20163810 김찬종

파이썬 포트폴리오

0. 머리말

파이썬을 배우기 전에 파이썬에 대해 들은 말들은 많았습니다.

지금도 많이 쓰이고 앞으로는 더 많이 쓰일 언어라던지, 사용하는 기능에 비해 쉽고 간단하다던지, 타 언어에 비해 폭넓게 쓰인다던지 등등 대부분이 파이썬의 장점이었습니다.

직접 배우면서 느낀 바로는 들여쓰기를 안하면 오류가 생긴다거나 else if가 아닌 elif를 쓰는 등 다른 언어와 비슷한 듯 다른 당황스러운 부분도 있었고, 리스트와 튜플 등 완전히 새로운 부분도 있어 처음엔 많이 낯설었지만 배우면서 확실히 코드의 간결함이 느껴지고 아직 배우지 못한 부분들에 대해 알고 싶은 생각이 많이 들었습니다.

전과 후 첫 학기를 진행하며 프로그래밍에 대해 모르는 점이 많고, 파이썬이라는 언어를 처음 접하며 아직은 미숙하지만 앞으로의 수업을 통해 더 많이 배웠으면 합니다.

코로나 사태로 인해 비대면 수업을 진행할 수밖에 없지만 더 많아진 개인 시간을 통해 부족한 부분은 따로 공부하고, 이번 포트폴리오 과제를 통해 다시 한 번 복습하고 정의와 사용법 위주로 정리하여 계속해서 볼 수 있는 자신만의 사전으로 만듦과 동시에 직접 코드를 실행해봄으로써 파이썬에 대해 다시 한 번 기본을 다지고자 합니다.

목차

1장 강의계획서

1. 강의계획서

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	• 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 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	• 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

1. 강의계획서

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 불허하는 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 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	• 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	• 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	• GUI를 이해하고 GUI 표준 모듈인 Tkinter를 사용해 필요한 위젯을 구성하고 윈도를 생성할 수 있다. • 이벤트 처리를 이해하고 Tkinter에서 이벤트 처리를 구현할 수 있다. • 써드파티 GUI 모듈인 pygame을 설치해 기본적인 윈도를 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	• 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	• 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	

2장 파이썬 개요

2. 파이썬 개요

파이썬(Python)은 1991년 프로그래머인 귀도 반 로섬이 발표한 고급 프로그래밍 언어로, 플랫폼에 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다.
파이썬은 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델을 가지고 있다. C 언어로 구현된 Cython 구현이 사실상의 표준이다.

파이썬은 초보자부터 전문가까지 사용자층을 보유하고 있다. 동적 타이핑(dynamic typing) 범용 프로그래밍 언어로, 펄 및 루비와 자주 비교된다. 다양한 플랫폼에서 쓸 수 있고, 라이브러리(모듈)가 풍부하여, 대학을 비롯한 여러 교육 기관, 연구 기관 및 산업계에서 이용이 증가하고 있다. 또 **파이썬**은 순수한 프로그램 언어로서의 기능 외에도 다른 언어로 쓰인 모듈들을 연결하는 풀언어(glue language)로써 자주 이용된다. 실제 **파이썬**은 많은 상용 응용 프로그램에서 스크립트 언어로 채용되고 있다. 도움말 문서도 정리가 잘 되어 있으며, 유니코드 문자열을 지원해서 다양한 언어의 문자 처리에도 능하다.
파이썬은 기본적으로 해석기(인터프리터) 위에서 실행될 것을 염두에 두고 설계되었다.

현대의 **파이썬**은 여전히 인터프리터 언어처럼 동작하나 사용자가 모르는 사이에 스스로 파이썬 소스 코드를 컴파일하여 바이트 코드(Byte code)를 만들어 냄으로써 다음에 수행할 때에는 빠른 속도를 보여 준다.

2. 파이썬 개요

주요 특징

-동적 타이핑(dynamic typing). (실행 시간에 자료형을 검사한다.)
-객체의 멤버에 무제한으로 접근할 수 있다. (속성이나 전용의 메서드 훅을 만들어 제한할 수는 있음.)
-모듈, 클래스, 객체와 같은 언어의 요소가 내부에서 접근할 수 있고, 리플렉션을 이용한 기술을 쓸 수 있다.

해석 프로그램의 종류

-**Cython**: C로 작성된 인터프리터.
-**스택리스** 파이썬: C 스택을 사용하지 않는 인터프리터.
-**자이썬**: 자바 가상 머신 용 인터프리터. 과거에는 제이파이썬(Jpython)이라고 불렸다.
-**IronPython**: .NET 플랫폼 용 인터프리터.
-**PyPy**: 파이썬으로 작성된 파이썬 인터프리터.

파이썬에서는 들여쓰기를 사용해서 블록을 구분하는 독특한 문법을 채용하고 있다. 이 문법은 파이썬에 익숙한 사용자나 기존 프로그래밍 언어에서 들여쓰기의 중요성을 높이 평가하는 사용자에게는 잘 받아들여지고 있지만, 다른 언어의 사용자에게서는 프로그래머의 코딩 스타일을 제한한다는 비판도 많다. 이 밖에도 실행 시간에서뿐 아니라 네이티브 이진 파일을 만들어 주는 C/C++ 등의 언어에 비해 수행 속도가 느리다는 단점이 있다. 그러나 사업 분야 등 일반적인 컴퓨터 응용 환경에서는 속도가 그리 중요하지 않고, 빠른 속도를 요하는 프로그램의 경우에도 프로토타이핑한 뒤 빠른 속도가 필요한 부분만 골라서 C 언어 등으로 모듈화할 수 있다(ctypes, SWIG, SIP 등의 래퍼 생성 프로그램들이 많이 있다).
또한 Pyrex, Psyco, NumPy 등을 이용하면 수치를 빠르게 연산할 수 있기 때문에 과학, 공학 분야에서도 많이 이용되고 있다. 점차적인 중요성의 강조로 대한민국에서도 점차 그 활용도가 커지고 있다.

