

파이썬 기초

Python Basic

류영표 강사

ryp1662@gmail.com

Copyright © "Youngpyo Ryu" All Rights Reserved.

This document was created for the exclusive use of "Youngpyo Ryu".

It must not be passed on to third parties except with the explicit prior consent of "Youngpyo Ryu".



류영표

Youngpyo Ryu

동국대학교 수학과/응용수학 석사수료

現 Upstage AI X 네이버 부스트 캠프 AI tech 1~5기 멘토

前 Innovation on Quantum & CT(IQCT) 이사

前 한국파스퇴르연구소 Image Mining 인턴(Deep learning)

前 (주)셈웨어(수학컨텐츠, 데이터 분석 개발 및 연구인턴)

강의 경력

- 현대자동차 연구원 강의 (인공지능/머신러닝/딥러닝/강화학습)
- (주) 모두의연구소 Aiffel 1기 퍼실리테이터(인공지능 교육)
- 인공지능 자연어처리(NLP) 기업데이터 분석 전문가 양성과정 멘토
- 공공데이터 청년 인턴 / SW공개개발자대회 멘토
- 고려대학교 선도대학 소속 30명 딥러닝 집중 강의
- 이젠 종로 아카데미(파이썬, ADSP 강사)
- 최적화된 도구(R/파이썬)을 활용한 애널리스트 양성과정(국비과정) 강사
- 한화, 하나금융사, 한전 KDN 교육
- 인공지능 신뢰성 확보를 위한 실무 전문가 자문위원
- 보건·바이오 AI활용 S/W개발 및 응용전문가 양성과정 강사
- Upstage AI X KT 융합기술원 기업교육 모델최적화 담당 조교

주요 프로젝트

및 기타사항

- 개인 맞춤형 당뇨병 예방·관리 인공지능 시스템 개발 및 고도화(안정화)
- 폐플라스틱 이미지 객체 검출 경진대회 3위
- 인공지능(AI)기반 데이터 사이언티스트 전문가 양성과정 1기 수료
- 제 1회 산업 수학 스터디 그룹 (질병에 영향을 미치는 유전자 정보 분석)
- 제 4,5회 산업 수학 스터디 그룹 (피부암, 유방암 분류)
- 빅데이터 여름학교 참석 (혼잡도를 최소화하는 새로운 노선 건설 위치의 최적화 문제)

Contents



사진 출처 : medium.com/swlh/5-free-python-courses-for-beginners-to-learn-online-e1ca90687caf

1. 파일 소개

2. 파일의 기본

3. 파일의 자료형

4. 조건문

5. 반복문

6. 파일 읽기/쓰기

7. 함수

8. 클래스 / 모듈

1. 파이썬 소개

- Why python?
- Programming Language Ranking
- What is programming?
- 코딩 교육
- 파이썬 이란?
- Python의 특징
- Python 2 VS python 3
- 컴파일러 VS 인터프리터
- Why 파이썬?
- 다양한 라이브러리 지원
- 파이썬 개발환경
- 개발 환경 개요
- 운영체제-Operation System
- 코드 편집기(Editor)
- ANACONDA 설치
- Jupyter Notebook 사용하기
- Google colab 사용법
- Goggle colab 런타임 방지-Python
- IDE Pycharm 이란?

Why Python

TIOBE Index						PYPL Index (Worldwide)				
May ▲ 2021	May ♦ 2020	Change ♦	Programming language	Ratings ♦	Change ♦	May ▲ 2021	Change ♦	Programming language	Share ♦	Trends
1	1		C	13.38%	-3.68%	1		Python	29.9 %	-1.2 %
2	3	↑	Python	11.87%	+2.75%	2		Java	17.72 %	-0.0 %
3	2	↓	Java	11.74%	-4.54%	3		JavaScript	8.31 %	+0.4 %
4	4		C++	7.81%	+1.69%	4		C#	6.9 %	-0.1 %
5	5		C#	4.41%	+0.12%	5	↑	C/C++	6.62 %	+0.9 %
6	6		Visual Basic	4.02%	-0.16%	6	↓	PHP	6.15 %	+0.1 %
7	7		JavaScript	2.45%	-0.23%	7		R	3.93 %	+0.0 %
8	14	↑↑	Assembly language	2.43%	+1.31%	8		Objective-C	2.52 %	+0.1 %
9	8	↓	PHP	1.86%	-0.63%	9		Swift	1.96 %	-0.2 %
10	9	↓	SQL	1.71%	-0.38%	10	↑	TypeScript	1.89 %	+0.0 %
11	15	↑↑	Ruby	1.50%	+0.48%	11	↓	Matlab	1.71 %	-0.2 %
12	17	↑↑	Classic Visual Basic	1.41%	+0.53%	12		Kotlin	1.62 %	+0.1 %
13	10	↓	R	1.38%	-0.46%	13	↑	Go	1.42 %	+0.1 %
14	38	↑↑	Groovy	1.25%	+0.96%	14	↓	VBA	1.33 %	-0.0 %
15	13	↓	MATLAB	1.23%	+0.06%	15	↑↑↑↑↑	Rust	1.13 %	+0.4 %
16	12	↓↓	Go	1.22%	-0.05%	16	↓	Ruby	1.12 %	-0.1 %
17	23	↑↑	Delphi/Object Pascal	1.21%	+0.60%	17	↑↑↑↑↑↑↑↑↑↑↑	Ada	0.72 %	+0.3 %
18	11	↓↓	Swift	1.14%	-0.65%	18	↓	Visual Basic	0.7 %	-0.2 %
19	18	↓	Perl	1.04%	+0.16%	19	↓↓↓	Scala	0.67 %	-0.4 %
20	34	↑↑	Fortran	0.83%	+0.51%	20	↓	Abap	0.61 %	+0.1 %
						21	↓	Dart	0.55 %	+0.0 %
						22	↑↑	Lua	0.49 %	+0.1 %
						23	↑↑↑	Julia	0.42 %	+0.1 %
						24	↓↓↓	Groovy	0.41 %	-0.0 %
						25	↓↓↓	Perl	0.4 %	-0.0 %
						26	↓↓↓	Cobol	0.36 %	-0.1 %
						27	↑	Delphi/Pascal	0.24 %	-0.0 %
						28	↓	Haskell	0.21 %	-0.1 %

What is programming?

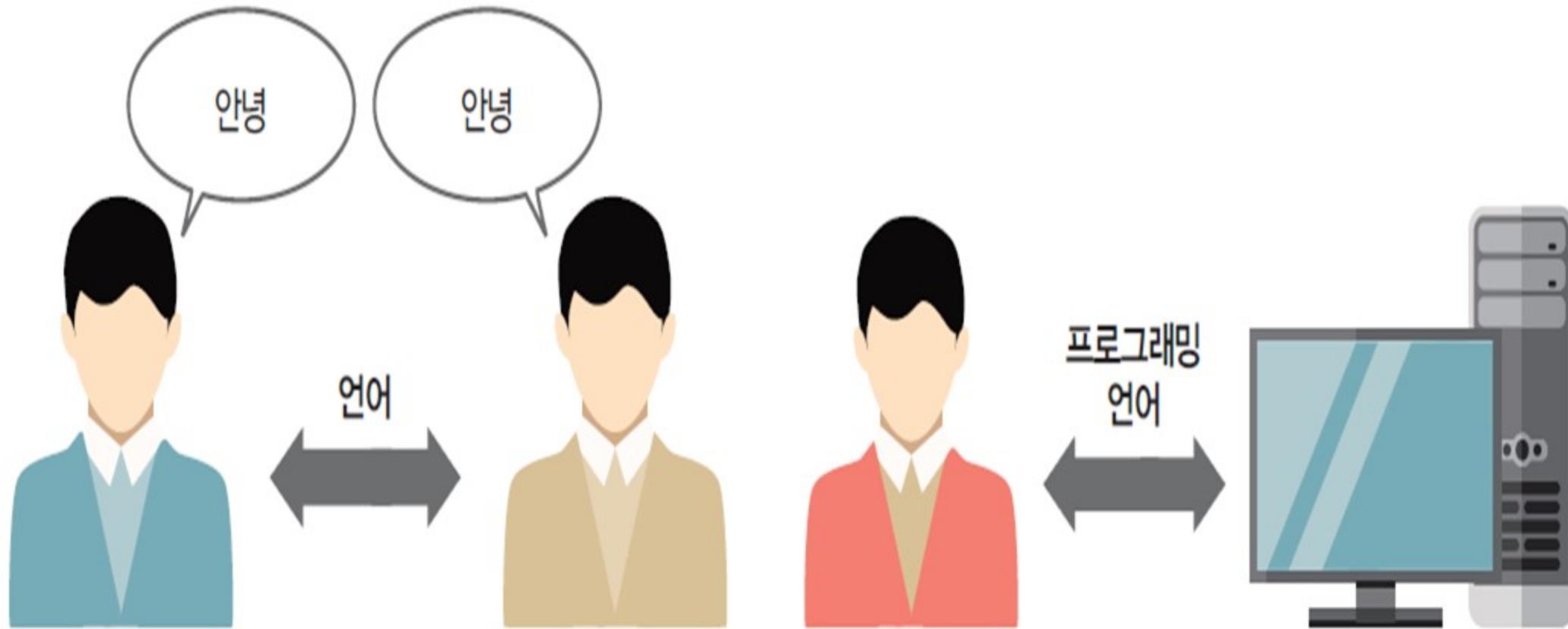


Fig. 1: 코딩교육은 왜 필요할까요?(코딩으로 준비하는 미래)



“세상의 모든 사람은 컴퓨터 프로그래밍을 배워야 합니다.
프로그래밍은 생각하는 법을 가르쳐 주기 때문입니다.”

“Everybody in this country should learn to program a computer,
because it teaches you how to think”

- 스티브 잡스 -



“프로그램은 사고력과 문제해결력을 향상시킵니다.”

“Learning to write programs stretches your mind,
and helps you think better.”

- 빌 게이츠 -



“코딩 교육은 당신의 미래일 뿐만 아니라 조국의 미래다.”

“Learning these skills isn't just important for your future,
it's important for our country's future.”

- 버락 오바마 -

파이썬 이란?

- 1989년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발
- C, C++, 자바 등 어떤 컴퓨터 프로그래밍 언어보다 배우기 쉬움
- 직관적이고 이해하기 쉬운 문법
- 객체 지향의 고수준 언어
- 앱(App)과 웹(WEB) 프로그램 개발 목적
- 웹 서버, 과학 연산, 사물 인터넷(IOT), 인공지능, 게임 등의 프로그램 개발하는 강력한 도구



파이썬 이란?

- 'Month Python's Flying Circus' 영국의 코미디에서 영감을 얻어 Python제작
- 무료한 크리스마스를 보내기 위하여 만든 프로그램.
- Python(피톤)은 그리스 신화 속의 괴물뱀



Python의 특징

"Life is too short, You need python." (인생은 너무 짧으니 파이썬이 필요해.)

- 1. **직관적이고 쉽다.**
 - 아주 간단한 영어 문장을 읽듯이 보고 쉽게 이해할 수 있도록 구성
- 2. **널리쓰인다.**
 - 구글, 아마존, 인스타그램, IBM, 디즈니, 야후, 유튜브, 노키아, NASA 등과 네이버, 카카오톡의 주력 언어 중 하나
- 3. **개발 환경이 좋다.**
 - 게임, 인공지능, 수치해석 등 다양한 라이브러리와 커뮤니티 활성화

Python의 특징

C++

```
#include <iostream>
using namespace std;
int main() {
    cout<<"Hello, gabia!";
    return 0;
}
```

Java

```
public class HelloGabia {
    public static void main(String args[]) {
        System.out.println("Hello, gabia!");
    }
}
```

Python

```
print("Hello, gabia!")
```

C++

```
#include <iostream>
using namespace std;
int main() {
    int arr[5] = {1, 2, 3, 4, 5};

    for(int i=0; i<5; i++) {
        cout << arr[i] << endl;
    }
    return 0;
}
```

Java

```
public class HelloGabia {
    public static void main(String args[]) {
        int arr[] = {1, 2, 3, 4, 5};

        for(int i=0; i<5; i++) {
            System.out.println(arr[i]);
        }
    }
}
```

Python

```
arr = [1, 2, 3, 4, 5]
for el in arr:
    print(el)
```

Python의 특징

- **플랫폼 = OS**

: 윈도우, 리눅스, 안드로이드, 맥OS, IOS 등 프로그램이 실행되는 운영체제를 플랫폼이라고 함.

- **독립적인 = 관계없는, 상관없는**

: OS에 상관없이 한번 프로그램을 작성하면 사용 가능.

- **인터프리터 = 통역기를 사용하는 언어**

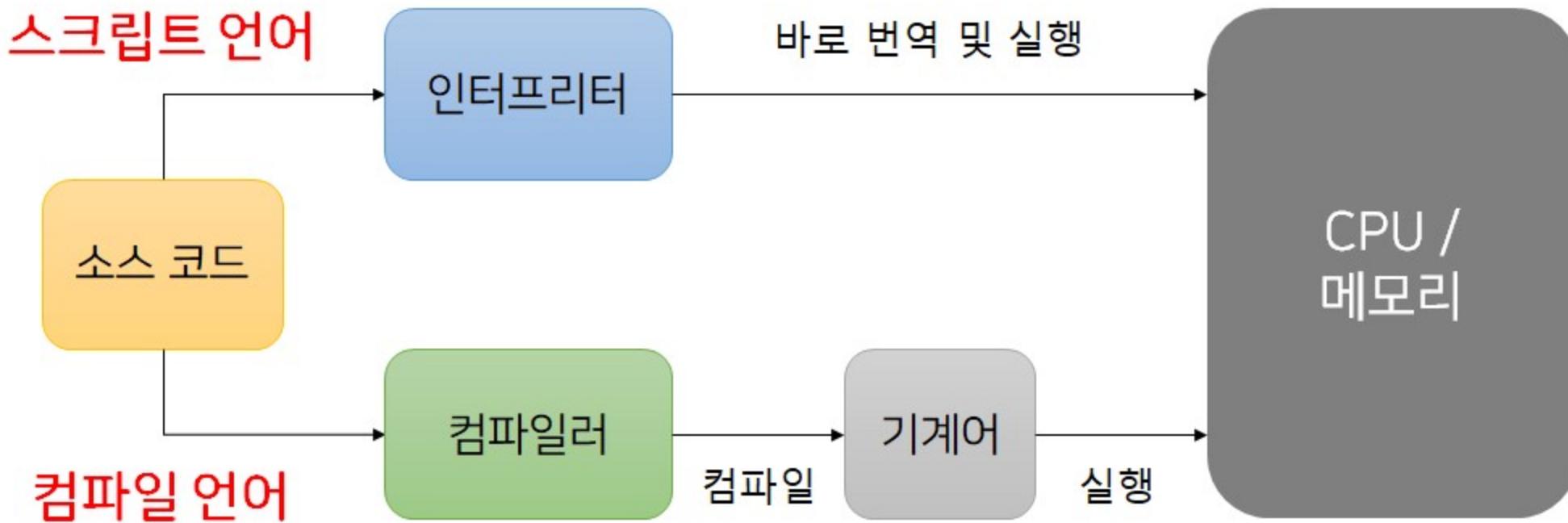
: 소스코드를 바로 실행 할 수 있게 지원하는 프로그램 실행 방법.

- **플랫폼 독립적인 인터프리터 언어**

컴파일러 VS 인터프리터

특징 \ 방식	컴파일러	인터프리터
번역 방법	프로그램 전체 번역	실행되는 줄(라인) 단위 번역
장점	한 번 컴파일한 후에는 매번 빠른 시간 내에 전체 실행 가능	번역 과정이 비교적 간단하고 대화형 언어에 편리함
단점	프로그램의 일부를 수정하는 경우에도 전체 프로그램을 다시 컴파일해야 함	실행할 때마다 매번 기계어로 바꾸는 과정을 다시 수행해야 하므로 항상 인터프리터가 필요함
출력물	목적 코드	즉시 실행
언어 종류	FORTRAN, COBOL, C 등	BASIC 등

[참고] 프로그램의 동작 과정



Python의 특징

- **객체 지향적 언어**

- 실행 순서가 아닌 단위 모듈(객체) 중심으로 프로그램을 작성
- 하나의 객체는 어떤 목적을 달성하기 위한 행동(method)와 속성(attribute)을 가지고 있음.

- **동적 타이핑 언어**

- 프로그램이 실행하는 시점에 프로그램이 사용해야 할 데이터에 대한 타입을 결정함.

자바 VS 파이썬

Dynamic Typing

: 코드 실행시점에 데이터의 Type을 결정하는 방법

Java	Python
<pre>public class Main { public static void main(String[] args) { int a = 10, b=20; int result = a+b; System.out.println("The result a+b = " + result); } }</pre>	<pre>a=10 b=20 print('The result a+b = ', a+b)</pre>

Output:

The result a+b = 30	The result a+b = 30
---------------------	---------------------

Java	Python
<pre>public class Main { public static void main(String[] args) { System.out.println("Hello World"); } }</pre>	<pre>print('Hello World')</pre>

Output:

Hello World	Hello World
-------------	-------------

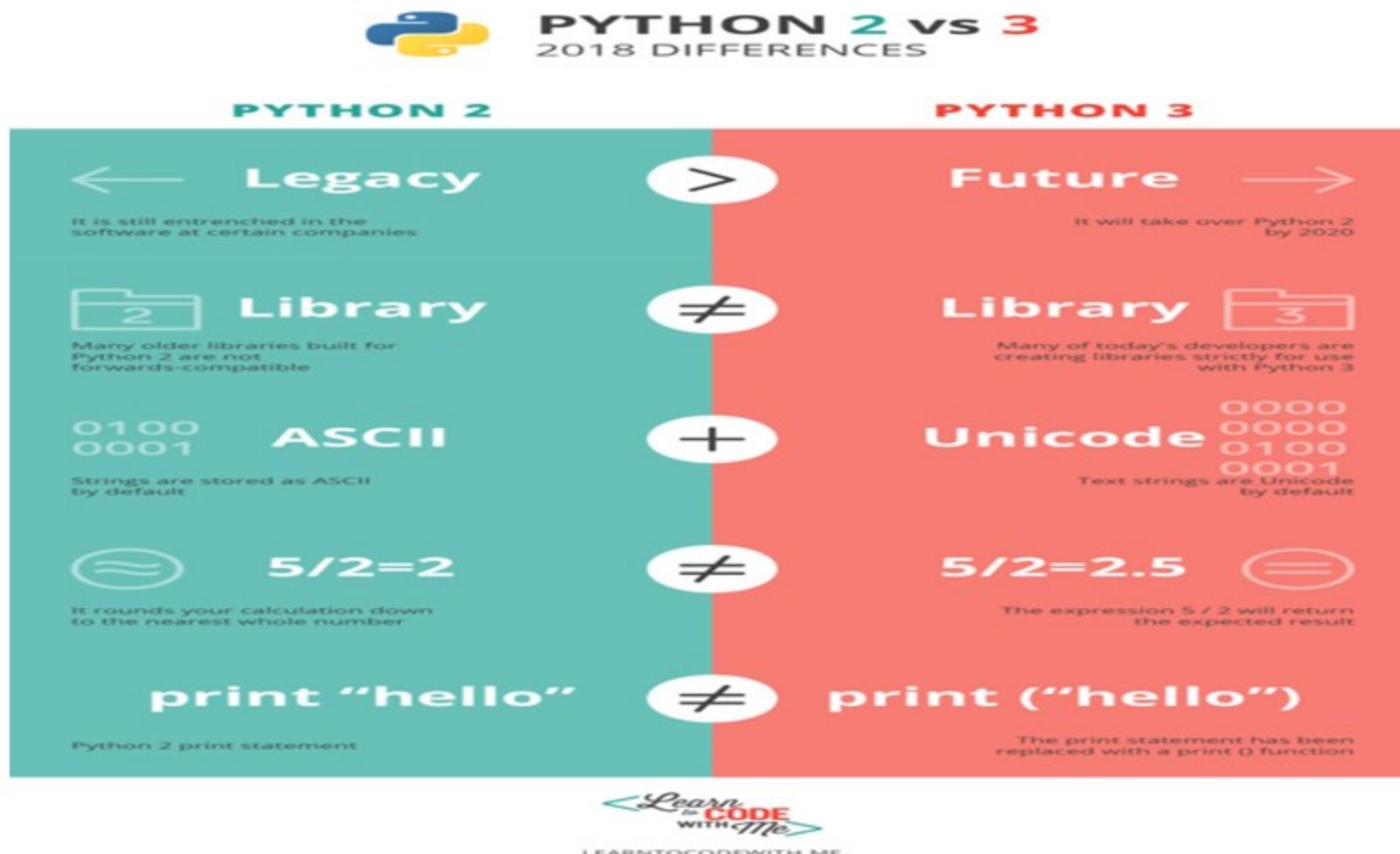
자바(JAVA)

: 데이터를 객체로 취급해서, 하나의 객체로 묶어서 프로그래밍하는 언어, C언어도 객체지향 언어.

파이썬(Python)

: 객체지향 언어이면서 인터프리터 언어(스크립터 언어), 객체 묶어서 프로그래밍하고 실행할 때마다 바로 해석

Python 2 VS python 3



다양한 라이브러리 지원



© 2016 Continuum Analytics - Confidential & Proprietary

개발 환경 개요

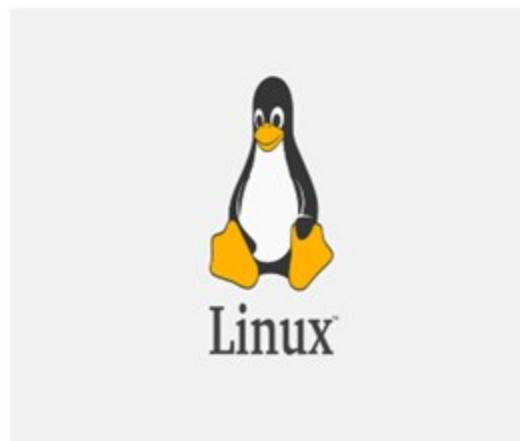
- 프로그램을 작성하고, 실행 시키는 환경
- 일반적으로 <코딩 환경> 이라고 부름
- 개발환경을 결정

1) 운영 체제(Operating System, OS)

2) python interpreter

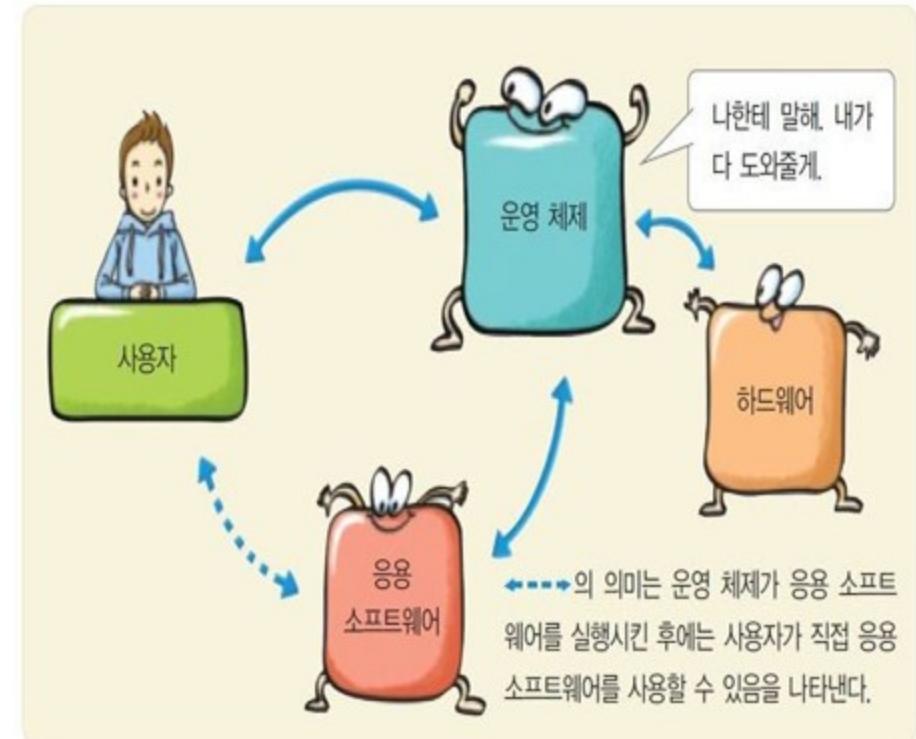
3) 코드 편집기(Editor)

운영 체제-Operation System



장점	친숙함 초기에 쉬움	모듈 설치 쉬움 무료, 참고문서 많음	모듈 설치 쉬움 참고문서 많음
단점	모듈 설치 어려움 / 유료	OS 자체 사용이 어려움	비쌈

