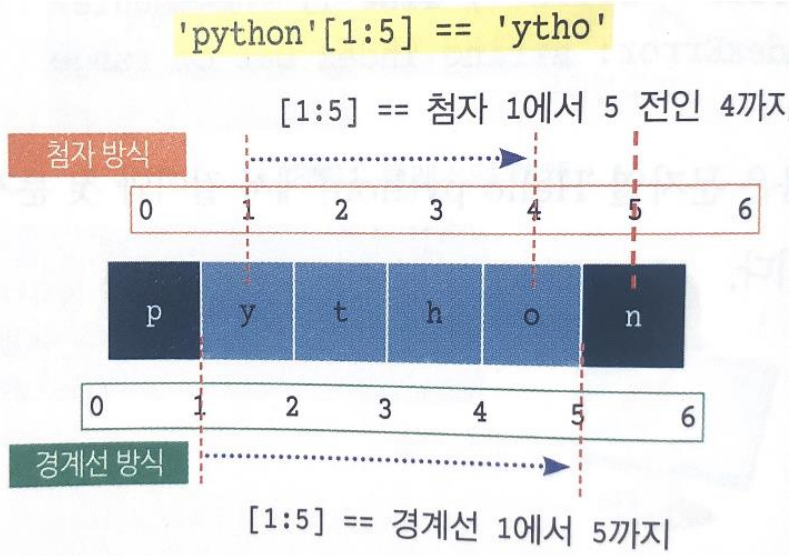
문자열의 부분 문자열 참조 방식
기존 문자열은 변함이 없고 일부분을 반환하는 것이다.

[start:end]로 부분 문자열을 반환하는데, start 첨자에서 end-1 첨자까지의 문자열을 반환한다.

```
>>> 'python'[1:5]
'ytho'
```

음수를 이용한 문자열 슬라이싱
'첨자 방식' 처럼 start 첨자에서 end-1 첨자까지의 문자열을 반환한다.

```
>>> 'python'[-4:-1]
'tho'
```



4. 문자열과 논리 연산

문자열 슬라이싱에서 순방향의 0, 양수, 역방향의 음수를 혼합해 사용할 수 있다.

Start와 end를 비우면 각각 '처음부터'와 '끝까지'를 의미한다.

Str[start:end:step]으로 문자 사이의 간격을 step으로 조정 가능하다.

문자열의 최대, 최소
내장함수 min(), max()는 인자의 최대값과 최솟값을 반환하는 함수이다.

```
>>> min('hjgfer')
'e'
>>> max('rhfhd')
'r'
```

```
>>> min(4, 89, 21)
4
```

4. 문자열과 논리 연산

함수 Ord()는 한글 문자의 유니코드 번호를 알 수 있다

함수 Char()는 유니코드 번호로 문자를 반환한다.

```
>>> ord('동')
46041
>>> chr(46041)
'동'
```

이스케이프 시퀀스 문자
 하나의 문자를 역슬래시(₩)로 시작하는 조합으로 표현하는 문자를 이스케이프 시퀀스 문자라고 한다.

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

4. 문자열과 논리 연산

클래스에 소속된 함수를 메소드라 한다.
메소드 호출은 '문자열 객체명.함수 이름(인자)' 와 같이 사용한다.

메소드 replace() : 문자열을 바꿔 반환하는 메소드

ex) str.replace('자바는', '파이썬은') -> '자바는' 이라는 문자 대신 '파이썬'을 반환한다

str.replace(' ', ' ') - > 문자열 str에서 빈 공백을 모두 없앤 문자열을 반환한다.