3-1 자료형과 연산자

3-1 자료형과 연산자

문자열 출력

- +를 통해 여러 개의 문자열을 출력 가능
- *를 통해 같은 문자열을 여러 번 출력 가능
- #을 통해 한 줄 단위로 주석처리 가능

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("원의 원주율 : " + '3.141592')
원의 원주율 : 3.141592
>>> print("별 " + "달 "*3)
별 달 달 달
>>> print("주석 설명입니다")#이것은 주석입니다
주석 설명입니다
>>>
```

실수, 정수 연산

- 문자 e를 통해 지수승 표현 가능
- + - * /으로 덧셈, 뺄셈, 곱셈, 나눗셈이 가능하다
- // % **으로 몫, 나머지, 지수승의 결과

```
>>> 1.2345e5
123450.0
>>> 5*3
15
>>> 8/3
2.6666666666666665
>>> 8//3
2
>>> 9%6
3
>>> 5**3
125
```

산술연산자의 계산 우선순위

- 거듭제곱 (**) > 부호를 나타내는 (+,-) > (* / % //) > 연산자(+,-)
- 거듭제곱은 오른쪽에서 왼쪽, 나머진 왼쪽에서 오른쪽으로 계산한다
- 2**2**3 = 2**(2**3) = 256
- -2**2 = -4

3-1 자료형과 연산자

최근 결과의 저장 장소 언더스코어 _

- _를 통해 대화형 모드에서 마지막에 실행된 결과값 호출 가능
- 숫자, 문자열 등 유효한 연산식이면 모두 가능

```
>>> 1+2
3
>>> 3*_
9
>>> '안녕하세요'
'안녕하세요'
>>> _*3
'안녕하세요안녕하세요안녕하세요'
>>>
```

연산자 //와 %를 이용한 단리, 복리 계산 활용

- 단리 총액 : 원금 * (1 + 이자율 * 기간)
- 복리 총액 : 원금 * (1 + 이자율)**기간
- 단리와 복리 모두 원금 100만원
  2.3%의 이자율, 5년 만기로 가정

```
>>> 1000000*(1+0.023*5) #5년 뒤 단리 총액
1115000.0
>>> 1000000*(1+0.023)**5 #5년 뒤 복리 총액
1120413.0756413424
>>>
```

3-1 자료형과 연산자

자료형과 type() 함수

- type()함수를 통해 상수나 변수의 자료형을 알 수 있음

```
>>> type(1)
<class 'int'>
>>> pi = 3.141592
>>> type(pi)
<class 'float'>
>>> type('helloworld')
<class 'str'>
```

키워드

- 문법을 위해 예약된 33개의 단어로 변수명으로 사용 불가

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for
', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not'
, 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

변수 이름의 규칙

- 대소문자의 영문자와 숫자, _로 구성하며 대소문자를 구별함
- 숫자는 맨 앞에 올 수 없음
- 키워드는 이름으로 사용할 수 없음

함수 divmod()

- divmod(a,b)로 몫과 나머지 두 개의 결과를 반환함
- python의 특징으로 여러 변수에 한번에 대입 가능

```
>>> a, b = divmod(12345,10000)
>>> print(a,"만",b,"원")
1 만 2345 원
```

3-2 표준 입출력과 문자열

3-2 표준 입출력과 문자열

입력함수 input()

- 사용자가 직접 값을 입력할 수 있게 해주는 함수
- input('~~ ')를 통해 입력받는 값과 별개로 메시지 출력
- a = input()을 통해 숫자를 입력하면 문자로 인식되기 때문에 a = int(a)를 통해 정수형으로 바꿔줄 수 있다

```
>>> name = input("이름 입력 : ")
이름 입력 : 김찬종
>>> print(name)
김찬종
```

2진수, 8진수, 16진수

- 0b/0B로 2진수를, 0o/0O로 8진수를, 0x/0X로 16진수 표현 가능
- bin(), oct(), hex()를 통해 10진수를 해당 진수로 변환 가능

```
>>> 0b11, 0o17, 0x1f
(3, 15, 31)
>>>
>>> bin(7), oct(12), hex(16)
('0b111', '0o14', '0x10')
```

Int를 이용한 문자열의 변환

- int('11', 2)를 통해 11을 2진수로 보고 10진수 정수로 변환
- 2를 8, 16으로 바꿔 해당 진수도 10진수로 변환 가능

```
>>> a = input("2진수 정수 입력 : ")
2진수 정수 입력 : 1111
>>> data = int(a, 2) #입력받은 2진수를 10진수로 바꿔 data에 저장
>>> print(data)
15
```

3-2 표준 입출력과 문자열

클래스와 객체

파이썬 또한 객체지향 프로그래밍 언어이며, 클래스와 객체를 구분한다.
문자열 str이 클래스라면, 문자열의 속성을 가진 "python", "helloworld"와 같은 문자열 상수들은 객체로 구분한다.

문자열 함수

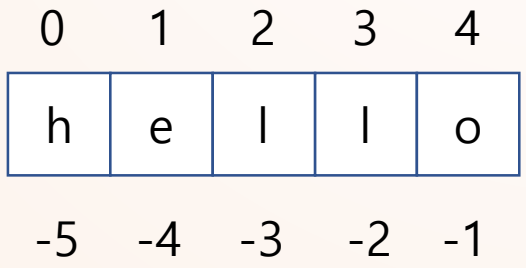
- len(문자열)을 통해 문자열의 길이를 알 수 있음
- 대괄호와 첨자를 통해 문자열의 구성 문자를 알 수 있음

```
>>> a = 'helloworld'
>>> len(a)
10
>>> a[0]
'h'
>>> a[0],a[3],a[5]
('h', 'l', 'w')
```

정수를 이용한 문자열 슬라이싱

- a가 양수라면 문자열[a1:a2]를 통해 a1번째~a2-1번째 까지의 문자열 참조 가능
  문자열[0:len(문자열)]을 통해 전체 참조 가능
- a가 음수라면 음수에 맞춰 문자열[a1:a2]를 통해 a1번째~a2-1까지 참조 가능
- 첨자를 비우면 앞의 첨자는 처음부터, 뒤의 첨자는 끝까지를 의미한다
- 문자열[start:end:step]에서 step을 통해 슬라이싱의 간격 설정 가능

```
>>> str = "hello world"
>>> str[0:3], str[-5:-1]
('hel', 'worl')
>>> str[::2]
'hlowrd'
```



3-2 표준 입출력과 문자열

문자함수 ord() chr()와 \ 함수, 최댓값 최솟값

- ord()를 통해 유니코드 번호를 알 수 있고, chr()를 통해 해당 유니코드 번호의 문자를 알 수 있다
- 역슬래쉬로 시작해 표현하는 문자를 이스케이프 시퀀스 문자라고 하며 다음과 같이 사용한다
- min() max()함수를 통해 숫자는 크기에 따라, 문자나 문자열은 코드값의 크기에 따라 최댓값 최솟값을 반환한다