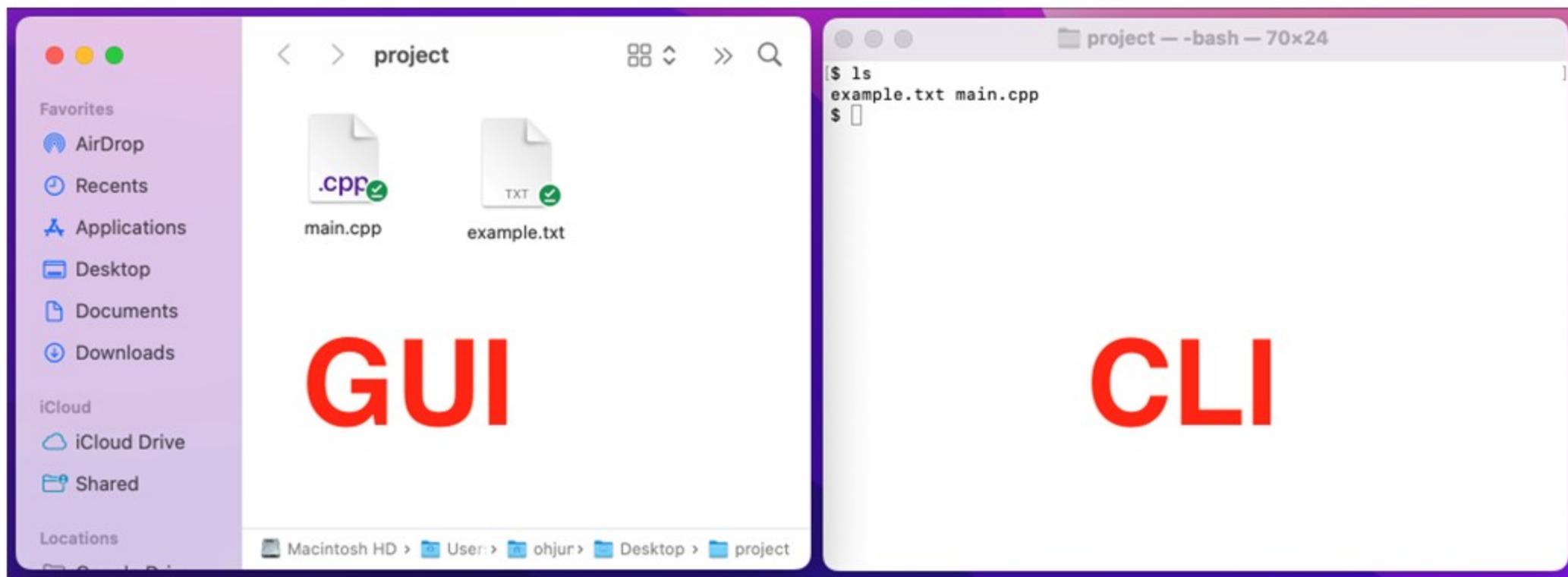
운영 체제-Operation System



- 하드웨어 : 컴퓨터 본체, 모니터, 마우스, 키보드, 하드디스크, CPU, RAM 등
- 소프트웨어 : 한글, 엑셀, 파워포인트, 크롬 등

터미널

- 터미널 - Mouse가 아닌 키보도로 명령을 입력 프로그램 실행



- Graphical User Interface

- Command Line Interface

Command Line Interface

- Graphic User Interface(GUI)와 달리 Text를 사용하여 컴퓨터에 명령을 입력하는 인터페이스 체계
- Windows – CMD Window, Windows Terminal
- Mac, Linux – Terminal

윈도우 cmder도 권장(<https://cmder.net/>)

The screenshot shows a Windows-style terminal window titled "cmd.exe" running under the "Cmder" interface. The window displays a command-line session with the following output:

```
E:\Program\_\Etc\cmder
λ dir
E 드라이브의 폴더 : HDD1
폴더 일련 번호 : F097-668F

E:\Program\_\Etc\cmder 디렉터리
2018-08-22 오후 10:18 <DIR> .
2018-08-22 오후 10:18 <DIR> ..
2018-08-22 오후 10:18 <DIR> bin
2018-05-30 오전 08:08 137,728 Cmder.exe
2018-08-22 오후 10:19 <DIR> config
2018-08-22 오후 10:18 <DIR> icons
2018-05-30 오전 08:07 1,069 LICENSE
2018-08-22 오후 10:18 <DIR> vendor
2018-05-30 오전 08:08 0 Version 1.3.6.678
3개 파일 138,797 바이트
6개 디렉터리 1,799,057,408 바이트 남음

E:\Program\_\Etc\cmder
λ ls
Cmder.exe* LICENSE 'Version 1.3.6.678' bin/ config/ icons/ vendor/

E:\Program\_\Etc\cmder
λ ls -l
total 152
-rwxr-xr-x 1 Alice 197121 137728 May 30 08:08 Cmder.exe*
-rw-r--r-- 1 Alice 197121 1069 May 30 08:07 LICENSE
-rw-r--r-- 1 Alice 197121 0 May 30 08:08 'Version 1.3.6.678'
drwxr-xr-x 1 Alice 197121 0 Aug 22 22:18 bin/
drwxr-xr-x 1 Alice 197121 0 Aug 22 22:19 config/
drwxr-xr-x 1 Alice 197121 0 Aug 22 22:18 icons/
drwxr-xr-x 1 Alice 197121 0 Aug 22 22:18 vendor/

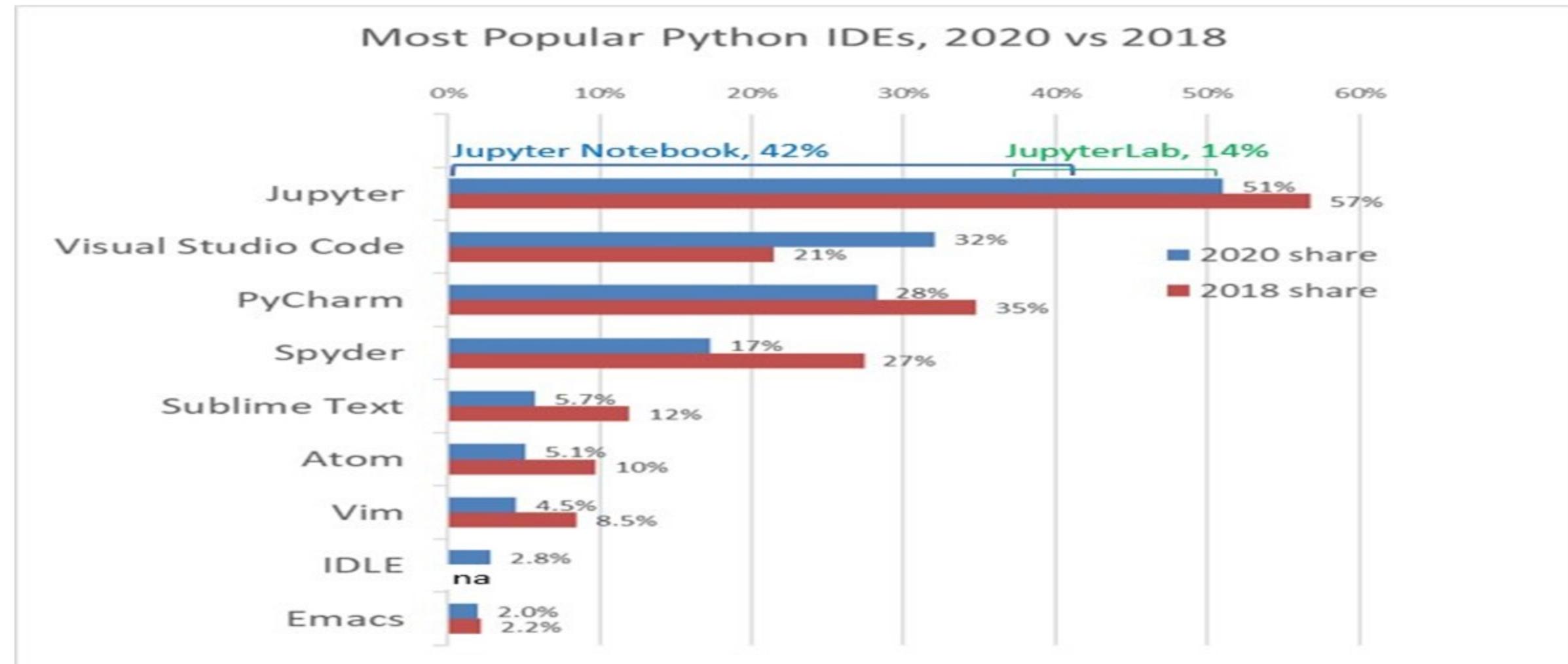
E:\Program\_\Etc\cmder
λ
```

코드 편집기(Editor)

- 파이썬 코드도 일종의 문서
- 한글, 워드처럼 코드를 입력할 문서 편집기가 필요함.
- **text** 타입의 문서를 저장하는 모든 편집기를 사용가능.

종류	설명	다운로드
메모장	윈도우의 기본 문서 편집도구	시작->메모장(Windows)
VI editor	리눅스의 기본 문서 편집도구	https://www.vim.org/download.php
Sublime Text Atom VS code	프로그래밍에 특화된 문서 편집도구	https://www.sublimetext.com/ https://atom.io/ https://code.visualstudio.com/
Pycharm	다양한 기능을 갖춘 파이썬 전용 개발 도구	https://www.jetbrains.com/pycharm/

파이썬 개발환경





- 손쉬운 패키지 설치
- 가상환경을 통한 패키지 버전 관리가 용이



- web을 통한 접근 가능
- Coding 결과를 실시간으로 확인 가능
- 자동완성 기능
- 다양한 언어 지원
- markdown을 통해 문서화 가능

ANACONDA 설치

<https://www.anaconda.com/products/individual>

Anaconda Installers



Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

Python 3.8

64-Bit Graphical Installer (462 MB)

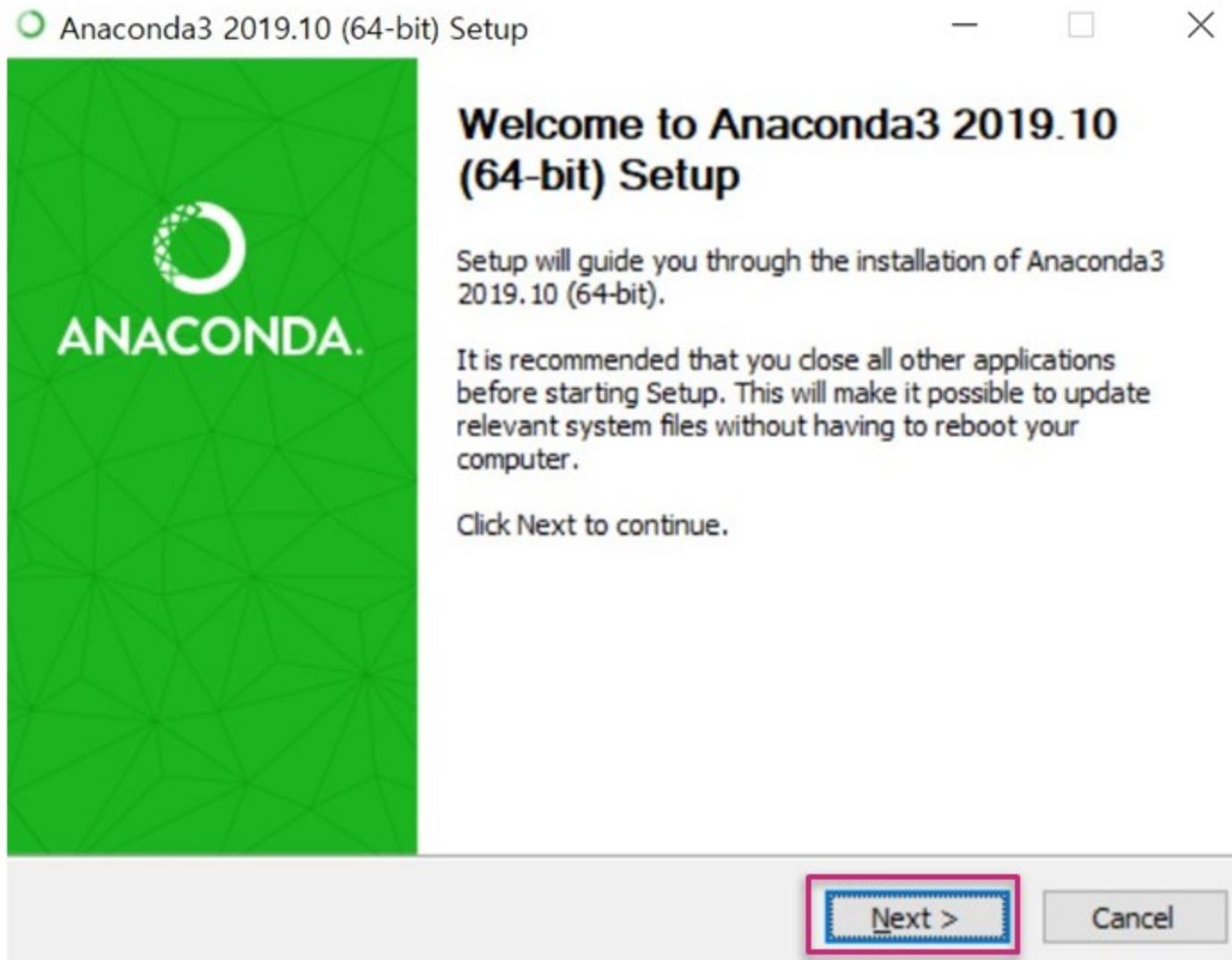
64-Bit Command Line Installer (454 MB)

Python 3.8

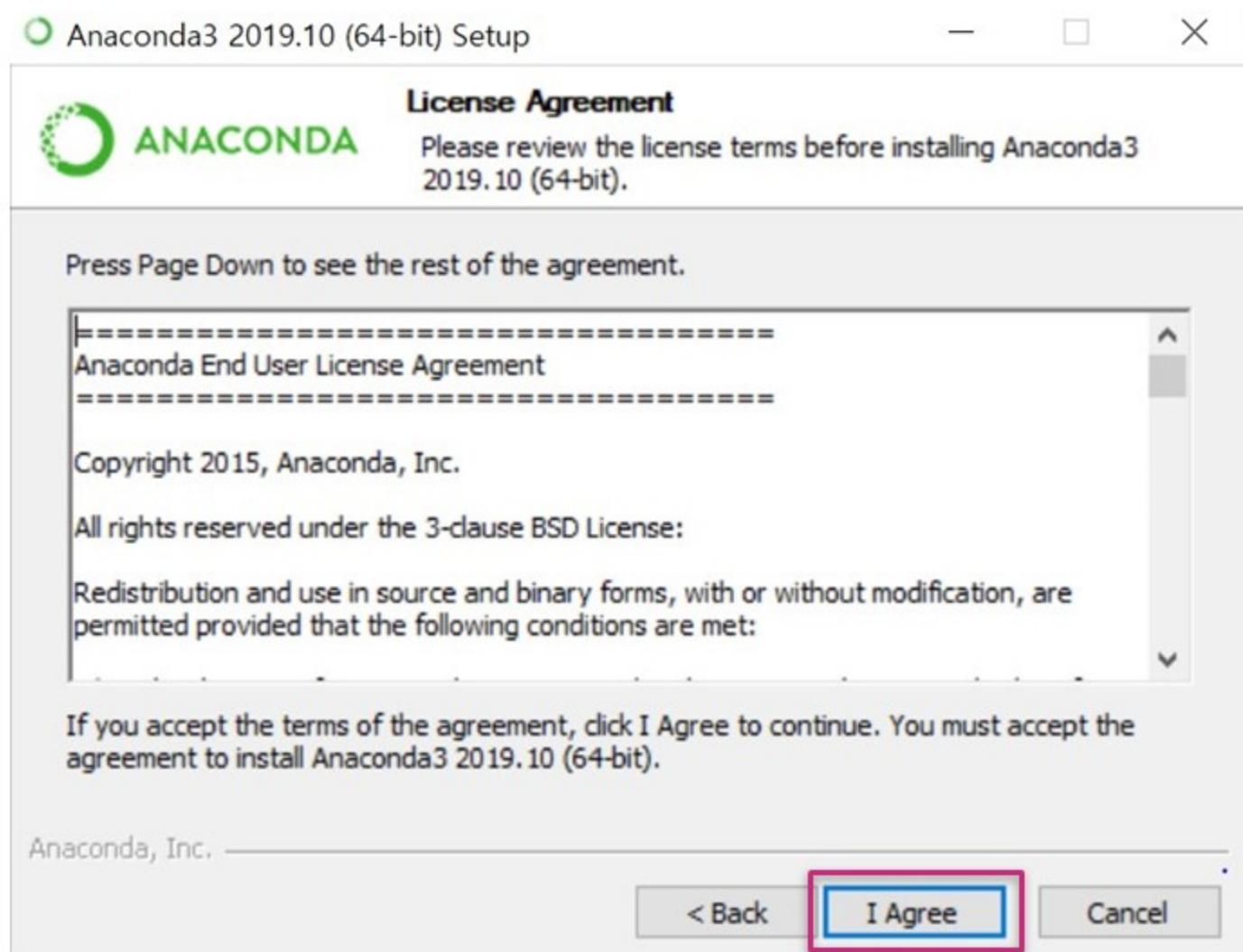
64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

