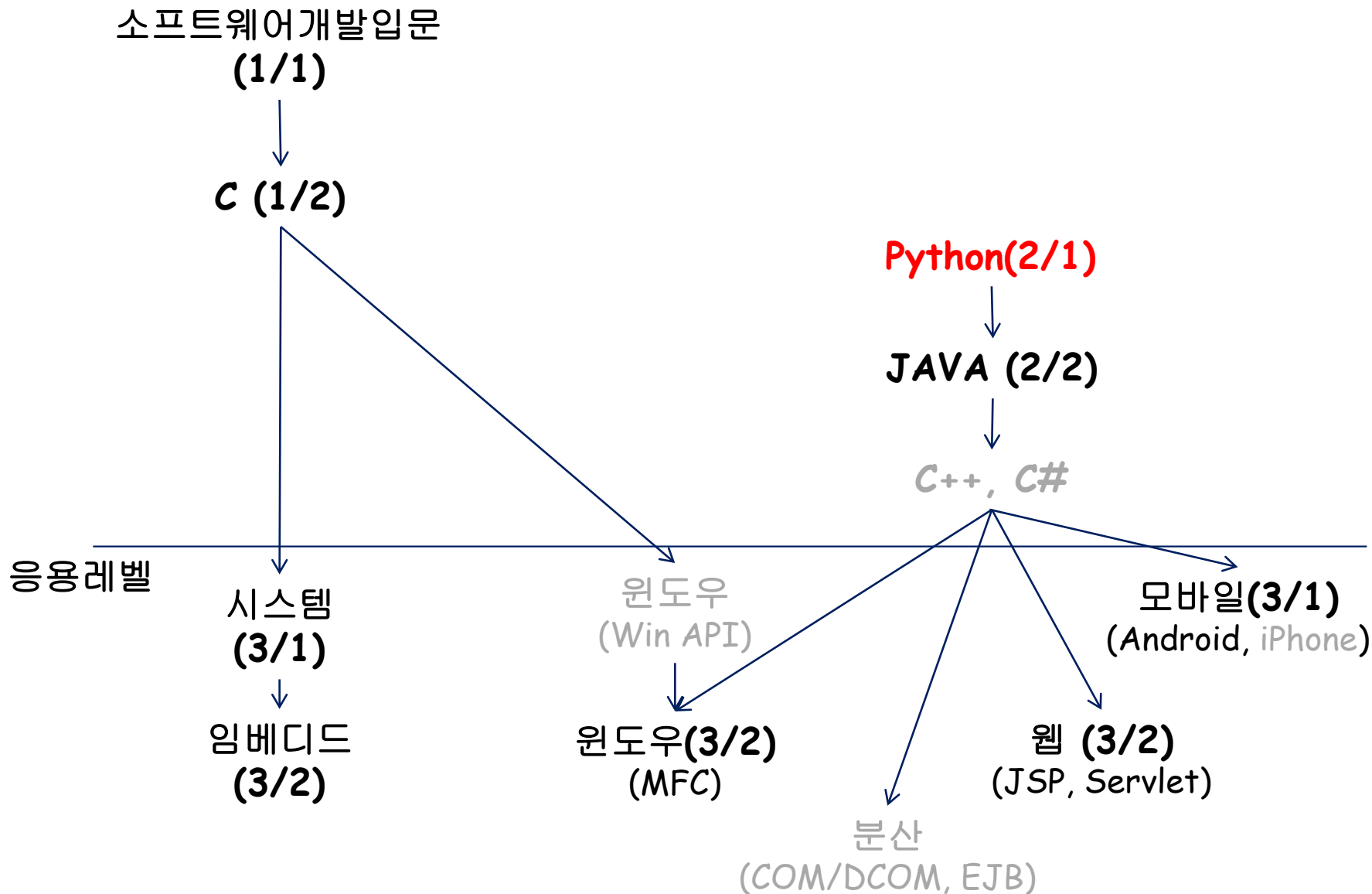


## <절차적 프로그래밍>

## <객체지향 프로그래밍>



# 수업계획서

## □ 주교재 & 참고교재

- ▣ 파이썬 **Express**, 천인국, 생능출판, 2020
- ▣ 파이썬에 참 좋은 PyCharm, 테리엇, 비제이퍼블릭, 2021
- ▣ 파이썬알고리즘 인터뷰, 박상길, 책만, 2020