```
>>> ord('다')
45796
>>> chr(45796)
'다'
>>> print('₩"큰따옴표₩"')
"큰따옴표"
>>> max('abcde')
'e'
>>> min('abcd','ef'), min(10,20,30)
('abcd', 10)
```

"\\"	역 슬래시 ("\")
"\'"	홑 따옴표 ("'") - 작은 따옴표
"\""	겹 따옴표 ('"') - 큰 따옴표
"\a"	경고음벨 (BEL)
"\b"	백 스페이스 (BS)
"\f"	폼 피드 (FF)
"\n"	줄 바꿈 (LF)
"\r"	캐리지 리턴 (CR)
"\t"	가로 탭 (TAB)
"\v"	세로 탭 (VT)
"\ooo"	8진수로 나타낸 문자 *ooo*
"\xhh"	16진수로 나타낸 문자 *hh*
"\N{name}"	유니코드 문자 데이터베이스의 문자 이름 Cloud, Spade, Sun, Star, ...
"\uxxxx"	16비트 16진수로 나타낸 문자 *xxxx*
"\Uxxxxxxxx"	32비트 16진수로 나타낸 문자 *xxxxxxxx*

문자열 관련 메소드

- 부분 문자열을 바꾸는 replace()
- 부분 문자열의 출현 횟수를 반환하는 count()
- 문자 사이사이에 문자열을 삽입하는 join()
- 부분 문자열의 첨자를 찾는 find()와 index()
- 문자열을 여러 문자열로 나누는 split()
  split('. ')로 사용하면 .을 기준으로 나눈다
- 알파벳 대문자소문자 변환 upper(), lower()
- 폭을 지정하고 중앙에 문자열 배치 center()

```
>>> str = '사과는 과일이다'
>>> str.replace('사과','바나나')
'바나나는 과일이다'
>>> str.count('과')
2
>>> '.'.join(str)
'사.과.는. .과.일.이.다'
```

```
>>> str="가나 다라 마바"
>>> str.find('가나'), str.index('마바')
(0, 6)
>>> #find는 찾는 문자열이 없으면 -1반환, index는 오류 발생
>>> str.split()
['가나', '다라', '마바']
```

```
>>> str = 'abcde'
>>> str.upper()
'ABCDE'
>>> str.lower()
'abcde'
>>> str.center(30, '*')
'*************abcde*************'
```

3-2 표준 입출력과 문자열

함수와 메소드

함수는 특정 작업을 수행하는 독립적인 코드 모임이고, 메소드는 클래스에 포함된 함수이다. len(str)처럼 함수는 독립적으로 사용 가능하고, 메소드는 str.split()처럼 객체를 통해 사용이 가능하다.

Format을 통한 출력

- print('{} * {} = {}'.format(2, 3, 2*3))으로 순서대로 중괄호에 대입
- print('{0} * {2} = {1}'.format(2, 2*3, 3))으로 대입 순서 조절 가능
- print('{0:5d} / {1:5d} = {2:10.3f}'.format(5, 3, 5/3))로 출력 형식과 전체 폭, 실수라면 .3f를 통해 소수점 이하 폭을 정할 수 있다.

```
>>> print('{} * {} = {}'.format(2, 3, 2*3))
2 * 3 = 6
>>> print('{0} * {2} = {1}'.format(2, 2*3, 3))
2 * 3 = 6
>>> print('{0:5d} / {1:5d} = {2:10.3f}'.format(5, 3, 5/3))
    5 /     3 =      1.667
```

C언어의 포맷팅 스타일 사용 %

- print("%d * %d = %d" % (2, 3, 6))형식으로 사용 가능
- print("%10s" % 'python')로 폭지정과 문자열 사용 가능
- print("%d%%" % 50)로 %출력 가능

```
>>> print("%d * %d = %d" % (2, 3, 6))
2 * 3 = 6
>>> print("%10s" % 'python')
    python
>>> print("%d%%"%50)
50%
```

3-3 논리자료와 연산

3-3 논리자료와 연산

논리유형 bool과 함수 bool

- 논리값의 자료형으로 bool이 주어진다
- 내장함수 bool로 인자의 논리 값을 알 수 있다

```
>>> type(True), type(False)
(<class 'bool'>, <class 'bool'>)
>>> bool(0), bool(''), bool('5'), bool('ab')
(False, False, True, True)
```

논리 연산자

- 논리곱 and, &는 두 항이 모두 참이어야 True 하나라도 틀리면 False
- 논리합 or, |은 두 항이 모두 거짓이어야 False 하나라도 참이면 True
- 배타적 논리합 ^는 두 항이 다르면 True, 같으면 False
- 단항연산자 not은 항이 True면 False. False면 True
- 연산의 우선순위는 not > and > or 순이다

관계연산자

- >, >=, <, <=, ==, !=가 있고 a<b<c등의 연산 가능

관계연산을 이용한 전기 요금 계산

- 200kwh 이하 시간당 730원, 201~400kwh 1260원
  400kwh이상 6060원

```
>>> usage = float(input('전기 사용량을 입력하세요 : '))
전기 사용량을 입력하세요 : 220
>>> less200 = usage <= 200
>>> less400 = 200 < usage <= 400
>>> greater400 = 400 < usage
>>> base = 730*less200 + 1260*less400 + 6060*greater400
>>> print('전기 요금 : %d원'%(base))
전기 요금 : 1260원
```

멤버십연산 in

- 결과를 bool형으로 반환하며 x in s로 s에 x문자열이 있는 지 확인
  x not in s로 s에 x문자열이 없으면 True 반환도 가능

```
>>> fruits = "'사과', '포도', '바나나'"
>>> fruit = input('과일을 검색하세요 : ')
과일을 검색하세요 : 포도
>>> print('해당 과일이 있습니까? {}'.format(fruit in fruits))
해당 과일이 있습니까? True
```

3-3 논리자료와 연산

비트 논리 연산

- 둘 다 1이어야 1인 비트 논리곱 &
- 둘 중 하나라도 1이면 1인 비트 논리합 |
- 둘이 같으면 0, 다르면 1인 비트 배타적 논리합 ^

```
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(23,57))
10진수 23, 2진수 00010111
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(57))
10진수 57, 2진수 00111001
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(23&57))
10진수 17, 2진수 00010001
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(23|57))
10진수 63, 2진수 00111111
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(23^57))
10진수 46, 2진수 00101110
```

비트 논리곱 &로 특정 비트값 알아내기

- 모든 비트가 1인 mask값을 원하는 정수와 논리곱해 a의 특정 비트를 뽑아낼 수 있다

