

2020 1학기

Python 중간고사

포트폴리오

학과 : 컴퓨터 정보 공학과

반 : PC

학번 : 20171586

이름 : 남궁승찬

선정 과목 : Python

목차

1. 강의계획서
2. 학습내용(Chapter1 ~ 6)
 - Chapter마다 실습내용 포함
3. 마무리
4. 후기

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.			



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	
학습주제	
목표및 내용	
미리읽어오기	
과제,시험,기타	
8 주차	
학습주제	
목표및 내용	
미리읽어오기	
과제,시험,기타	
9 주차	
학습주제	
목표및 내용	
미리읽어오기	
과제,시험,기타	

Chapter.1

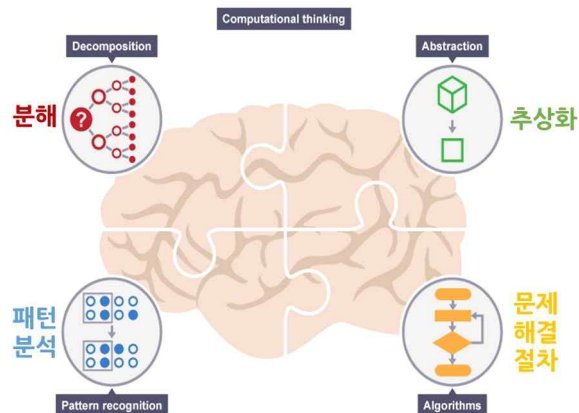
파이썬 언어의 개요와 첫 프로그래밍

1-1. 파이썬 언어란?

- 파이썬은 배우기 쉽고 누구나 무료로 사용 할 수 있는 오픈 소스 프로그래밍 언어다. 파이썬은 1991년 네덜란드의 귀도 반 로섬이 개발했으며, 현재는 비영리 단체인 파이썬 소프트웨어 재단이 관리하고 있다.

1-2. 컴퓨팅 사고력과 파이썬

- 문제 해결 능력과 창의,융합 사고 능력에 필요한 컴퓨팅 사고력이 필요하며 컴퓨팅 사고력이란 컴퓨터 과학의 기본 개념과 원리 및 컴퓨팅 시스템을 활용해 실생활 및 다양한 학문 분야의 문제를 이해하고 창의적 해법을 구현해 적용할 수 있는 능력으로 1.추상화 능력, 2.자동화 능력, 3.창의,융합능력이 있으며 컴퓨팅 사고력 구성 요소로는 분해: 데이터, 프로세스 또는 문제를 작고 관리 가능한 부분으로 나눔, 패턴 인식: 데이터의 패턴,추세 및 정규성 관찰, 추상화: 패턴을 생성하는 일반 원칙을 규정, 알고리즘 설계: 이 문제와 유사한 문제 해결을 위한 단계별 지침을 개발을 가지고 있다.

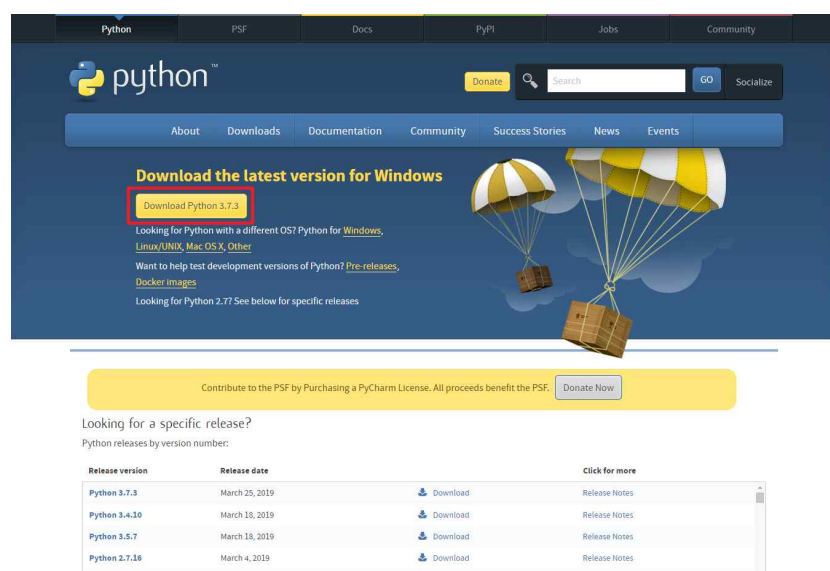


또한 프로그래밍의 절차로는 이해(U:understand), 설계(D:design), 구현(I:Implement), 공유(S:share)의 절차인(UDIS)를 따르며, 각 단계에서 컴퓨팅 사고력의 요소를 활용하고 체험하게 된다.



2-1. 파이썬 설치

1. 우선 파이썬 공식 홈페이지의 다운로드 페이지(<http://www.python.org/downloads>)에서 윈도우용 파이썬 언어 패키지를 다운로드한다. 다음 화면에서 Python 3.x로 시작하는 버전 중 가장 최근의 윈도우 인스톨러를 다운로드 한다 (이 글을 작성하는 시점의 최신 버전은 3.7.3이다).



2. 인스톨러를 실행한 후에 "Install Now"를 선택하면 바로 설치가 진행된다. 파이썬이 어느 곳에서든지 실행될 수 있도록 "Add Python 3.7 to PATH" 옵션을 반드시 선택해야 한다.



3. 설치가 완료되면 [close]를 클릭하여 종료한다.

파이썬이 정상적으로 설치되었다면 다음 그림과 같이 프로그램 메뉴에서 확인할 수 있다.