메소드 replace(old, new, count)는 문자열 old를 new로 대체하는데 count는 대체 횟수를 지정한다.
옵션 count가 없으면 모두 바꾸고, 있으면 앞에서부터 지정한 횟수만큼 바꾼다.

```
>>> str = '파이썬 파이썬 파이썬'
>>> str.replace('파이썬', 'python', 2)
'python python 파이썬'
```

4. 문자열과 논리 연산

문자열을 표준 입력으로 받아 소속 문자를 참조할 범위를 출력하고
다시 첨자를 입력 받아 문자열의 전체와 참조 문자를 출력하는 프로그램 (과제 3-1)

```
str = input('문자열 입력 >> ')
str[:]
n =len(str)
print('참조할 첨자: 0 ~', n-1)
num = input('참조할 첨자 입력 >> ')
print('문자열 :', str, '길이:', n)
print('참조 문자:', str[int(num)])
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:4
Type "help", "copyright", "credits" or "license()" f
>>>
= RESTART: C:/Users/PJY/AppData/Local/Programs/Pythc
문자열 입력 >> Python is a good language!
참조할 첨자: 0 ~ 25
참조할 첨자 입력 >> 12
문자열 : Python is a good language! 길이: 26
참조 문자: g
>>>
```

4. 문자열과 논리 연산

메소드 replace()를 이용해 문자열을 바꿔 출력하는 프로그램 (과제 3-2)

str.replace('a', 'b') -> 'a' 이라는 문자 대신 'b'을 반환한다

```
str = 'Beautiful is better than ugly.'
print('위 철학을 메소드 replace()를 사용해 다음 철학으로 다시 저장')
str.replace('Beautiful is better than ugly.', 'Explicit is better than implicit.')
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (Intel)] on w
Type "help", "copyright", "credits" or "license()" for more information.
>>> str = 'Beautiful is better than ugly.'
>>> print('위 철학을 메소드 replace()를 사용해 다음 철학으로 다시 저장')
위 철학을 메소드 replace()를 사용해 다음 철학으로 다시 저장
>>> str.replace('Beautiful is better than ugly.', 'Explicit is better than implicit.')
'Explicit is better than implicit.'
>>>
```

4. 문자열과 논리 연산

문자열 split() 메소드를 사용해 시작 정보를 표준으로 입력받아 다시 시, 분, 초로 출력하는 프로그램 (과제 3-3)

split() 메소드 : 문자열에서 공백을 기준으로 문자열을 나눠준다.
split(','), split(':') 처럼 괄호 안에 특정한 문자열 값이 있을 경우에 구분자를 이용하여 문자열을 나눠준다

```
hours, mins, secs = input('시각 정보(16:30:15) 입력 >>')split(':')
print('입력 시각 정보:', hours, ':', mins, ':', secs)
print(hours, '시', mins, '분', secs, '초')

= RESTART: C:/Users/PJY/AppData/Local/
시각 정보(16:30:15) 입력 >> 17:26:52
입력 시각 정보: 17 : 26 : 52
17 시 26 분 52 초
>>>
```

4. 문자열과 논리 연산

2진수, 8진수, 16진수를 정형화해 출력하는 프로그램 (과제 3-4)

- Format() : 출력을 정형화하는 함수
문자열 중간중간에 변수나 상수를 함께 출력하는 방법이다.

- 문자열 '{} + {} = {}'에서 {}이 위치한 부분에 Format(3, 4, 3+4) 인자인 3, 4, 7이 순서대로 출력된다.

- {n:md}로 정수를 형식 유형으로 출력할 수 있다.