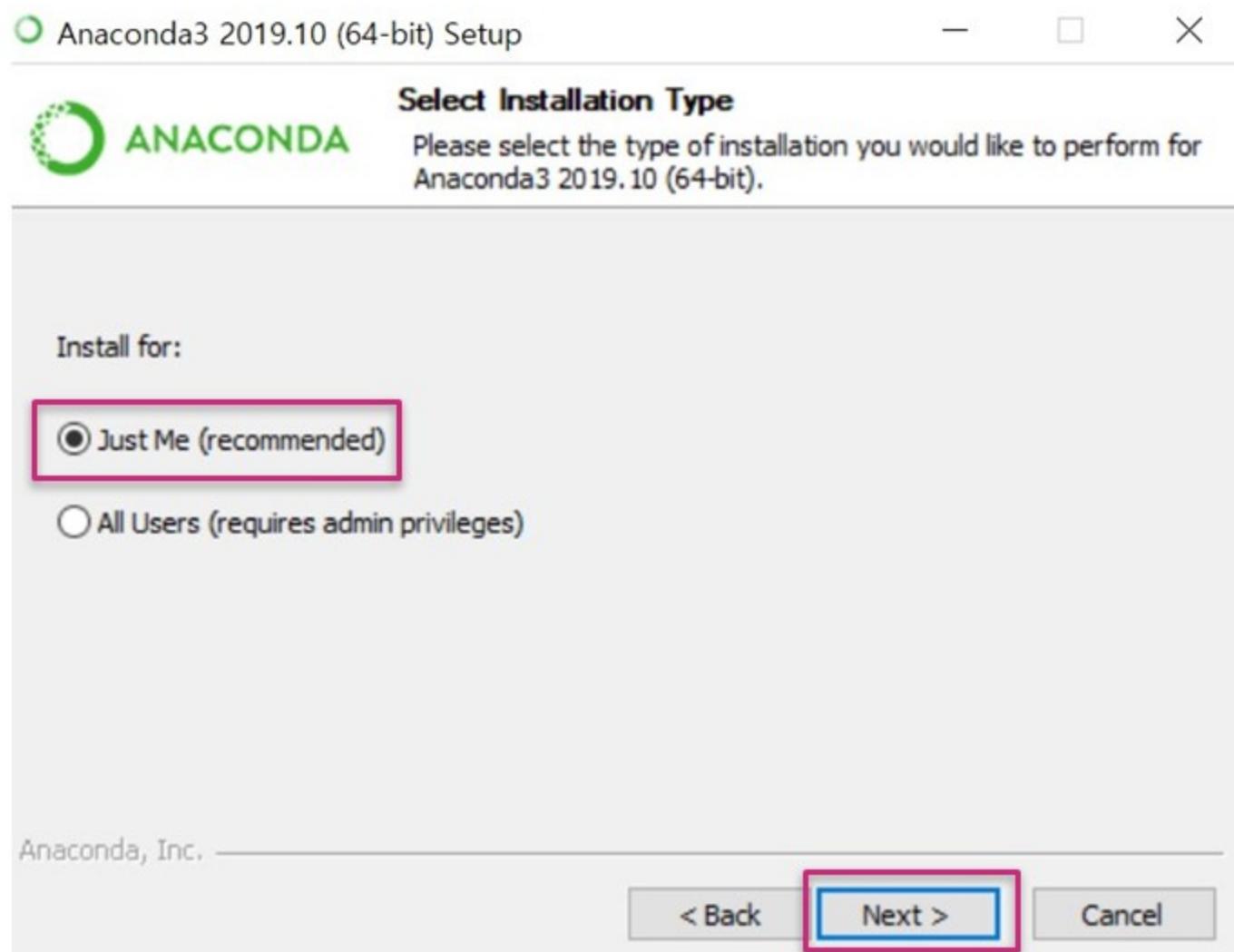
ANACONDA 설치



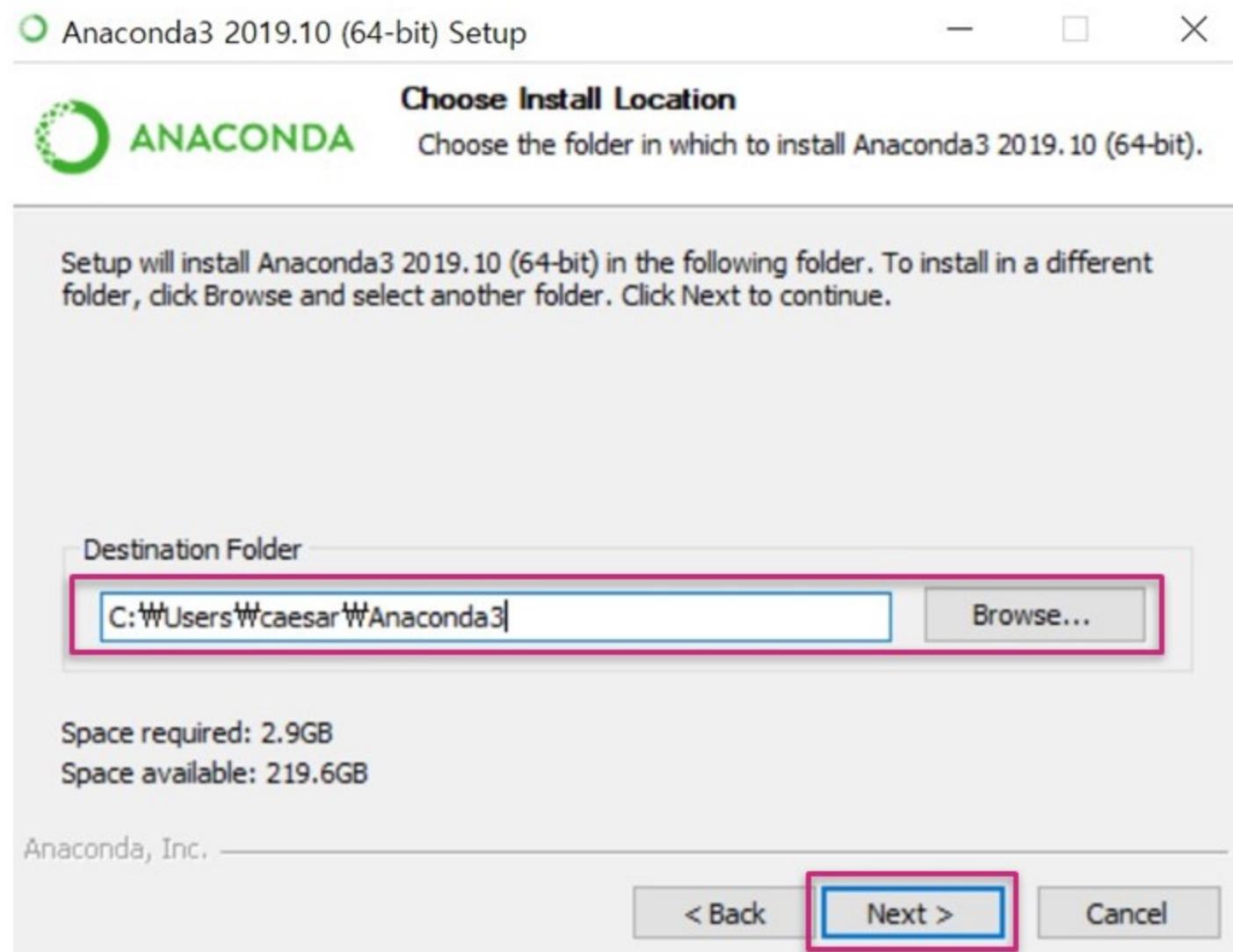
ANACONDA 설치



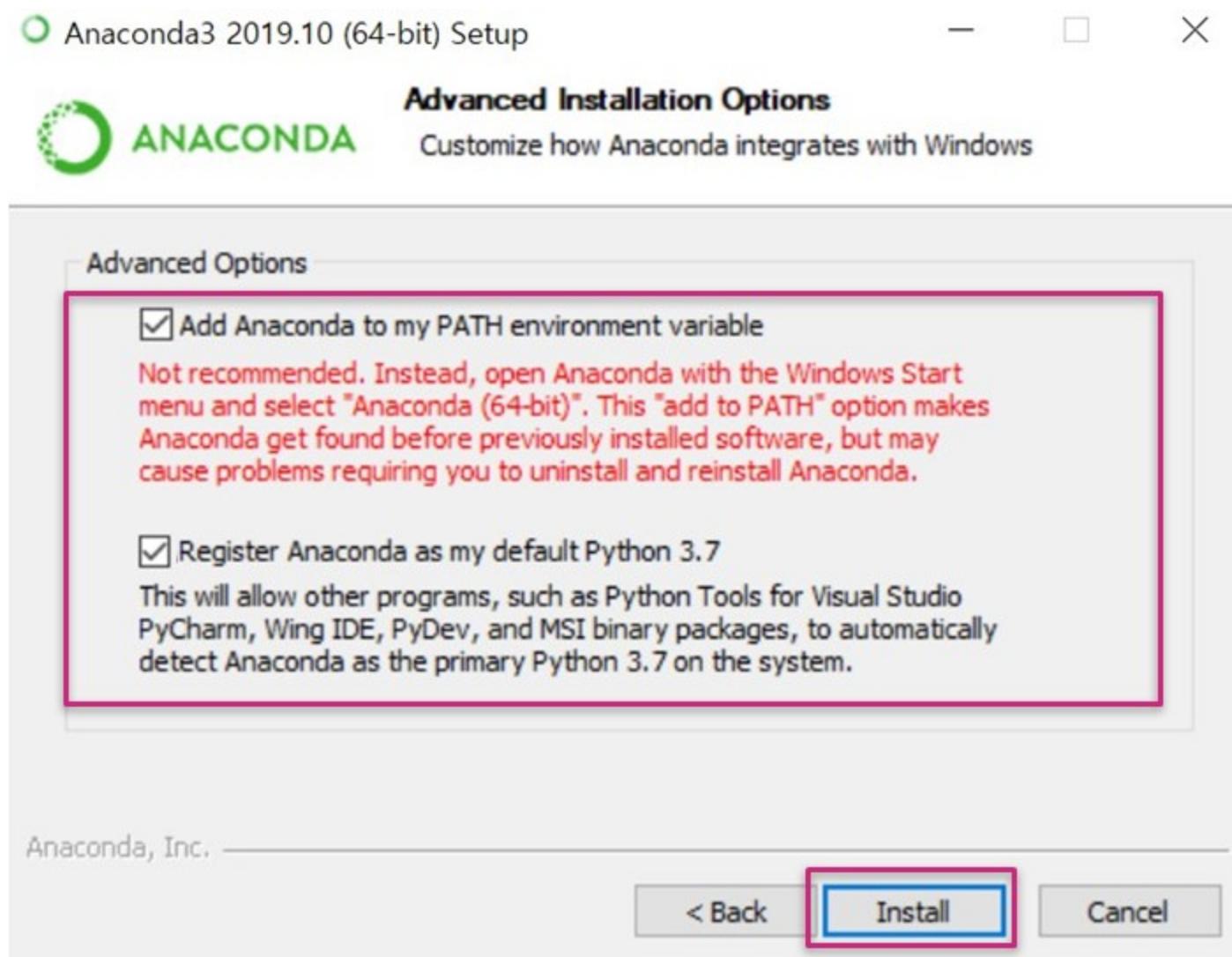
ANACONDA 설치



ANACONDA 설치



ANACONDA 설치



Jupyter Notebook 사용하기

- 아나콘다네비게이터실행



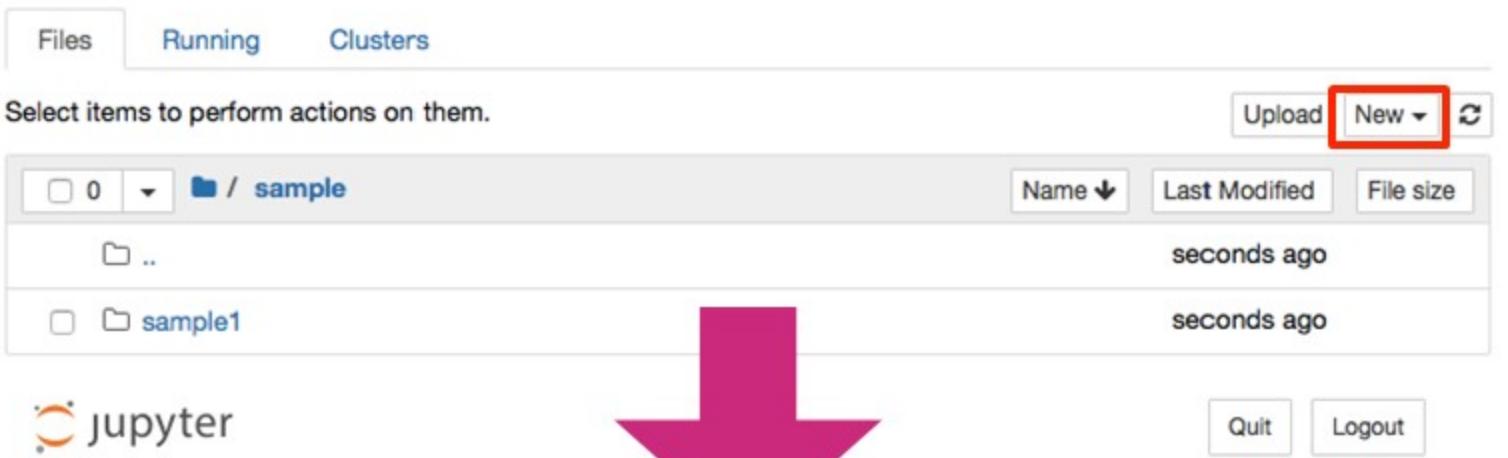
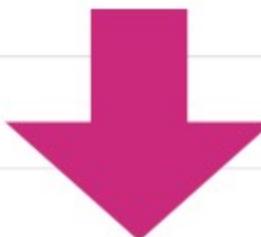
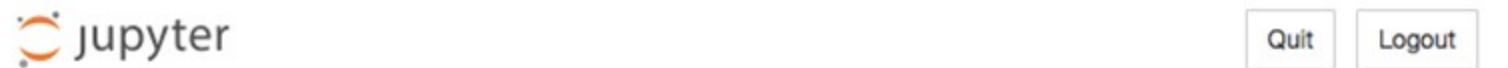
Jupyter Notebook 사용하기

- Anaconda Prompt 실행후 아래 명령어 실행
→ jupyter notebook

```
Last login: Wed Aug  5 15:41:48 on ttys000
→ ~ jupyter notebook
```

Jupyter Notebook 사용하기

- 새로운 노트북 생성하기



Jupyter Notebook 사용하기

- 유용한 단축키

```
In [ ]: # Edit mode : Enter  
# Command mode : ESC
```

```
In [ ]: # 셀 추가 (Command mode에서)  
# 현재 셀 위로 : a  
# 현재 셀 아래로 : b
```

```
In [ ]: # 셀 삭제 (Command mode에서) : dd
```

```
In [ ]: # 선택한 셀을  
# 복사하기 : c  
# 잘라내기 : x
```

```
In [ ]: # 셀 붙여넣기  
# 선택한 셀 위로 : Shift + v  
# 선택한 셀 아래로 : v
```

```
In [ ]: # 셀에 라인 추가하기 (Command mode에서) : l
```

Jupyter Notebook 사용하기

- 유용한 단축키
 - 함수나 변수에 대한 설명1: Shift + Tab

In []: str

In []: Init signature: str(self, /, *args, **kwargs)
Docstring:
In []: str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

A screenshot of a Jupyter Notebook interface. In the top cell, the word 'str' is typed. A tooltip-like pop-up window appears below it, containing the function signature 'Init signature: str(self, /, *args, **kwargs)', the docstring 'Docstring:', and two examples of how 'str' can be used: 'str(object="") -> str' and 'str(bytes_or_buffer[, encoding[, errors]]) -> str'. The background shows other cells and a toolbar.

- 함수나 변수에 대한 설명2: ? 셀을 실행해야함

In [1]: str?

Init signature: str(self, /, *args, **kwargs)
Docstring:
str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

A screenshot of a Jupyter Notebook interface. In the top cell, the question mark '?' is added after 'str' to form 'str?'. A tooltip-like pop-up window appears below it, containing the same information as the previous screenshot: the function signature, the docstring, and two usage examples. The background shows other cells and a toolbar.

Jupyter Notebook 사용하기

Jupyter 맛보기

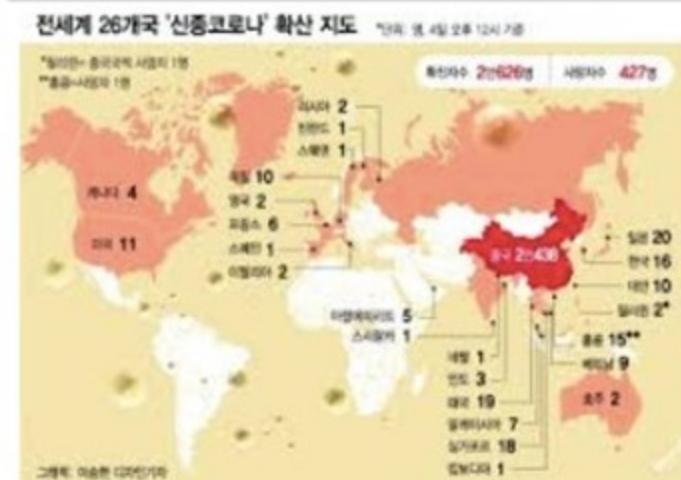
markdown

문서처럼 작성할 수도 있습니다.

```
In [2]: print("당연히 코딩도 할 수 있습니다.")
```

당연히 코딩도 할 수 있습니다.

```
In [7]: from IPython.display import Image  
Image(url="data:image/jpeg;base64
```



Google colab 사용법

- 풀 네임은 Google Colaboratory
- Google Drive + Jupyter Notebook
 - Google Drive처럼 협업 가능(동시에 수정 가능)
- <https://colab.research.google.com/>로 접속시 사용 가능
- 컴퓨터 사양(21년 01월 기준)
 - Ubuntu 18.04.5 LTS
 - CPU 제논 2.3GHz
 - 메모리 13G
 - GPU : K80 또는 T4 :
 - TPU도 사용 가능
- GPU 사용시 최대 12시간 (**Colab은 GPU를 무료로 사용가능**)
- Github의 소스 코드를 Colab에서 사용 가능

Google colab

OS 확인

- !cat /etc/issue.net

하드웨어 사양

1. CPU

- !cat /proc/cpuinfo

2. Memory

- !cat /proc/meminfo

3. Disk

- !df -h

4. GPU

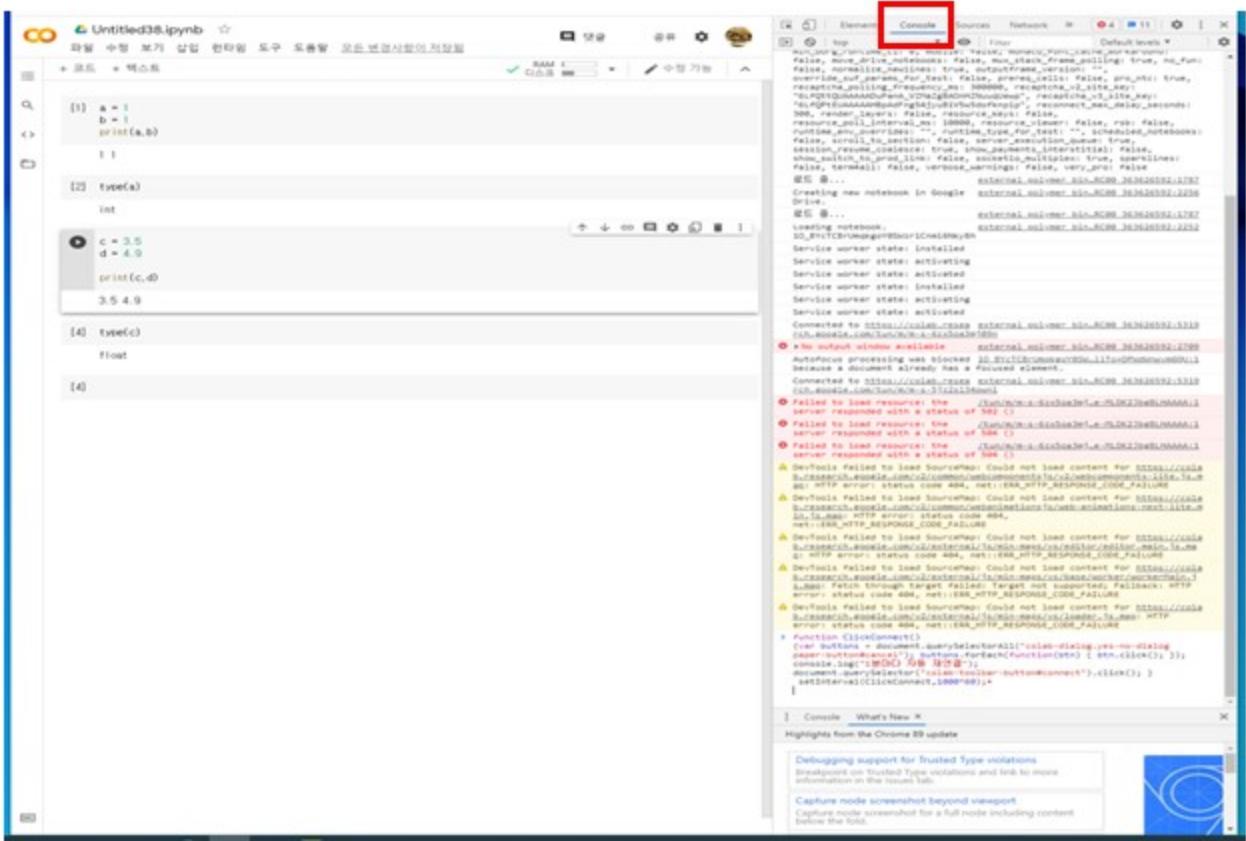
- !nvidia-smi

Google colab 런타임 방지

Google colab의 전체 세션 유지 시간은 12시간이고, 90분 이상 비 활성화 되어 있으면 자동으로 끊기고, 몇 분 동안 사용을 안하면 자동으로 세션이 끊긴다.

->방법 : 크롬 브라우저에서 F12(혹은 Ctrl+shift+i)를 눌러서 개발자 도구 창을 열고 console 창에서 입력해 주면 된다.

```
function ClickConnect()
{var buttons = document.querySelectorAll("colab-
dialog.yes-no-dialog paper-button#cancel");
buttons.forEach(function(btn){ btn.click(); });
console.log("1분마다 자동 재연결");
document.querySelector("colab-toolbar-
button#connect").click();
setInterval(ClickConnect,1000*60);}
```



Python IDE VS code 이란?

- VS Code의 장점

1. MS의 친절한 관리

- MS의 프로그램은 비싸지만 그만큼 친절함.

2. 무료 정책

3. 점유율 1위

4. 마이크로 소프트가 직접 배포하는 확장성

5. 언어 확장성이 좋음(C,C++,JAVA 등 지원함)

The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a tree view of a project structure. The root folder is 'POKEMONGO-BOT'. Inside it, there are several subfolders and files: 'api_wrapper.py', 'update_live_inventory.py', 'update_live_stats.py', 'update_web_inventory.py', 'use_licenses.py', 'utils.py', 'event_handlers.py', 'health_record.py', 'migrations.py', 'plugins.py', 'services.py', 'socketio_server.py', 'test.py', 'walkers.py', and '_init_.py'. The file 'api_wrapper.py' is currently open in the main editor area. The code in the editor is as follows:

```
from __future__ import absolute_import
import time
import logging
import random, base64, struct
import hashlib
import os
from pgoapi.exceptions import (ServerSideRequestThrottlingException,
                                NotLoggedInException, ServerBusyOrOfflineException,
                                NoPlayerPositionSetException, EmptySubrequestChainException,
                                UnexpectedResponseException)
from pgoapi.pgoapi import PGoApi
from pgoapi.pgoapi import PGoApiRequest
from pgoapi.protos.POGOProtos.Networking.Requests.RequestType_pb2 import RequestType
from pgoapi.utilities import get_time
from .human_behaviour import sleep, gpa_noise_rng
from pokemongo_bot.base_dir import _base_dir
class PermaBannedException(Exception):
    pass

class ApiWrapper(PGoApi, object):
    DEVICE_ID = None

    def __init__(self, configuration):
        self.config = config
        self.gen_device_id()
        device_info = {
            "device_id": ApiWrapper.DEVICE_ID,
            "device_brand": 'Apple',
            "device_model": 'iPhone',
            "device_model_boot": 'iPhone8,2',
            "hardware_manufacturer": 'Apple',
            "hardware_node": 'M66AP',
            "firmware_brand": 'iPhone OS',
            "firmware_type": '9.3.3'
        }
```

At the bottom of the status bar, it shows 'In 18, Col 1' and 'Python'.

Python IDE Pycharm 이란?

- Pycharm은 Python 언어에 사용되는 JetBrains사에 의해 개발된 IDE입니다.
- IDE는 컴파일, 코드 편집기, 디버그, 배포 등의 모든 작업을 하나의 프로그램에서 할 수 있도록 해주는 통합 개발 환경

Pycharm 의 특징

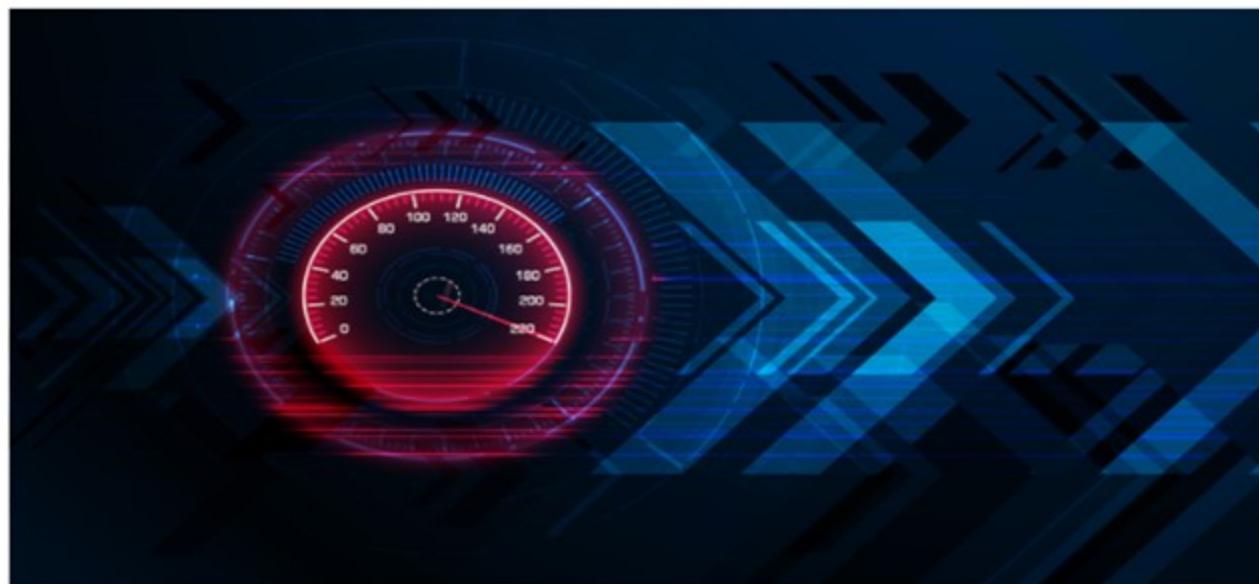
1. 코드 분석 및 코딩 지원
2. 프로젝트 및 코드 탐색
3. 파이썬
4. 웹 프레임 워크 지원(Professional 전용, 유료버전)
5. 통합 파이썬 디버그
6. 라인 단위 테스트
7. Google App Engine python(Professional 전용, 유료버전)
8. 과학적 도구 지원(Professional 전용, 유료버전)

Python 속도

“2~5배 더 빠르게”… 귀도 반 로섬, 파이썬 속도 개선한다

Paul Krill | InfoWorld

파이썬(Python) 창시자 귀도 반 로섬이 파이콘 2021(Pycon 2021)에서 진행된 **파이썬 랭귀지 서밋(Python Language Summit)**에서 파이썬 속도를 2배에서 5배까지 더 빠르게 만들기 위한 단기 및 장기 계획을 발표했다.



©Getty Images

파이썬에는 이미 파이파이(PyPy)와 같은 대체 런타임부터 C/C++로 작성된 랩핑 모듈까지 더 빠르게 실행할 수 있는 여러 방법이 있다. 하지만 C로 작성된 파이썬 참조 구현체이자 가장 널리 사용되는 언어 버전인 C파이썬(CPython) 자체의 속도를 높이는 방법은 거의 없다는 게 반로섬의 설명이다.

Python



프론트 엔드 VS 백엔드

	프론트엔드	백엔드
정의	사용자들에게 보여지는 부분 Ex) 레이아웃, 텍스트, 그림 등	사용자들에게 보이지 않는 시스템 안쪽 부분 Ex) 내부적 DB
기술	HTML&CSS *HTML : 콘텐츠에 구조와 의미를 주기 위한 것 *CSS : 콘텐츠의 모양과 스타일을 주기 위한 것 JAVA script *웹페이지에 기능을 더해 HTML 웹페이지를 동작하고 살아있게 만드는 것	스프링프레임워크 *동적인 웹사이트 개발을 위한 여러가지 서비스 제공 JAVA *컴퓨터가 이해할 수 있는 0과 1로 이루어진 기계어
직무	UI/UX 디자인 *UX디자인 : 사용자가 경험하게 될 전반적 디자인 *UI디자인 : 겉으로 시각화 되는 디자인 Front-end 개발 *웹상의 표현을 도와주는 여러 기술을 활용하여 UX와 Web 디자인 요소를 구현	서버코딩 *시스템이 운용되기 위해 필요한 시스템의 관련 기능 구현 등을 담당 DB 활용 *데이터베이스를 구축하고 운영

프론트엔드 VS 백엔드



2. 파이썬의 기본

- 파이썬의 기본
 - 연습문제 1
 - 파이썬의 기본
 - 컴퓨터의 구조 - 폰노이만 아키텍처
 - 중앙처리장치
 - 파이썬의 기본
 - 연습문제 2-1
 - 연습문제 2-2

파이썬의 기본

● 산술 연산자

1. 더하기 : +

$$[] \quad 1 + 2$$

 3

$$[] \quad 250.7 + 300.2$$

 550.9

▼ 2. 빼기 : -

$$[] \quad 5 - 3$$

 2

$$[] \quad 250.5 - 100.3$$

 150.2

▼ 3. 곱하기 : *

$$[] \quad 5 * 3$$

 15

$$[] \quad 20.5 * 2$$

 41.0

▼ 4. 나누기 : /

$$[] \quad 10 / 2$$

 5.0

$$[] \quad 7 / 3$$

 2.3333333333333335

파이썬의 기본

- 산술 연산자

- ▼ 4. 나누기 : /

[] 10 / 2

5.0

[] 7 / 3

2.3333333333333335

- ▼ 5. //

[] 10 // 2

5

[] 7 // 3

2

/의 연산 결과: 소수점 표시 → 실수형

//의 연산 결과: 소수점 없음(나누기의 몫) → 정수형

파이썬의 기본

- 산술 연산자

- ▼ 6. 나머지 : %

[] 10 % 2



0

[] 7 % 3



1

- ▼ 7. 그 밖에 연산자

제곱 : **

[] 2 ** 3



8

[] 5 ** 3



125

파이썬의 기본

- 연산자의 우선순위

사칙연산의 연산자 우선순위와 같음
괄호()가 있다면 가장 우선처리

$$[] \ 5 + 3 * 4$$

17

$$[] [8 - 3] * 2$$

10

1번

10의 제곱을 출력해보자

```
] : ► # 정답을 적어주세요
```

```
)ut[1]: 100
```

```
] : ► # 정답을 적어주세요
```

```
)ut[3]: 100
```

파이썬의 기본

- 가장기초적인 프로그래밍 문법 개념
- 데이터(값)을 저장하기 위한 메모리 공간의 프로그래밍상 이름
- 변수는 메모리 주소를 가지고 있고, 변수에 들어가는 값을 메모리 주소에 할당됨.
- 할당연산자(Assignment) : =

변수 = 연산/조건/함수

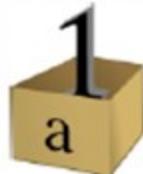
할당 연산자 우측을 연산을 한 후, 그 결과를 좌측에 있는 변수에 할당(저장)

파이썬의 기본

- 할당연산자(Assignment) : =

- Variables in C: Call-by-value

```
int a = 1;
```



Putting the value in a box with the variable name.

```
a = 2;
```



If you change the value of the variable, the box will be updated with the new value.

```
int b = a;
```



Assigning one variable to another makes a copy of the value and put that value in the new box.

- Names in Python: Call-by-object

```
a = 1
```



It tags the value with the variable name.

```
a = 2
```



It just changes the tag to the new value in memory
Garbage collection free the memory automatically

```
b = a
```



It makes a new tag bound to the same value.



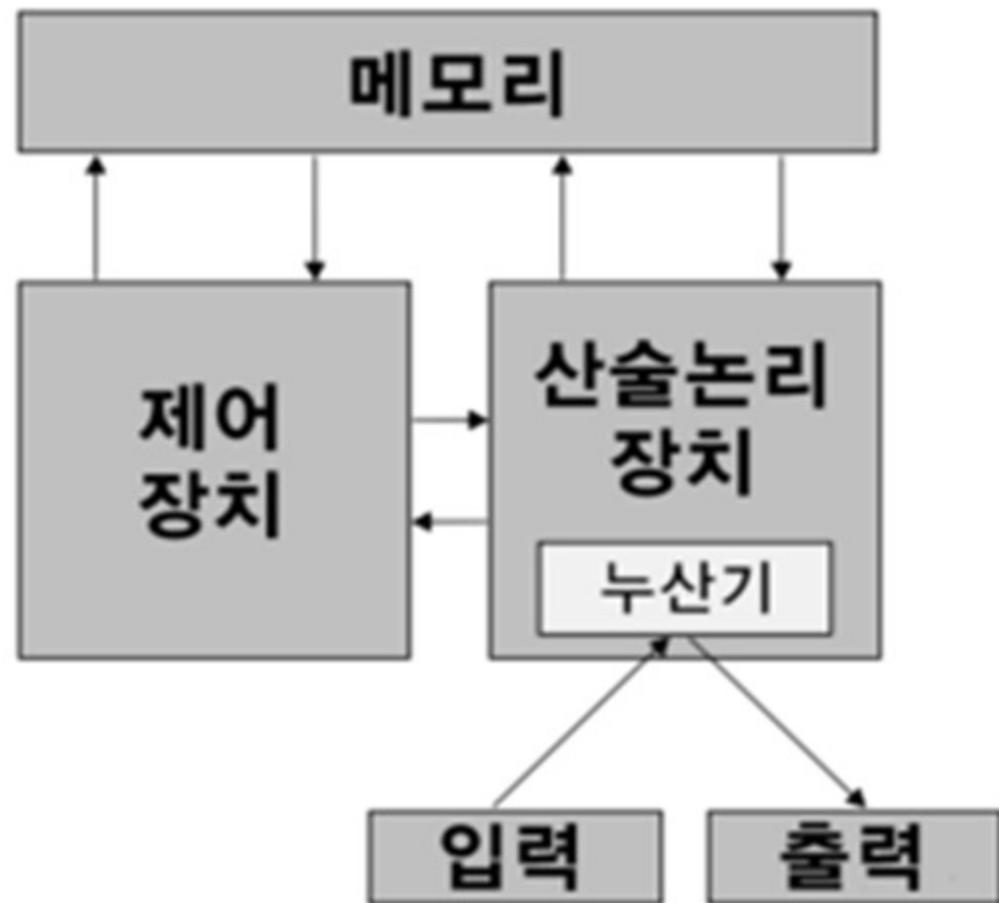
영문, 숫자 사용 가능

대소문자 구분

숫자부터 시작할 수 없음.