[시작 → 모든 프로그램 → Python 3.7]



1-1. 파이썬 셸에서 첫 대화형 코딩.

```
File Edit Format Run Op ==
print('Hello World!') Hello World!
>>> |
```

1-2. 두 줄의 출력이 있는 파이썬 두 번째 프로그램 작성.

```
print('Hello World!') Hello World!
print('Hi,python') Hi,python
>>> |
```



chapter.1의 실습문제

교과서 p.29의 3문제

1.

```
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.
>>> print('안녕, 파이썬!')
안녕, 파이썬!
>>>
```

2.

 hellopython.py - C:\Users\Wt\Desktop\python\hellopython.py

File Edit Format Run Options Window Help

```
print('파이썬은 재미있는 언어이다.')
```

```
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\Wt\Desktop\python\hellopython.py =====
파이썬은 재미있는 언어이다.
>>> |
```

3.

 introduce.py - C:\Users\Wt\Desktop\python\introduce.py (3.8.2)

File Edit Format Run Options Window Help

```
print('이름: 남궁승찬')
print('대학: 중앙미래대학교')
print('전공: 컴퓨터공학과')
```

```
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\Wt\Desktop\python\introduce.py =====
이름: 남궁승찬
대학: 중앙미래대학교
전공: 컴퓨터공학과
>>> |
```

Chapter.2

파이썬 프로그래밍을 위한 기초 다지기

1. 다양한 자료: 문자열과 수

1-1. 자료의 종류와 문자열 표현

-1. 글자의 모임인 문자열 :

파이썬에선 문자 하나 또는 문자가 모인 단어나 문장 또는 단락 등을 문자열(string)이라 칭함.

문자열의 따옴표는 앞뒤를 동일하게 사용

```
1-1 print("Hello World!") #실행 불가
```

1-2. 문자열 연산자+,*와 주석

-1. 문자열 연결과 반복 연산자 :

더하기 기호인 +는 문자열의 연결을 뜻하며 *는 곱하기를 정수나 실수 등의 앞뒤에 따옴표를 사용하면 문자열이 됨.

-2. 문법과 상관 없는 주석 :

#이후는 주석을 의미하며 “ ” 사이의 글 또한 주석처리가 됨.

```
2 print(3 * "원의 원주율 "+"3.141592") #>> 3.141592는 문자열이 됨.  
#>> 3* "원의 원주율"은 3번 입력됨.  
3 ''' 이곳은 주석처리가 됨 '''  
#( # 또한 주석처리가 되며 한줄에서 문장 이후도 사용 가능)
```

1-3. 정수와 실수의 다양한 연산

-1.수의 +,-,*,/ 연산자 :

파이썬 셸은 간단한 계산기로 사용할 수도 있으며 수의 연산인 +,-,*,/ 가 사용 가능하다.

```
>>> 5+8
13
>>> 5-3
2
>>> 5*3
15
>>> 5/3
1.6666666666666667
```

-2.수의 //,%,** 연산자 :

정수 나눗셈 연산자 // 는 나눈 몫이 결과이기에 나누기 양수에서는 나누기/연산에서 소수부 없이 정수만 남는다 나머지 연산자%는 나눈 나머지가 결과이며 연속한 별표 **는 지수승 연산자다

```
>>> 8/5
1.6
>>> 8//5
1
>>> 5%3
2
>>> 5**3
125
```

-3.최근 결과의 특별한 저장 장소(_):

대화형 모드에서 마지막에 실행된 결과값은 특별한 저장공간인 _에 대입된다.

```
>>> 5**3
125
>>> _
125
```

-4.표현식 문자열 실행함수 eval() :

함수 eval('expression')은 실행 가능한 연산식 문자열인 expression을 실행한 결과를 반환한다 이에 함수 인자는 반드시 연산 가능한 문자열이어야 하며 내부에서 문자열 표현은 인자인 문자열 작은따옴표와 달리 큰따옴표를 사용해야 하며, 결과도 문자열이다.

```
>>> eval('4*3%5')
2
>>> eval('"'java"*3')
'javajavajava'
```

2.변수와 키워드, 대입 연산자

2-1. 자료형

-1.자료형과 type() 함수 :

지금까지 알아본 정수와 실수,문자열 등을 자료형 이라 한다. 대화형 모드에서 자료형을 직접 알아보려면 type() 함수를 이용해야 한다.

```
>>> type(3)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type('python')
<class 'str'>
>>> type(3+4j)
<class 'complex'>
```

2-2. 변수와 대입연산자

-1.변수의 이해와 =를 이용한 값의 저장 :

변수란 말 그대로 변하는 자료를 저장하는 메모리 공간이다. 그러므로 변수 이름은 저장 값에 의미 있는 이름을 붙이며 대

입 연산자의 왼쪽에는 반드시 저장 공간인 변수가 와야 한다.

```
>>> unit =3
>>> 8 = unit
SyntaxError: cannot assign to literal
```

-2.변수 이름을 붙일 때의 규칙 :

문자는 대소문자의 영문자,숫자 그리고 _로 구성되며,대소문자는 구별되고. 숫자는 맨 앞에 올 수 없으며 import, True, False 등과 같은 키워드는 사용할 수 없다.

종 류	의 미
=	우항을 좌항에 대입
+=	좌항과 우항을 더한 값을 좌항에 대입
-=	좌항에 우항을 뺀 값을 좌항에 대입
*=	좌항에 우항을 곱한 값을 좌항에 대입
**=	좌항에 우항을 제곱한 값을 좌항에 대입
/=	좌항에 우항을 나눈 값을 좌항에 대입(실수)
//=	좌항에 우항을 나눈 값을 좌항에 대입(정수)
%=	좌항을 우항으로 나눈 나머지를 좌항에 대입

-3.한 번에 여러 자료 대입 :

파이썬에서는 콤마로 구분된 여러 변수에 순서대로 값을 대입할 수 있으며 두 변수에 저장된 값을 교환(swap)하는 코드가 있다. 또한 함수 divmode(a,b)는 나누기 몫 연산//와 연산%를 함께 수행한다.

```
>>> a,b=5,9
>>> temp=a
>>> a=b
>>> b=temp
>>> print(a,b)
9 5
>>> divmod(28,3)
(9, 1)
```

3.자료의 표준 입력과 자료 변환 함수

3-1. 표준 입력과 다양한 변환 함수

-1.함수 input()으로 문자열 표준 입력 :

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

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

-2. 문자열과 정수,실수 간의 자료 변환 함수 :

함수 str()은 주로 정수와 실수를 문자열로 변환하는데 사용함.
함수 int()는 정수 형태의 문자열등을 정수로 float()는 소수점이 있는 실수 형태의 문자열을 실수로 변환한다.

```
>>> str(235)
'235'
>>> int('200')
200
>>> float('3.1415')
3.1415
```

-3. 16,8,2진수의 활용 :

수의 상수를 표현하는 방법으로는 16진수는 0x, 8진수는 0o , 2진수는 0b이다.

```
>>> 0x1f,0o17,0b11
(31, 15, 3)
```