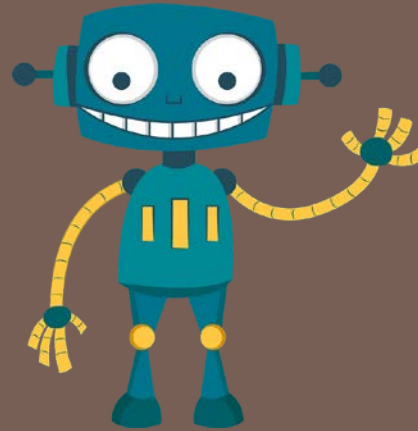
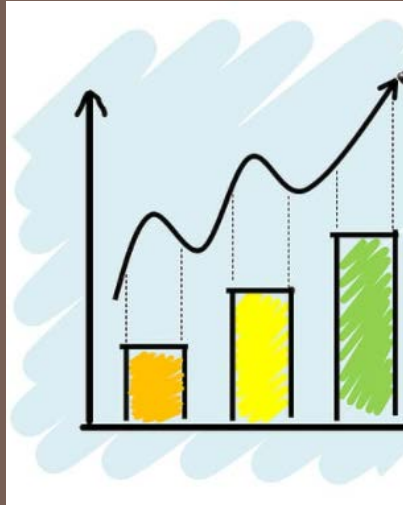
## □ 평가