10진수 정수는 d, 부동소수는 f로 표기한다.
m : 출력 폭을 지정한다.
ex) 5.3f은 소수점이 포함된 전체 폭 5,
소수점 이하 폭 3을 의미한다.

```
a = -7 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-7, -7 & 0xff, a, a))
a = -6 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-6, -6 & 0xff, a, a))
a = -5 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-5, -5 & 0xff, a, a))
a = -4 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-4, -4 & 0xff, a, a))
a = -3 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-3, -3 & 0xff, a, a))
a = -2 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-2, -2 & 0xff, a, a))
a = -1 & 0xff
print('10진수: {}  2진수: {:08b} 8진수: {:2o} 16진수: {:2x}'.format(-1, -1 & 0xff, a, a))
a = 0 & 0xff
print('10진수: {}  2진수: {:8b} 8진수: {:2o} 16진수: {:2x}'.format(-0, -0 & 0xff, a, a))
```

```
= RESTART: C:₩Users₩PJY₩AppData₩Local₩Programs₩Python₩P
10진수: -7  2진수: 11111001 8진수: 371 16진수: f9
10진수: -6  2진수: 11111010 8진수: 372 16진수: fa
10진수: -5  2진수: 11111011 8진수: 373 16진수: fb
10진수: -4  2진수: 11111100 8진수: 374 16진수: fc
10진수: -3  2진수: 11111101 8진수: 375 16진수: fd
10진수: -2  2진수: 11111110 8진수: 376 16진수: fe
10진수: -1  2진수: 11111111 8진수: 377 16진수: ff
10진수: 0  2진수:        0 8진수:  0 16진수:  0
>>>
```

CHAPTER. 04

4. 문자열과 논리 연산

논리 자료와 다양한 연산

- 논리 유형 bool과 함수 bool() : True와 False를 키워드로 제공한다.
- 곱과 합 연산자 and 와 or : and와 &는 두 항이 모두 참이어야 True 이고 or과 |는 하나라도 참이면 True이다.
- 베타적 논리 합 연산자 ^와 not 연산자 : ^는 두 항이 다르면 True이다. Not은 뒤에 위치한 논리 값을 바꾼다.
-

5. 조건과 반복

월(month)를 표준 입력으로 입력받아 계절을 출력하는 프로그램 (과제 4-1)

- 조건에 따른 선택을 결정하는 if문 : 조건인 논리 표현식의 결과가 True이면 이후 구성된 블록 구문인 문장을 실행한다.

- if~else문 : 조건 if의 논리 표현식 결과는 True, false 이며, else : 블록은 논리표현식 결과가 false일 때 실행된다.

- if ~elif문 : 여러 조건 중에서 하나를 선택하는 구문이다. 논리 표현식1이 false여야 elif 논리 표현식2 가 실행된다.

```
month = int(input('월 입력 ? '))
if  3 >= month:
    print('{}월 겨울'.format(month))
elif 8 >= month:
    print('{}월 여름'.format(month))
elif 10 >= month:
    print('{}월 가을'.format(month))
elif 12 >= month:
    print('{}월 겨울'.format(month))
```

```
= RESTART: C:/L
월 입력 ? 3
3월 겨울
>>>
```

5. 조건과 반복

난수인 임의의 정수 3개를 대상으로 가장 큰 정수를 출력하는 프로그램 (과제 4-2)

- 함수 randint(시작, 끝)를 사용해 임의의 정수를 얻을 수 있다.

+ from 모듈 import randint
  //구문 사용하면 randint() 바로 사용가능

Ex) from 모듈 import randint
    randint(1,8) // 모듈 없이 바로 함수() 사용 가능

```
from random import randint
a = randint(1, 99)

from random import randint
b = randint(1, 99)

from random import randint
c = randint(1, 99)

print(a , b, c)

if a > b >= c:
    print(a)
elif a > c > b:
    print(a)
elif b > a >= c:
    print(b);
elif b > c > a:
    print(b);
else:
     print(c)
```

```
= RESTART:
85 68 78
85
>>>
```

5. 조건과 반복

섭씨 온도 20도에서 41도까지 3도씩 증가하면서 화씨로 변환해 출력하는 프로그램 (과제 4-3)

반복을 제어 하는 for문과 while문

while <반복 조건>:
    반복 몸체인 문장들

for 변수 in <시퀀스>:
    반복 몸체인 문장들

내장 함수 range()를 사용한 for문
= for i in range(start, stop, step)