```
>>> a = int(input('정수 하나를 입력하세요 : '))
정수 하나를 입력하세요 : 27
>>> mask = 0b1111 #0xf도 가능
>>> print('정수 {0}의 2진수는 {0:b}'.format(a))
정수 27의 2진수는 11011
>>> print('가장 오른쪽 4비트 : {0:04b} 정수로는 {0}'.format(a&mask))
가장 오른쪽 4비트 : 1011 정수로는 11
```

비트 배타적 논리합 ^을 이용한 암호화

- 비트 배타적 논리합은 특징이 있다

  a ^ a == 0, a ^ 0 == a, a ^ 1 == ~a

  a ^ b == b ^ a, (a ^ b) ^ c == a ^ (b ^ c)

  (a ^ b) ^ b == a ^ (b ^ b) == a ^ 0 == a

  마지막 식을 활용해 a ^ b로 암호화한 뒤, 다시 ^b해주면 a값

```
>>> orgpwd = int(input('비밀번호를 입력하세요 : '))
비밀번호를 입력하세요 : 369369
>>> keymask=135791
>>> encpwd = keymask^orgpwd
>>> print('암호화된 비밀번호 : %d'%encpwd)
암호화된 비밀번호 : 503990
>>> inpwd = int(input('입력했던 비밀번호를 정확히 입력하세요 : '))
입력했던 비밀번호를 정확히 입력하세요 : 369369
>>> result = encpwd ^ keymask
>>> print('로그인 성공시 1, 실패시 0이 나옵니다 : %d'%(inpwd==result))
로그인 성공시 1, 실패시 0이 나옵니다 : 1
```

비트 보수 연산 ~

- 단항연산자로 비트0은 1로, 1은 0으로 변환한다. ~(~n)은 자기 자신이다

3-3 논리자료와 연산

비트 이동 연산자 >>와 <<

- 비트 값이 오른쪽 또는 왼쪽으로 지정한 횟수만큼 이동한다
- <<에 따른 빈자리는 0으로, >>에 따른 빈자리는 원래 정수 부호에 따라 양수면 0, 음수면 1이 채워진다
- <<특성에 따라 a<<n은 a에 2**n만큼 곱한 결과가 나온다

```
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(23))
10진수  23, 2진수 00010111
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(23<<2))
10진수  92, 2진수 01011100
>>> print('10진수 {0:3d}, 2진수 {0:08b}'.format(23>>3))
10진수   2, 2진수 00000010
```

4	**	거듭제곱
5	+x, -x, ~x	단항 덧셈(양의 부호), 단항 뺄셈(음의 부호), 비트 NOT
6	*, @, /, //, %	곱셈, 행렬 곱셈, 나눗셈, 버림 나눗셈, 나머지
7	+, -	덧셈, 뺄셈
8	<<, >>	비트 시프트
9	&	비트 AND
10	^	비트 XOR
11	|	비트 OR
12	in, not in, is, is not, <, <=, >, >=, !=, ==	포함 연산자, 객체 비교 연산자, 비교 연산자
13	not x	논리 NOT
14	and	논리 AND
15	or	논리 OR

연산자의 우선순위

- 지수 연산 > 단항 연산 > 산술 연산 > 비트 연산 > 관계 소속 > 대입 연산 > 논리연산 순이다

음수의 비트 표현

- 정수값을 사용하면 사용자를 위해 2진수에 –값을 표현한 출력된다. 마스크 값을 이용해 논리곱해주면 컴퓨터에 저장되는 값을 볼 수 있다

```
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(-20))
10진수 -20, 2진수 -0010100
>>> print('10진수 {0:2d}, 2진수 {0:08b}'.format(20))
10진수 20, 2진수 00010100
>>> print('10진수 {0:2d}, 2진수 {1:08b}'.format(-20, -20&0xff))
10진수 -20, 2진수 11101100
>>> #0xff는 마스크값이고, -20앞의 8비트는 모두 111111110이다
```

3-4 조건문

3-4 조건문

조건에 논리 값에 따른 선택 if와 elif, else

- if 조건식 :
  문장 — 왼쪽의 형태로 사용하며 첫째 줄의 콜론과 둘째 줄의 들여쓰기는 반드시 사용해야 한다

- if 조건식:
  문장
  elif 조건식:
  문장
  else:
  문장 — 왼쪽의 형태로 사용하며 if조건식에 맞지 않을 때 elif 조건식을 수행하며 elif에도 맞지 않는 나머지 경우에는 else의 문장을 실행한다 elif는 여러 개일 수 있다

```
>>> if a >= 50:
        print("a는 500|상")
elif a >= 40:
        print("a는 400|상")
else:
        print("a는 400|만")


a는 400|만
```

컴퓨터의 현재 시간 얻어오기

- from time import localtime
- localtime().tm_hour은 현재 시 반환
- localtime().tm_min은 현재 분 반환

```
>>> from time import localtime
>>> print(localtime().tm_hour)
1
>>> print(localtime().tm_min)
21
```

중첩 if문

- if문 내에 if문을 사용함으로써 이중으로 조건 처리 가능

```
category = int(input('원하는 음료는? 1. 커피 2. 주스 '))

if category == 1:
    menu = int(input('번호 선택? 1. 아메리카노 2. 카페라테 3. 카푸치노 '))
    if menu == 1:
        print('1. 아메리카노 선택')
    elif menu == 2:
        print('2. 카페라테 선택')
    elif menu == 3:
        print('3. 카푸치노 선택')
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:
tel)] on win32
Type "help", "copyright", "credits" or "license()"
>>>
= RESTART: C:₩Users₩Chanjong Kim₩Desktop₩2020 pytho
y
원하는 음료는? 1. 커피 2. 주스 1
번호 선택? 1. 아메리카노 2. 카페라테 3. 카푸치노 2
2. 카페라테 선택
>>>
```

3-4 조건문

반복을 제어하는 for문

- for 변수 in 시퀀스:　　왼쪽의 형태로 사용하며 콜론과 반복될 문장　　들여쓰기는 필수
- 여러 값을 갖는 시퀀스에서 시퀀스의 마지막 항목까지 변수에 하나씩 값을 할당한다
- 마찬가지로 else: 문장을 사용해 반복이 끝난 후 실행 가능

```
>>> for s in 'python':
        print(s)


p
y
t
h
o
n
```

```
>>> for i in 3,4,5,6,7:
        print(i, end = ' ')


3 4 5 6 7
```

내장함수 range

- for 변수 in range(숫자)를 통해 숫자만큼 반복이 가능
- 변수에 저장되는 값은 0~숫자-1까지
- range(a,b,c)로 a부터 b까지 c의 간격으로 얻어올 수 있다

```
>>> for i in range(5):
        print(i, end = ' ')

0 1 2 3 4
>>> for i in range(5,10):
        print(i, end = ' ')


5 6 7 8 9
>>>
```

지정된 최소 한 자릿수가 포함된 두 자릿수 찾기

- for문과 문자열이 포함되는지 확인하는 함수 n in snum을 통해 찾아낼 수 있다.