- ▣ 중간시험 : 30%, 기말시험 : 30%, 과제 : 20%, 출석 : 20%

## □ 카톡 오픈채팅방

- ▣ LMS 공지사항에서 접속 링크 조회 가능
- ▣ 실명으로 참여할 것

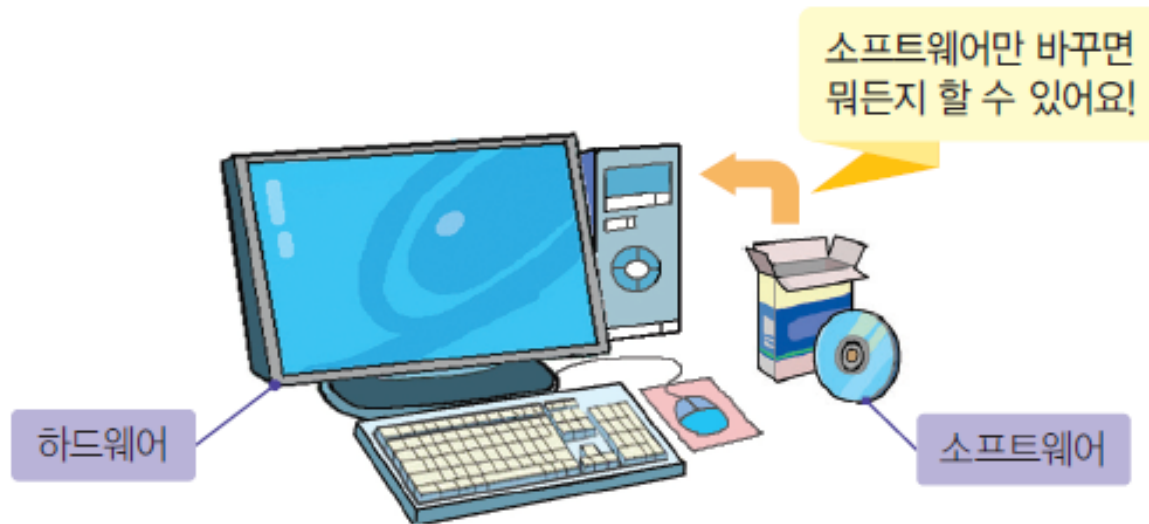
# 파이썬 익스프레스



## 1장 프로그래밍과 파이썬 소개

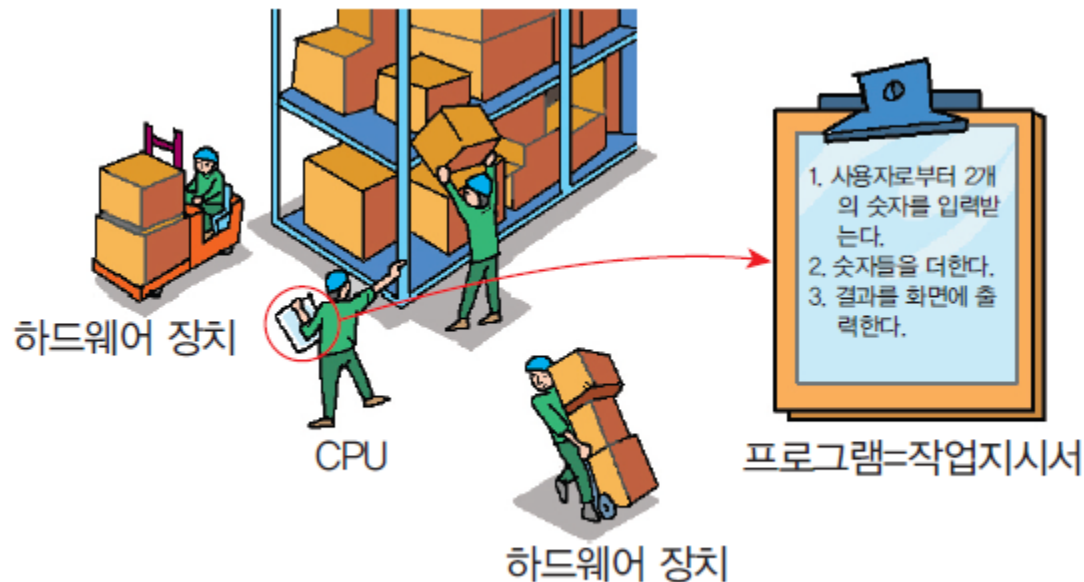
# 컴퓨터가 많이 사용되는 이유

- 컴퓨터는 대단히 유연한 기계
- 컴퓨터로 리포트를 작성할 수도 있지만, 게임도 할 수 있다



# 컴퓨터 프로그램

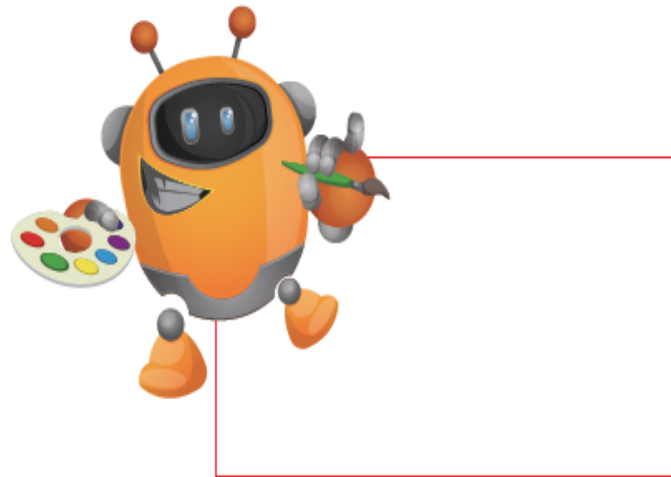
- 컴퓨터에 일을 시키려면 인간이 컴퓨터에게 자세한 **명령어 (instruction)**들의 리스트를 주어야 한다.
- **프로그램 (program)** : 컴퓨터가 수행할 명령어를 적어놓은 문서



# 명령어들의 예

- 붓을 들고 있는 로봇에게 사각형을 그리게 하는 작업은 다음과 같은 지시사항들로 이루어질 수 있다.

- ▶ 100 픽셀만큼 앞으로 이동한다.
- ▶ 90도 회전한다.
- ▶ 100 픽셀만큼 앞으로 이동한다.
- ▶ 90도 회전한다.
- ▶ 100 픽셀만큼 앞으로 이동한다.
- ▶ 90도 회전한다.
- ▶ 100 픽셀만큼 앞으로 이동한다.



# 우리는 왜 프로그래밍에 대하여 알아야 할까?

- 우리가 어떤 일을 하든지 상관없이 프로그래밍은 필수적인 기술이 되었다. 이와 학생들에게는 물론, 인문사회계 학생들도 프로그래밍에 대하여 어느 정도는 알아야 한다

저는 인문계인데도  
프로그래밍에 대하여  
알아야 하나요?



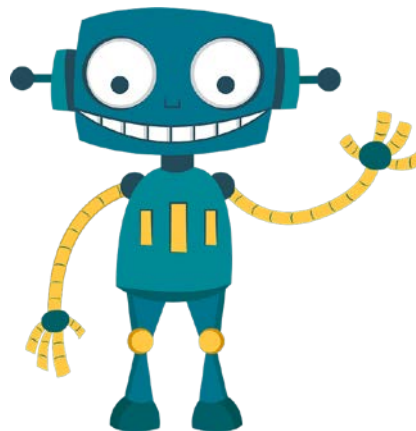
인문계 출신들이 주로 하는 각종  
업무도 파이썬을 이용하여  
자동화하는 경우가 많습니다.



앞으로 단순 반복 작업은  
컴퓨터가 맡게 될 것입니다.