-4. 10진수의 변환 함수bin(),oct(),hex() :

```
>>> bin(7),oct(8),hex(14)
('0b111', '0o10', '0xe')
>>> #각각 2진수,8진수,16진수로 바뀐다.
```

chapter.2의 실습문제

교과서 p.68의 4문제

1.

```
no2.py - C:/Users/t/Desktop/2020 1학기/python/과제2/no2.py (3.8.2) Python 3.8.2 Shell
File Edit Format Run Options Window Help
km = input('차의 속도를 입력(km) >>')
print(km, '(km)은', int(km) / 1.61, '이다.')

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/t/Desktop/2020 1학기/python/과제2/no2.py =====
차의 속도를 입력(km) >>135
135 (km)은 83.85093167701862 이다.
>>>
```

2.

```
no4.py - C:/Users/t/Desktop/2020 1학기/python/과제2/no4.py (3.8.2) Python 3.8.2 Shell
File Edit Format Run Options Window Help
print('알려진 지구 둘레: 40120')
all = 2 * 3.141592 * 6378.1
print('지구와 같은 원둘레: ', float(all))
print('차이: ', 40120 - float(all), '(km)')

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/t/Desktop/2020 1학기/python/과제2/no4.py =====
알려진 지구 둘레: 40120
지구와 같은 원둘레: 40074.77587040001
차이: 45.2241295999201 (km)
>>>
```

3.

```
no6.py - C:/Users/t/Desktop/2020 1학기/python/과제2/no6.py (3.8.2) Python 3.8.2 Shell
File Edit Format Run Options Window Help
fn = int(input('Enter First number: '))
sn = int(input('Enter Second number: '))

print(fn, '/', sn, '=>', fn / sn)
print(fn, '%', sn, '=>', fn % sn)
print(fn, '//', sn, '=>', fn // sn)
print(fn, '**', sn, '=>', fn ** sn)

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/t/Desktop/2020 1학기/python/과제2/no6.py =====
Enter First number: 12
Enter Second number: 5
12 / 5 => 2.4
12 % 5 => 2
12 // 5 => 2
12 ** 5 => 248832
>>>
```

4.

```
no8.py - C:/Users/t/Desktop/2020 1학기/python/과제2/no8.py (3.8.2) Python 3.8.2 Shell
File Edit Format Run Options Window Help
fon = input('네 자릿수 정수 입력 >> ')
print('역순 >> ', fon[::-1])

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/t/Desktop/2020 1학기/python/과제2/no8.py =====
네 자릿수 정수 입력 >> 5432
역순 >> 2345
>>>
```

Chapter.3

일상에서 활용되는 문자열과 논리 연산

1. 문자열 다루기

1-1. 문자열 str 클래스와 부분 문자열 참조 슬라이싱

-1. 문자열은 클래스 str의 객체 :

파이썬에서 문자열은 '문자의 나열'로, 텍스트 시퀀스 라고도 함.

-2. 함수 len()으로 문자열 길이 참조 :

함수 len()으로 문자열의 길이를 알 수 있다.

-3. 문자열의 문자 참조 :

문자열을 구성하는 문자는 0부터 시작되는 첨자를 대괄호 안에 기술해 참조가 가능함.

```
>>> a='python'
>>> len(a)
6
>>> 'python'[2]
't'
```

1-2. 문자열 부분 문자열 참조 방식

-1.0과 양수를 이용한 문자열 슬라이싱 :

문자열에서 일부분을 참조하는 방법을 슬라이싱이라고 함.

-2. 음수를 이용한 문자열 슬라이싱 :

슬라이스 에서는 음수도 사용이 가능함.

```
>>> 'python'[2:4]
'th'
>>> 'python'[-6:-1]
'pytho'
```


-3.start와 end를 비우면 처음부터와 끝까지를
의미 :

슬라이싱에서 start와 end를 비울 수 있으며 각각 '처음부터'와
'끝까지'를 의미한다.

-4.str[start:end:step]의로 문자 사이의 간격을
step으로 조정 가능 :

부분 반환 문자열에서 문자 사이의 간격을 step으로 조정하려
면 위를 사용해야 하며 step을 생략하면 1이다

```
>>> 'python'[:3]
'pyt'
>>> 'python'[::-3]
'ph'
```

1-3.문자 함수와 이스케이프 시퀀스

-1.문자 함수 ord()와 chr() :

파이썬은 유니코드 문자를 사용하므로 한글을 비롯한 세계의
언어를 사용하는 데 아무런 지장이 없다.

-2.이스케이프 시퀀스 문자 :

하나의 문자를 역슬래시로 시작하는 조합으로 표현하는 문자이
다.

```
>>> ord('가')
44032
>>> '\n'
'\n'
```

2. 문자열 관련 메소드

2-1. 문자열 대체와 부분 문자열 출현 횟수, 문자열 삽입

-1. 문자열 바꿔 반환하는 메소드 replace() :

클래스에 소속된 함수를 메소드라 한다. 문자열 클래스 str에 속한 메소드는 매우 다양한데 메소드의 호출은 '문자열 객체명.함수 이름(인자)'와 같이 사용한다.

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

-2. count()와 join()메소드 :

문자나 부분 문자열의 출현 횟수를 알려면 메소드 str.count()를 문자와 문자 사이에 원하는 문자열을 삽입하려면 메소드 join()을 사용해야 한다.

```
>>> str = '단순한 것이 복잡한 것보다 낫다.'
>>> str.count('복잡')
1
>>> str.count('것')
2
>>> num = '12345'
>>> '->'.join(num)
'1->2->3->4->5'
```

2-2. 문자열 찾기와 나누기

-1. 문자열을 찾는 find(), index()와 나누는 split()

메소드:

클래스 str에서 부분 문자열 sub가 맨 처음에 위치한 첨자를 반환받으려면 메소드 str.find() 또는 str.index를 사용한다.

str.split()은 공백을 기준으로 문자열을 나눠 준다.

```
>>> str = '자바 C 파이썬 코틀린'
>>> str.find('자바')
0
>>> str.index('자바')
0
>>> str.split()
['자바', 'C', '파이썬', '코틀린']
~~~
```

2-3. 출력을 정형화하는 함수 format()

-1. format() 메소드를 이용한 간결한 출력 처리:

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

```
>>> str = '{}+{}={}'.format(3,4,3+4)
>>> print(str)
3+4=7
~~~
```

-2.문자열의 {n:md}로 정수와 실수를 형식유형으로 처리

{1:5d}처럼 중괄호의 내부에서 인자의 순번 다음 콜론 이후에 5d,10f등의 출력 폭과 출력 형식을 정할 수 있다. 실수는 f와 F 모두 실수를 소수점 형태로 출력한다.