```
n = input("10진수의 한 자릿수 입력 >> ")
print('두 자릿수 정수에서 최소 한 자릿수가 %s인 정수 찾기 : '%n)
print(' 결과 '.center(50,'='))

for i in range(10, 100):
    snum = str(i)
    if n in snum:
        print(i, end= ' ')
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45
tel)] on win32
Type "help", "copyright", "credits" or "license()" fo
>>>
= RESTART: C:#Users#Chanjong Kim#Desktop#2020 python
10진수의 한 자릿수 입력 >> 2
두 자릿수 정수에서 최소 한 자릿수가 2인 정수 찾기 :
========== 결과 ==========
12 20 21 22 23 24 25 26 27 28 29 32 42 52 62 72 82 92
>>>
```

3-4 조건문

간단한 반복구조 while

- while 조건: 문장 — 왼쪽 형태로 사용하며 조건이 참인 동안 무한히 실행한다. 조건이 False가 되면 else의 문장이 실행된다
  else: 문장

```
>>> n=1
>>> while n<=5:
        print(n, end = ' ')
        n+=1
else:
        print("₩n 반복 종료")


1 2 3 4 5
 반복 종료
```

이중 for문을 이용한 구구단

- for문 내에 for문을 넣어 이중으로 반복할 수 있다

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

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

난수를 생성하는 방법

- import random 후에 random.randint(1, 10)을 사용하면 1~10 사이의 수를 반환
- from random import randint를 통해 randint(1,10)으로 바로 사용 가능

```
>>> import random
>>> random.randint(1, 10)
8
>>> from random import randint
>>> randint(1,10)
3
```

3-4 조건문

반복을 제어하는 break와 continue

- 반복문 내부에 break를 통해 반복문을 탈출할 수 있고 continue를 통해 continue 아래의 값을 무시하고 다음 반복으로 넘어갈 수 있다

Continue와 break로 여러 영단어중 하나의 철자 맞히기

- 아래의 값들을 무시하고 반복을 진행하는 continue를 이용한다

```
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,'*'))
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020
tel)] on win32
Type "help", "copyright", "credits" or "licens
>>>
= RESTART: C:₩Users₩Chanjong Kim₩Desktop₩2020
.py
월, 화, 수 중 하나 영어 단어 입력 >> mondayy
잘못 입력했어요!
월, 화, 수 중 하나 영어 단어 입력 >> monday
입력: monday, 철자가 맞습니다.
****** 종료 *****
```

Random함수와 반복을 이용한 복권 모의 실험

```
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')
```

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb
tel)] on win32
Type "help", "copyright", "credits" or
>>>
== RESTART: C:₩Users₩Chanjong Kim₩Desk
======= 모의 로또 당첨 번호 ========
(11, 17, 28, 30, 33, 35)

---------- 내 번호 확인 ----------
16 X 34 X 15 X 24 X 28 O 42 X
1 개 맞음
>>>
```

3-5 리스트와 튜플

3-5 리스트와 튜플

관련된 항목을 나열하는 리스트

- 리스트명 = ['항목1', '항목2' ...]의 형태이며 대괄호를 사용
- 타입은 리스트이며, 리스트 또한 시퀀스이다

```
>>> fruits = ['사과', '포도', '바나나']
>>> print(fruits)
['사과', '포도', '바나나']
>>> type(fruits)
<class 'list'>
>>> for s in fruits:
        print(s)


사과
포도
바나나
```

리스트의 생성함수 list()와 항목 수 반환함수 len()

- 리스트명 = ['항목1'...]로 생성하거나, 리스트명 = []로 빈 리스트를 만든 후 수정한다
- 내장함수 list()를 통해 리스트명 = list() 빈 리스트를 생성한 후 리스트명.append('항목')메소드로 값을 추가한다
- 항목 모두가 문자인 리스트를 만드려면 py = ['p', 'y', 't', 'h', 'o', 'n'] 또는 py = list('python')을 쓰면 항목이 문자로 이루어진다
- len(리스트명)으로 리스트의 항목 수를 반환받을 수 있다

```
>>> fruit = list()
>>> fruit.append('사과')
>>> fruit.append('포도')
>>> print(fruit)
['사과', '포도']
>>> len(fruit)
2
>>> py = list('python')
>>> py
['p', 'y', 't', 'h', 'o', 'n']
```

리스트의 항목 참조

- 첫 항목이 0으로 시작해 1씩 증가시켜 첨자로 참조할 수 있다
- 반대 방향도 가능하고 리스트[a:b:c]로 c간격으로 참조 가능

```
>>> py = list('python')
>>> py[0], py[5]
('p', 'n')
>>> py[-1], py[-2]
('n', 'o')
```

3-5 리스트와 튜플

리스트를 참조하는 메소드 count()와 index()

- count()로 해당 값을 갖는 항목의 개수를 센다
- index()로 해당 값을 갖는 항목이 위치한 첨자를 반환한다
- 리스트의 첨자로 항목 수정이 가능

```
>>> py = list('python')
>>> py.count('t')
1
>>> py.index('t')
2
```

```
>>> py = list('python')
>>> py[py.index('t')]='y'
>>> print(py)
['p', 'y', 'y', 'h', 'o', 'n']
```

리스트의 부분 수정

- 슬라이싱을 통해 여러 항목의 수정이 가능하다
- 여기서 ['수박']이 아닌 '수박'으로 대입했다면 항목 수박이 아니라 리스트 수박으로 대입해서 '수', '박', '바나나', '딸기 ' 가 된다

```
>>> fruits = ['사과', '포도', '바나나', '딸기']
>>> fruits[0:2]=['수박']
>>> print(fruits)
['수박', '바나나', '딸기']
```

리스트의 삽입메소드 insert와 삭제 remove, pop(첨자), pop()