# 우리는 왜 프로그래밍에 대하여 알아야 할까?

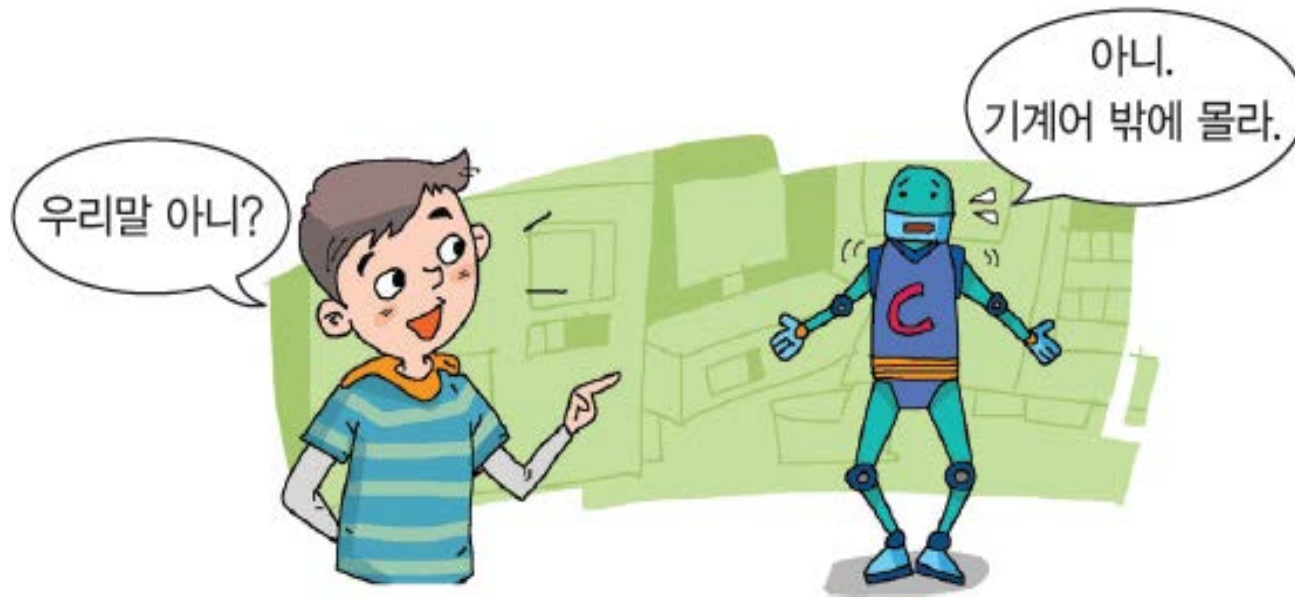
- 보고서 쓰는 인공지능 로봇
- 인공지능 알고리즘 주식 매매
- 로봇 프로세스 자동화(RPA)





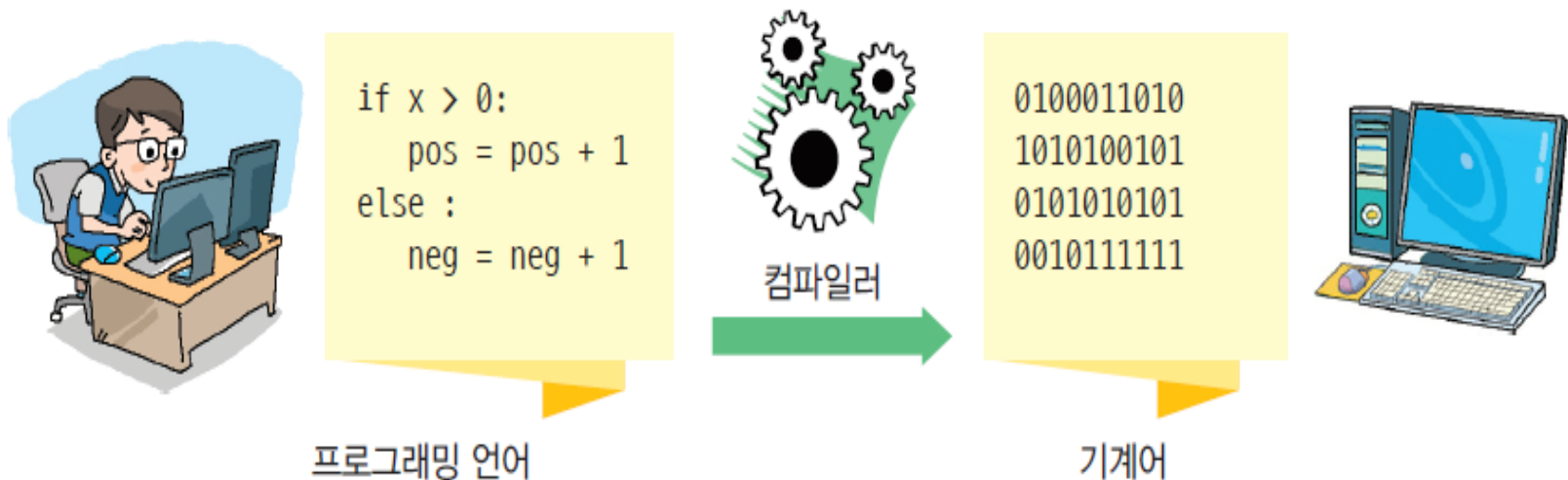
# 프로그래밍 언어

- 컴퓨터는 사람의 언어를 이해할 수 없다!
- “프로그래밍 언어”는 컴퓨터가 이해하는 언어이다.



# 컴파일러(인터프리터)

- 인간이 프로그래밍 언어를 배워서 프로그램을 작성하면 **컴파일러(또는 인터프리터)**라고 하는 통역 소프트웨어가 프로그램을 기계어로 바꾸어준다.