Start : 시퀀스의 시작, stop: 시퀀스 끝(stop-1 까지), step: 증가 값

```
for d in range(20, 42, 3):
    print('섭씨:', int(d), '화씨:', float((int(d)*9/5)+32), '화씨(약식):', int((int(d)*2)+30),
          '차이:%.2f' %abs((float((int(d)*9/5)+32)-(int((int(d)*2)+30)))))
```

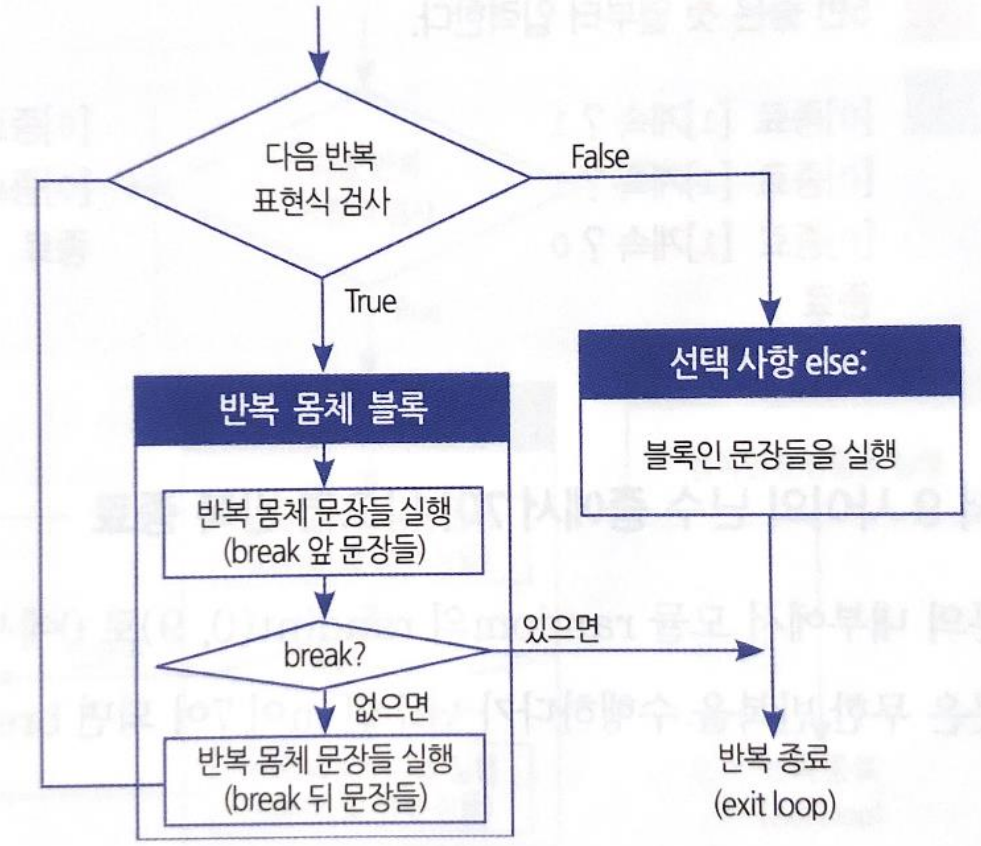
```
섭씨: 20 화씨: 68.0 화씨(약식): 70 차이:2.00
섭씨: 23 화씨: 73.4 화씨(약식): 76 차이:2.60
섭씨: 26 화씨: 78.8 화씨(약식): 82 차이:3.20
섭씨: 29 화씨: 84.2 화씨(약식): 88 차이:3.80
섭씨: 32 화씨: 89.6 화씨(약식): 94 차이:4.40
섭씨: 35 화씨: 95.0 화씨(약식): 100 차이:5.00
섭씨: 38 화씨: 100.4 화씨(약식): 106 차이:5.60
섭씨: 41 화씨: 105.8 화씨(약식): 112 차이:6.20
>>>
```

5. 조건과 반복

횟수를 정하지 않은 반복에 적합한 while 반복

While문은 for문에 비해 간결하며 반복 조건인
논리 표현식의 값에 따라 반복을 수행한다.