- insert(첨자위치, 항목)으로 삽입한다
- 메소드 remove(항목)은 항목을 삭제하고, pop(첨자)는 해당 첨자의 항목을 지운 뒤 그 항목을 반환한다
  pop()은 마지막 항목을 삭제하고 반환한다

```
>>> fruits = ['사과', '포도', '바나나', '딸기']
>>> fruits.insert(1,'수박')
>>> print(fruits)
['사과', '수박', '포도', '바나나', '딸기']
>>> fruits.remove('바나나')
>>> print(fruits)
['사과', '수박', '포도', '딸기']
>>> fruits.pop()
'딸기'
>>> print(fruits)
['사과', '수박', '포도']
```

3-5 리스트와 튜플

변수나 항목의 삭제 del과 모든 항목 제거 clear()

- del 리스트명[첨자]로 항목을 삭제한다 슬라이스로 삭제 가능
  del 리스트명은 리스트 자체를 메모리에서 삭제한다
- 리스트명.clear()로 리스트를 빈 리스트로 만든다

```
>>> fruits = ['사과', '포도', '바나나', '딸기']
>>> del fruits[0:2]
>>> print(fruits)
['바나나', '딸기']
>>> del fruits
>>> print(fruits)
Traceback (most recent call last):
  File "<pyshell#88>", line 1, in <module>
    print(fruits)
NameError: name 'fruits' is not defined
>>> fruits = ['사과', '포도', '바나나', '딸기']
>>> fruits.clear()
>>> fruits
[]
```

리스트끼리 추가하는 메소드 extend와 + 연산자

- 리스트1.extend(리스트2)로 리스트1 뒤에 리스트2 연결
- 리스트3 = 리스트1+리스트2로 문자열 연결 *연산자도 가능

```
>>> a = ['가', '나', '다']
>>> b = ['라', '마', '바']
>>> a.extend(b)
>>> print(a)
['가', '나', '다', '라', '마', '바']
>>> c = a + ['사', '아','자']
>>> print(c)
['가', '나', '다', '라', '마', '바', '사', '아', '자']
```

리스트의 항목 순서 변경 reverse(), sort(), 함수 sorted()

- 항목 순서를 뒤집는 메소드 reverse()
- 오름차순 정렬 메소드 sort() sort(reverse = True)로 내림차순
- 오름차순 정렬 함수 sorted() sorted(reverse = True)로 내림차순

```
>>> a = ['나', '라', '다', '가']
>>> a.reverse()
>>> a
['가', '다', '라', '나']
>>> a.sort()
>>> a
['가', '나', '다', '라']
>>> sorted(a,reverse=True)
['라', '다', '나', '가']
```

3-5 리스트와 튜플

리스트 컴프리헨션

- 조건의 만족으로 리스트를 간결히 생성
- a = []
  for i in range(2, 11, 2): 를 a = [i for i in range(2,11,2)]로 축소
      a.append(i)

- 홀수 리스트 생성
  odd = []
  for i in range(10): 를 a = [i for i in range(10) if i%2 ==1]
      if i%2 == 1: 로 축소
          odd.append(i)

```
>>> a = [i for i in range(10) if i%2 == 1]
>>> a
[1, 3, 5, 7, 9]
```

리스트 = [리스트에 들어갈 값 for 항목 in 시퀀스 if 조건식] 형태

리스트의 얕은 복사와 깊은 복사

- 리스트 f1이 있을 때 f2 = f1은 얕은 복사로 같은 메모리 공간을 써 f2의 변화가 f1에 적용됨
- f2 = f1[:]슬라이싱을 이용하거나 copy(), list()를 이용하면 깊은 복사로 f2는 새로운 리스트

변수의 동일 객체 여부를 검사하는 is

- 피연산자 변수 2개가 동일한 메모리를 공유하는지 검사 같으면 True

```
>>> a = ['가', '나', '다']
>>> b = a
>>> print(a is b)
True
>>> c = a[:]
>>> print(a is c)
False
```

3-5 리스트와 튜플

수정할 수 없는 항목의 나열 튜플

- 튜플명 = ('항목', '항목'...)으로 구성되며 각 항목의 타입은 제한이 없다
- 타입은 튜플이며, 하나의 항목을 가진 튜플을 만들 땐 tup = 1, 과 같이 콤마를 붙인다
- 슬라이스로 읽을 순 있으나 튜플 특성상 수정은 불가능하다

```
>>> tup = ("a", 2, "c")
>>> type(tup)
<class 'tuple'>
>>> tupp = 1,
>>> type(tupp)
<class 'tuple'>
>>> tup[0:1]
('a',)
>>> tup[0:2]
('a', 2)
```

튜플의 연결 +와 반복 *

- 튜플을 연결하거나, 반복 횟수만큼 반복된 튜플을 반환한다

```
>>> tup1 = ("사과", "과일")
>>> tup2 = ("양파", "야채")
>>> tup1 + tup2
('사과', '과일', '양파', '야채')
>>> tup1*3
('사과', '과일', '사과', '과일', '사과', '과일')
```

튜플의 순서정렬 함수 sorted()와 튜플 제거 del()

- sorted(튜플)또는 sorted(튜플, reverse=True)로 순서 정렬
- del(튜플)로 튜플을 메모리에서 완전히 제거

```
>>> tup = ("나", "다", "가", "라")
>>> sorted(tup)
['가', '나', '다', '라']
>>> sorted(tup, reverse=True)
['라', '다', '나', '가']
>>> del(tup)
>>> tup
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    tup
NameError: name 'tup' is not defined
```

수정 가능 객체와 불가능 객체

- 문자열과 튜플은 수정이 불가능하다
- id()로 저장된 메모리 위치를 알 수 있는데, 문자열 내의 각 문자
  라던지 튜플은 동일한 리스트라면 같은 공간을 사용한다
  리스트 등은 동일한 리스트여도 다른 공간을 사용한다

3-6 딕셔너리와 집합

3-6 딕셔너리와 집합

키와 값의 쌍을 항목으로 관리하는 딕셔너리

- 딕셔너리 = {키:값, 키:값,...}의 형태
- 항목 순서는 의미가 없고, 키는 중복될 수 없음
- 딕셔너리명[키]='값'을 통해 삽입 가능
- 키 값을 이용해 값을 참조할 수 있다

```
>>> food = {'한식':'불고기', '일식':'초밥', '양식':'스테이크'}
>>> print(food)
{'한식': '불고기', '일식': '초밥', '양식': '스테이크'}
>>> food['중식']='탕수육'
>>> print(food)
{'한식': '불고기', '일식': '초밥', '양식': '스테이크', '중식': '탕수육'}
>>> food['한식']
'불고기'
```