# 프로그래밍 언어

- 프로그램은 ‘프로그래밍 언어’로 작성된다. 프로그램을 만드는 사람을 ‘**프로그래머**’라고 한다.



→  
프로그래밍 언어

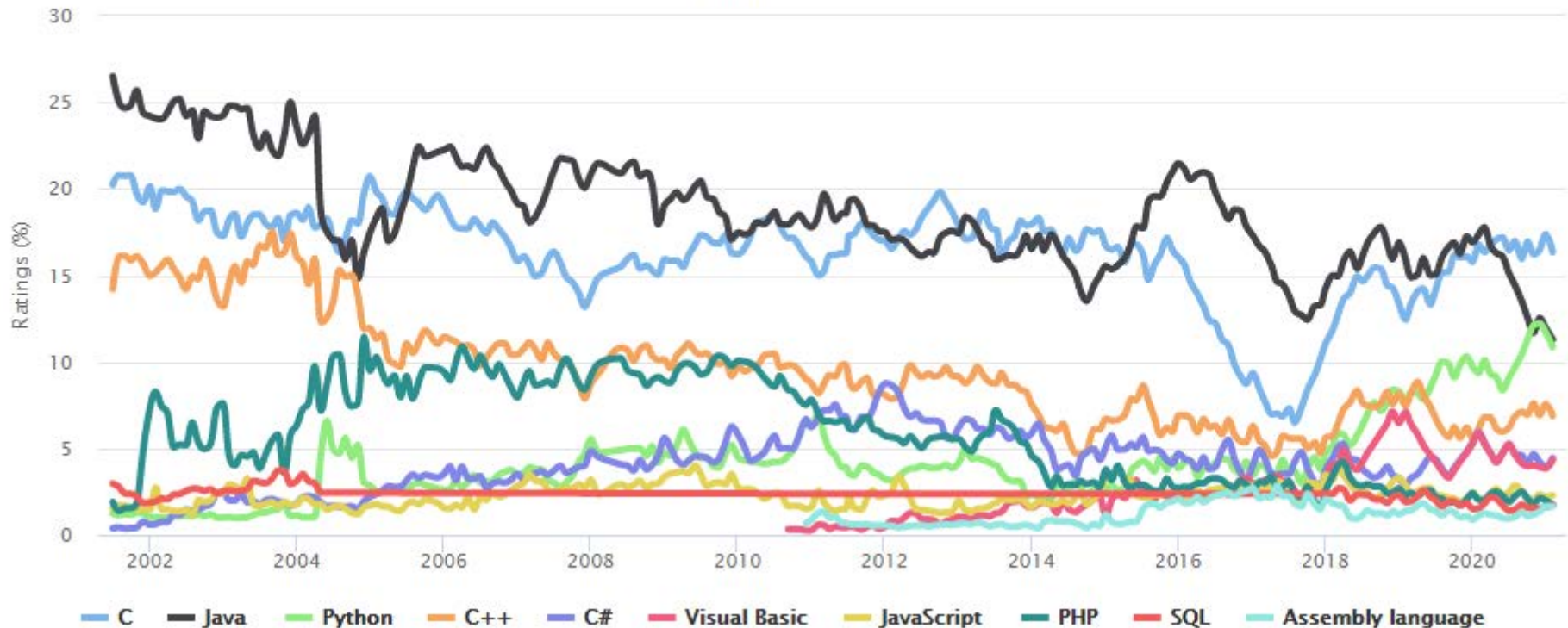


# 프로그래밍 언어의 종류

- 많이 사용되는 언어들에는 'Python', 'Java', 'C' 등이 있다.

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# 파이썬(Python)