특수문자 사용 불가(+, -, *, /, \$ 등)

컴퓨터의 구조 - 폰노이만 아키텍처

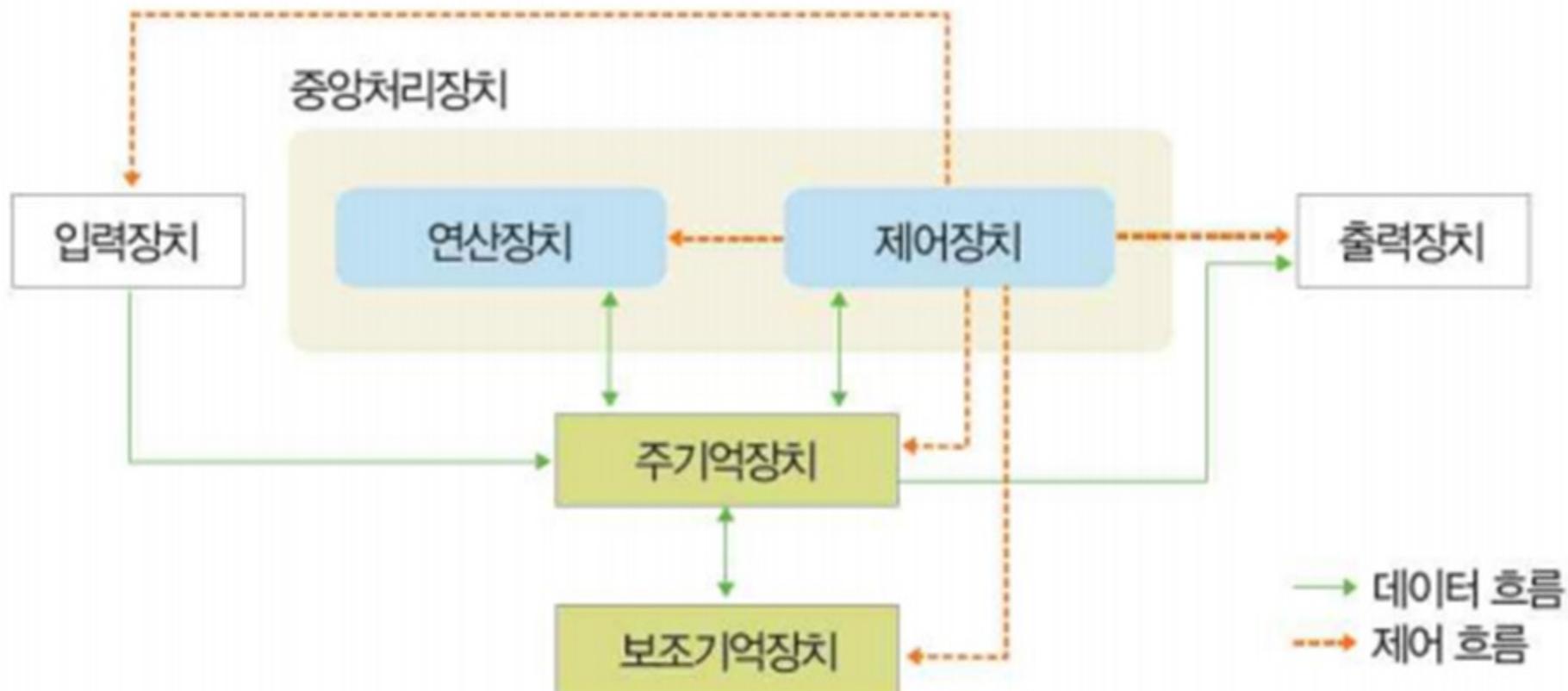


폰 노이만 아키텍처에서는 사용자가 컴퓨터에 값을 입력하거나 프로그램을 실행하는 경우, 그 정보를 먼저 메모리에 저장시키고 CPU가 순차적으로 그 정보를 해석하고 계산하여 사용자에게 결과값 전달

- 변수는 프로그램에서 사용하기 위한 특정한 값을 저장하는 공간.
- 선언 되는 순간 메모리 특정영역에 물리적인 공간이 할당됨.

중앙처리장치

그림 3-2 하드웨어의 구성



파이썬의 기본

- 할당연산자(Assignment) : **=**

```
In [ ]: ┏ a = 3
```

```
In [ ]: ┏ b = 5
```

```
In [ ]: ┏ print(a+b)
```

```
In [ ]: ┏ print("a+b")
```

```
In [ ]: ┏ a=8
```

```
In [ ]: ┏ print(a+b)
```

할당연산자 우측을 연산을 한 후, 그 결과를 좌측에 있는 변수에 할당(저장)

- 문제) Teacher = 'Youngpyo Ryu'의 의미는?
 - Teacher의 이름은 Youngpyo Ryu이다.
 - Teacher의 Youngpyo Ryu이다.
 - Teacher와 Youngpyo Ryu는 같다.
 - Teacher에 Youngpyo Ryu를 넣어라.
 - Teacher에 Youngpyo Ryu 를 넣어라
-> 정확히는 Teacher라는 **변수**에 'Youngpyo Ryu'라는 **값**을 넣으라는 의미.

파이썬의 기본

- 할당연산자(Assignment) : `=`

```
In [3]: a += 1  
a
```

```
Out[3]: 4
```

```
In [5]: a = a+1  
a
```

```
Out[5]: 4
```

산술 연산자와 할당 연산자를 함께 쓸 수 있음
코드를 더욱 간결하게 쓸 수 있음

파이썬의 기본

- 할당연산자(Assignment) : `=`

In [6]: `a +=1`
a

Out[6]: 5

In [10]: `a *= -1`
a

Out[10]: 0

In [12]: `a *= -1`
a
File "<ipython-input-12-
a * = -1
^
SyntaxError: invalid syntax

In [7]: `a -=5`
a

Out[7]: 0

In [11]: `a /= 5`
a

Out[11]: 0.0

In [13]: `a /= 5`
a
File "<ipython-input-13-
a / = 5
^
SyntaxError: invalid syntax

파이썬의 기본

- 알파벳, 숫자, 언더스코어(_)로 선언이 가능
ex) data = 0, a12_ = 2, tt_ = 'afdf'
- 변수명은 **의미 있는 단어로 표기**하는 것이 좋다.
Ex) teacher = 'Youngpyo Ryu'
- 변수명은 **대소문자가 구분** 된다.
Ex) ABC와 Abc는 같지 않다.
- 특별한 의미가 있는 **예약어는 쓰지 않는다.**
Ex) for, if, elif, else 등

파이썬의 기본

파이썬 예약어는 파이썬에서 이미 키워드로 지정되어 있어 변수나 함수의 이름으로 사용할 수가 없다.

```
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']
```

파이썬의 기본

- 할당연산자(Assignment) : **=**

단순 산술 연산이 아닌 변수간의 연산도 가능

```
[ ] a = 5  
    b = 7  
    c = a + b  
    print(c)
```

12

주의 파이썬은 대소문자를 구분함

```
[ ] c = A + B
```



```
NameError Traceback (most recent call last)
```

```
<ipython-input-26-b698ae2737d5> in <module>
```

```
----> 1 c = A + B
```

```
NameError: name 'A' is not defined
```

파이썬의 기본

- 할당연산자(Assignment) : `=`

파이썬은 들여쓰기(Indent)에 굉장히 민감(jupyter에서는 보정)

```
(base) C:\Users\USER>python
Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+1
File "<stdin>", line 1
    1+1
          ^
IndentationError: unexpected indent
xxx
```

연습문제 2-1

1번

사과가 5개 오렌지가 3개 있을 때 총 과일의 갯수를 구해보자

사과는 apple이라는 변수, 오렌지는 orange라는 변수에 할당 한 후, 총 과일의 갯수를 total이라는 변수에 저장해보자

- : ┌ # 정답을 적어주세요

- : ┌ # 아래 주석을 풀고 실행시켜서 원하는 값이 나왔는지 확인하세요
total

[10]: 8

연습문제 2-2

2번

국어는 100점, 영어는 88점, 수학은 94점 일 때, 평균을 구하려고 한다.

각각의 점수를 kor, eng, math라는 변수에 저장한 후 평균을 구해 avg라는 변수에 할당해보자.

```
[ ]: ┌ # 정답을 적어주세요
```

```
[ ]: ┌ # 아래 주석을 풀고 실행시켜서 원하는 값이 나왔는지 확인하세요  
# avg
```

```
Out[13]: 94.0
```

3. 파이썬의 자료형

- 숫자(int, Float)
- 숫자의 자료형
- 문자(String)
- 리스트(list)
- 튜플(tuple)
- 딕셔너리(dict)
- 셋(set)
- 불(bool)

1. 숫자의 자료형



데이터 형식	설명	크기(Byte)	값의 범위
bool	논리 형식	1(8bit)	true, false
sbyte	signed byte 정수	1(8bit)	-128 ~ 127
byte	부호 없는 정수	1(8bit)	0 ~ 255
short	정수	2(16bit)	-32,768 ~ 32,767
ushort	unsigned short 부호 없는 정수	2(16bit)	0 ~ 65,535
char	유니코드 문자	2(16bit)	U+0000 ~ U+ffff
int	정수	4(32bit)	-2,147,483,648 ~ 2,147,483,647
uint	unsigned int 부호 없는 정수	4(32bit)	0 ~ 4,294,967,295
float	단일 정밀도 부동 소수점 형식	4(32bit)	-3.402823e38 ~ 3.402823e38
long	정수	8(64bit)	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
ulong	unsigned long 부호 없는 정수	8(64bit)	0 ~ 18,446,744,073,709,551,615
double	복수 정밀도 부동 소수점 형식	8(64bit)	-1.79769313486232e308 ~ 1.79769313486232e308
decimal	29자리 데이터를 표현할 수 있는 소수 형식	16(128bit)	$\pm 1.0 \times 10^{-28} \sim \pm 7.9 \times 10^{28}$

허수(i)

1. 숫자(int, float)

- int와 float

```
[1] a = 1  
    b = 1  
    print(a,b)
```

```
1 1
```

```
[3] type(a)
```

```
int
```

```
[3] c = 3.5  
    d = 4.9  
    print(c,d)
```

```
3.5 4.9
```

```
[4] type(c)
```

```
float
```

int는 정수, float는 실수

type(변수) : 변수의 자료형을 알려주는 함수

Print(변수) : 모니터화면에 결과물을 출력해주는 함수

1. 숫자(int, float)

- int와 float

```
[4] d = int(c)
print(d)
type(d)
```

3
int

```
[8] a = 10
type(a)
```

int

```
[9] float(a)
```

10.0

Quiz) 현재 a의 자료형은?

Ans) 현재 a는 int이다. 그 이유는 형 변환 해주지 않음

숫자는 형 변환을 할 수 있음

```
[11] a = float(a)
```

type(a)

float

- 5.3과 5.7 정수형으로
형 변환 후 덧셈하면 결과값은?

```
a = 5.3
b = 5.7

a = int(a)
b = int(b)

print(a+b)
```

2. 문자(String)

- string

‘ ’(작은 따옴표) 혹은 “ ”(큰 따옴표)을 이용하여 문자형(string)임을 표시

```
[11] f = '문자형 예시입니다.'  
      print(f)  
      type(f)
```

☞ 문자형 예시입니다.
str

2. 문자(String)

```
[12] 따옴표 = '문장안에 따옴표를 쓰고 싶다면? """ 번갈아 쓰면 됩니다.'  
print(따옴표)
```

☞ 문장안에 따옴표를 쓰고 싶다면? "" 번갈아 쓰면 됩니다.

```
[13] escape = "escape 문자인 \" \\ \"(역슬래시 혹은 원화표시)도 사용가능"  
print(escape)
```

☞ escape 문자인 " \ "(역슬래시 혹은 원화표시)도 사용가능

2. 문자(String)

```
print(' 실패란 넘어지는 것이 아니라,  
 넘어진 자리에 머무는 것이다.  
 - 도서 '프린세스, 라 브라바!' 中 ' )
```

실패란 넘어지는 것이 아니라,
 넘어진 자리에 머무는 것이다.
 - 도서 '프린세스, 라 브라바!' 中

```
print('₩t탭₩n다음줄')
```

탭

다음줄

2. 문자(String)

```
[21] g = 3  
     h = '5'
```

Quiz) g,h의 자료형은?

```
[22] type(g)
```

⇨ int

```
[23] type(h)
```

⇨ str

2. 문자(String)

```
[21] g = 3  
     h = '5'
```

```
[24] i = str(g)  
      print(i)  
      type(i)
```

↳ 3
str

```
[ ] j = int(h)  
print(j)  
type(j)
```

↳ 5
int

str와 int도 형 변환이 가능

2. 문자(String)

- string의 연산

```
[26] str_1 = '문자'  
     str_2 = '의 연산'  
     result = str_1 + str_2  
     result
```

⇒ '문자의 연산'

```
[27] str_3 = '더하기'  
     result += str_3  
     result
```

⇒ '문자의 연산더하기'

더하기를 하면 두 문자열을 붙여줌

2. 문자(String)

- string의 연산

```
[28] str_4 = '곱하기'  
      print(str_4 * 5)
```

⇨ 곱하기곱하기곱하기곱하기곱하기

```
[29] str_5 = '반복'  
      print((str_4 + str_5)*2)
```

⇨ 곱하기반복곱하기반복

곱하기는 그 수만큼 문자열을 반복함

2. 문자(String)

- string의 연산

단, 자료형이 다르다면 연산이 불가

```
int_1 = 8  
str_6 = '문자'  
  
error = str_6 + int_1
```

```
-----  
-----  
TypeError                                     Traceback (most  
<ipython-input-1-9ac98546a9bd> in <module>  
      2 str_6 = '문자'  
      3  
----> 4 error = str_6 + int_1  
  
TypeError: can only concatenate str (not "int") to str
```

2. 문자(String)

- string의 indexing(인덱싱)

인덱싱? 순서가 있는 변수의 특정 위치에 접근 할 수 있게 해줌

변수[위치]

(*주의*)파이썬의 인덱싱은 0부터 시작

2. 문자(String)

- string의 indexing(인덱싱)

인덱싱? 순서가 있는 변수의 특정 위치에 접근 할 수 있게 해줌

변수[위치]

```
[47] str_5 = '인덱싱을 하기 위한 string입니다.'  
      len(str_5)      len(변수)는 변수의 길이를 재는 함수
```

↳ 21

```
[48] str_5[2]
```

↳ '싱'

2. 문자(String)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작: 끝: 간격]

주의

- 슬라이싱은 시작부터 끝 -1 까지 가져옴
- 간격이 1 이라면 생략해도됨

2. 문자(String)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작: 끝: 간격]

```
[49] str_5 = '인덱싱을 하기 위한 string입니다.'  
      str_5[2:7]
```

⇒ '싱을 하기'

2. 문자(String)

- string의 slicing(슬라이싱)

슬라이싱? 변수의 일정 부분에 접근하여 잘라옴

변수[시작: 끝: 간격]

```
[50] str_6 = '안녕하세요. 홍길동 고객님!'
```

```
[52] str_6[7:10]
```

↳ '홍길동'

2. 문자(String)

- string의 slicing(슬라이싱)

```
[54] str_6 = '안녕하세요. 홍길동 고객님!' → 시작점이 없다면 처음부터  
      print(str_6[:2])
```

↳ 안녕

```
[57] print(str_6[11:]) → 끝이 없다면 끝까지
```

↳ 고객님!

```
[58] print(str_6[:]) → 시작과 끝이 없다면 전체
```

↳ 안녕하세요. 홍길동 고객님!

2. 문자(String)

- string의 slicing(슬라이싱)

파이썬은 마이너스 인덱싱을 지원함

```
str_6 = '안녕하세요. 흥길동 고객님!'
```

```
print(str_6[-1])
```

```
!
```

```
print(str_6[-4:])
```

```
고객님!
```

```
print(str_6[: -2])
```

```
안녕하세요. 흥길동 고객
```

→ -1은 끝에서부터 거꾸로

2. 문자(String)

- string의 slicing(슬라이싱)
파이썬은 마이너스 인덱싱을 지원함

```
[65] str_6 = '안녕하세요. 흥길동 고객님!'
      print(str_6[::-1]) →[::-1]은 str 전체를 뒤집음
```

⇨ !님객고 동길흥 .요세하녕안

2. 문자(String)

- string의 method

method? 파이썬의 객체가 가지고 있는 고유의 함수
.을 이용하여 사용

1 대소문자변환: lower, upper

```
[66] str_7 = 'Alphabet'  
     print(str_7.lower())
```

↳ alphabet

```
[67] print(str_7.upper())
```

↳ ALPHABET

2. 문자(String)

- string의 method
- 2. 해당문자의갯수세기: count

```
[68] str_7 = 'Alphabet'  
     str_7.count('a')
```

☞ 1

→ count는 대소문자를 구분함

Quiz) str_7에서 모든 것을 소문자바꾼다음에 a문자의갯수를 세려면?

```
[71] str_7.lower().count('a')
```

☞ 2

2. 문자(String)

- string의 method

3. 문자열의 위치 찾기

- 1) find → 해당하는 문자열의 위치(인덱스)를 반환

```
[72] str_7 = 'Alphabet'  
      str_7.find('t')
```

↳ 7

- 2) index

```
[73] str_7.index('t')
```

↳ 7

2. 문자(String)

- string의 method

3. 문자열의 위치 찾기

- 1) find

```
[74] str_7 = 'Alphabet'  
      str_7.find('z')
```

→문자열에 없으면 -1 반환

↳ -1

- 2) index

```
[75] str_7.index('z')
```

→문자열에 없으면 error

↳ -----

ValueError: substring not found

2. 문자(String)

- string의 method

4. 문자열 바꾸기: replace

```
[82] str_8 = 'Life is C between B and D'  
      str_8.replace('C', 'Chicken')
```

↳ 'Life is Chicken between B and D'

5. 문자열 나누기: split

```
[83] str_8.split(' ')
```

↳ ['Life', 'is', 'C', 'between', 'B', 'and', 'D']

→ 따옴표 ' 사이에

기준이 되는 문자를 넣기
생략 시 기준은 띄어쓰기

2. 문자(String)

- string의 method

6. 문자열삽입: join

```
[84] str_9 = 'abcd'  
     ', '.join(str_9)
```

→ 따옴표 사이에 삽입할 문자를 넣기

⇒ 'a,b,c,d'

7. join과 split

```
[85] str_10 = str_8.split()
```

→ split한 결과에 join을 사용하면
다시 문자열로 만들수있음

```
[86] ' '.join(str_10)
```

⇒ 'Life is C between B and D'

2. 문자(String)

- string의 method

더 많은 메소드는 [python 공식문서](#)나 아래와 같은 방법으로 확인 가능

```
[21] print(dir(str_8))
```

```
'__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__geattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

3. 리스트(list)

- 시퀀스 자료형, 여러 데이터들의 집합
- int, float 같은 다양한 데이터 타입 포함

[]을 이용하여 리스트(list)임을 선언

```
[1] list_1 = [3, 2, 5]
    list_2 = list()
```

```
[2] type(list_1)
```

↳ list

3. 리스트(list)

- list의 특징

리스트는 모든 자료형을 담을 수 있음

리스트는 삽입, 수정, 삭제가 자유로움

리스트는 순서가 있는 자료형 → 인덱싱과 슬라이싱이 가능

```
[4] list_3 = [1, 2, '문자', ['이중 리스트', '가능'], ('리스트 속', '튜플')]
```

```
[5] list_3[3]
```

⇒ ['이중 리스트', '가능']

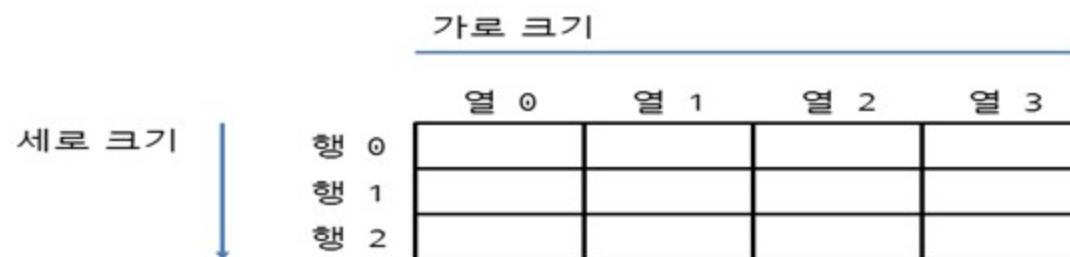
```
[6] list_3[4][1]
```

⇒ '튜플'

주의 python의 인덱스는 0부터 시작

3. 리스트(list)

- 평면 구조의 2차원 리스트를 배워보자.
- 2차원 리스트는 가로X세로 형태로 이루어져 있으며,
행(row)과 열(column) 모두 0부터 시작



3. 리스트(list)

- 2차원 리스트는 리스트 안에 리스트를 넣어서 만듦.
- ,(콤마)로 구분
- 리스트 = [[값, 값],[값, 값],[값, 값]]
 - 요소 접근하기
- 리스트[세로인덱스][가로인덱스]

```
a = [[10,20],[30,40],[50,60]]
```

```
a
```

```
[[10, 20], [30, 40], [50, 60]]
```

```
a[0][0]
```

```
10
```

```
a[1][0]
```

```
30
```

3. 리스트(list)

- list의 인덱싱

```
[7] list_4 = [[1,2,3,4],  
              [2,4,6,8],  
              [3,6,9,12],  
              [4,8,12,16],  
              [5,10,15,20]]
```

가장바깥쪽부터 괄호부터

```
[9] list_4[1][3]
```

앞에서부터 순서대로

→ 8

3. 리스트(list)

- list의 인덱싱

```
[ ] list_5 = [[[1,2,3],[4],[5,[6,7,8,9]]],  
             [10,11,[12,13,[14]]]]
```

```
[ ] list_5[1][2][2]
```

```
⇒ [14]
```

3. 리스트(list)

- list의 슬라이싱

```
[7] list_4 = [[1,2,3,4],  
              [2,4,6,8],  
              [3,6,9,12],  
              [4,8,12,16],  
              [5,10,15,20]]
```

```
[ ] list_4[1][:3]
```

```
↳ [2, 4, 6]
```

```
[ ] list_4[:2][1]
```

```
↳ [2, 4, 6, 8]
```

3. 리스트(list)

- list의 슬라이싱

```
[9] list_5 = [[[1,2,3],[4],[5,[6,7,8,9]]],  
             [10,11,[12,13,[14]]]]
```

```
[13] list_5[:2][0]
```

```
⇒ [[1, 2, 3], [4], [5, [6, 7, 8, 9]]]
```

```
[14] list_5[0][:2]
```

```
⇒ [[1, 2, 3], [4]]
```

3. 리스트(list)

- list의 연산

```
[15] list_6 = ['a', 'b', 'c']
```

```
[16] list_7 = ['가', '나', '다']
```

```
[17] list_6 + list_7
```

⇒ ['a', 'b', 'c', '가', '나', '다']

```
[18] list_6 * 3
```

⇒ ['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']

```
[15] list_6 = ['a', 'b', 'c']
```

```
[16] a in list_6
```

False

3. 리스트(list)

- list의 메소드

리스트는 **삽입, 수정, 삭제**가 자유로움

- 1 삽입: append, insert

```
[19] list_8 = ['파이썬', 'C', 'java']
```

→ insert는 원하는 위치에

```
[20] list_8.append('R')  
list_8
```

```
↪ ['파이썬', 'C', 'java', 'R']
```

→ append는 리스트 제일 뒤에

```
[21] list_8.insert(2, 'C++')  
list_8
```

```
↪ ['파이썬', 'C', 'C++', 'java', 'R']
```

```
[22] list_8.insert(5, 'C')  
list_8
```

```
↪ ['파이썬', 'C', 'C++', 'java', 'R', 'C'] 105
```

3. 리스트(list)

- list의 메소드

2. 수정: 덮어쓰기 → 인덱스나 슬라이싱을 통해 수정

```
[23] list_8[0] = 'Python'  
list_8
```

```
↪ ['Python', 'C', 'C++', 'java', 'R', 'C']
```

2-1) 확장

```
[23] list_8.extend(['MATLAB', 'Julia'])  
list_8
```

```
['Python', 'C', 'C++', 'java', 'R', 'C', 'MATLAB', 'Julia']
```

3. 리스트(list)

- list의 메소드

- 3. 삭제: remove

```
[24] list_8
```

```
↳ ['Python', 'C', 'C++', 'java', 'R', 'C']
```

```
[25] list_8.remove('C')  
list_8
```

```
↳ ['Python', 'C++', 'java', 'R', 'C']
```

→ **remove**는 지우고자하는 요소를 명시해야 함

→ 가장 앞에 있는 요소가 삭제됨

```
[34] del list_8[6]  
list_8
```

```
['Python', 'C', 'C++', 'java', 'R', 'C']
```

3. 리스트(list)

- list의 메소드

4. 꺼내오기: pop

```
[26] list_8.pop()
```

```
⇒ 'C'
```

```
[28] list_8
```

```
⇒ ['Python', 'C++', 'java', 'R']
```

- pop은 괄호()안 인덱스에 해당하는 요소를 반환
- 인덱스를 생략하면 가장 마지막 요소를 꺼내옴

3. 리스트(list)

- list의 메소드

- 5. 정렬: sort

```
[29] list_9 = [1,5,7,2,6]
```

```
[30] list_9.sort()  
list_9
```

→ 리스트 자체를 정렬

↳ [1, 2, 5, 6, 7]

list.sort(key=None, reverse=False)

- key : 정렬해주고 싶은 기준, 기본적으로 < 비교에 의해 정렬(오름차순)
- reverse : True로 설정하게 되면 > 비교에 의해 정렬(내림차순)

3. 리스트(list)

- list의 메소드

- 5. 정렬: sort

```
[32] list_10 = [1,47,12,6]
     list_10.sort(reverse=True)
     list_10
```

```
↪ [47, 12, 6, 1]
```

3. 리스트(list)

- list의 메소드

6. 뒤집기: reverse

```
[37] list_9
```

```
↳ [1, 2, 5, 6, 7]
```

```
[38] list_9.reverse()  
list_9
```

```
↳ [7, 6, 5, 2, 1]
```

→ 현재리스트 요소 순서를 뒤집음

→ 리스트 자체를 뒤집음

3. 리스트(list)

- list의 메소드

6. 뒤집기: reverse

```
[32] list_10 = [1, 47, 12, 6]
      list_10.sort(reverse=True)
      list_10
```

```
↪ [47, 12, 6, 1]
```

```
[41] list_10.reverse()
      list_10
```

```
↪ [1, 6, 12, 47]
```

3. 리스트(list)_참고

```
a = [10, 9, 8, 7, 6]
b = [5, 7, 8, 9, 10]
```

```
print(id(a), id(b))
```

```
140486843016208 140486843685856
```

```
b=a
```

```
print(id(a), id(b))
```

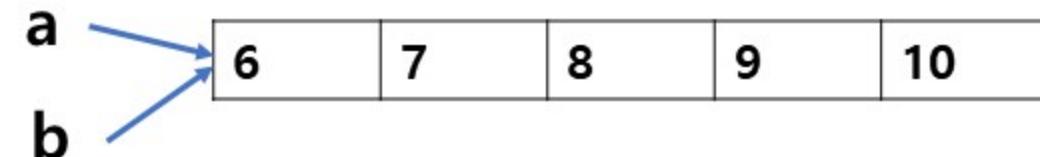
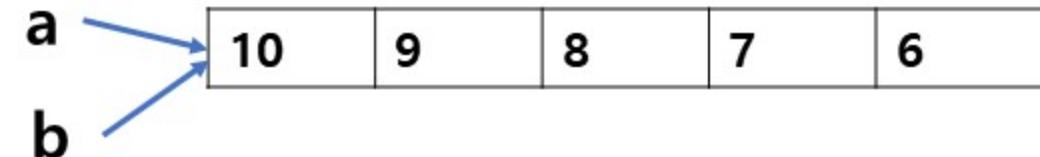
```
140486843016208 140486843016208
```

```
a.sort()
```

```
print(b)
```

```
[6, 7, 8, 9, 10]
```

a - >	10	9	8	7	6
b - >	6	7	8	9	10



a - >	6	7	8	9	10
b - >	6	7	8	9	10

3. 리스트(list)_참고

같은 값은 복사하고 싶으면?