딕셔너리의 다양한 인자 사용

- 값으로 리스트, 튜플 등을 사용할 수 있다
- 키:값의 형태가 아니라 [키,값]리스트 형식과 (키,값)튜플 형식 가능
- 키가 단순 문자열이면 월='Monday'처럼 단순 나열할 수 있다
- 딕셔너리의 키는 수정 불가능한 객체인 정수, 실수 등을 사용
- 튜플은 키로 가능하지만 리스트는 불가능

```
>>> dic = dict()
>>> dic['abc']=[1,2,3,4]
>>> dic['abc']
[1, 2, 3, 4]
>>> day=dict(())
>>> day=((('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')))
>>> print(day)
(('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday'))
>>> day2 = dict(월='monday', 화='tuesday', 수='wednesday')
>>> day2
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
```

딕셔너리의 메소드

- keys() 메소드로 키로만 구성된 리스트 반환
- items() 메소드로 (키,값)쌍의 튜플이 들어있는 리스트 반환
- values() 메소드로 값으로만 구성된 리스트 반환

```
>>> day2 = dict(월='monday', 화='tuesday', 수='wednesday')
>>> print(day2.keys())
dict_keys(['월', '화', '수'])
>>> print(day2.items())
dict_items([('월', 'monday'), ('화', 'tuesday'), ('수', 'wednesday')])
>>> print(day2.values())
dict_values(['monday', 'tuesday', 'wednesday'])
```

3-6 딕셔너리와 집합

- for문으로 모든 키를 순회하는 방법

```
>>> for key in day2:
        print('%s: %s'%(key, day2[key]))


월: monday
화: tuesday
수: wednesday
```

- get(키, 키가 없을 때 반환값) 메소드로 값 조회
- pop(키, 키가 없을 때 반환값) 메소드로 항목 삭제 후 삭제한 값 반환
- popitem() 메소드로 임의의 항목 삭제

```
>>> day2 = dict(월='monday', 화='tuesday', 수='wednesday')
>>> day2.get('화','해당 키가 없습니다')
'tuesday'
>>> day2.pop('목','해당 키가 없습니다')
'해당 키가 없습니다'
>>> print(day2.popitem())
('수', 'wednesday')
>>> print(day2)
{'월': 'monday', '화': 'tuesday'}
```

- del 딕셔너리[키]로 항목 삭제 또는 del 딕셔너리로 메모리에서 딕셔너리 자체를 삭제

```
>>> day2 = dict(월='monday', 화='tuesday', 수='wednesday')
>>> del day2['수']
>>> day2
{'월': 'monday', '화': 'tuesday'}
>>> del day2
>>> day2
Traceback (most recent call last):
  File "<pyshell#70>", line 1, in <module>
    day2
NameError: name 'day2' is not defined
```

- update() 메소드로 괄호의 딕셔너리 합병 (동일한 키가 있었다면 새로운 값으로 대체)
- clear() 메소드로 딕셔너리 모든 항목 삭제

```
>>> day = dict(월='monday',화='tuesday')
>>> day2 = dict(수='wednesday',목='thursday')
>>> day.update(day2)
>>> day
{'월': 'monday', '화': 'tuesday', '수': 'wednesday', '목': 'thursday'}
>>> day.clear()
>>> day
{}
```

- 문장 in을 이용해 해당 키가 존재하는지 검사

```
>>> '월' in day
True
>>> '금' not in day
True
```

3-6 딕셔너리와 집합

중복되는 요소가 없고 순서가 없는 집합

- s1={1,2,3} s2={'hi','go'} 형태로 사용
- 원소로 정수, 실수, 문자열, 튜플 등 수정 불가능한 항목 사용
- 집합의 원소는 중복을 허용하지 않으므로 멤버십 검사 또는 중복 제거에 주로 사용

```
>>> s1 = {1,2,3}
>>> s2 = {'hi', ('a','b')}
>>> type(s1), type(s2)
(<class 'set'>, <class 'set'>)
>>> s1 = {1,2,2,3}
>>> s1
{1, 2, 3}
```

내장함수 set()

- s = set(원소1, 원소2, ...)으로 사용하며 {}는 빈 딕셔너리라서 set()가 공집합을 의미
- 함수 set으로는 인자로 리스트, 튜플 자체(원소x)를 사용할 수 있고 결과는 시퀀스 항목에서 중복을 제거한 원소로 구성됨 원소로 리스트나 딕셔너리를 사용할 순 없음.

```
>>> s = set([1,2,3])
>>> s
{1, 2, 3}
>>> d = set((4,5,6))
>>> d
{4, 5, 6}
>>> e = set('abcd')
>>> e
{'c', 'd', 'a', 'b'}
```

집합의 메소드

- add(원소) 메소드로 원소 추가
- remove(원소), discard(원소), pop() 메소드로 항목 삭제
- clear()메소드로 모든 원소 삭제

```
>>> a = {1, 2, 3}
>>> a.add(4)
>>> a.add(5)
>>> a
{1, 2, 3, 4, 5}
>>> a.remove(1)
>>> a.discard(2)
>>> a
{3, 4, 5}
>>> print(a.pop())
3
>>> a
{4, 5}
```

3-6 딕셔너리와 집합

- 합집합 |(메소드 union(), update()), 교집합 &(메소드intersection())
  차집합 –(메소드 difference()), 여집합 ^(메소드 symmetric_difference())
  |=, &=, -=, ^=로 축약 대입 가능하다
- len()함수로 집합의 원소 개수 확인
- 멤버십 연산자 in으로 특정 원소가 집합에 있는지 확인

```
>>> a={1,2,3}
>>> b={3,4,5}
>>> a|b
{1, 2, 3, 4, 5}
>>> a.union(b)
{1, 2, 3, 4, 5}
>>> a&b
{3}
>>> a-b
{1, 2}
>>> a^b
{1, 2, 4, 5}
>>> a^=b
>>> a
{1, 2, 4, 5}
```

```
>>> a={1,2,3,"hi"}
>>> len(a)
4
>>> 2 in a
True
>>> 6 in a
False
>>> 'hi' in a
True
```

내장함수 zip()

- zip() 함수로스트나 튜플 항목으로 조합된 항목 생성
- zip()의 결과를 list()등의 내장함수 인자로 사용하면 리스트 생성 가능

```
>>> a = ['A', 'B', 'C']
>>> b = (80, 90, 85)
>>> z = zip(a,b)
>>> type(z)
<class 'zip'>
>>> list(z)
[('A', 80), ('B', 90), ('C', 85)]
>>> tuple(zip(a,b))
(('A', 80), ('B', 90), ('C', 85))
>>> dict(zip(a,b))
{'A': 80, 'B': 90, 'C': 85}
```