- 1991년에 귀도 반 로섬(Guido van Rossum)이 개발한 대화형 프로그래밍 언어



# 파이썬의 특징

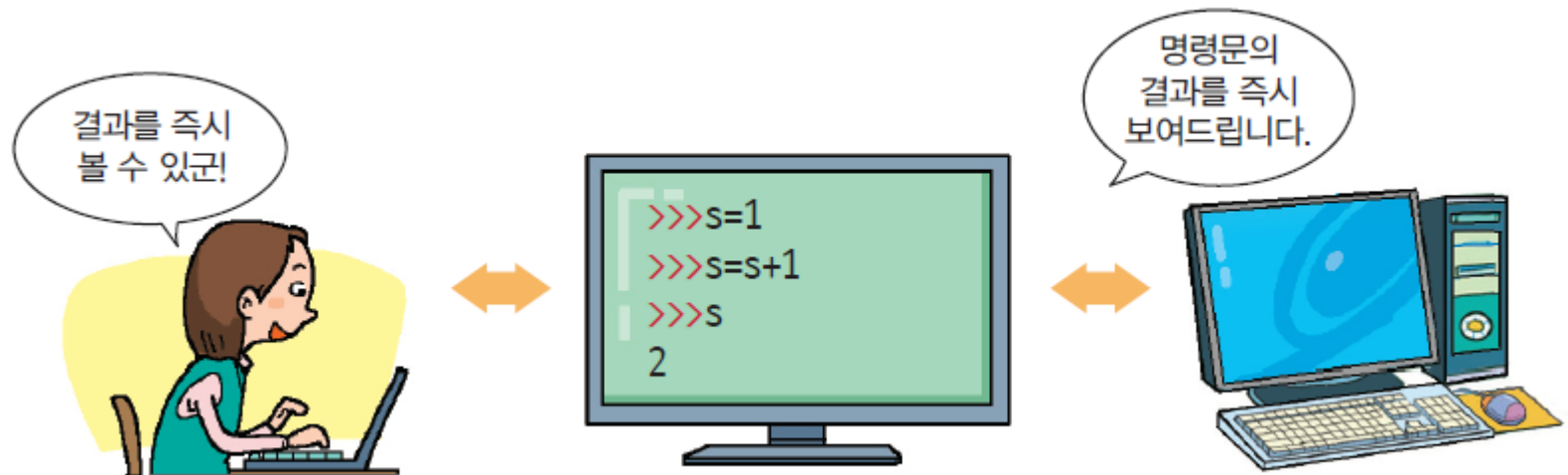
- 생산성이 뛰어나다.
- 간결하면서도 효율적인 프로그램을 빠르게 작성

C 언어	파이썬
<pre>#include &lt;stdio.h&gt;  int main(void) {     printf("Hello World! \n");     return 0; }</pre>	<pre>print("Hello World!")</pre>

# 파이썬의 특징

## □ 인터프리터 언어

- ▣ 파이썬 프로그래머는 자신이 작성한 명령문의 결과를 즉시 볼 수 있기 때문에 초보 프로그래머한테는 아주 바람직



# 파이썬의 특징

- 라이브러리가 풍부
- 라이브러리 설치가 쉽다.