-3.메소드 format()의 다양한 형식 지정

3. 주요 포맷스트링 종류

매개변수	형식
%d	정수형 10진수 상수(Integer)
%f	실수형 상수(float)
%lf	실수형 상수(double)
%c	문자 값(char)
%s	문자 스트링((const)(unsigned) char *)
%u	10진수 양의 정수
%o	8진수 양의 정수
%x	16진수 양의 정수
%s	문자열
%n	*Int(총 바이트 수) 지금까지 출력한 총 바이트 수
%hn	%n의 반인 2바이트 단위

3.논리 자료와 다양한 연산

3-1. 논리 값과 논리 연산

-1.논리 유형bool과 함수 bool() :

참과 거짓을 의미하는 True와 False를 키워드로 제공하고 이러한 논리 값의 자료형을 클래스 bool로 제공한다. 또한 내장함수 bool()로 인자인 변수나 상수의 논리 값을 알 수 있다.

-2.논리곱과 논리합 연산자 and와 or :

논리곱인 and와 &연산자는 두 항이 모두 참이어야 True다

-3.배타적 논리합 역산자 ^와 not 연산자 :

배타적 논리합 연산자인^은 두 항이 다르면 True,같으면False 이다.

-4:논리 연산 우선순위 not and or

논리 연산은,not 연산,논리곱 and,논리합 or 순이다.

3-2. 관계 연산

-1.수학에서 자주 다루웠던 관계 연산자 :

※ 비교 연산자의 종류

구분	연산자	의미
비교 연산자	>	크다.
	<	작다.
	>=	크거나 같다.
	<=	작거나 같다.
	==	피연산자들의 값이 같다.
	!=	피연산자들의 값이 같지 않다.

-2. 논리 값 True와 False의 산술 연산:

파이썬은 논리 값 True와 False를 각각 1과 0으로 산술 연산에 활용할 수 있다..

```
>>> 40 * True, 30 * False
(40, 0)
```

3-3. 멤버십 연산 in

-1.문자열에서의 부분 문자열인지 검사 :

x in s = 문자열 s에 부분 문자열 x가 있으면 True, 없으면 False,

x not in s = 문자열 s에 부분 문자열x가 없으면 True, 있으면 False.

3-4. 비트 논리 연산

-1.비트 논리곱 &, 논리합 |, 배타적 논리합 ^, 이동 연산 >>, << :

종류	기호	문법	의미
비트 연산자	&	a&3	a와 3의 비트 논리곱 연산
		a 3	a와 3의 비트 논리합 연산
	^	a^3	a와 3의 비트 배타적 논리합 연산
	>>	a>>3	a의 비트열을 오른쪽으로 3칸 이동
	<<	a<<3	a의 비트열을 왼쪽으로 3칸 이동

-2.연산자 우선순위

우선순위	연산자 (연산기호)	연산자 명
1	() , [] , -> , ,	괄호, 포인터, 도트
2	!, ~, ++, --, *, &, sizeof(), (자료형)	단항 연산자
3	*, /, %	산술 연산자
4	+, -	
5	<<, >>	비트이동 연산자
6	<, >, <=, >=	관계 연산자
7	==, !=	
8	&	비트논리 연산자
9	^	
10		
11	&&	논리 연산자
12		
13	?	조건연산자
14	=, +=, *=, , <<=	할당 및 복합 할당연산자
15	,	coma 연산자

2.

6

8

```
a = int(input('정수 입력 >> '))
b = int(input('2의 지수승 입력 >>'))

print('정수', a, '<<', b, '결과:', a << b)
print('정수', a, '* 2**', b, '결과:', a * 2**b)

print('2진수(32비트){0.092b} 정수:{0.8d}'.format(a))
print('2진수(32비트){2.092b} 정수:{2.8d}:{0} << {1}'.format(a,b,a<b))
print('2진수(32비트){2.092b} 정수:{2.8d}:{0} * 2**{1}'.format(a,b,a*2**b))
```


Chapter.4

일상생활과 비유되는 조건과 반복

1.조건식의 논리 값에 따른 선택 if

1-1.조건에 따라 해야 할 일을 처리해야 하는 경우 if문을 사용.

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

```
// = RESTART: C:\Users\wt\Desktop\2020 13
04-1rideif.py
롤러코스터 T-Express, 즐기세요!
>>> |
```

1-2조건에 따라 하나를 선택하는 if else ;

조건 if의 논리 표현식 결과는 True 또는 False이며 else: 블록은 논리 표현식 결과가 False일 때 실행된다.

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

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

```
teiji on wincc
Type "help", "copyright", "credits"
>>>
= RESTART: C:\Users\wt\Desktop\2020
04-3earlybirddiscount.py
지금 시각: 7시 26분, 조조 할인 된다
>>> |
```

1-3조건에 따라 하나를 선택하는 if elif :

다중 택일 결정 구조인 if elif는 조건의 여러 경로 중에서 하나를 선택하는 구문이다.

```
PM = float(input('미세먼지(10마이크로그램)의 농도는 ? '))
if 151 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM,'매우 나쁨'))
elif 81 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM,'나쁨'))
elif 31 <= PM:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM,'보통'))
else:
    print('미세먼지 농도: {:.2f}, 등급: {}'.format(PM,'좋음'))
```

```
미세먼지(10마이크로그램)의 농도는 ? 78
미세먼지 농도: 78.00, 등급: 보통
>>> |
```

2.반복을 제어하는 for문과 while문

2-1.정해져 있는 시퀀스의 항목 값으로 반복을 실행

하는 for문 :

여러 자료 값이 순서대로 구성된 시퀀스에서 자료 값의 개수만큼 반복적인 작업을 수행하기 위한 구문이 for문이다.