```
a = [5,4,3,2,1]
```

```
b=a[:]
```

```
print(id(a),id(b))
```

```
140486843034272 140486843033392
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a.sort()
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a = [5,4,3,2,1]
```

```
b = a.copy()
```

```
print(id(a),id(b))
```

```
140486843511744 140486843484752
```

```
a.sort()
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
import copy  
a = [5,4,3,2,1]
```

```
b=copy.deepcopy(a)
```

```
print(id(a),id(b))
```

```
140486843203744 140486843046240
```

```
a.sort()  
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a
```

```
[1, 2, 3, 4, 5]
```

3. 리스트(list)_참고

같은 값을 복사할 때, 주의사항.

```
a = [5,4,3,2,1]
```

```
b=a[:]
```

```
print(id(a),id(b))
```

```
140486843034272 140486843033392
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a.sort()
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[5, 4, 3, 2, 1]
```

```
a = [[1,2],[3,4]]
```

```
a
```

```
[[1, 2], [3, 4]]
```

```
b=a[:]
```

```
a[0][1]=100
```

```
a
```

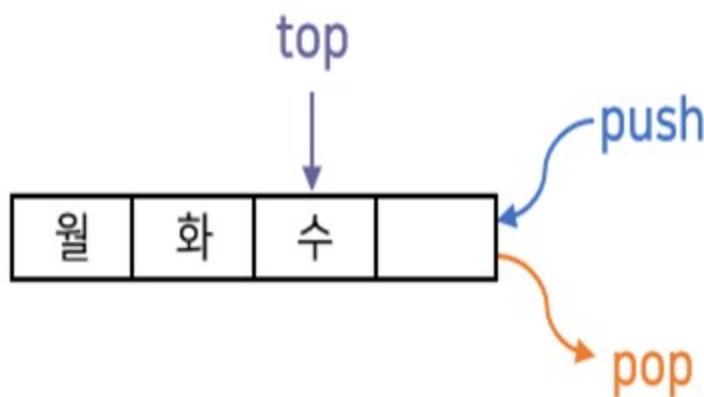
```
[[1, 100], [3, 4]]
```

```
b
```

```
[[1, 100], [3, 4]]
```

3. [자료구조] Stack(스택)

- 나중에 넣은 데이터를 먼저 반환하도록 설계된 메모리 구조
- 스택(Stack)은 데이터를 차곡차곡 쌓아 올린 형태
- 후입선출(LIFO, Last-in First-Out)



```
list_5 = [1, 2, 3, 4, 5]  
list_5.append(6)  
list_5.append(7)
```

```
list_5
```

```
[1, 2, 3, 4, 5, 6, 7]
```

```
list_5.pop()
```

```
7
```

```
list_5.pop()
```

```
6
```

```
list_5
```

```
[1, 2, 3, 4, 5]
```

3. [자료구조] Queue(큐)

- 먼저 넣은 데이터를 먼저 반환하도록 설계된 메모리 주소
- 스택(Stack)과 비슷하지만 다른 자료 구조
- 선입선출(Frist-in First-Out)
- 데이터의 삽입 연산(enQueue), 삭제 연산(deQueue)



```
list_5 = [1, 2, 3, 4, 5]  
list_5.append(6)  
list_5.append(7)
```

```
list_5  
[1, 2, 3, 4, 5, 6, 7]
```

```
list_5.pop(0)  
1
```

```
list_5.pop(0)  
2
```

```
list_5  
[3, 4, 5, 6, 7]
```

3. 리스트(list)_참고

- 패킹 : 한 변수에 여러 개의 데이터를 넣는 것
- 언패킹 : 한 변수의 데이터를 각각의 변수로 반환

```
e = [1,2,3] # 1,2,3이라는 변수를 t에 패킹  
e
```

```
[1, 2, 3]
```

```
f,g,h = e # e에 있는 값 1,2,3을 변수 a,b,c에 언 패킹  
print(f,g,h)
```

```
1 2 3
```

1번

다음과 같은 출력값을 만들어보자

```
In [ ]: ► a = '아메리카노를'  
          b = '좋아한다'  
          c = '많이'
```

```
In [ ]: ► # 정답을 적어주세요
```

```
Out [2]: '아메리카노를많이많이많이좋아한다'
```

연습문제 3

2번

좋아하는 음식 5개를 food라는 리스트에 담아보자

```
In [ ]: # 정답을 적어주세요
```

3번

리스트 food의 가장 앞에 파이썬이, 가장 뒤에는 집이라는 단어가 오도록 수정해보자

```
In [ ]: # 정답을 적어주세요
```

```
In [ ]: # 아래의 주석을 풀고 실행시켜보세요  
# food
```

4번

pop을 이용하여 파이썬을 삭제해보자

```
In [ ]: # 정답을 적어주세요
```

```
In [ ]: # 아래의 주석을 풀고 실행시켜보세요  
# food
```

5번

remove를 이용하여 집을 삭제해보자

4. 튜플(tuple)

- tuple
()을 이용하여 튜플(tuple)임을 선언

```
[42] tuple_1 = ()  
     tuple_2 = tuple()
```

```
[43] tuple_3 = (1, 2)  
     tuple_4 = (3,)  
     tuple_5 = (4, 5, (6, 7))  
     tuple_6 = 8, 9, 10
```

한 개의 요소만 사용할 때에는 콤마(,)를 반드시 사용해야 함
괄호가 없어도 튜플로 선언할 수 있음

4. 튜플(tuple)

- tuple

인덱싱과 슬라이싱이 가능

리스트와 달리 요소를 **삽입/삭제/수정** 할 수 없음

```
[44] tuple_3 = (1, 2)
     tuple_3[0] = 3
```

```
-----  
TypeError                                         Traceback (most recent call last)  
<ipython-input-44-c8372825bae8> in <module>()  
      1 tuple_3 = (1, 2)  
----> 2 tuple_3[0] = 3
```

TypeError: 'tuple' object does not support item assignment

SEARCH STACK OVERFLOW

4. 튜플(tuple)

- tuple의 연산

```
[45] tuple_3 = (1, 2)
     tuple_4 = (3, )
     tuple_3 + tuple_4
```

↳ (1, 2, 3)

```
[46] tuple_4 * 2
```

↳ (3, 3)

5. 딕셔너리(dictionary)

- dictionary

{ }을 이용하여 딕셔너리(dictionary)임을 선언

```
[47] dict_1 = {'key' : 'value'}  
     dict_2 = dict()
```

- 딕셔너리는 **key**와 **value** 값으로 이루어진 자료형
- 순서가 있는 자료형이 아니며, **key**를 통해 **value**값에 접근이 가능
- **key**는 고유한 값으로 중복될 수 없음
- **value**는 중복 가능

5. 딕셔너리(dictionary)

- dictionary 추가/삭제/수정

1. 요소추가하기: 새로운 **key**에 **value**를 할당

```
[48] dict_3 = {'홍길동' : 100, '홍계월' : 200}
```

```
[49] dict_3['슈퍼맨'] = 300  
dict_3
```

```
↪ {'슈퍼맨': 300, '홍계월': 200, '홍길동': 100}
```

주의 인덱스 대신 **key** 사용

2. 요소수정하기: 덮어쓰기

```
[51] dict_3['홍길동'] = 1  
dict_3
```

```
↪ {'홍계월': 200, '홍길동': 1}
```

5. 딕셔너리(dictionary)

- dictionary 추가/삭제/수정

3. 요소삭제하기

```
[50] del dict_3['슈퍼맨']  
dict_3
```

☞ {'홍계월': 200, '홍길동': 100}

주의 인덱스대신**key** 사용

del : 파이썬자체의명령어로메모리자체에서삭제

5. 딕셔너리(dictionary)

- key와 value에 접근하기

- key에 접근하기: keys()

```
[1] dict_3 = {'홍길동' : 100, '홍계월' : 200, '슈퍼맨' : 150, '배트맨' : 250}
    print(dict_3.keys())
    print(type(dict_3.keys()))
```

```
↳ dict_keys(['홍길동', '홍계월', '슈퍼맨', '배트맨'])
    <class 'dict_keys'>
```

- value에 접근하기: values()

```
[2] print(dict_3.values())
    print(type(dict_3.values()))
```

```
↳ dict_values([100, 200, 150, 250])
    <class 'dict_values'>
```

5. 딕셔너리(dictionary)

- key와 value에 접근하기

- 3. key와 value에 함께 접근하기

```
[3] print(dict_3.items())
    print(type(dict_3.items()))
```

```
↪ dict_items([('홍길동', 100), ('홍계월', 200), ('슈퍼맨', 150), ('배트맨', 250)])
<class 'dict_items'>
```

주의

리스트처럼 보이나 실제로는 **dict의 개체이므로**
리스트의 메소드를 사용할 수 없음

리스트처럼 사용하고 싶다면 리스트로 **형 변환**을 해야함

```
[4] list(dict_3.keys())[0]
```

```
↪ '홍길동'
```

5. 딕셔너리(dictionary)

- key와 value에 접근하기

4. key로 value에 접근하기

- 인덱싱처럼 키값을 이용하여 바로 접근
- 메소드get을 사용

```
[5] dict_3
```

```
↳ {'배트맨': 250, '슈퍼맨': 150, '홍계월': 200, '홍길동': 100}
```

```
[6] print(dict_3['배트맨'])  
print(dict_3.get('배트맨'))
```

```
↳ 250  
250
```

5. 딕셔너리(dictionary)

- key와 value에 접근하기

4. key로 value에 접근하기

```
[7] print(dict_3['펭수'])
```

```
→ -----  
      KeyError                                         Traceback (most recent call last)  
      <ipython-input-7-43368d90b0e9> in <module>()  
      ----> 1 print(dict_3['펭수'])  
      KeyError: '펭수'
```

→ 키값을 이용하여 접근할 경우, 해당 키가 없으면 오류 발생

SEARCH STACK OVERFLOW

```
[8] print(dict_3.get('펭수'))
```

→ get을 통해 접근할 경우, None을 반환

```
→ None
```

6. 셋(집합)

- set
 - {}을 이용하여 set을 선언

```
[9] set_1 = set()  
set_2 = set('Hello')  
set_3 = set([1, 2, 3])  
set_4 = {3,5, 'hi'}
```

중복을 허용하지 않음
순서가 없음



```
[10] set_2  
↳ {'H', 'e', 'l', 'o'}
```

```
[11] set_3
```

```
↳ {1, 2, 3}
```

```
[12] set_4
```

```
↳ {3, 5, 'hi'}
```

6. 셋(집합)

- set의 연산

1. 교집합

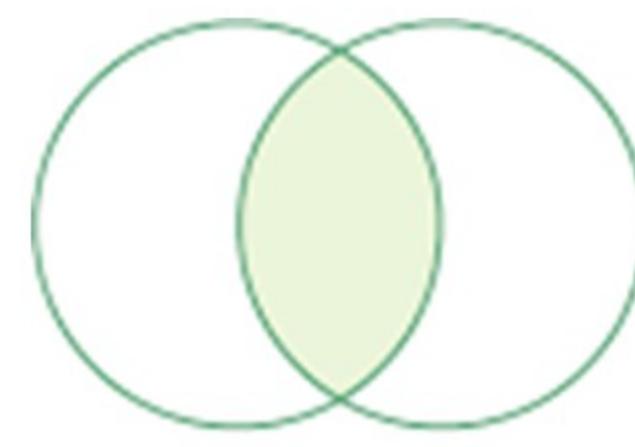
```
[13] set_5 = {1,2,3,4}  
      set_6 = {4,5,6,7}
```

```
[14] set_5 & set_6
```

```
↪ {4}
```

```
[15] set_5.intersection(set_6)
```

```
↪ {4}
```



집합 1 집합 2
(a) 교집합

6. 셋(집합)

- set의 연산

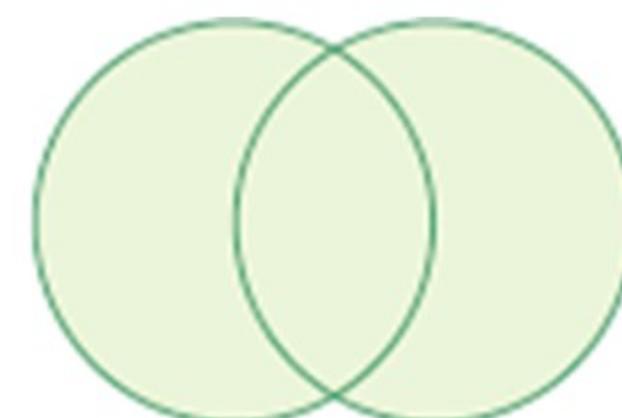
2. 합집합

```
[16] set_5 | set_6
```

↳ {1, 2, 3, 4, 5, 6, 7}

```
[17] set_5.union(set_6)
```

↳ {1, 2, 3, 4, 5, 6, 7}



집합 1 집합 2
(b) 합집합

6. 셋(집합)

- set의 연산

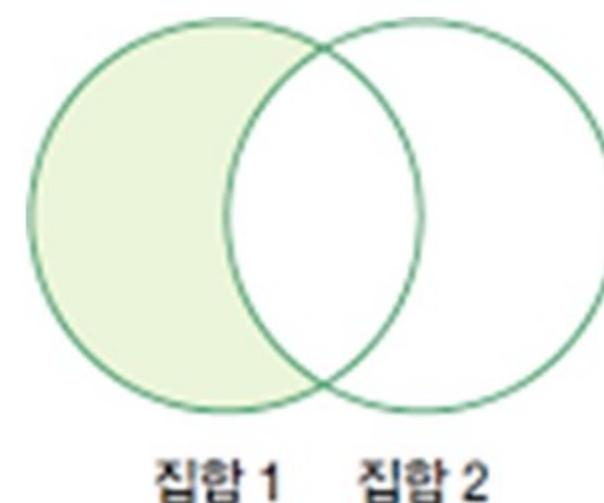
3. 차집합

```
[18] set_5 - set_6
```

☞ {1, 2, 3}

```
[19] set_5.difference(set_6)
```

☞ {1, 2, 3}



(c) 차집합

6. 셋(집합)

- set의 메소드

1. 한 개 추가하기: add

```
[22] set_6 = {4, 5, 6, 7}  
      set_6.add(8)  
      set_6
```

↳ {4, 5, 6, 7, 8}

2. 여러 개 추가하기: update

```
[23] set_6.update([1,2,3])  
      set_6
```

↳ {1, 2, 3, 4, 5, 6, 7, 8}

6. 셋(집합)

- set의 메소드

3. 한 개 지우기: remove

```
[24] set_6.remove(3)  
      set_6
```

```
↳ {1, 2, 4, 5, 6, 7, 8}
```

7. 불(bool)

- bool : 참과거짓을 나타내는 자료형

참: True

거짓: False

[26] True

↳ True

[27] type(False)

↳ bool

[28] 1 == 1

↳ True

[29] 1 > 2

↳ False

→ 비교 연산자와 조건문의 반환값으로 불자료형이 사용됨

비교 연산자	의미
==	같음
!=	같지 않음
>, >=, <, <=	크거나 같음/ 작거나 같음

7. 불(bool)

- bool : 참과거짓을 나타내는 자료형
자료형에도 참/거짓이 있음

자료	참/거짓
'python'	참
" "	거짓
[1,2,3]	참
[]	거짓
()	거짓
{ }	거짓
1	참
0	거짓
None	거짓

```
[30] bool(1)
```

↳ True

```
[31] bool('')
```

↳ True

```
[32] bool('')
```

↳ False

bool()함수를 통해 해당 자료형의 참/거짓을 확인할 수 있음

연습문제 4

사용자로 점수를 3개 입력받아
모든 점수가 65점보다 클 경우 True 아닐경우
False 를 출력하세요

연습문제 4

1번

국어, 영어, 수학 점수가 키로 하는 딕셔너리를 만들어보자.
각각의 점수는 다음과 같다.

국어 : 87

영어 : 88

수학 : 92

▶ # 정답을 적어주세요

2번

다음 연산들의 값을 예측해보자

] : ▶ # (1)
3 <= 1

] : ▶ # (2)
6 % 3 == 0

] : ▶ s1 = {'a', 'b', 'c'}
s2 = {'b', 'e', 'f', 'g'}
(3)
s1 & s2 == 'b'

] : ▶ # (4)
s1 & s2 == {'b'}

연습문제 4

a="20190505chicken19000"는 잘못 쓰여진 변수이다.

2019는 year이라는 변수에, 그 다음 0505를 day라는 변수에,

chicken을 menu라는 변수에, 19000을 money라는 변수에

인덱싱을 사용하여 각각 저장하고 출력하시오.

```
a = "20190505chicken19000"
```

```
# code 작성
```

```
year = ?
```

```
day = ?
```

```
menu = ?
```

```
money = ?
```

```
# code 종료
```

4. 조건문

- 문제해결 프로세스
- 조건문이란 ?
- if / elif / else
- 비교연산자
- 논리연산자 : and / or / not
- 요소인지 파악 하기 : in / not in
- if문 안에 if문
- Pass
- Fluent python

문제해결 프로세스

과학적 문제해결 프로세스

PART1.
문제발생의 원인 규명 과정



PART2.
해법의 수립과 적용 과정



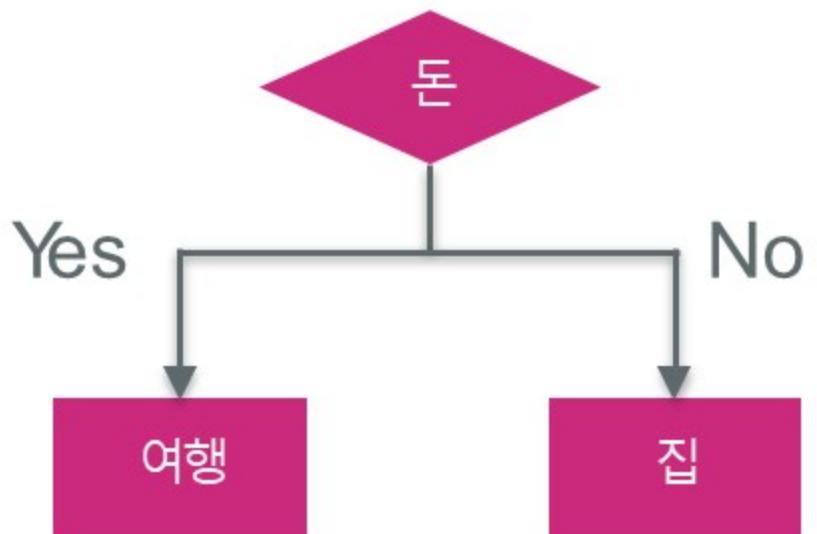
【문제해결사】 윤정식 2011.지회_참조

조건문이란 ?

- 조건에 따라 특정한 동작을 하게하는 명령어
- 프로그램 예시 in 생활
 - 자동차 앞차 간격이 100M 이하면 속도를 50km 이하로 낮춰라
 - 사용자가 어린이인 경우에는 VOD를 플레이 하지 마라
 - 패턴이 5회 틀리면 10초 동안 대기 상태로 만들어라
- 조건문은 조건을 나타내는 기준과 실행해야 할 명령으로 구성됨
- 조건의 참, 거짓에 따라 실행해야 할 명령이 수행되거나 되지 않음
- 파이썬은 조건문으로 if, else, elif 등의 명령 키워드를 사용함.

조건문이란 ?

- 1. if / else



if 조건식:

코드

들여쓰기 or 탭

공식 4칸

```
[ ] money = True  
  
if money :  
    print("여행을 간다")  
else :  
    print("집에서 쉰다")
```

여행을 간다

조건문이란 ?

- if / else

```
[ ] if 조건 :  
    수행 할 문장1  
    수행 할 문장2  
else :  
    수행 할 문장3  
    수행 할 문장4
```

```
[ ] money = True  
  
if money :  
    print("여행을 간다")  
else :  
    print("집에서 쉰다")
```



여행을 간다

주의

콜론(:)과 들여쓰기에 주의할 것

조건문이란 ?

- if / elif / else

```
[ ] if 조건1 :  
    수행 할 문장1  
elif 조건2 :  
    수행 할 문장2  
else :  
    수행 할 문장3
```

```
[ ] money = 20000  
  
if money < 5000 :  
    print('라면을 먹는다')  
elif 5000 < money < 25000 :  
    print('치킨을 먹는다')  
elif 25000 < money < 50000 :  
    print('삼겹살을 먹는다')  
else :  
    print('소고기를 먹는다')
```



치킨을 먹는다

조건문이란 ?

1. 비교연산자

```
[ ] a = 3  
    b = 5
```

a < b



True

```
[ ] c = 3  
    d = 3
```

c <= d



True

```
[ ] c != d
```



False

조건문이란 ?

1. 비교연산자

```
a=10  
b=10
```

```
a==b
```

```
True
```

```
a is b
```

```
True
```

```
a=300  
b=300
```

```
a==b
```

```
True
```

```
a is b
```

```
False
```

```
a=-5  
b=-5
```

```
a is b
```

```
True
```

```
a=-6  
b=-6
```

```
a is b
```

```
False
```

```
a = [1,2,3,4,5]  
b=a[:]
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[1, 2, 3, 4, 5]
```

```
a is b
```

```
False
```

```
import copy  
a = [1,2,3,4,5]  
b=copy.deepcopy(a)
```

```
a
```

```
[1, 2, 3, 4, 5]
```

```
b
```

```
[1, 2, 3, 4, 5]
```

```
a is b
```

```
False
```

조건문이란 ?

2. 논리연산자: and / or / not

[] True and True



True

[] True and False



False

1) and

비교하는 대상이 모두 참이여야만
True 반환

```
boolean_list = [True, False, True, False, True]
```

```
all(boolean_list)
```

False

조건문이란 ?

2. 논리연산자: and / or / not

[] True or True



True

2) or

비교하는 대상 중 하나만 참이여도
True 반환

[] True or False



True

```
boolean_list = [True, False, True, False, True]
```

```
any(boolean_list)
```

True

[] False or False



False

조건문이란 ?

2. 논리연산자: and / or / not

[] not True



False

[] not False



True

3) not

참이면 False를, 거짓이면 True를 반환

조건문이란 ?

2. 논리연산자: and

```
a=100  
if a>0 and (a%2)==0:  
    print('a is even')
```

a is even

5 and 7

7

7 and 13

13

- and : 조건이 True/ False로 반환이 됨.
- & : 비트연산자로 작동 됨.

7 & 13 # 0111&1101

5

13 & 7# 0111&1101

5

조건문이란 ?

2. 논리연산자: or

```
a=100  
if a>0 or (a%2)!=0:  
    print('a is even')
```

a is even

- or : 조건이 True/ False 로 반환이 됨.
- | : 비트연산자로 작동 됨.

5 or 7

5

7 | 13 # 0111|1101

15

7 or 13

7

13 | 7# 0111|1101

15

조건문이란 ?

3. 요소인지 파악하기 : in/not in

```
[ ] e = [1, 3, 5, 7]
```

```
[ ] 0 in e
```

 False

```
[ ] 1 in e
```

 True

1) x in s

x가 s의 요소인가

```
[ ] 2 not in e
```

 True

```
[ ] 3 not in e
```

 False

2) x not in s

x가 s의 요소가 아닌가

조건문이란 ?

- if문 안에 if문

조건 안에 조건을 만들 수 있음

```
[ ] money = 20000  
card = True
```

```
[ ] if card :  
    if money < 30000 :  
        print("삼겹살을 먹는다")  
    else :  
        print("소고기를 먹는다")  
    else :  
        if money <= 1000 :  
            pass  
        else :  
            print("라면을 먹는다")
```



삼겹살을 먹는다

조건문이란 ?

- pass

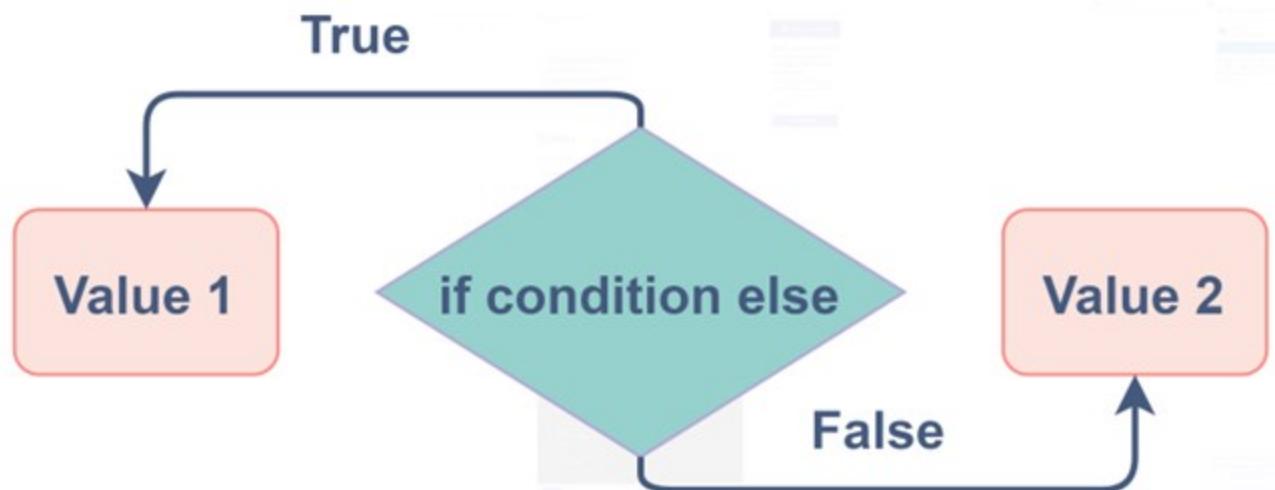
조건을 만족해도 아무 일도 일어나지 않게 하려면?