matplotlib

 Keras  
A deep learning library

  
OpenCV

파이썬의 막강한  
라이브러리

  
Requests  
HTTP for Humans

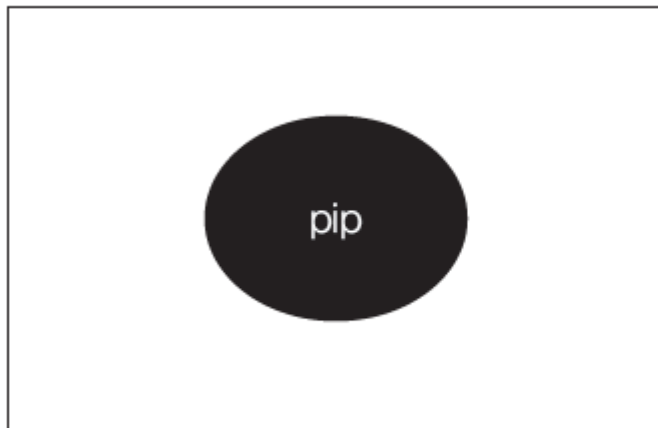
  
scikit  
learn

BeautifulSoup  

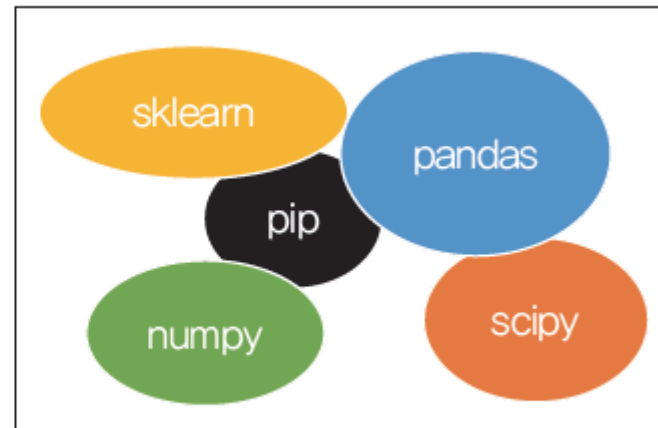



# 아나콘다(Anaconda)

- 인기 있는 라이브러리가 거의 모두 포함된 배포판



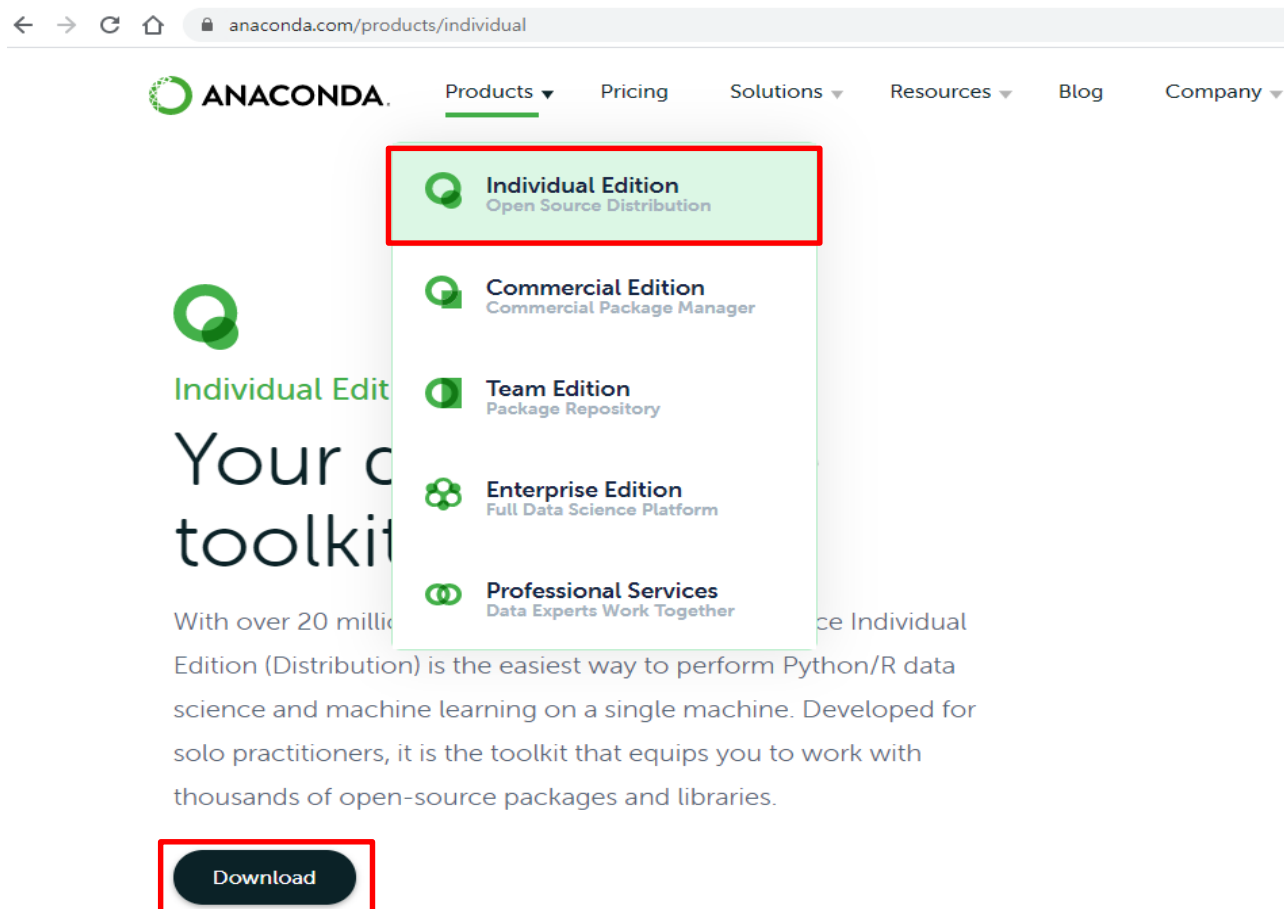
파이썬



아나콘다

# 아나콘다 다운로드

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



# 아나콘다 다운로드

## Anaconda Installers

Windows 

Python 3.8

64-Bit Graphical Installer (457 MB)


32-Bit Graphical Installer (403 MB)

MacOS 

Python 3.8

64-Bit Graphical Installer (435 MB)

64-Bit Command Line Installer (428 MB)

Linux 

Python 3.8

64-Bit (x86) Installer (529 MB)

64-Bit (Power8 and Power9) Installer (279 MB)


다운로드 받은 파일을 더블 클릭하여 실행

# 아나콘다 vs 미니콘다

미니콘다	아나콘다
- 개별 패키지를 각각 설치하는데 거부감이 없거나	- 콘다 혹은 파이선에 익숙하지 않거나
- 아직 사용여부를 모르는 패키지를 설치하는데 디스크 공간이 부족하거나 시간이 아까워서	- 파이선과 함께 한번에 주요 패키지를 자동적으로 설치하고 싶거나
- 빠르게 파이선과 콘다를 설치해서 사용해보고 싶다면	- 디스크 공간이 충분하다면 (3기가 이상)

# Miniconda

다운로드: <https://docs.conda.io/en/latest/miniconda.html>

 **Conda**  
latest

[Conda](#)  
[Conda-build](#)

**Miniconda**  
[Windows installers](#)  
[MacOSX installers](#)  
[Linux installers](#)  
[Installing](#)  
[Other resources](#)

[Help and support](#)  
[Contributing](#)  
[Conda license](#)

[Docs](#) » [Miniconda](#)

[Edit on GitHub](#)

## Miniconda

Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, packages they depend on, and a small number of other useful packages, including pip, zlib and a few others. Use the `conda install command` to install 720+ additional conda packages from the Anaconda repository.