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

U4- /numseq.py

```
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
>>> |
```

2-2.내장함수 range()를 사용한 for문 :

내장 함수 range(4)는 정수 0에서 4까지 5개의 항목인 정수로 구성되는 시퀀스를 만든다. 그러므로 반복 구문에서 변수 I에 할당되는 0에서 4까지의 5개의 정수로 반복을 수행할 수 있다.

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

U4-8coffeeorder.py

```
주문하세요! [아메리카노] [카페라테] [카푸치노] >> 아메리카노
아메리카노 주문
주문하세요! [아메리카노] [카페라테] [카푸치노] >> 카페라테
카페라테 주문
주문하세요! [아메리카노] [카페라테] [카푸치노] >> 카푸치노
카푸치노 주문
주문하세요! [아메리카노] [카페라테] [카푸치노] >> 
모르겠어요.
주문을 마치겠습니다.
>>> |
```

2-3.횟수를 정하지 않은 반복에 적합한 while문:

while문은 for문에 비해 간결하며 반복 조건인 논리 표현식의 값에 따라 반복을 수행한다. while문은 횟수를 정해놓지 않고 어떤 조건이 False가 될 때까지 반복을 수행하는데 적합하다.

```
MAXNUM = 4
MAXHEIGHT = 130

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

```
키는 ? 134
커서 못 들어갑니다.
키는 ? 123
들어가요. 1명
키는 ? 121
들어가요. 2명
키는 ? 88
들어가요. 3명
키는 ? 133
커서 못 들어갑니다.
키는 ? 123
들어가요. 4명
4명 모두 찾습니다. 다음 번에 이용하세요.
>>> |
```

3.임의의 수인 난수와 반복을 제어하는 break, continue 문

3-1.임의의 수를 발생하는 난수

-1.임의의 수를 발생하는 난수 : 모듈 random의 함수 randint(시작,끝)를 사용해 정수 시작과 끝 수 사이에서 임의의 정수를 얻을 수 있다.

```
winnumber = 11, 17, 28, 30, 33, 35
print(' 모의 로또 당첨 번호 '.center(28, '='))
print(winnumber)
print()
print(' 내 번호 확인 '.center(30, '-'))
cnt = 0
import random
for i in range(6):
    n = random.randint(1, 45)
    if n in winnumber:
        print(n, 'O ', end = ' ')
        cnt += 1
    else:
        print(n, 'X ', end = ' ')
print()
print(cnt, '개 맞음')
```

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

----- 내 번호 확인 -----
35 O  18 X  34 X  43 X  28 O  37 X
2 개 맞음
>>>
```

3-1.반복을 제어하는 break 문과 continue문

-1. 반복을 종료하는 break문

반복 while의 논리 표현식이 True라면 무한히 반복된다. 문장 break는 else 블록을 실행시키지 않고 반복을 무조건 종료한다.

-2. continue이후의 반복 몸체를 실행하지 않고 다음 반복을 계속 실행:

반복 for와 while문 내부에서 continue문장은 이후의 반복 몸체를 실행하지 않고 다음 반복을 위해 논리 조건을 수행한다.

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

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

chapter.4의 실습문제

교과서 p.156의 짝수 4문제

2.

```
oh = 12000
import random
for i in range(5):
    t = random.randint(35,50)
    if t>=40 :
        print('근로시간',t,'시간, 주급 ',int(t*oh*1.5))
    else :
        print('근로시간',t,'시간, 주급 ',t*oh)
```

```
===== RESTART: C:/Users/
=====
근로시간 42 시간, 주급 756000
근로시간 37 시간, 주급 444000
근로시간 40 시간, 주급 720000
근로시간 39 시간, 주급 468000
근로시간 35 시간, 주급 420000
>>> |
```

4.

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

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
tel)] on win32
Type "help", "copyright", "credits" or
>>>
===== RESTART: C:/Users/t/Desktop
=====
당신의 키(cm)와 몸무게(kg)는? >>171 72
키: 171 (cm), 몸무게: 72 (kg)
BMI: 24.6 정상
>>> |
```

6.

```
from random import randint
while True:
    n = randint(1,99)
    n1 = randint(1,99)
    print(n,'*',n1,'=',n*n1)

    tf = input('계속 y / n ? ')
    if tf == 'n' :
        break
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
tel)] on win32
Type "help", "copyright", "credits" or
>>>
===== RESTART: C:/Users/t/Desktop
=====
33 * 61 = 2013
계속 y / n ? y
10 * 37 = 370
계속 y / n ? y
8 * 57 = 456
계속 y / n ? n
>>> |
```

8.

```
from random import randint
```

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

```
print('첫 값은 ',n, ' 이다')
```

```
while True:
```

```
    p = input('산술 연산의 종류를 입력하세요. >>')
```

```
    if p == '+' :
```

```
        nt = int(input('두 번째 피연산자를 입력하세요. >>'))
```

```
        print(n, ' + ',nt, ' = ', n+nt)
```

```
    elif p == '-' :
```

```
        nt = int(input('두 번째 피연산자를 입력하세요. >>'))
```

```
        print(n, ' - ',nt, ' = ', n-nt)
```

```
    elif p == '*' :
```

```
        nt = int(input('두 번째 피연산자를 입력하세요. >>'))
```

```
        print(n, ' * ',nt, ' = ', n*nt)
```

```
    elif p == '/' :
```

```
        nt = int(input('두 번째 피연산자를 입력하세요. >>'))
```

```
        print(n, ' / ',nt, ' = ', n/nt)
```

```
    elif p == '%' :
```

```
        nt = int(input('두 번째 피연산자를 입력하세요. >>'))
```

```
        print(n, ' % ',nt, ' = ', n%nt)
```

```
    else :
```

```
        print(' 종료 '.center(30,'*'))
```

```
        break
```

File Edit Shell Debug Options Window

Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 2020) on win32

Type "help", "copyright", "credits" or "quit()"

>>>

===== RESTART: C:\Users\t/Des

>>>

첫 값은 91 이다

산술 연산의 종류를 입력하세요. >>+

두 번째 피연산자를 입력하세요. >>20

91 * 20 = 1820

산술 연산의 종류를 입력하세요. >>-

두 번째 피연산자를 입력하세요. >>45

91 - 45 = 46

산술 연산의 종류를 입력하세요. >>/

두 번째 피연산자를 입력하세요. >>5

91 / 5 = 18.2

산술 연산의 종류를 입력하세요. >>*

***** 종료 *****

>>> |

Chapter.5

할목의 나열인 리스트와 튜플

1. 여러 자료 값을 편리하게 처리하는 List

1-1. 리스트의 개념과 생성

-1. 관련된 나열 항목을 관리하는 List :

프로그래밍 언어는 일반적으로 여러 항목을 한 번에 관리할 수 있는 자료형을 지원한다. 파이썬도 여러 항목을 하나의 단위로 묶어 손쉽게 사용하는 복합 자료형을 여러 개 제공하는데 그중 대표적인 것이 List이다.

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

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