```
[ ] money = 1000
    card = False

[ ] if card :
    if money < 30000 :
        print("삼겹살을 먹는다")
    else :
        print("소고기를 먹는다")
else :
    if money <= 1000 :
        pass
    else :
        print("라면을 먹는다")
```

조건문이란 ?

- 삼항 연산자(Ternary Operators)
 - Condition(조건)이 참이면 true_value 값을 취하고 거짓이면 false_value 값을 취함
- [true_value] if [condition] else [false_value]



```
value = 20  
  
is_even = True if value %2 ==0 else False  
  
print(is_even)  
  
True
```

Fluent Python

• Formatting

1. %

코드	설명
%d	int
%f	float
%s	string
%c	문자 1개
%%	% 자체

```
[4] num = 1
```

```
[5] print('문자열 포맷팅 하기 예시 %d' %num)
```

↳ 문자열 포맷팅 하기 예시 1

```
[6] ch = '이름'
```

```
[7] print('내 이름은 : %s' %ch)
```

↳ 내 이름은 : 이름

```
[8] fl = 2.566132
```

```
[9] print('반올림 하기 : %0.1f' %fl)
```

↳ 반올림 하기 : 2.6

Fluent Python

• Formatting

1. %

코드	설명
%d	int
%f	float
%s	string
%c	문자 1개
%%	% 자체

괄호로 묶어주면 두 개 이상도 포맷팅 할 수 있음

```
[10] a = 3  
     b = 5
```

```
[11] print('%d 와 %d 를 더하면 %d' %(a,b,a+b))
```

⇒ 3 와 5 를 더하면 8

- Formatting

2. format

'{}'.format(포맷팅 할 변수)

```
[12] print('문자열 포맷팅 하기 예시 {}'.format(2))
```

↳ 문자열 포맷팅 하기 예시 2

```
[14] print('내 이름은 : {}'.format(ch))
```

↳ 내 이름은 : 이름

- **Formatting**

2. format

- 괄호로 묶어주면 두 개 이상도 포맷팅 할 수 있음

```
a = '류'  
b = '영표'
```

```
print('성은 {}, 이름은 {}'.format(a,b))
```

성은 류, 이름은 영표

- 인덱스를 통해 순서를 지정할 수도 있음

```
print('성은 {0}, 이름은 {1}'.format(a,b))
```

성은 류, 이름은 영표

```
print('Frist name : {1}, Last name : {0}'.format(a,b))
```

Frist name : 영표, Last name : 류

- **Formatting**

2. format

- 이름을 통해서 포맷팅할 수도 있음

```
[20] print('나는 오늘 {음료}를 {개수} 잔이나 마셨다'.format(음료='커피', 개수=3))
```

↪ 나는 오늘 커피를 3 잔이나 마셨다

- 이 밖에도 정렬이나 공백 이용하는 등 다양한 포맷팅이 가능

- Formatting

3. f-string(f 문자열 포맷팅)

f ‘{ }’

```
[28] year = 2020
```

```
[29] f'올해는 {year}'
```

→ '올해는 2020'

- **Formatting**

3. f-string(f 문자열 포맷팅)

- 다른 포맷팅과는 달리 표현식을 지원함

```
[39] drink = '커피'  
      nums = 3
```

```
[40] f'나는 오늘 {drink}를 {nums + 1} 잔이나 마셨다'
```

⇨ '나는 오늘 커피를 4 잔이나 마셨다'

```
# 문자열 맨 앞에 f를 붙이고, 출력할 변수, 값을 중괄호 안에 넣습니다.  
s = 'coffee'  
n = 5  
result1 = f'저는 {s}를 좋아합니다. 하루 {n}잔 마셔요.'  
print(result1)
```

저는 coffee를 좋아합니다. 하루 5잔 마셔요.

Fluent Python

• Formatting

```
[37] '나는 오늘 %s 를 %d + 1 잔이나 마셨다' %('커피', 3)
```

↳ '나는 오늘 커피 를 3 + 1 잔이나 마셨다'

```
[38] '나는 오늘 {음료}를 {개수+1} 잔이나 마셨다'.format(음료='커피', 개수=3)
```

↳ -----
KeyError Traceback (most recent call last)
<ipython-input-38-21356a636e64> in <module>()
----> 1 '나는 오늘 {음료}를 {개수+1} 잔이나 마셨다'.format(음료='커피', 개수=3)

KeyError: '개수+1'

SEARCH STACK OVERFLOW

```
[39] drink = '커피'  
      nums = 3
```

```
[40] f'나는 오늘 {drink}를 {nums + 1} 잔이나 마셨다'
```

↳ '나는 오늘 커피를 4 잔이나 마셨다'

• Padding

: 여유 공간을 지정하여 글자배열+소수점 자릿수를 맞추기

```
print('제품:%8s, 제품가격:%.5f.' %('Peer',3.241)) #8칸을 비우고, 5개의 소수점을 찍어라.  
print('제품:{0:8s}, 제품가격:{1:.5f}'.format('Peer',3.241))
```

제품: Peer, 제품가격:3.24100.
제품:Peer , 제품가격:3.24100.

```
print('제품:%16s, 제품가격:%10.5f.' %('Peer',3.241)) #10칸 공간,  
print('제품:{0:<8s}, 제품가격:{1:>.10f}'.format('Peer',3.241)) #제품 : 8칸을 비우면서 왼쪽으로 정렬,  
#제품가격은 10칸을 비우면서 오른쪽으로 정렬.
```

제품: Peer, 제품가격: 3.24100.
제품:Peer , 제품가격:3.2410000000.

연습문제 5-1

b로 a를 나눈 나머지가 3 초과면 실패, 3이면 무승부, 3 미만이면 성공이 출력되도록 만들어 보자

: ┌ a = 34
 b = 4

: ┌ # 정답을 적어주세요

성공

연습문제 5-2

[홀수 짝수 판별기]

사용자로부터 정수를 하나 입력받아
입력한 정수가 홀수인지 짝수인지 판별하여라.
(** 0은 짝수라 하자.)

[출력결과]

정수를 입력해주세요: 5

입력하신 5는 홀수입니다.

짝수를 입력해주세요: 10

입력하신 10은 짝수입니다.

무슨 학교 다니세요?

태어난 연도를 계산하여 학교 종류를 맞추는 프로그램

당신이 태어난 연도를 입력하세요.

2000

학생이 아닙니다.

- 나이는 현재년도-태어난 연도+1로 계산
- 26세 이하 20세 이상이면 “대학생”, 20세 미만 17세 이상이면 “고등학생”,
17세 미만 14세 이상이면 “중학생”, 14세 미만 8세 이상이면 “초등학생”
 - 그 외의 경우는 “학생이 아닙니다” 출력

연습문제 5-4

사용자로 점수를 3개 입력받아

각 점수가 65점보다 클 경우 합격 아닐경우 불합격을 출력하세요

단, 0~100 이 아닌 숫자가 입력된경우 잘못된
"잘못된 점수가 입력되었습니다" 를 출력하세요

연습문제 5-5

양의 정수 하나를 입력 받아 이 정수가 2의 배수인지 3의 배수인지 작성하시오

5. 반복문

- while 문
- for 문
- break, continue

- 정해진 동작을 반복적으로 수행하게 하는 명령문
- 프로그램 예시 in 생활
 - 100명의 학생에 성적을 산출할 때
 - 검색 결과를 출력해 줄 때
 - 워드에서 단어 바꾸기 명령 실행
- 반복문은 반복 시작 조건, 종료 조건, 수행 명령으로 구성됨
- 반복문 역시 반복 구문은 들여쓰기와 block으로 구분됨
- 파이썬은 반복문으로 for, while 등의 명령 키워드를 사용함

반복문 - while

조건문이 참인 동안 반복해서 문장을 수행함

```
[ ] while 조건 :  
    수행할 문장1  
    수행할 문장2
```

```
[ ] # 10이하의 짝수 프린트하기  
    i = 1  
    → 조건문에쓰일변수  
  
    while i <= 10 :  
        if i % 2 == 0 :  
            print(i)  
        i += 1  
    → 조건문  
    → 변수를 증감
```

2
4
6
8
10

반복문 - while

- break

반복문에서 빠져나오고 싶다면?

```
[ ] # 100번째 방문자 찾기  
i = 90
```

```
while i :
```

```
    i += 1
```

```
    if i == 100 :
```

```
        print("축하합니다. %d번째 방문자입니다." % i)
```

```
        break
```

```
print("감사합니다. 이벤트가 종료되었습니다.") → 반복문이 종료되었으므로 수행됨
```

→ 변수는 0이 아니라면 항상 참(무한반복)



축하합니다. 100번째 방문자입니다.

감사합니다. 이벤트가 종료되었습니다.

break를 사용하면 반복문이 더 이상 작동하지 않고 멈춤

반복문- while

- continue
반복문의 첫부분으로 돌아가고 싶다면?

```
[ ] i = 0

while i < 11 :
    i += 1
    if i == 6 : → 6일 때는 continue를 사용해 반복문의 가장 처음으로 가도록 함
        continue
    if i % 2 == 0 :
        print(i)
```



2

4

8

10

→ 6일 때는 짹수임에도 불구하고 프린트가 안됨

반복문 - for

- for문

```
[ ] for 변수 in range(변수가 속한 자료형, 혹은 변수의 범위) :  
    수행해야할 문장1  
    수행해야할 문장2
```

주의

콜론(:)과 들여쓰기에 주의할 것

```
[ ] for x in range(0,5) :  
    print(x)
```



0
1
2
3
4

반복문- for

- for문

```
[ ] for x in range(0,5) :  
    print(x)
```



0
1
2
3
4



- 변수의 범위 지정하기
- 1) 숫자로 범위 주기
 - 2) 자료형으로 범위 주기

- 변수의 범위 지정하기
 - 1) 숫자로 범위주기: range()

range(시작, 끝, 간격)

- 시작부터 끝- 1 까지
- 간격이 1이라면 생략 가능
- 0부터 시작하여 1씩 증가한다면 시작과 간격 생략 가능

반복문 - for

- 변수의 범위 지정하기

- 숫자로 범위주기: range()

연습문제7! 5에서부터 0까지 카운트다운을 세어 볼까요?

```
[ ] # 카운트 다운
for count in range(5, -1, -1):
    print(count) → 하나씩 작아지므로 간격은 1
                           → 0까지 출력해야 하므로 -1이 끝인덱스가 되어야 함
```



5
4
3
2
1
0

반복문 - for

- 변수의 범위 지정하기
- 2) 자료형으로 범위

```
[ ] word = 'Hello!'  
      for w in word:  
          print(w)
```



H
e
l
l
o
!

```
[ ] for a, b in [(2,1), (2,2), (2,3), (2,4)]:  
    print(a*b)
```



2
4
6
8

반복문 - for

- 이중으로 반복문을 사용하기

```
[ ] # 구구단 2단과 3단을 출력하기  
  
[5] for i in range(2,4) :  
    print('===' , i, '단 ===')  
    for j in range(1, 10) :  
        print(i * j)
```



👤 === 2 단 ===
2
4
6
8
10
12
14
16
18
==== 3 단 ====
3
6
9
12
15
18
21
24
27

break, continue

for, while 에서 제어흐름을 벗어나기 위해 사용

break

for, while 을 완전히 중단

continue

처음으로 돌아가 다음반복 수행

break, continue

for 문에서의 예제

break

```
for i in range(5):
    if (i==3):
        break
    print(i,end = '')
```

012

continue

```
for i in range(5):
    if (i==3):
        continue
    print(i,end = '')
```

0124

```
for i in range(5):
    if (i==3):
        pass
    print(i,end = '')
```

01234

break, continue

while 문에서의 예제

break

```
i=0  
  
while i<30:  
    if i==20:  
        break  
    print(i,end = '')  
    i +=1
```

012345678910111213141516171819

continue

```
i=0  
  
while i<30:  
    if i==20:  
        continue  
    print(i,end = '')  
    i +=1
```

012345678910111213141516171819

Toy example

- 1부터 10까지 합을 구하시오. (range 함수를 이용하시오)

연습문제 1

- (1) 사용자로부터 정수를 입력받아, 해당 정수만큼 “안녕”을 출력하세요.
- (2) 사용자로부터 정수를 입력받아, 입력된 정수 만큼 별 찍기

(1) [출력결과]

```
정수를 입력해주세요: 5
안녕
안녕
안녕
안녕
안녕
```

(2) [출력결과]

```
정수를 입력해주세요: 5
*
**
***
****
*****
```

연습문제 2

- (1) 계단형, n을 입력받아, n 만큼 줄을 만들고 계단 형태로 별찍기
- (2) 삼각형, 왼쪽 아래가 직각인 n만큼의 높이를 가지는 직각삼각형
- (2)-(1) 오른쪽 아래가 직각인 n만큼의 높이를 가지는 직각삼각형
- (3) 역삼각형, 왼쪽 위가 직각인 n만큼의 높이를 가지는 직각삼각형
- (3)-(1) 오른쪽 위의 직각인 n만큼의 높이를 가지는 직각삼각형
- (4) 피라미드, n만큼의 높이를 가지는 홀수개의 별 피라미드

(1) [출력결과]	(2) [출력결과]	(2-1) [출력결과]	(3) [출력결과]	(3-1) [출력결과]	(4) [출력결과]
n:5 ***** *	n : 5 * ** *** **** *****	n : 5 * ** *** **** *****	n : 5 *	n : 5 *	n : 5 * *** ***** *****

연습문제 3

- (1) $x = [3, 6, 9, 20, -7, 5]$ 의 값의 모든 요소에 10을 곱한뒤 출력하세요
 - 심화 : 출력과 리스트 x 의 값에도 모두 10을 곱해주세요
- (2) $y = \{"math": 70, "science": 80, "english": 20\}$ 의 값의 모든 요소에 10을 더한뒤 출력하세요
- 심화 : 출력과 딕셔너리 y 의 값에도 모두 10을 더해주세요
- (3) 숫자를 입력받고 입력받은 정수의 구구단을 출력하세요

(3) [출력결과]

정수를 입력해주세요: 5

5*1 = 5

5*2 = 10

5*3 = 15

...

...

연습문제 4

- 1~100 임의의 숫자를 맞추시오.

```
import random

true_value = random.randint(1,100)
input_value=99999 #임의의 값 할당.

print('숫자를 맞춰보세요(1~100)')
```

숫자를 맞춰보세요(1~100)

50

숫자가 너무 큽니다.

30

숫자가 너무 큽니다.

10

숫자가 너무 큽니다.

1

숫자가 너무 작습니다.

4

숫자가 너무 작습니다.

5

숫자가 너무 작습니다.

정답입니다. 입력한 숫자는 5입니다.

연습문제 5

- (1) word = [“school”, “game”, “piano”,
“science”, “hotel”, “mountain”] 중 글자수가 6
글자 이상인 문자를 모아 새로운 리스트를 생성
하세요
- (2) 구구단을 1단부터 9단까지 출력하세요

연습문제 6

1-100 까지 숫자중

3과 5의 공배수일경우 “3과 5의 공배수”

나머지 숫자중 3의배수일경우 “3의배수”

나머지 숫자중 5의배수일경우 “5의배수”

모두 해당되지 않을경우 그냥숫자

를 출력하세요

심화 : 1-입력한숫자까지의 숫자중

연습문제 7

사용자로부터 숫자를 계속 입력받다가

값을 입력해주세요30

s or S 를 입력하면 합계출력

값을 입력해주세요20

값을 입력해주세요50

값을 입력해주세요40

값을 입력해주세요30

값을 입력해주세요s

합계는 ? 170

연습문제 8

사람 별 평균을 구하라.

- kor_score = [39,69,20,100,80]
- math_score = [32,59,85,30,90]
- eng_score = [49,70,48,60,100]
- midterm_score = [kor_score,math_score,eng_score]

	A	B	C	D	E
국어점수	39	69	20	100	80
수학점수	32	59	85	30	90
영어점수	49	70	48	60	100

6. 파일 읽기/쓰기

- 파일 쓰기
 - 파일 읽기

- open

open은 파일을 여는데 사용하는 함수

open(file_path, mode=' ', encoding=' ')

- **file_path** : 파일의 주소
- **mode** : 파일을 열 때 필요한 파일 모드(읽기/쓰기/수정)
- **encoding** : 파일을 열 때 필요한 인코딩(생략 가능)

```
[ ] new_file = open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'w')

[ ] print(new_file)

↳ <_io.TextIOWrapper name='./drive/My Drive/Colab Notebooks/day2/toy_data/new.txt' mode='w' encoding='UTF-8'>
```

* 주의* open으로 연 파일은 파일 객체를 반환함

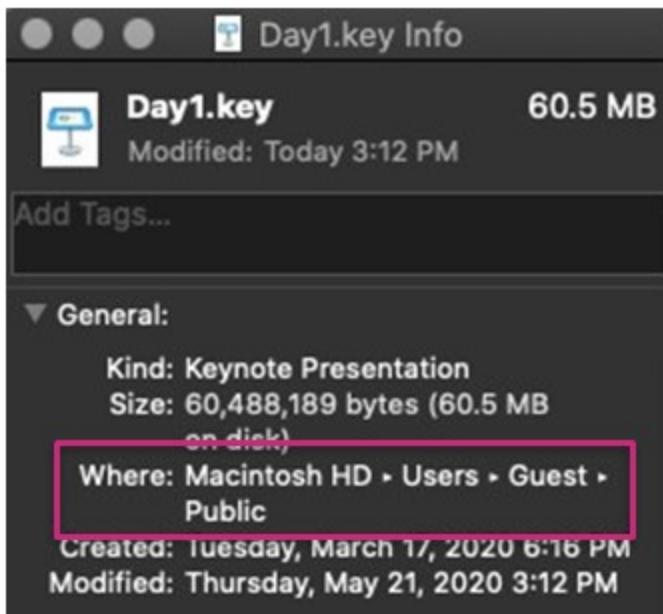
- open

1. file path

- file path(파일 경로)는 string
- 파일은 두 가지 방식으로 표현할 수 있음
 - 절대 경로: 파일이 가지고 있는 고유한 경로
 - 상대 경로: 현재 내 위치에서부터 해당 파일까지의 경로

- open
 - 1. file path
 - 1) 절대경로: 파일이 가지고 있는 고유한 경로

현재 위치와 무관하게 항상 해당 파일에 접근 가능



→ 'Users/Guest/Public/Day1.key'

파일쓰기

- open
 - 1. file path
 - 2) 상대 경로 : 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

현재파일의 위치	./
현재파일의 상위폴더	../

파일쓰기

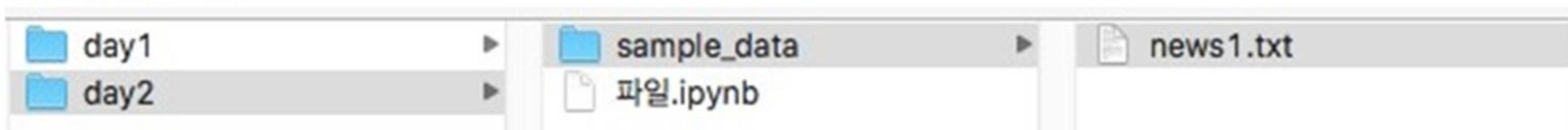
- open

1. file path

- 2) 상대 경로: 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

현재파일의 위치	./
현재파일의 상위폴더	../

Quiz. 현재 '파일.ipynb'에서 작업중이다. 'news1.txt' 파일을 열기 위해 필요한 상대 경로는?



→ './sample_data/news1.txt'

파일쓰기

- open

1. file path

- 2) 상대 경로: 지금 내가 작업하고 있는 위치에서부터 해당 파일까지의 경로
내 위치에 따라 달라질 수 있음

현재파일의 위치	./
현재파일의 상위폴더	../

Quiz. 현재 '파일.ipynb'에서 작업중이다. 'day1' 디렉토리(폴더)에 있는 'quiz.txt' 파일을 열기 위해 필요한 상대 경로는?



→ '../day1/quiz.txt'

파일쓰기

- open

1. file path

- file path(파일 경로)는 **string**
- **윈도우 환경**에서 파일 열 때 주의사항

윈도우환경에서는 '\U'가 파일경로에 포함된 경우 안 열리는 경우(유니코드 에러)가 존재함.

→ 이럴땐, **r‘파일경로’**를 사용하거나 **\를 두번씩 써주면 됨.**

경로: ‘Users/Guest/Public/Day1.key’

→ r‘Users/Guest/Public/Day1.key’
→ ‘Users\\Guest\\Public\\Day1.key’

파일쓰기

- open

2. mode

파일 모드	설명	비고
r	읽기 모드	파일에 새로운 내용을 쓰거나 수정이 불가능
w	쓰기 모드	파일에 새로운 내용을 덮어씀(기존의 내용 모두 삭제)
a	수정 모드	파일에 새로운 내용이 추가됨(기존의 내용 + 새로운 내용)
rb	바이너리 읽기 모드	바이너리 파일을 읽을 때 사용
wb	바이너리 쓰기 모드	바이너리 파일을 쓸 때 사용

- open

3. encoding

지정하지 않으면 플랫폼에 따르게 됨

- UTF-8 : 대표적인 조합형 유니코드 인코딩 방식

- CP949: 현재 윈도우 커널에는 유니코드가 적용되었지만, 한글 윈도우의 명령 프롬프트가 사용하는 기본 인코딩은 여전히 **CP949**

- write

파일.write(내용)

- 파일: open으로 열었던 파일 객체
- 내용: 파일에 쓰고 싶은 내용

```
[6] data = '새로운 내용을 쓰고 싶다면, \n 이렇게 작성하면 됩니다.'
```

```
[7] new_file.write(data)
```

↳ 29

→ 파일 안의 문자의 갯수를 반환

* 주의 * 파일을 열고, 작업을 마쳤다면 반드시 닫을 것

```
[8] new_file.close()
```

- with

with open(file_path, mode=) as 변수:

수행할 문장1

수행할 문장2

- with문을 사용하면 with블록을 벗어나는순간 파일을 자동으로 close해줌
- **as 변수**: open으로 연 파일 객체를 **변수**로 받아줌

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'w') as f :  
    data = '이렇게도 작성이 가능합니다. close를 따로 안해도 되요.'  
    f.write(data)
```

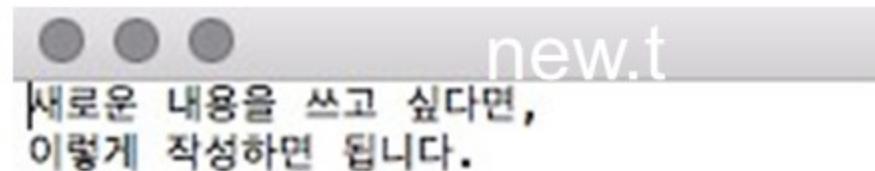
파일읽기

- read / readline / readlines

함수	설명
read	파일 전체를 한번에 읽어서 string 으로 반환
readline	파일을 한줄만 읽어서 string 으로 반환
readlines	파일을 줄단위로 모두 읽어서 list 로 반환

파일읽기

- read / readline / readlines



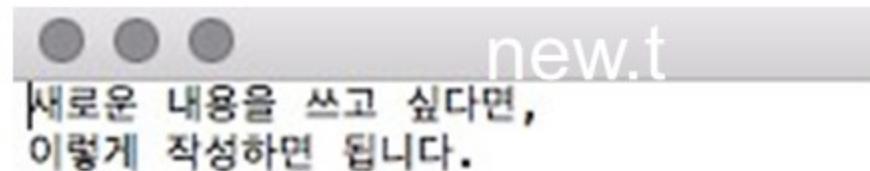
1. read

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:  
    lines = f.read()  
    print(lines)  
    print(type(lines))
```

⇨ 새로운 내용을 쓰고 싶다면,
이렇게 작성하면 됩니다.
<class 'str'>

→ 파일 **전체**를 **string**으로 불러옴

- read / readline / readlines



2. readline

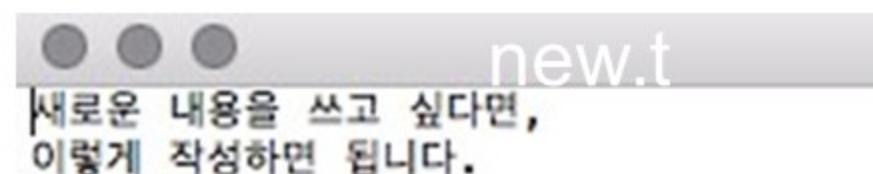
```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:  
    line = f.readline()  
    print(line)  
    print(type(lines))
```

⇨ 새로운 내용을 쓰고 싶다면,

```
<class 'str'>
```

→ 파일에서 **한줄**만 **string**으로 불러옴

- read / readline / readlines



2. readline

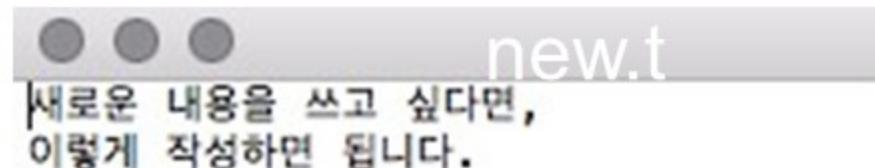
(참고) readline + while문으로 전체파일을 불러올수 있음

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:  
    while True :  
        line = f.readline()  
        if not line : break  
        print(line)
```

⇨ 새로운 내용을 쓰고 싶다면,

이렇게 작성하면 됩니다.