=> 어떤 조건문이 false가 될 때까지 반복을 수행한다.



반복 break문과 continue문

- break는 else: 블록을 실행시키지 않고 반복을 무조건 종료한다.
- continue는 반복 몸체를 실행시키지 않고 다음 반복을 위해
  논리 조건을 수행한다.

6. 리스트와 튜플

리스트란?

리스트는 항목의 나열인 시퀀스이다. <각 항목은 모두 같은 자료형일 필요는 없다.>

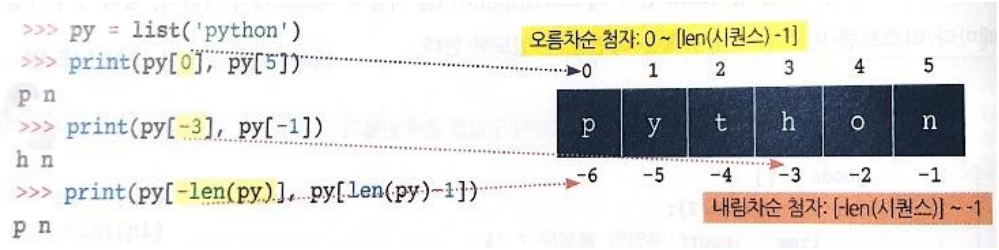
Ex) menu = ['치킨', '피자', '떡볶이'] //대괄호 내부의 항목의 나열은 리스트이다.

리스트의 항목 추가

메소드 append(삽입할 항목) : 리스트 가장 뒤에 항목을 추가하기 위한 메소드이다.

리스트의 항목 참조

리스트에서 첨자(index)를 사용해 항목을 참조할 수 있다.



6. 리스트와 튜플

매소드 count(값) : 값을 갖는 항목의 개수를 알 수 있다.

메소드 indext : 인자인 값의 항목이 위치한 첨자를 반환한다.

리스트의 첨자를 이용하여 리스트의 항목을 수정할 수 있다.

ex) top = [‘태연’, ‘엑소’, ‘하이라이트 ']
top[1] = ‘엔시티드림’
=>[‘태연’, ‘엔시티드림’, ‘하이라이트 ']

리스트 내부에 다시 리스트를 포함하는 중첩 리스트

```
animal = [['사자', '코끼리', '호랑이'], '조류', '어류']
bird = ['독수리', '참새', '까치']
fish = ['갈치', '붕어', '고등어']
animal[1:] = [bird, fish]
```

실행 결과

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

6. 리스트와 튜플

리스트의 부분 참조인 슬라이싱

리스트[start:stop:step] : 첨자 start에서 첨자 stop-1까지 step 간격으로 부분 리스트 반환

리스트의 부분 수정

```
>>> sports = ['풋살', '족구', '비치사커', '야구', '농구', '배구']
>>> sports[0:3] = ['축구']
>>> print(sports)
['축구', '야구', '농구', '배구']
```

리스트의 일부분을 다른 리스트로 수정하려면 슬라이스 방식에 대입해야 한다.
즉 리스트의 일부분이 대입한 리스트로 대체된다.

※ 리스트에 항목을 대입하면 오류가 발생한다.

※ sports[0:3] = '축구' 로 대입하면 '축구'를 리스트로 만들어 대입하므로
결과는 항목으로 '축 ' '구 ' 가 대입된다.

6. 리스트와 튜플

리스트의 항목 삽입과 삭제

리스트 메소드 insert(첨자, 항목)

=>리스트의 첨자 위치에 항목을 삽입한다.

리스트 메소드 remove(값), pop(첨자), pop()

=> 리스트에서 지정된 값의 항목을 삭제한다.