```
05-1coffee.py
['에스프레소', '아메리카노', '카페라테', '카페모카']
<class 'list'>
1. 에스프레소
2. 아메리카노
3. 카페라테
4. 카페모카
>>> x'xxxxx
```

-2. 빈 리스트의 생성과 항목 추가, 리스트의 항목 수를 반환하는 함수 len() :

빈 대괄호로 빈 리스트를 만들 수 있다. len() 함수는 리스트의 항목 수를 알 수 있다.

-3.일상코딩: 리스트의 메소드 append() :

메소드 append()는 리스트에 품목을 표준입력으로 입력 받을 수 있다.

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

구입할 품목은 ? 과자
['과자']
구입할 품목은 ? 우유
['과자', '우유']
구입할 품목은 ? 세면 도구
['과자', '우유', '세면 도구']
길이: 3

1-2.리스트의 항목 참조

-1.문자열 시퀀스와 같이 첨자로 리스트의 항목 참조 :

리스트에서 index를 사용해 항목 하나하나를 참조할 수 있다. 물론 첨자는 정수여야 하며, 이는 문자열에서 배운 첨자 방식과 동일하다.

```
pl = ['C', 'C++', 'Python', 'JAVA']
print(pl[0])
print(pl[2])
print()
for i in range(len(pl)):
    print(pl[i])
```

C
Python

C
C++
Python
JAVA

1-3.리스트의 항목 수정

-1.리스트의 메소드 count()와 index() :

리스트의 메소드 count(값)은 값을 갖는 항목의 수, index(값)은 인자인 값의 항목이 위치한 첨자를 변환한다. 동일한 값이 여러 개이면 첫 번째로 나타난 위치의 첨자다.

```
food = ['짜장면', '짬뽕', '우동', '울면']
print(food)

food.append('탕수육')
print(food)

food[1] = '굴짬뽕'
print(food)

food[food.index('우동')] = '물만두'
print(food)
```

['짜장면', '짬뽕', '우동', '울면']
['짜장면', '짬뽕', '우동', '울면', '탕수육']
['짜장면', '굴짬뽕', '우동', '울면', '탕수육']
>>> |

2-1.리스트의 부분 참조인 슬라이싱

시퀀스인 리스트도 콜론을 사용해 리스트 전체나 일부분을 참조할 수 있다.

-3.-1부터 시작되는 역순 첨자 :

-4.0또는 양수인 정순 첨자와 역순인 음수 첨자를 함께 사용하는 슬라이싱

2-2.리스트의 항목 삽입과 삭제

-2.문장 del에 의한 항목이나 변수의 삭제

```

item = ['연필', '볼펜']
print(item)

item.insert(1,2)
item.insert(3,3)

item.insert(4, '지우개')
item.insert(5,1)
print(item)

print(item.pop(1))
item.remove('연필')
del item[2:]

print(item)

```


2-3.리스트의 추가,연결과 반복

- 1.리스트에 추가하는 메소드 extend()
- 2.리스트를 연결하는 +연산자 반복하는 *연산자

2-4.리스트 항목의 순서와 정렬

- 1.리스트 항목 순서를 뒤집는 메소드 reverse()
- 2.리스트 항목 순서를 정렬하는 메소드 sort()
- 3.리스트 항목 순서를 정렬한 리스트를 반환하는 내장 함수 sorted()

```
word=list('샬롬정')
word.extend('복빛')
print(word)
word.sort()
print(word)

fruit=['복숭아','자두','골드키위','귤']
print(fruit)
fruit.sort(reverse=True)
print(fruit)

mix = word+fruit
print(sorted(mix))
print(sorted(mix,reverse=True))
```

```
['샬롬', '복빛', '정', '복숭아', '자두', '골드키위', '귤']
['복숭아', '자두', '골드키위', '귤']
['복숭아', '자두', '골드키위', '귤']
>>> |
```

2-5.리스트 컴프리헨션

- 1.조건을 만족하는 항목으로 리스트를 간결히 생성하는 컴프리헨션
- 2.조건이 있는 컴프리헨션

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

seq=[i for i in range(10)]
print(seq)

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

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

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 9, 25, 49, 81]
[1, 9, 25, 49, 81]
>>> |
```

2-6. 리스트 대입과 복사

-1.리스트 대입에 의한 동일 리스트의 공유 :

리스트의 대입에는 주의가 필요하며 리스트에서 대입 연산자 = 은 얇은 복사라고 하여 대입되는 변수가 동일한 시퀀스를 가지게 된다.

-2.리스트의 깊은 복사에 의한 대입으로 새로운 리스트의 생성 :

리스트에서 완전히 새로운 리스트를 만들어 복사하려면 슬라이스[:]나 copy()또는 list()함수를 이용해야 한다.

-2.변수의 동일 객체 여부를 검사하는 is :

문장 is는 피연산자인 변수 2개가 동일한 메모리를 공유하는지 검사한다. 그러므로 같으면 True,다르면 False를 반환한다.

3.항목의 순서나 내용을 수정할 수 없는 튜플

3-1.괄호로 정의하는 시퀀스 튜플

-1.수정할 수 없는 항목의 나열인 튜플:

튜플은 문자열,리스트와 같은 항목의 나열인 시퀀스이며 튜플은 리스트와 달리 항목의 순서나 내용의 수정이 불가능하다.