- read / readline / readlines



2. readlines

```
[ ] with open('/content/drive/My Drive/Colab Notebooks/python/toy_data/new.txt', 'r') as f:  
    lines = f.readlines()  
    print(lines)  
    print(type(lines))
```

```
↳ ['새로운 내용을 쓰고 싶다면,\n', '이렇게 작성하면 됩니다.'][  
<class 'list'>]
```

- 파일에서 줄단위로 읽어서 list로 불러옴
- 개행문자(\n)가 남아 있음

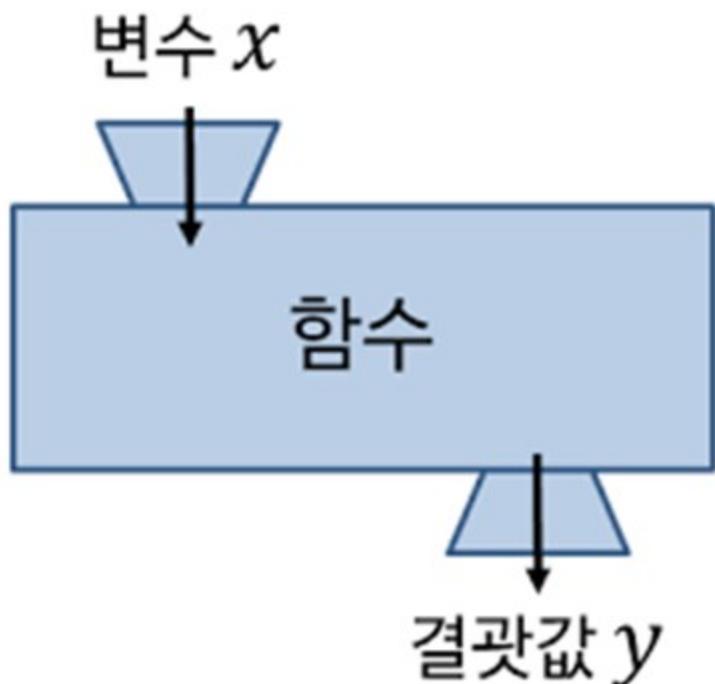
7. 함수

- 함수란?
- 함수 정의하기
- 함수(Function)
- 연습문제

- 더 알아보기
- 연습문제
- 함수와 변수

함수(Function)

- 함수란?



```
[ ] def 함수이름(함수인자) :  
    수행할 문장1  
    수행할 문장2  
    return 반환값
```

- 일정한 프로세스를 할 수 있도록 하는 것
- 코드를 논리적인 단위로 분리
- 여러 개의 문장을 하나로 묶어주는 역할

함수(Function)

- 함수 정의하기

```
def 함수이름(파라미터):  
    함수의 내용  
    Return 반환값
```

- **함수이름**: 사용자가 정의하는 함수 이름, 기존에 사용되는 예약어들을 제외하고 사용
- **파라미터**: 함수안에서 사용할 변수들(생략 가능)
- **return** : 함수안에서 모든 연산을 마친 후 반환할 값(생략 가능)

함수(Function)

- 함수 정의하기

pop quiz! 두개의 input를 받으면 더하기를 하여 값을 돌려주는 함수를 만들어봅시다.

```
[ ] def add(a, b) :  
    result = a + b  
    return result
```



```
[ ] c = add(3, 5)  
c
```



8

→ 함수정의

→ 함수호출

함수(Function)

- 함수 정의하기

- 매개변수와 return 값이 모두 있는 함수

```
[ ] def add(a, b) :  
    result = a + b  
    return result
```

```
[ ] c = add(3, 5)  
c
```



8

함수(Function)

- 함수 정의하기

2. return값이 비어있는 함수

```
[ ] def sub(a, b) :  
    print('뺄셈의 결과는 %d입니다.' %(a-b))  
    return
```

```
[ ] d = sub(1, 2)
```

 뺄셈의 결과는 -1입니다.

```
[ ] print(d)
```

 None

함수(Function)

- 함수 정의하기

- 3. return이 없는 함수

```
[ ] def mul(a, b) :  
    print('%d 와 %d의 곱은 %d입니다.' %(a, b, a*b))
```

```
[ ] mul(3, 2)
```

❸ 3 와 2의 곱은 6입니다.

```
def f(x) :  
    print(x+10)
```

```
f(10)
```

20

```
c = f(10)
```

20

```
print(c)
```

None

함수(Function)

- 함수 정의하기

4. 매개변수와 return 값이 모두 없는 함수

```
[ ] def start():
    print("Hello World!")
```

```
[ ] start()
```



Hello World!

함수(Function)

예제) 구구단. 함수를 사용하지 않을 때 VS 함수를 사용할 때

```
for i in range(1,10):
    print(f'===== {i} 단=====')
    for j in range(1,10):
        print(i*j)
```

===== 1 단=====

1
2
3
4
5
6
7
8
9
--

```
def gugu(num):
    for i in range(1, 10):
        print(f'{num} x {i} = {num * i}')
```

gugu(3)

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27

함수(Function)

3가지 전달 방식이 존재함.

값에 의한 호출 (Call by Value)	참조에 의한 호출 (Call by Reference)	객체 참조에 의한 호출 (Call by Object Reference)
함수에 인자를 넘길 때 값만 넘김	함수에 인자를 넘길 때 메모리 주소를 넘김	객체의 주소가 함수로 전달됨.
함수 내에 인자 값 변경시에 호출자에게 영향을 주지 않음	함수 내에 인자 값 변경이 되면 호출자의 값도 변경됨.	전달된 객체를 참조하여 변경 시 호출자에게 영향을 주나, 새로운 객체를 만들 경우 호출자에게 영향을 주지 않음.

함수(Function)

```
def Sandwich(eggs):
    eggs.append(1) # 기존 객체의 주소값에 [1] 추가
    eggs = [2, 3]

ham = [0]
Sandwich(ham)
print(ham) # [0, 1]
print(eggs)

[0, 1]
-----
NameError                                 Traceback (most r
<ipython-input-4-15a09db8233a> in <cell line: 8>()
      6 Sandwich(ham)
      7 print(ham) # [0, 1]
----> 8 print(eggs)

NameError: name 'eggs' is not defined
```

```
# 1번 함수
def swap_value(x, y):
    temp = x
    x = y
    y = temp

# 2번 함수
def swap_offset(offset_x, offset_y):
    temp = a[offset_x]
    a[offset_x] = a[offset_y]
    a[offset_y] = temp

# 3번 함수
def swap_reference(list, offset_x, offset_y):
    temp = list[offset_x]
    list[offset_x] = list[offset_y]
    list[offset_y] = temp
```

```
# 테스트
a = [1,2,3,4,5]
swap_value(a[1], a[2])
print(a) # [1,2,3,4,5] swap 발생 안함.
swap_offset(1,2)
print(a) # [1,3,2,4,5] swap 발생.
swap_reference(a, 1, 2)
print(a) # [1,3,2,4,5] swap 발생.
```

```
[1, 2, 3, 4, 5]
[1, 3, 2, 4, 5]
[1, 2, 3, 4, 5]
```

연습문제

함수를 만들어 보자.

- 이 함수는 두 개의 숫자를 input으로 받으면 앞의 숫자를 뒤의 숫자로 나누는 함수이다.
- 나눗셈을 한 후에는 몫과 나머지 순으로 된 튜플 값을 반환한다.

```
def div(a, b):  
    # 정답을 적어주세요
```

div(10, 3)

(3, 1)

div(3, 5)

(0, 3)

div(1, 10)

(0, 1)

연습문제

함수를 만들어보자.

- 이 함수는 두 개의 숫자를 input으로 받으면 작은 수로 큰 수를 나눈 몫과 나머지를 반환하는 함수이다.
- 반환 값은 튜플로 되어 있으며 몫, 나머지 순으로 되어 있다.
- 단, 0으로 나누는 것은 불가하기 때문에 두 수 중에 작은 수가 0이라면 화면에 '0은 사용할 수 없습니다.'를 출력하고 종료되어야 한다.

```
div2(3, 5)
```

(1, 2)

```
div2(0, 5)
```

0은 사용할 수 없습니다.

```
div2(10, 1)
```

(10, 0)

함수(Function)

- 더 알아보기

- 1 매개변수의 초기값 미리 설정하기

```
[ ] def function(a, n=2) :  
    print("%d의 제곱은 %d 입니다." %(a, a**n))
```

```
[ ] function(4)
```



4의 제곱은 16 입니다.

→ 초기값을 설정하는 경우, 매개변수의 순서에 주의!
초기값 없는 변수, 초기값 있는 변수 순서로 배치

함수(Function)

- 더 알아보기

2. 매개변수가 몇 개가 필요할지 모를 때

```
[3] def test(*args) :  
    print(args)
```

```
[4] test(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

⇒ (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

```
[5] test([1,2],3)
```

⇒ ([1, 2], 3)

→ 매개변수에 *args를 붙이면
함수가 받은 input 모두를 tuple로 묶어줌

Quiz! 왜 tuple로 묶어줄까요?

함수(Function)

- 더 알아보기
2. 매개변수가 몇 개가 필요할지 모를 때

```
def add(**kwargs):  
    print(kwargs)|
```

```
add(a=1, b=2, c=3)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

→ 매개변수에 *kwargs를 붙이면 함수가 받은 input 모두를 dict로 묶어줌

Quiz! 왜 dict로 묶어줄까요?

연습문제

어떠한 string을 받으면 일정한 단위로 끊어서 화면에 출력하는 함수를 짜보자.
끊는 단위는 따로 정하지 않으면 2로 설정해보자.

Hint : input을 string 과 unit = 2로 받고, while을 사용하고, 길이는 len 함수를 사용하도록 하자.

```
: func('테스트를 위한 문장입니다.')
```

```
테스  
트를  
위  
한  
문장  
입니  
다.
```

```
: func('테스트를 위한 문장입니다.', 4)
```

```
테스트를  
위한  
문장입니  
다.
```

연습문제

add_all 함수를 짜봅시다

```
add_all(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

55

Hint : *args를 input으로 받으세요.

함수(Function)

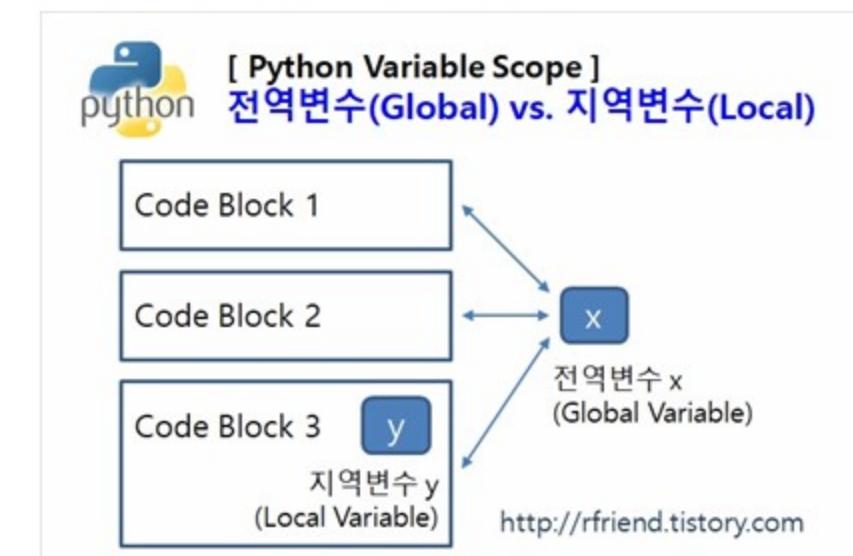
- 함수와 변수

Quiz! 함수 안에 선언한 변수를 함수 밖에서도 사용할 수 있을까?

NO!

함수 안의 블록은 함수 고유의 영역임.

따라서, 함수 안에서 선언한 변수는 함수 밖에서 사용할 수 없음.



함수(Function)

- 함수와 변수

예시 1)

```
[ ] def myfunction():
    in_val = '함수 안의 변수'
```

```
[ ] myfunction()
```

```
[ ] in_val
```



```
NameError Traceback (most recent call last)
```

```
<ipython-input-30-a23ad0090d3> in <module>
```

```
----> 1 in_val
```

```
NameError: name 'in_val' is not defined
```

함수(Function)

- 함수와 변수

예시 2)

```
[ ] a = 1
```

```
[ ] def myfunction2(a) :  
    a += 1  
    print(a)
```

```
[ ] myfunction2(a)
```

2

```
[ ] a
```

1

함수 실행 시 출력값은 무엇일까요?

a는 어떤 값일까요?

함수(Function)

- 함수와 변수

함수 안의 블록은 함수 고유의 영역임.

따라서, 함수 안에서 선언한 변수는 함수 밖에서 사용할 수 없음.

```
[ ] def 함수이름(함수인자) :  
    수행할 문장1  
    수행할 문장2  
    return 반환값
```

2. global 사용하기 ← 권장하지 않음

- 함수와 변수
- 함수 안에 있는 변수를 밖에서도 쓰고 싶다면?
 1. return 사용하기
 2. global 사용하기

함수(Function)

- 함수와 변수

함수 안에 있는 변수를 밖에서도 쓰고 싶다면?

1. return 사용하기

```
a = 1
def myfunction3(a) :
    a += 1
    print(a)
    return a
```

함수(Function)

- 함수와 변수

함수안에 있는 변수를 밖에서도 쓰고 싶다면?

2. global 사용하기

```
[ ] a = 1  
  
[ ] def myfunction4():  
    global a  
    a += 1  
    print(a)
```

```
[ ] myfunction4()
```

2

```
[ ] a
```

2

```
a=1  
print('처음에 있는 전역변수',a)  
def myfunction():  
    global a  
    a+=1  
    print('안에 있는 a',a)  
myfunction()  
print('밖에 있는 a',a)
```

처음에 있는 전역변수 1
안에 있는 a 2
밖에 있는 a 2

함수(Function)

- 함수와 변수

return 사용하기

```
[21] for _ in range(5) :  
    c = myfunction3(a)  
    print('함수의 결과 : %d' %c )  
    print('a의 값 : %d' %a)
```

```
↳ 3  
함수의 결과 : 3  
a의 값 : 2  
3  
함수의 결과 : 3  
a의 값 : 2  
3  
함수의 결과 : 3  
a의 값 : 2  
3  
함수의 결과 : 3  
a의 값 : 2  
3  
함수의 결과 : 3  
a의 값 : 2
```

global 사용하기

```
[25] for _ in range(5) :  
    myfunction4()  
    print('a의 값 : %d' %a)
```

```
↳ 2  
a의 값 : 2  
3  
a의 값 : 3  
4  
a의 값 : 4  
5  
a의 값 : 5  
6  
a의 값 : 6
```

← 함수밖의변수(global variable)이
함수의작동에따라그값이변함

연습문제

- 팩토리얼(Factorial)을 구하시오.

1부터 시작하여 어떤 범위에 있는 모든 정수를 곱하는 것.

$$\text{Ex)} \ 5! = 120$$

Function annotation

```
def func_name(params):  
    return 0
```

```
def func_name(param1 : int, param2 : str) -> int:  
    return 0
```

- 오른쪽 같은 표현들은 말 그대로 주석의 역할을 한다.
- 즉, 코드상 기능을 하지 않고 보충 설명을 해주는 것으로, 다른 형태도 사용하더라도 문제 없이 작동
- 함수명 뒤 화살표(->)의 경우, 해당 함수의 return 값의 형태에 대한 주석

```
def func_name(params) -> Tuple[float, float, float]:  
    return test_loss, test_f1, test_acc
```

- :의 경우 매개변수(Parameter)의 형태에 대한 주석

8. 파이썬 심화

- enumerate
- Zip/Lambda/map
- List comprehension
- Class
- 모듈 /패키지

파이썬 심화

pop quiz! 문제를 먼저 풀어보자. 여기는 동네 유명한 빵집이다. 사람들에게 먼저 온 순서대로 번호표를 나누어주려고 한다. 번호표를 나누어주는 함수를 작성해보자.

- 함수는 사람 이름으로 되어 있는 리스트를 받아서 "대기번호 x번 : 사람 이름"를 화면에 출력하고 (번호표, 사람 이름)을 원소로 이루어진 리스트를 반환한다.

- input : 리스트
- output : 리스트

▶ `people = ['펭수', '뽀로로', '뚝딱이', '텔레토비']`

▶ `# func1을 만들어주세요`

▶ `lines = func1(people)`

대기번호 1번 : 펭수
대기번호 2번 : 뽀로로
대기번호 3번 : 뚝딱이
대기번호 4번 : 텔레토비

파이썬 심화

```
[ ] people = [ '펭수', '뽀로로', '뚝딱이', '텔레토비' ]
```

```
[ ] def func1(line) :
    new_lines = []
    i = 1
    for x in line :
        print("대기번호 %d번 : %s" %(i, x))
        new_lines.append((i, x))
        i += 1
    return new_lines
```

→ 대기번호를 트래킹하는 변수 i
→ 별도로 업데이트해야 함

```
[ ] lines = func1(people)
```



대기번호 1번 : 펭수
대기번호 2번 : 뽀로로
대기번호 3번 : 뚝딱이
대기번호 4번 : 텔레토비

파이썬 심화_enumerate

- enumerate

- 반복 가능한 객체의 인덱스와 원소에 함께 접근할 수 있는 함수
- tuple(인덱스, 원소)의 형태로 객체를 반환
- 보통 리스트와 함께 쓰임(tuple도 가능)

```
[ ] lst = [ 'a' , 'b' , 'c' ]
      for x in enumerate(lst) :
          print(x)
```



(0, 'a')
(1, 'b')
(2, 'c')

파이썬 심화_enumerate

- enumerate

```
[ ] st = 'abcd'  
  
for x in enumerate(st) :  
    print(x)
```

(0, 'a')
(1, 'b')
(2, 'c')
(3, 'd')

```
[ ] se = {'a', 'b', 'c'}  
for x in enumerate(se) :  
    print(x)
```

(0, 'c')
(1, 'a')
(2, 'b')

```
[ ] dic = {0 : 'p', 1 : 'b' , 2 : 'd'}  
  
for x in enumerate(dic):  
    print(x)
```

(0, 0)
(1, 1)
(2, 2)

주의

set은 순서가없는자료형

파이썬 심화_enumerate

앞에서 풀었던 퀴즈를 enumerate를 이용해서 다시 작성해보자
(빵집문제)

```
[ ] def func1_with_enu(line) :  
    new_lines = []  
    for idx, val in enumerate(line):  
        print("대기번호 %d번 : %s" %(idx+1, val))  
        new_lines.append((idx+1, val))  
    return new_lines
```

```
[ ] lines = func1_with_enu(people)
```



대기번호 1번 : 펭수
대기번호 2번 : 뽀로로
대기번호 3번 : 뚝딱이
대기번호 4번 : 텔레토비

파이썬 심화_zip

- zip
 - 반복가능한 객체들을(2개 이상) 병렬적으로 묶어주는 함수
 - 각 원소들을 튜플의 형식으로 묶어줌

```
[1] str_list = ['one', 'two', 'three', 'four']
      num_list = [1, 2, 3, 4]

      for i in zip(num_list, str_list) :
          print(i)
```

```
↪ (1, 'one')
  (2, 'two')
  (3, 'three')
  (4, 'four')
```

```
[2] s1 = '123'
      s2 = 'abc'
      s3 = 'ㄱㄴㄷ'
      list(zip(s1,s2,s3))
```

```
↪ [('1', 'a', 'ㄱ'), ('2', 'b', 'ㄴ'), ('3', 'c', 'ㄷ')]
```

파이썬 심화_lambda

- lambda
 - lambda(람다)는 식형태로 되어 있어서 lambda expression(람다 표현식)이라고도 불림
 - 람다는 익명의 함수로서 함수를 간편하게 작성할 수 있게 해줌
 - python3에서 사용이 권장되지는 않지만 머신러닝이나 데이터분석시 많이 사용됨

lambda 매개변수: 리턴 값

파이썬 심화_lambda

- lambda : 한 줄을 실행한 결과 값이 바로 반환값이 됨.

```
[ ] def plus_two(num) :  
    return num + 2
```

```
[ ] a = 2  
b = plus_two(a)  
print(b)
```



4

```
[ ] lambda x : x + 2
```

👤 <function __main__.<lambda>(x)>

→ 람다는함수를생성함

```
[ ] func2 = lambda x : x + 2
```

```
[ ] c = func2(2)
```

```
[ ] c
```



4

파이썬 심화_map

- map
 - 리스트, 튜플, 스트링등 자료형 각각의 원소에 동일한 함수를 적용

map (함수, 자료형)

```
items = [1, 2, 3, 4, 5]
```

```
squared = []
for i in items :
    squared.append(i*i)
```

```
print(squared)
```

```
[1, 4, 9, 16, 25]
```

```
squared_map = list(map(lambda x : x**2, items))
```

```
print(squared_map)
```

```
[1, 4, 9, 16, 25]
```

map



연습 문제 1

lambda와 map을 이용하여 items의 요소들을 string(문자)로 바꾸는 것을 짜봅시다.

▶ items = [1, 24, 3, 6, 7]

▶ # 정답을 적어주세요

▶ print(str_items)

['1', '24', '3', '6', '7']

연습 문제 2

1~10까지의 정수를 항목으로 갖는 리스트 객체에서 map 함수와 람다식을 이용해

항목의 제곱 값을 갖는 리스트를 반환하는 프로그램을 작성하십시오.

파이썬 심화_list comprehension

- 파이썬만의 독특한 문법으로 간결한 코딩을 할 수 있음

1) for 문

0부터 9까지를 순서대로 가지고 있는 리스트를 만드세요.

```
[ ] list_1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[ ] list_2 = []
    for x in range(10) :
        list_2.append(x)
print(list_2)
```



```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

파이썬 심화_list comprehension

1) for 문

0부터 9까지를 순서대로 가지고 있는 리스트를 만드세요.(한줄로)

```
[ ] lc_1 = [x for x in range(10)]  
print(lc_1)
```



[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[저장할 값 **for** 원소 **in** 반복 가능한 객체]

파이썬 심화_list comprehension

1) for 문

Quiz 1) 구구단2단을 list comprehension을 이용하여 구현하고 리스트를 화면에 출력해보자.

Quiz 2) 다음의 문장을 분석해보자.

"코로나 바이러스를 예방하기 위해 사회적 거리두기를 실천합시다.
마스크를 끼고 손씻기를 생활화합시다."라는 문장을 띄어쓰기별로
분석하려고 한다. 띄어쓰기별로 문장을 나눈 후 각 요소의 길이를 리스
트에 저장하라.

(Hint : 띄어쓰기는 split 함수를 써라.)

파이썬 심화_list comprehension

2) for 문+ if 문

10부터 20 사이의 숫자들 중에서 짝수만을 담은 리스트를 만들어보자.
(List Comprehension을 사용하시오.)

```
[33] list_3 = []
      for x in range(10, 21) :
          if x % 2 == 0 :
              list_3.append(x)
      print(list_3)
```

```
↪ [10, 12, 14, 16, 18, 20]
```

파이썬 심화_list comprehension

2) for 문+ if 문

10부터 20 사이의 숫자들 중에서 짝수만을 담은 리스트를 만들어보자.

```
[34] lc_2 = [x for x in range(10, 21) if x % 2 == 0]
```

```
[35] lc_2
```

```
↳ [10, 12, 14, 16, 18, 20]
```

[저장할 값 **for 원소 **in** 반복 가능한 객체 **if 조건**]**

2) for 문+ if 문

Quiz 1) 1부터 10의 제곱수중, 50 이하인수만 리스트에 저장하라.

Quiz 2) 다음의 문장을 분석해보자.

"코로나바이러스를 예방하기 위해 사회적 거리두기를 실천합시다. 마스크를 끼고 손씻기를 생활화합시다."라는 문장을 띠어쓰기별로 분석하려고 한다. 띠어쓰기별로 문장을 나눈 후 각 요소의 길이가 5미만인 것들만 리스트에 저장하라.