내장함수 enumerate

- 0부터 시작하는 첨자와 항목 값의 튜플 리스트를 만들어줌
- 반복으로 편리하게 사용 가능(*tp는 자동으로 첨자/값을 구별)
- dict(enumerate(튜플or리스트))로 첨자:값의 딕셔너리 생성 가능

```
>>> a = [10, 20, 30]
>>> list(enumerate(a))
[(0, 10), (1, 20), (2, 30)]
>>> sub = ['국어', '영어', '수학', '과학']
>>> for tp in enumerate(sub):
        print('lst[{}]: {}'.format(tp[0], tp[1]))
        print('lst[{}]: {}'.format(*tp))


lst[0]: 국어
lst[0]: 국어
lst[1]: 영어
lst[1]: 영어
lst[2]: 수학
lst[2]: 수학
lst[3]: 과학
lst[3]: 과학
```

시퀀스 간의 변환

- list, tuple, set, dict 내장함수를 사용해 시퀀스 간의 변환 가능
- 딕셔너리를 다른 시퀀스로 변환하면 키만 받아옴

```
>>> a = ('가','나','다','라')
>>> print(list(a))
['가', '나', '다', '라']
>>> a
('가', '나', '다', '라')
>>> a=list(a)
>>> a
['가', '나', '다', '라']
>>> tuple(a)
('가', '나', '다', '라')
```

```
>>> day = dict(월='monday', 화='tuesday', 수='wednesday')
>>> day
{'월': 'monday', '화': 'tuesday', '수': 'wednesday'}
>>> day2 = list(day)
>>> day2
['월', '화', '수']
>>> tuple(day)
('월', '화', '수')
```

3-7 사용자 정의 함수와 내장 함수

특정 기능을 수행하는 함수

- 사용자 정의 함수와 내장 함수가 있으며, 내장 함수는 파이썬 내에 설치되어 있는 input, int 등의 함수로 마무리 부분에 정리
- 사용자 정의 함수는 함수 정의와 함수 호출이 필요하다
- def 함수이름(): 형태로 함수를 정의하며 함수이름()으로 호출한다.
  문장

```
>>> def sum(a, b):
        print("합 : ",a+b)


>>> sum(3, 5)
합 :  8
```

- 괄호 사이에 인자를 넣어 호출 시 해당 인자를 함수로 넘겨줌으로써 함수 내의 문장에서 해당 인자를 사용할 수 있다
- 인자를 안 넣었을 때 인자로 사용할 기본값을 설정할 수 있다
- 함수 내에 return문을 넣음으로써 함수가 return문을 만나면 함수가 종료되며 return에 반환값이 있다면 해당 값을 반환한다
  파이썬에서는 return값을 여러 개 반환할 수 있다
  함수 gett의 반환값이 네개라면 a, b, c, d = get()로 받아올 수 있다

```
>>> def sum(a, b=5):
        print("합 : ",a+b)


>>> sum(6)
합 :  11
```

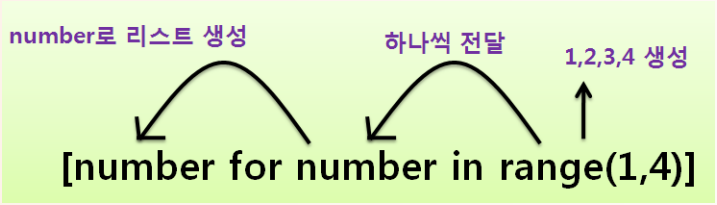
- 함수 내부에서 대입에 사용되었다면 지역 변수로 인식되고(새로운 변수)
  함수 내부에서 대입 없이 참조만 하면 전역 변수로 인식된다(기존 변수)
  global 변수명 으로 선언해 항상 전역변수로 사용 가능

- 함수 내에 """ 쓰고 싶은 문장 """을 통해 출력 결과에 영향을 주지 않는 주석을 달 수 있다. """내부에서 공백, 줄바꿈 등 모두 그대로 출력된다

4. 마무리

시퀀스 : 리스트, 튜플, range 등 연속적으로 이어진 자료형

컴프리헨션 : 조건을 만족하는 항목으로 간결하게 생성
range 뒤에 if 조건으로 조건 추가 가능



Random 함수 : from random import..로 바로 사용 가능
..에 randint(a,b)로 a이상 b이하 값 반환
randrange(a,b,c)로 a이상 b미만 c숫자의 간격으로 랜덤하게 반환

출력 : format사용
print('{0:5d} * {1:5d} = {2:6d}'.format(2, 3, 2*3))
%사용
print("%d * %d = %d" % (2, 3, 6))

abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

파이썬 내장함수

포트폴리오 후기

기계에서 컴퓨터정보공학과로 2학년 전과를 진행하며 프로그래밍에 대해 거의 무지했습니다.
심지어 모든 수업이 비대면 강의로 이루어지며 필요한 부분을 바로바로 물어보지 못하는 상태였고,
그런 부분에서 각 개념에 대한 자세한 설명과 많은 예제, 책 중간중간에 있는 팁 부분은 아직 모르는 점이 많던 저에게 큰 도움이 되었습니다.

포트폴리오를 작성해 모든 개념들을 다시 한 번 돌아보면서, 기억하지 못하고 있었던 몇몇 기능들이 눈에 들어왔습니다. 웬만한 함수, 메소드들은 다 기억하고 있다고 생각했지만 이런 개념을 까먹고 있었다고? 싶은 몇가지 항목들을 보면서 지금은 다시 한 번 정리해보길 잘했다는 생각이 듭니다.

각 내용들을 직접 찾아보고, 간단한 예제로나마 직접 코딩을 해보며 다시 한 번 모든 개념들을 되짚어볼 수 있었고, 포트폴리오 작성을 통해 개념 위주로 정리해놓아 언제든지 다시 확인할 수 있는 좋은 정리본을 만들어 추후에도 필요하다면 사용하고자 합니다.

아직 파이썬을 어디까지 활용할 수 있을 지는 감이 잘 오지 않지만 앞으로 있을 수업에 더 진지하게 임해 파이썬에 관한 다양한 기능들과 모듈, 활용방안들을 완벽하게 익힐 기회가 되었으면 합니다.

감사합니다.