리스트 메소드 pop(첨자), pop()

=> 지정된 첨자 항목, 마지막 항목을 삭제하고 반환한다.

리스트의 추가, 연결과 반복

리스트 메소드 extend(list)

⇒ 리스트에 인자인 list를 가장 뒤에 추가한다

리스트와 정수를 *로 곱하면 항목이 지정된 정수만큼 반복된 리스트를 반환한다.

```
>>> days = ['월', '화']
>>> print(days * 3)
['월', '화', '월', '화', '월', '화']
```

6. 리스트와 튜플

시퀀스 튜플이란?

튜플은 문자열, 리스트와 같은 항목의 나열인 시퀀스이다.

❖ 튜플은 리스트와 달리 항목의 순서나 내용의 수정이 불가능하다.

튜플은 괄호() 사이에 항목을 기술한다. (괄호는 생략할 수 있다.)

Ex) singer = ('태연', '엔시티드림', '아이유', '하이라이트')
food = ('피자', '치킨', '버블티')
space = '밤', '낮', '해'

튜플 항목 참조와 출력

튜플도 리스트와 같이 첨자 참조 tuple[index]와 tuple[start:stop:step] 슬라이스가 가능하다.

❖ 튜플은 수정이 불가능하므로 첨자와 슬라이스로 수정할 수 없다.

```
>>> nation = '대한민국', '뉴질랜드', '캐나다'
>>> city = ('부산', '웰링톤', '몬트리올')
>>> nation[1]
'뉴질랜드'
>>> city[1:3]
('웰링톤', '몬트리올')
```

CHAPTER. 07

7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

딕셔너리란?

딕셔너리는 키와 값의 쌍인 항목을 나열한 시퀀스이다.
즉, 딕셔너리는 콤마(,)로 구분된 항목들의 리스트로 표현된다.
각각의 항목은 키 : 값과 같이 키와 값을 콜론으로 구분하고 전체는 중괄호{}로 묶는다.

- 딕셔너리 항목 순서는 의미가 없으며, 키는 중복될 수 없다.
- 키는 수정도리 수 없지만, 값은 수정될 수 있다.
- 값은 키로 참조된다.

Ex) mycar = {<key>:<value>, <key>:<value>, <key>:<value>}

함수 dict()로 생성하는 딕셔너리

내장 함수 dict() 함수로 빈 딕셔너리를 생성할 수 있고, 인자로 리스트나 튜플 1개를 사용할 수 있다.
키 = 값 항목 나열로도 지정할 수 있다.
Ex) food = dict(피자='pizza', 치킨='chicken')

- 딕셔너리 키는 수정 불가능한 객체로 사용한다.
- 튜플은 키로 가능하지만 리스트는 키로 사용할 수 없다.

7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

딕셔너리 항목의 순회

❖ 딕셔너리 메소드 keys() : 키로만 구성된 리스트를 반환한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> print(food.keys())
dict_keys(['피자', '치킨'])
```

❖ 딕셔너리 메소드 items() : (키, 값) 쌍의 튜플이 들어있는 리스트를 반환한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> print(food.items())
dict_items([('피자', 'pizza'), ('치킨', 'chicken')])
```

❖ 딕셔너리 메소드 values() : 값으로 구성된 리스트를 반환한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> print(food.values())
dict_values(['pizza', 'chicken'])
```

7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

딕셔너리 항목의 참조와 삭제

메소드 get(키[, 키가 없을 때 반환 값]) : 키의 해당 값을 반환한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> food.get('피자')
'pizza'
```

메소드 pop(키[, 키가 없을 때 반환 값]) : 키인 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> print(food.pop('치킨'))
chicken
>>> food
{'피자': 'pizza'}
```

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

❖ 만약 데이터가 하나도 없다면 오류가 발생한다.