파이썬 심화_list comprehension

- 2) for 문+ if 문
- 1부터 10 까지의 숫자들 중 홀수이면 제곱수를, 짝수이면 세제곱수를 담은 리스트를 만들어보자.

```
[40] list_4 = []
      for x in range(1, 11) :
          if x % 2 == 1 :
              list_4.append(x ** 2)
          else :
              list_4.append(x ** 3)
```

```
[41] list_4
```

```
↪ [1, 8, 9, 64, 25, 216, 49, 512, 81, 1000]
```

【 저장할 값 if 조건 else 저장할 값 for 원소 in 반복 가능한 객체】

파이썬 심화_list comprehension

- 3) for 문+ for 문

```
word_1 = 'Hello'  
word_2 = 'World'
```

```
result = [i+j for i in word_1 for j in word_2]  
result
```

```
[ 'Hw',  
  'Ho',  
  'Hr',  
  'Hl',  
  'Hd',  
  'ew',  
  'eo',  
  'er',  
  'el',  
  'ed',  
  'lw',  
  'lo',  
  'lr',  
  'll',  
  'ld',  
  'lw',  
  'lo',  
  'lr',  
  'll',  
  'ld',  
  'ow',  
  'oo',  
  'or',  
  'ol',  
  'od' ]
```

파이썬 심화_list comprehension

2) for 문+ if 문

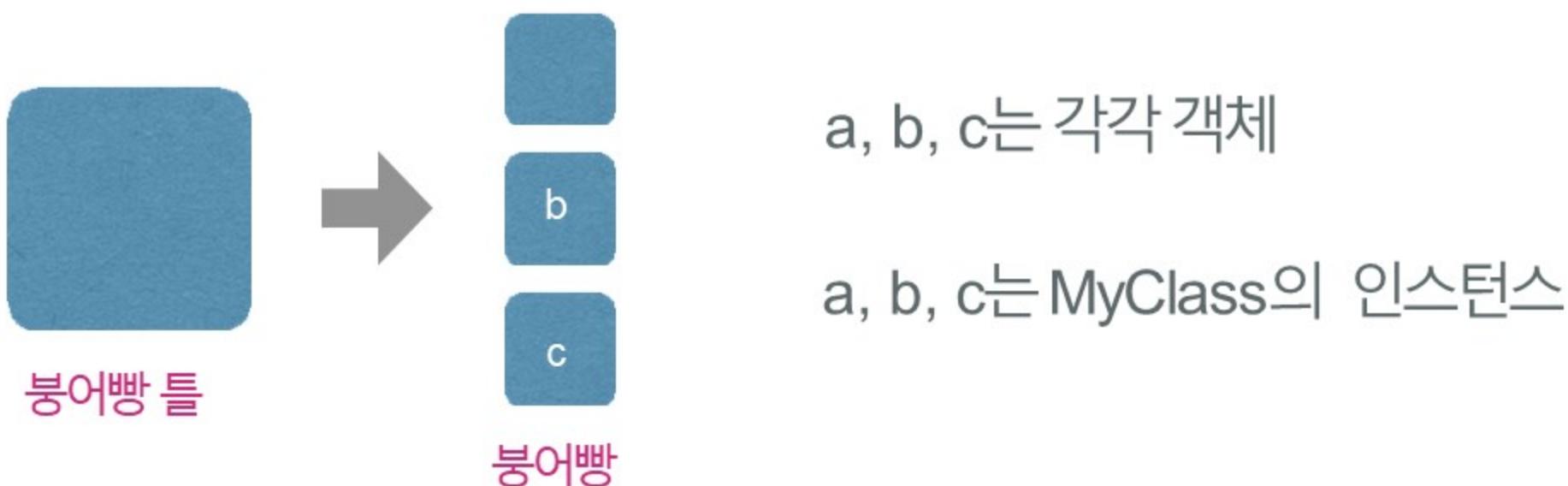
Quiz 1) 40 이하의숫자는5를 더하고, 40 초과의숫자는41로 바꾸어리스트로저장하고,리스트를출력하라.

Quiz 2) 컷트라인이60점일때, 사람이름과통과여부를리스트로담아서출력하라. 이름과통과여부는튜플로묶어있는자료이다.

```
[ ] students = {"보라돌이" : 61,  
                 "뚜비" : 35,  
                 "나나" : 78,  
                 "뽀" : 88}
```

파이썬 심화_class

- Class
 - 객체(Object)
 - 클래스(Class) : 객체를 만드는 구조/틀
 - 인스턴스(Instance) : 클래스가 실질적으로 객체를 만들었을 때 그 객체를 부르는 용어



파이썬 심화_class

class 클래스이름():
 클래스변수,
 메소드 인스턴스변수,
 메소드등 클래스의 구조와 내용

```
class 클래스 이름:  
    변수1  
    변수2  
    .....필요한 만큼 변수를 제공.....  
    def 메소드1(인수):  
        .....메소드의 처리.....  
    def 메소드2(인수):  
        .....메소드의 처리.....  
    필요한만큼 메소드를 제공.....
```

```
class MyClass():  
    class_var = '클래스 변수'  
  
    @classmethod  
    def class_method(cls):  
        print('클래스의 메소드')  
  
    def instance_method(self):  
        self.instance_var = '인스터스 변수'  
        print("인스턴스의 메소드")
```

파이썬 심화_class

- 인스턴스(instance)

클래스가 실질적으로 객체를 만들었을 때 그 객체를 부르는 용어

```
[5] class MyClass():
    pass
```

→ 클래스를 선언

```
▶ a = MyClass()
  b = MyClass()
  c = MyClass()
```

→ 함수처럼 콜(call)하여 사용

```
▶ print(a,b,c)
```

```
◀ <__main__.MyClass object at 0x7ff058193eb8>
  <__main__.MyClass object at 0x7ff058193e80>
  <__main__.MyClass object at 0x7ff058193ef0>
```

객체(object) 생성 →

- 인스턴스(instance)

클래스가 실질적으로 객체를 만들었을 때 그 객체를 부르는 용어

1. instance variable(인스턴스 변수)

인스턴스 고유의 변수

2. instance method(인스턴스 메소드)

인스턴스에 적용되는 함수

파이썬 심화

- 인스턴스(instance)

```
[1] class Account():
    def make_account(self):
        self.balance = 0

    def deposit(self, money) :
        self.balance += money

    def draw(self, money) :
        self.balance -= money
```

인스턴스

```
[2] a1 = Account()
```

self : 인스턴스를 위한 placeholder

→ 클래스를 선언

파이썬 심화_class

- 인스턴스(instance)

```
[1] class Account():
    def make_account(self) :
        self.balance = 0

    def deposit(self, money) :
        self.balance += money

    def draw(self, money) :
        self.balance -= money
```

```
[2] a1 = Account()
    a1.make_account()
    a1.deposit(1000)
    print(a1.balance)
```

→ 1000

인스턴스.변수

인스턴스.메소드

→ 위와 같은 방식으로 클래스 선언 시에
만들었던 변수들, 메소드들을 사용할 수 있음

파이썬 심화

- 인스턴스메소드(Instance Method)

주의 인스턴스 메소드의 매개변수 중 가장 첫번째는 인스턴스의 자리

C++	Python
<pre>class Person{ int age; public: Person(){ this->age = 17; } }</pre>	<p>[▶] <code>class person(): def __init__(self): self.age = 17</code></p> <p>[51] <code>p1 = person()</code></p> <p>[52] <code>p1.age</code></p>

파이썬 심화

- 인스턴스메소드(Instance Method)

주의 인스턴스 메소드의 매개변수 중 가장 첫번째는 인스턴스의 자리

```
[1] class TestClass1():
        def __init__(self, in1, in2):
            self.v1 = in1
            self.v2 = in2
```

인스턴스

```
[2] t_cls1 = TestClass1(10, 20)
```

인스턴스메소드의 매개변수

```
[3] t_cls1.v1
```

→ 파이썬은 인스턴스를 자동으로 넘겨줌

⇨ 10

- 인스턴스메소드(Instance Method)

주의 인스턴스 메소드의 매개변수 중 가장 첫번째는 인스턴스의 자리

```
[4] class TestClass2():
    def __init__(in1, in2):
        self.v1 = in1
        self.v2 = in2
```

```
[5] t_cls2 = TestClass2(10, 20)
```

```
-----  
[5] t_cls2 = TestClass2(10, 20)
-----  
TypeError                                         Traceback (most recent call last)
<ipython-input-5-32e19ab7347c> in <module>()
      1 t_cls2 = TestClass2(10, 20)
-----
```

TypeError: __init__() takes 2 positional arguments but 3 were given

SEARCH STACK OVERFLOW

- Instance Method(인스턴스 메소드)

- 1) `__init__`(초기화자): 인스턴스가 생성될 때, 객체의 초기 값을 설정하는 메소드로, 자동으로 호출됨
- 2) `__del__`(소멸자): 인스턴스가 소멸될 때, 자동으로 호출되는 메소드

```
[1] class TestClass1():
    def __init__(self, in1, in2):
        self.v1 = in1
        self.v2 = in2
```

```
[2] t_cls1 = TestClass1(10, 20)
```

```
[3] t_cls1.v1
```

→ 초기화자를 통해서 객체의 초기값을 자동으로 설정

파이썬 심화

- 인스턴스(instance)

```
[1] class Account():
    def make_account(self) :
        인스턴스변수 self.balance = 0

        def deposit(self, money) :
            self.balance += money

        def draw(self, money) :
            self.balance -= money
```

인스턴스 메소드

인스턴스

```
[2] a1 = Account()
    a1.make_account()
    a1.deposit(1000)
    print(a1.balance)
```

⇒ 1000



통장1에 1000원을 넣음

```
[3] a2 = Account()
    a2.make_account()
    a2.deposit(5000)
    print(a2.balance)
```

⇒ 5000



통장2에 5000원을 넣음

파이썬 심화

- 인스턴스(instance)

```
[1] class Account():
    def make_account(self) :
        인스턴스변수 self.balance = 0

        def deposit(self, money) :
            self.balance += money

        def draw(self, money) :
            self.balance -= money
```

클래스이름



Account.balance

```
-----  
AttributeError                                     Traceback (most recent  
<ipython-input-4-35da668b8893> in <module>()  
      ----> 1 Account.balance  
  
AttributeError: type object 'Account' has no attribute 'balance'
```



→ 통장끼리는 잔고를 공유하지 않음

파이썬 심화

● 인스턴스(instance)

```
class test():
    name = '아무나'
    age = 0
    def __init__(self, name, age):
        print('생성자 호출!')
        self.name = name
        self.age = age
    def __del__(self):
        print('소멸자 호출!')
    def info(self):
        print('나의 이름은', self.name, '입니다!')
        print('나이는', self.age, '입니다!')
```

```
r = test('류영표', 7)
```

생성자 호출!

```
r.info()
```

나의 이름은 류영표 입니다!
나이는 7 입니다!

```
[14] test('류영표', 7)
```

생성자 호출!
<__main__.test at 0x7ff089935e80>

```
[15] test.info()
```

```
-----  
TypeError Traceback (most recent call last)  
<ipython-input-15-a72a0779be47> in <module>()  
----> 1 test.info()
```

TypeError: info() missing 1 required positional argument: 'self'

파이썬 심화

- 인스턴스(instance)

```
▶ class ClassExample:  
    def __init__(self, name, age): # 생성자(Constructor) 생성 시 자동 수행  
        self.name = name  
        self.age = age + 5 # 변수 생성 후 파라미터로 보낸 값 할당  
    print("[생성자] self와 변수 비교 " + str(self.age) + " age : " + str(age))  
    print("[생성자] 이름 : " + self.name + " 나이 : " + str(self.age))  
  
    def ten_year_call(self, val): # 함수  
        return self.age + 10 # 반환
```

```
[2] a = ClassExample("철수", 20) # 객체화 (ClassExample의 정보를 a에 담음)  
print(a.age) # a의 age 출력  
print(a.ten_year_call(50)) # a의 ten_year_call 메소드 수행
```

```
[생성자] self와 변수 비교 25 age : 20  
[생성자] 이름 : 철수 나이 : 25  
25  
35
```

파이썬 심화

- 객체 슬라이싱과 `__getitem__`

```
class CustomNumbers:  
    def __init__(self):  
        self._numbers = [n for n in range(1, 11)]  
  
    a = CustomNumbers()
```

a[2:5]

a._numbers[2:5]
[3, 4, 5]

```
TypeError                                     Traceback (most recent call last)  
<ipython-input-37-cd92fdb5c05> in <module>  
----> 1 a[2:5]
```

TypeError: 'CustomNumbers' object is not subscriptable

파이썬 심화

- 객체 슬라이싱과 `__getitem__`
- 인스턴스 변수에 직접 접근하지 말고 객체 자체를 통해서 슬라이싱을 구현하기 위해서는 `__getitem__` 특별 메서드를 정의해야 함.

```
class CustomNumbers:  
    def __init__(self):  
        self._numbers = [n for n in range(1, 11)]  
  
    def __getitem__(self, idx):  
        return self._numbers[idx]  
  
a = CustomNumbers()  
a[2:7]  
# [3, 4, 5, 6, 7]
```

파이썬 심화

- 메소드 오버로딩(method overriding)

```
class Adder{  
    static int add(int a,int b)  
    {  
        return a+b;  
    }  
    static int add(int a,int b,int c)  
    {  
        return a+b+c;  
    }  
}
```

자바는 함수이름이 같아도 허용.

```
class Korea:  
  
    def __init__(self, name,population, captial):  
        self.name = name  
        self.population = population  
        self.capital = captial  
  
    def show(self):  
        print(  
            """  
            국가의 이름은 {} 입니다.  
            국가의 인구는 {} 입니다.  
            국가의 수도는 {} 입니다.  
            """.format(self.name, self.population, self.capital)  
        )  
  
    def show(self, abc):  
        print('abc :', abc)  
  
>>> from inheritance import *  
>>> a = Korea('대한민국',50000000, '서울')  
>>> a.show()  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: show() missing 1 required positional argument: 'abc'
```

- Class Variable과 Class Method

1. class variable(클래스 변수)

클래스와 인스턴스 전체가 공유하는 변수

2. class method(클래스 메소드)

클래스와 인스턴스 전체가 공유하는 함수

인스턴스와 상관없이 사용(인스턴스가 없어도 사용 가능) 클래스와 인스턴스 전체에 공유됨

파이썬 심화

- Class Variable과 Class Method

1. class variable(클래스 변수)

클래스와 인스턴스 전체가 공유하는 변수

2. class method(클래스 메소드)

클래스와 인스턴스 전체가 공유하는 함수

```
[8] class MyClass2():
    class_var = '클래스 변수'

    @classmethod
    def class_method(cls):
        print('클래스의 메소드')
```

데코레이터(decorator) : wrapping을
통해 특정 코드를 재사용할 수 있게
해주는 역할

cls : 클래스를 위한 placeholder
(self와 유사한 역할)

→ 파이썬은 클래스 또한 자동으로 넘겨줌

• Class Variable과 Class Method

```
[8] class MyClass2():
    class_var = '클래스 변수'

    @classmethod
    def class_method(cls):
        print('클래스의 메소드')
```

인스턴스.변수

인스턴스.메소드

→ 위와 같은 방식으로 클래스 변수와
클래스 메소드를 사용할 수 있음

클래스이름.변수

클래스이름.메소드

```
[10] e = MyClass2()
      print(e.class_var)
      print(MyClass2.class_var)
```

⇨ 클래스 변수
클래스 변수

[▶] e.class_method()
MyClass2.class_method()

⇨ 클래스의 메소드
클래스의 메소드

파이썬 심화

• Class Variable과 Class Method

```
[6] class Account2():
    bank = '모두은행'
    total = 0
```

클래스 변수

```
@classmethod
def merge(cls, acc1, acc2):
    cls.total = acc1.balance + acc2.balance
    print("당신의 재산은 %d" %cls.total)
```

```
def __init__(self):
    self.balance = 0
```

클래스 메소드

```
def deposit(self, money):
    self.balance += money
```

```
def draw(self, money):
    self.balance -= money
```

```
[7] Account2.bank
```

☞ '모두은행'



은행

내자산

```
[8] b1 = Account2()
b1.deposit(4000)
print(b1.balance)
print(b1.bank)
```



☞ 4000
모두은행

```
[9] b2 = Account2()
b2.deposit(7000)
print(b2.balance)
print(b2.bank)
```



☞ 7000
모두은행

```
[10] Account2.merge(b1, b2)
```

☞ 당신의 재산은 11000

```
[11] Account2.total
```

☞ 11000

파이썬 심화

```
[22] class Hotel():
    def __init__(self):
        self.room = []

    def add_person(self, name):
        self.room.append(name)
```

```
[23] r1 = Hotel()
      r2 = Hotel()

      r1.add_person('뽀로로')
      r2.add_person('펭수')
```

```
[26] Hotel.room
```

```
□ -----  
AttributeError                                     Traceback (most recent call last)  
<ipython-input-26-b87892c63c8a> in <module>()  
----> 1 Hotel.room
```

```
AttributeError: type object 'Hotel' has no attribute 'room'
```

파이썬 심화

```
[27] class GuestHouse():
    guest = []

    def __init__(self):
        self.room = []

    @classmethod
    def check_in(cls, name):
        cls.guest.append(name)

    def add_person(self, name):
        self.check_in(name)
        self.room.append(name)
```

```
[28] r3 = GuestHouse()
r4 = GuestHouse()

r3.check_in('뚝딱이')
r4.check_in('뽕뽕이')
```

```
[29] GuestHouse.guest
```

```
↳ ['뚝딱이', '뽕뽕이']
```

```
[30] r3.room
```

```
↳ []
```

```
[31] r4.room
```

```
↳ []
```

```
[32] r3.add_person('홍길동')
r4.add_person('텔레토비')
```

```
[33] r3.room
```

```
↳ ['홍길동']
```

```
[34] r4.room
```

```
↳ ['텔레토비']
```

```
[35] GuestHouse.guest
```

```
↳ ['뚝딱이', '뽕뽕이', '홍길동', '텔레토비']
```

- 클래스 상속(inheritance)
 - 상속: 부모의 클래스를 물려받음
→ 부모 클래스가 가지고 있는 함수/변수를 그대로 사용할 수 있음
 - 상속의 장점: 기존 클래스를 변형하지 않고 추가/변경이 가능함

상속받고자 하는 클래스(부모 클래스(superclass))

```
[ ] class MyPhone2(MyPhone):  
    def has_case(self, val=False) : 자식 클래스(subclass)의 새로운 메소드  
        self.case = val
```

• 클래스상속(inheritance)

```
[ ] class MyPhone():
    def __init__(self, model, color):
        self.model = model
        self.color = color

    def set_name(self, name):
        self.user = name
        print("사용자의 이름은 : %s" %self.user)

    def set_number(self, number):
        self.number = number

[ ] class MyPhone2(MyPhone):
    def has_case(self, val=False):
        self.case = val
```

```
[48] p2 = MyPhone2('iphone', 'red')
```

```
[49] p2.set_name("MJ")
```

↳ 사용자의 이름은 : MJ

부모 클래스의 메소드 사용 가능

```
[50] p2.has_case(True)
```

```
[51] p2.case
```

↳ True

자식 클래스의 새로운 메소드도 사용 가능

- 메소드 오버라이딩(overriding)
 - 상속을 받은 자식클래스가 상속해 준 부모 클래스의 메소드를 변형하는 방법
 - 원본(부모클래스)의 소스코드는 그대로 유지한채 소스코드를 확장, 개인화 할 수 있다는 장점이 있음

파이썬 심화

```
class Calculate:  
    type = 'low'  
  
    def __init__(self,n1,n2):  
        self.n1 = n1  
        self.n2 = n2  
  
    def sum(self):  
        print(self.n1+self.n2)
```

```
c = Calculate(4,2)  
print(c.type)  
c.sum()
```

low
6

```
class Calculate_1(Calculate):  
    type = 'high'  
  
    def sub(self):  
        print(self.n1-self.n2)  
  
    def mul(self):  
        print(self.n1*self.n2)
```

```
c1 = Calculate_1(4,2)  
print(c1.type)  
c1.sum()  
c1.sub()  
c1.mul()
```

high
6
2
8

파이썬 심화

- 메소드 오버라이딩(overriding)

```
[46] class MyPhone():
    def __init__(self, model, color):
        self.model = model
        self.color = color

    def set_name(self, name):
        self.user = name
        print("사용자의 이름은 : %s" %self.user)

    def set_number(self, number):
        self.number = number
```

```
[52] p1 = MyPhone('iphone', 'red')
    p1.set_number("010-xxxx-xxxx")
```

```
[55] class MyPhone3(MyPhone) :
    def set_number(self, num):
        self.number = num
        print("이 핸드폰의 번호는 : %s" %self.number)
```

```
[56] p3 = MyPhone3('iphone', 'red')
    p3.set_number("010-xxxx-xxxx")
```

⇨ 이 핸드폰의 번호는 : 010-xxxx-xxxx

1번

Human이라는 클래스를 만들어보자.

- Human 클래스는 아래와 같은 특징이 있다.
- Human 클래스는 인스턴스 생성 시 birth_date, sex, nation을 변수로 가지고 있다.
- give_name은 인스턴스 메소드로 이름을 input으로 받아서 name이라는 인스턴스 변수로 저장하고 화면에 이름을 출력하는 역할을 하는 함수이다.
- can_sing이라는 함수는 True/False값을 input으로 받으며, 참이면 “Sing a song”을 화면에 출력하는 함수이다.

2번

Human이라는 클래스를 상속하는 Child라는 클래스를 만들어보자.

- Child의 클래스에는 아래와 같은 변수와 함수들이 추가된다.
- 눈동자 색깔을 나타내는 변수 eye를 인스턴스 선언시 사용할 수 있게 추가해보자.
- Child의 클래스는 노래하는 능력이 없다. 따라서 can_sing이라는 메소드가 호출되면 무조건 “Can't Sing”이고 출력하도록 바꿔보자.
- Child는 노래 대신 춤을 출 수 있다. can_dance라는 메소드가 호출되면 “Dance Time!”을 출력하도록 메소드를 작성해보자.

- 모듈(module)

- 1 모듈

- 함수나 변수, 클래스 등을 가진 파일(.py) / jupyter notebook(.ipynb)
- 모듈 안에는 함수, 클래스 또는 변수들이 정의되어 있음
- 파이썬은 많은 표준 라이브러리 모듈을 제공함

```
[ ] import math
```

```
[ ] math.factorial(4)
```



24

```
[ ] from math import factorial
```



24

import 모듈 이름

from 모듈 이름

import 변수/함수/클래스 이름

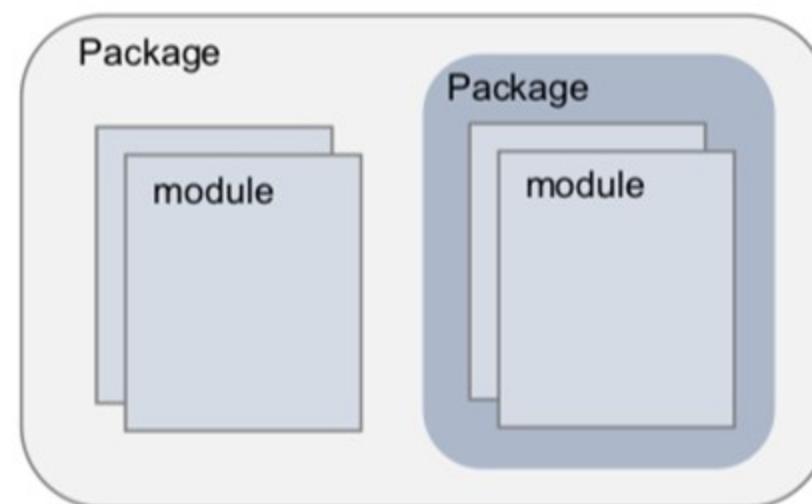
- 모듈(module)

- 2. 패키지(package)

- 모듈을 효율적으로 관리하기 위한 모듈의 상위 개념
 - 공동 작업이나 코드의 유지 보수 등에 유리

```
[ ] import 패키지.모듈  
      import 패키지.모듈.변수  
      import 패키지.모듈.함수  
      import 패키지.모듈.클래스
```

```
[ ] from 패키지.모듈 import 변수/함수/클래스
```



예외처리

프로그램의 제어 흐름을 조정하기 위해 사용됨.

1. `try` : 실행코드(오류가 발생하나 보자!)
2. `except` : 예외처리 코드(오류가 발생하면 이걸 실행하자!)
3. `else` : 예외처리할 오류가 없을때 실행되는 코드
4. `finally` : 오류 발생여부 상관 없이 무조건 실행되는 코드
5. `raise` : 오류를 일부러 발생시키기

예외처리

```
int('number')
```

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-1-3146e5e02b83> in <module>()  
----> 1 int('number')
```

```
ValueError: invalid literal for int() with base 10: 'number'
```

SEARCH STACK OVERFLOW

```
try:  
    int('number')  
except:  
    print('Not number!')
```

```
Not number!
```

```
try:  
    int('number')  
except ValueError as m:  
    print(m)
```

```
invalid literal for int() with base 10: 'number'
```

예외처리

```
try:  
    int('11')  
except ValueError as m:  
    print(m)  
else:  
    print('No Error!')
```

No Error!

```
for _ in ['11', 'number']:  
    try:  
        int(_)  
    except ValueError as m:  
        print(m)  
    else:  
        print('No Error!')  
finally:  
    print('Finally Code')  
print('##')
```

No Error!
Finally Code

invalid literal for int() with base 10: 'number'
Finally Code

예외처리

```
try:  
    int('11')  
except ValueError as m:  
    print(m)  
else:  
    print('No Error!')
```

No Error!

```
for _ in ['11', 'number']:  
    try:  
        int(_)  
    except ValueError as m:  
        print(m)  
    else:  
        print('No Error!')  
finally:  
    print('Finally Code')  
print('##')
```

No Error!
Finally Code

invalid literal for int() with base 10: 'number'
Finally Code

에러를 직접 일으키는 방법

```
# 190이 넘는 학생을 발견하면 반복을 종료한다.  
school = {'1반' : [150, 156, 179, 191, 199], '2반' : [150, 195, 179, 191, 199]}  
  
try:  
    for class_number, students in school.items():  
        for student in students:  
            if student > 190:  
                print(class_number, '190을 넘는 학생이 있습니다.')  
                # break # 바로 상위 for문은 종료되지만 최고 상위 for문은 종료되지 않는다.  
                raise StopIteration  
            # 예외가 try 문 안에 있지 않으면 에러 발생시 프로그램이 멈춘다.  
except StopIteration:  
    print('정상종료')
```

1반 190을 넘는 학생이 있습니다.

정상종료

```
a = 10  
  
# True 인 경우  
print("1. assert True 인 경우")  
assert a == 10, "a is not 10"  
  
print("끝1")  
# False 인 경우  
print("2. assert False 인 경우")  
assert a == 999, "a is not 999"  
  
print("끝2")
```

1. assert True 인 경우

끝1

2. assert False 인 경우

```
AssertionError                                     Traceback (most recent call last)  
<ipython-input-11-2b467e8c26a9> in <module>()  
      10 # False 인 경우  
      11 print("2. assert False 인 경우")  
--> 12 assert a == 999, "a is not 999"  
      13  
      14 print("끝2")
```

AssertionError: a is not 999

Thank you.

파이썬 기초

ryp1662@gmail.com

Copyright © "Youngpyo Ryu" All Rights Reserved.
This document was created for the exclusive use of "Youngpyo Ryu".
It must not be passed on to third parties except with the explicit prior consent of "Youngpyo Ryu".