-2.튜플 항목 참조와 출력

```
singer=('BTS', '불뽕간사춘기', 'BTS', '블랙핑크', '태연')
song=('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
print(singer)
print(song)

print(singer.count('BTS'))
print(singer.index('불뽕간사춘기'))
print(singer.index('BTS'))
print()

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

```
= HCSIAH11: C:\Users\HCSIAH11\AppData\Local\Microsoft\Windows\Python\Python38-32\python.exe
05-12kptuple.py
('BTS', '불뽕간사춘기', 'BTS', '블랙핑크', '태연')
('작은 것들을 위한 시', '나만, 봄', '소우주', 'Kill This Love', '사계')
2
1
0

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

3-2. 튜플 연결과 반복, 정렬과 삭제

- 1. 튜플 연결 + 연산자와 반복*연산자:
- 2. 튜플 항목의 순서를 정렬한 리스트를 반환하는 내장함수 `sorted()`
- 3. 문장 `del`에 의한 튜플 변수의 제거

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

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

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

```
03_03daytuple.py
<<class 'tuple'>
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday')
('monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday',
sunday', 'sunday')
>>> |
```

Chapter.6

일상에서 활용되는 문자열과 논리 연산

1.키와 값인 쌍의 나열인 딕셔너리

1-1.딕셔너리의 개념과 생성

-1.키와 값의 쌍을 항목으로 관리하는 딕셔너리:

딕셔너리란 말 그대로 ‘사전’을 생각하면 이해하기 쉽다. 사전은 여러 낱말 각각의 의미, 어원 따위를 설명한 내용을 담고 있다. 여기서 낱말을 키(key),해설을 값(value)라고 보면 된다.

-1.빈 딕셔너리의 생성과 항목 추가 :

빈 중괄호로 빈 딕셔너리를 만들 수 있다.

1-2.다양한 인자의 함수 dict()로 생성하는 딕셔너리

-1.리스트 또는 튜플로 구성된 키-값을 인자로 사용:

내장 함수 dict()함수에서 인자로 리스트나 튜플 1개를 이용하여 딕셔너리를 만들 수 있다.

-2.키가 문자열이면 키 = 값 항목의 나열로도 딕셔너리 생성

```
bts1={'그룹명':'방탄소년단','인원수':7,'리더':'김남준'}
bts1['소속사']='빅히트 엔터테인먼트'
print(bts1)
bts2=dict([['그룹명','방탄소년단'],[7,'인원수']])
print(bts2)
bts3=dict([('리더','김남준'),('소속사','빅히트 엔터테인먼트')])
print(bts3)

bts=dict(그룹명='방탄소년단',인원수=7,리더='김남준',소속사='빅히트 엔터테인먼트',
bts['구성원']=['RM','진','슈가','제이홉','지민','뷔','정국'])
print(bts)
print(bts['구성원'])
```

```
tel)) on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\wt\Desktop\2020 1학기\python\MiddleTest\2020 python code\ch6\
06-3btsdict.py
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔테
인먼트'}
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔테
인먼트'}
{'그룹명': '방탄소년단', '인원수': 7, '리더': '김남준', '소속사': '빅히트 엔테
인먼트', '구성원': ['RM', '진', '슈가', '제이홉', '지민', '뷔', '정국']}
{'RM', '진', '슈가', '제이홉', '지민', '뷔', '정국'}
>>>
```

1-3.딕셔너리 키는 수정 불가능한 객체로 사용

- 1.딕셔너리의 키로 정수, 실수 사용 가능
- 2.튜플은 키로 가능하지만 리스트는 키로 사용 불가능 :

수정 불가능한 튜플은 딕셔너리의 키로 사용될 수 있다.

1-4.딕셔너리 항목의 순회

- 1.딕셔너리 메소드 keys() :

딕셔너리 메소드 keys()는 키로만 구성된 리스트를 반환한다
그러므로 for문에서 시퀀스 위치에 메소드 keys()를 사용하면
딕셔너리의 모든 항목을 참조하는 구문을 사용할 수 있다.

- 2.딕셔너리 메소드 items() :

딕셔너리 메소드 items()는 (키,값) 쌍의 튜플이 들어 있는 리스트를 반환한다.

- 3.딕셔너리 메소드 values() :

딕셔너리 메소드 values()는 값으로 구성된 리스트를 반환한다.

- 4.반복 for문에서 딕셔너리 변수로만 모든 키 순회 :

반복 for문 에서는 시퀀스 위치에 있는 딕셔너리 변수만으로도 모든 키를 순회할 수 있다.

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

for key in season.keys():
    print('%s %s' % (key, season[key]))

for item in season.items():
    print('%s %s' % (item[0], item[1]), end=' ')
    print()

for item in season.items():
    print('%s %s' % (item[0], item[1]), end=' ')
    print()

dict_keys(['봄', '여름', '가을', '겨울'])
dict_items([('봄', 'spring'), ('여름', 'summer'), ('가을', 'actumn'), ('겨울', 'winter')])
dict_values(['spring', 'summer', 'actumn', 'winter'])
봄 spring
여름 summer
가을 actumn
겨울 winter
봄 spring 여름 summer 가을 actumn 겨울 winter
여름 summer
가을 actumn
겨울 winter
>>>
```

1-5.딕셔너리 항목의 참조와 삭제

-1.키로 조회하는 딕셔너리 메소드 get() :

딕셔너리 메소드 get(키)는 키의 해당 값을 반환한다.

-2.키로 삭제하는 딕셔너리 메소드 pop() :

딕셔너리 메소드 pop(키)는 항목을 삭제하고, 삭제되는 키의 해당 값을 반환한다.

-3.임의의 항목을 삭제하는 딕셔너리 메소드 popitem() :

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

-4.문장 del로 딕셔너리 항목 삭제

1-6.딕셔너리 항목 전체 삭제와 변수 제거

-1.모든 항목을 제거하는 clear() :

-2.문장 del로 딕셔너리 변수 자체 제거

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

print(color.get('녹색'))
print(color.get('노란색'))
print()

color['노란색']='yellow'
print(color)
print()

c='흰색'
print('삭제: %s %s' % (c,color.pop('흰색')))
print(color)
c='빨간색'
print('삭제: %s %s' % (c,color.pop(c,'없어요'))))

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

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

color.clear()
print(color)
```

```
{'검은색': 'black', '흰색': 'white', '녹색': 'green', '파란색': 'blue'}
green
None
{'검은색': 'black', '흰색': 'white', '녹색': 'green', '파란색': 'blue', '노란색': 'yellow'}
삭제: 흰색 white
{'검은색': 'black', '흰색': 'white', '녹색': 'green', '파란색': 'blue', '노란색': 'yellow'}
삭제: 빨간색 없어요
임의의 삭제: ('노란색', 'yellow')
임의의 삭제 후: {'검은색': 'black', '녹색': 'green', '파란색': 'blue'}
검은색삭제 후: {'녹색': 'green', '파란색': 'blue'}
{}
>>>|
```

1-7.딕셔너리 결합과 키의 멤버십 검사 연산자

- 1.딕셔너리를 결합하는 메소드update()
- 2.문장 in으로 쉽게 검사.