```
>>> food = dict(피자='pizza', 치킨='chicken')
>>> print(food.popitem())
('치킨', 'chicken')
```

딕셔너리를 문장 del에 이어 키로 지정하면 해당 항목이 삭제된다.

CHAPTER. 07

7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

중복과 순서가 없는 집합

파이썬에서 집합은 수학과 같이 원소를 콤마로 구분하며 중괄호로 둘러싸 표현한다.

Ex) s1 = {12, 13, 14, 15}

- 원소는 불변의 값으로 중복될 수 없고 서로 다른 값이어야 한다.
- 원소의 순서는 의미가 없다.

내장함수 set()을 활용한 집합 생성

set(원소로 구성된 리스트 / 튜플/ 문자열)

- 인자가 없으면 공집합이 생성된다.
- 수정 가능한 리스트와 딕셔너리는 집합의 원소로 사용할 수 없다.

집합 메소드 add(원소) : 이미 만들어진 집합에 원소를 추가한다.
집합 메소드 remove(원소) : 원소를 삭제한다. (삭제하려는 원소가 없으면 오류가 발생한다.)
집합 메소드 discard(원소) : 원소를 삭제한다. 원소가 없어도 오류가 발생하지 않는다
집합 메소드 pop() : 임의의 원소를 삭제하려면 pop()을 사용해야 한다.

CHAPTER. 07

7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

집합의 주요 연산인 합집합, 교집합, 차집합, 여집합

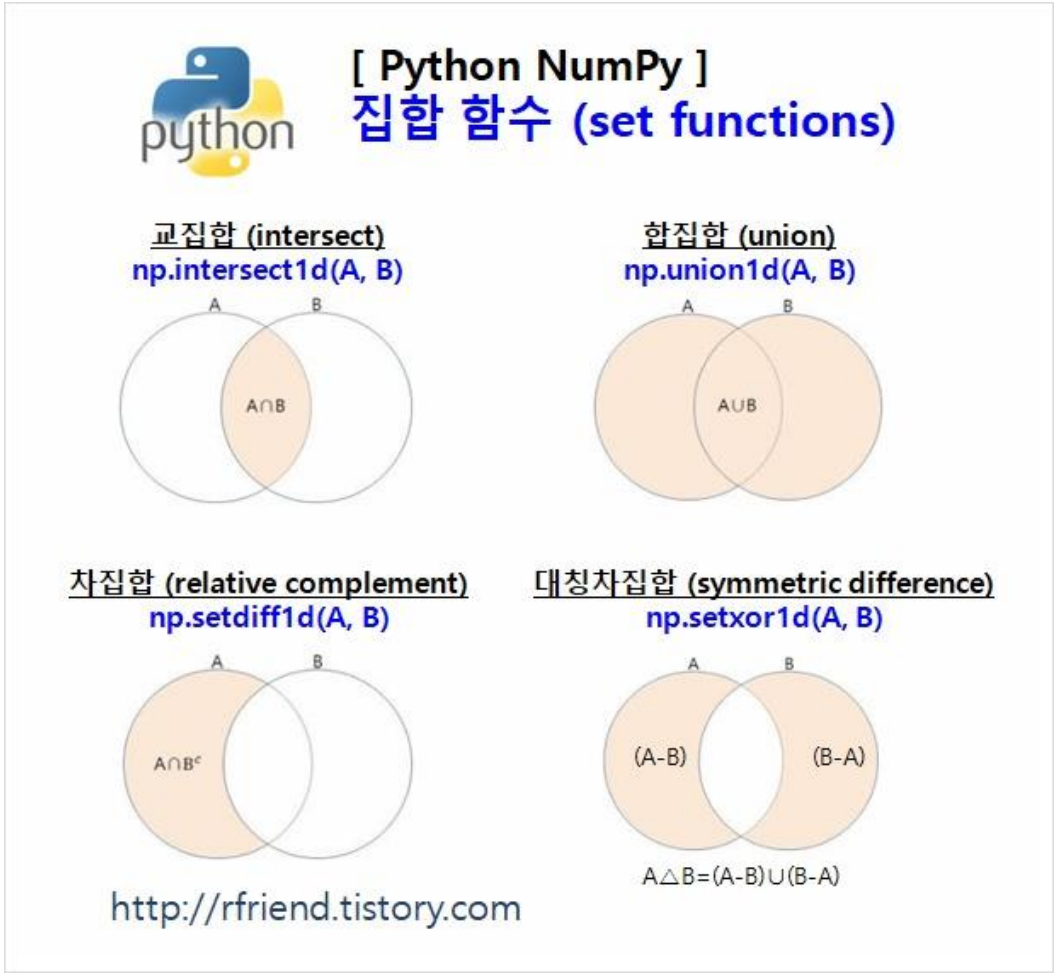
- 합집합 연산자 | 와 메소드 union(), update()
  -> union()은 합집합의 결과가 반영되어 수정되지 않지만
  update()은 합집합의 결과가 호출하는 집합에 반영된다.

- 교집합 연산자 &와 메소드 intersection()

- 차집합 연산자 –와 메소드 difference()
  -> 피연산자의 순서에 따라 결과가 달라지므로
  교환법칙이 성립하지 않는다.

- 여집합 연산자 ^와 메소드 symmetric_difference()

- 집합 원소 수 함수 len()
  : 함수 len()으로 집합에 들어있는 원소의 개수를 확인한다.
- 소속 연산자 in
  : 특정 원소가 집합에 있는지 확인한다. (true/false 로 반환)



7. 키와 값의 나열인 딕셔너리와 중복을 불허하는 집합

내장 함수 zip()란?

동일한 수로 이뤄진 여러 개의 튜플 항목 시퀀스를 각각의 리스트로 묶어주는 역할을 한다.
(리스트나 튜플 항목으로 조합된 항목을 생성하는 내장 함수이다.)