2.중복과 순서가 없는 집합.

2-1.수학에서 배운 집합을 처리하는 자료형

- 1.원소는 유일하고 순서는 의미 없는 집합.

2-2.내장함수 set()을 이용한 집합 생성.

- 1.집합을 만드는 내장 함수 set()
- 2.함수 set()호출로 공집합 만들기
- 3.리스트나 튜플을 인자로 사용하는 함수 set()
- 4.문자열을 인자로 사용하는 함수 set()
- 5.수정 가능한 리스트와 딕셔너리는 집합의 원소로 사용 불가.

2-3.중괄호로 직접 원소를 나열해 집합 생성.

- 1.{원소1,원소2,...}로 생성
- 2.{원소1,원소2,...}에서 리스트나 딕셔너리는 원소로 사용 불가.

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

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

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

2-4. 집합의 원소 추가와 삭제

- 1. 집합 메소드 add(원소)로 추가
- 2. remove(원소)와 discard(원소), pop()으로 항목 삭제
- 3. clear()로 집합의 모든 원소 삭제

```
from random import randrange
from random import sample

mylotto = set()
while True:
    num = randrange(1, 46)
    print(num, end=' ')
    mylotto.add(num)
    if len(mylotto) == 6:
        break

print()
print('집합: {}'.format(mylotto))
print('정렬 리스트: {}'.format(sorted(mylotto)))
print()

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

```
45 8 29 39 1 11
집합: {1, 39, 8, 11, 45, 29}
정렬 리스트: [1, 8, 11, 29, 39, 45]

sample 함수 리스트: [9, 19, 42, 23, 14, 41]
sample 함수 정렬 리스트: [9, 14, 19, 23, 41, 42]
>>>
```

- 4. 합집합 연산자 |와 메소드 union(), update()
- 5. 교집합 연산자 &와 메소드 intersection()
- 6. 차집합 연산자 -와 메소드 difference()
- 7. 여집합 연산자 ^와 메소드 symmetric_difference()
- 8. 집합 연산의 축약 대입 연산자 |=, &=, -=, ^=와 메소드 intersection_update()


```

daysA = {'월', '화', '수', '목'}
daysB = set(['수', '목', '금', '토', '일'])
weekends = set(['토', '일'])

alldays = daysA | daysB
print(alldays)

workdays=alldays-weekends
print(workdays)

print(daysA&daysB)
print(daysA.symmetric_difference(daysB))

```

```

{'수', '토', '목', '화', '금', '일'}
{'수', '토', '목'}
{'수', '목'}
{'월', '일', '화', '금', '토'}
>>>

```

2-5.함수 len()과 소속 연산 in

- 1.집합 원소 수 함수 len()
- 2.멤버십 연산자 in

3.내장 함수 zip()과 enumerate(),시퀀스 간의 변환.

3-1.내장 함수 zip()

- 1.리스트나 튜플 항목으로 조합된 항목을 생성하는 내장 함수 zip() :

내장 함수zip()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 함수다.

- 2.2개의 리스트나 튜플로 키-값 항목인 딕셔너리를 생성하는 내장 함수zip()

3-2.내장 함수 enumerate()

- 1.첨자를 자동으로 만들어 첨자와 값과의 쌍인 튜플을 만들어 주는 내장 함수 enumerate()
- 2.반복에서 사용하는 내장 함수 enumerate()

3-3.시퀀스 간의 변환

- 1.튜플과 시퀀스 간의 변환 :

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

- 2.리스트와 집합 간의 변환 :

- 3.딕셔너리를 다른 시퀀스로 변환하면 항목이 키로만 구성.

마무리

Chapter 1.

파이썬 언어란 무엇인지 이해하고 개발 도구를 설치해 프로그램을 구현할 수 있으며 특징과 활용분야를 설명할 수 있다.

Chapter 2.

파이썬의 재료인 문자열과 수에 대해 이해하고 코드로 구현할 수 있으며 변수를 이해하고 다양한 대입 연산자를 활용할 수 있다. 또한 표준입력으로 문자열을 입력받은 후 원하는 자료로 변환해 활용할 수 있다.

Chapter 3.

문자열에서 문자나 부분 문자열을 반환하는 여러 방법을 구현할 수 있으며 문자열 객체에 소속된 다양한 메소드를 이해하고 활용할 수 있다. 또한 논리값을 이해하고 다양한 연산을 사용해 실생활에서의 표현에 활용할 수 있다.

Chapter 4.

조건에 따라 하나를 결정하는 if문을 구현할 수 있고 반복을 수행하는 while문과 for문을 구현할 수 있으며 임의의 수인 난수를 이해하고 반복을 제어하는 break문과 continue문을 사용할 수 있다.

Chapter 5.

다양한 종류의 항목을 쉽게 나열하는 리스트를 구현할 수 있고 리스트에서 부분 참조 방법, 이를 이용한 수정,리스트 연결,삽입과 삭제 그리고 리스트 컴프리헨션 등을 구현할 수 있으며 수정할 수 없는 다양한 종류의 항목 나열을 쉽게 처리하는 튜플을 구현할 수 있다.

Chapter 6.

키와 값의 쌍인 항목을 관리하는 딕셔너리를 생성하고 수정하는 방법을 이해하고, 다양한 방법으로 딕셔너리를 구현할 수 있다. 또한 집합의 특징을 이해하고 합집합 등과 같은 다양한 집합의 연산을 구현할 수 있으며 내장 함수 zip()과 enumerate(),시퀀스 간의 변환을 이해하고,구현할 수 있다.

후기

Phython 중간고사 포트폴리오로 Phython을 선택함으로서 Phython을 인강으로 듣고 직접 과제도 풀어 보았지만 내가 스스로 찾아가며 글을 쓰며 배움으로서 본질적인 이해를 할 수 있어 조금 더 Phython에 다가갈 수 있었던 것 같습니다.