```
>>> a = ['nana', 'hihi', 'haha']
>>> b = (12, 23, 34)
>>> z = zip(a, b)
>>> type(z)
<class 'zip'>
>>> list(zip(a, b))
[('nana', 12), ('hihi', 23), ('haha', 34)]
```

내장 함수 enumerate()란?

0부터 시작하는 첨자와 항목의 값의 튜플 리스트를 생성한다.
(첨자를 자동으로 만들어 첨자와 값과의 쌍인 튜플을 만들어주는 내장 함수이다.)

```
>>> s1 = 'good'
>>> list(enumerate(s1))
[(0, 'g'), (1, 'o'), (2, 'o'), (3, 'd')]
```

시퀀스 간의 변환

내장 함수 list()와 tuple(), set(), dict()등을 사용하여 시퀀스를 간편하게 변환할 수 있다.

8. 맺음말

맺음말.

파이썬 포트폴리오를 제작하면서 그동안 공부했던 내용들을 한 번 더 돌이켜 보는 시간이 되었습니다.
수업을 듣고 과제를 하며 어느 정도의 개념은 알고 있었다고 생각했는데 개념을 한 번 더 자세히 보니
생각보다 모르고 있던 부분들이 많았다는 것을 알게 되는 좋은 시간이었습니다.
또한 책의 예제들을 구현해 보며 알고 있던 내용도 한 번 더 곱씹어 볼 수 있는 시간이었습니다.
제대로 된 포트폴리오를 아직 한 번도 못 만들어 봤는데 이 기회에 어떤 형식으로 만들어야 되는지 구상을 하며
나중에 더 나은 퀄리티의 포트폴리오를 제작할 수 있는 경험이 되었습니다.
또한 전에는 그나마 익숙한 내용을 공부했다면 앞으로 배울 내용은 생소한 내용이 많아 수업에 더 집중해야겠다고
느꼈습니다.
지금까지 제 포트폴리오를 읽어주셔서 감사합니다.