[See if Miniconda is right for you.](#)

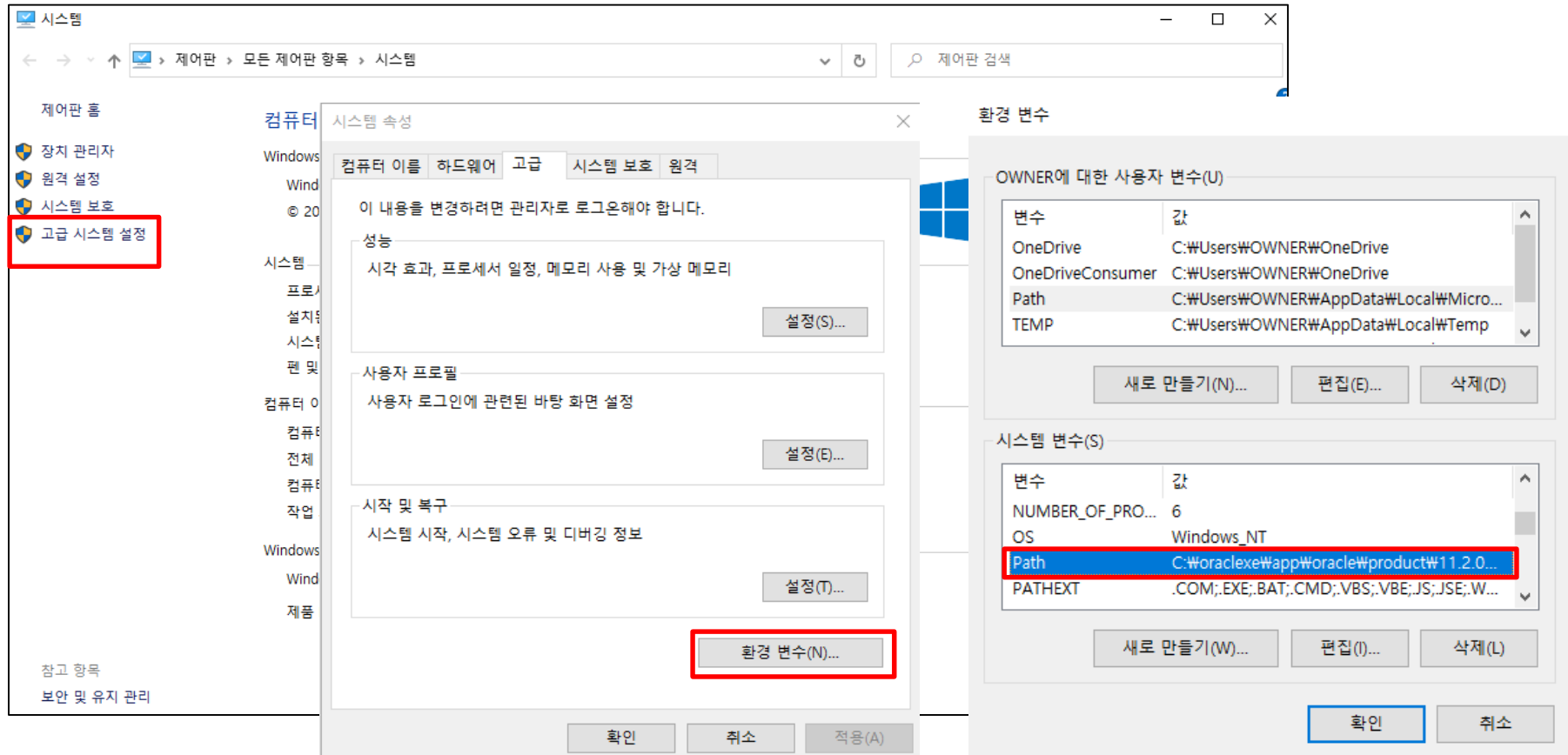
## Windows installers

*Windows*

Python version	Name	Size	SHA256 hash
Python 3.8	Miniconda3 Windows 64-bit	57.0 MiB	4fa22bba0497babb5b6608cb8843545372a99f5331c8120099ae1d803f627c61
	Miniconda3 Windows 32-bit	54.2 MiB	9c2ef76bae97246c85c206733ca30fd1feb8a4b3f90a2a511fea681ce7ebc661
Python 2.7	Miniconda2 Windows 64-bit	54.1 MiB	6973025404832944e074bf02bda8c4594980eed4707bb51baa8fbdab4bf326c
	Miniconda2 Windows 32-bit	47.7 MiB	c8049d26f8b6b954b57bcd4e99ad72d1ffa13f4a6b218e64e641504437b2617b

# 환경 변수 설정

- 바탕화면 → “내 PC” 우클릭 → 속성 → 고급 시스템 설정  
→ 환경변수 → 시스템변수에서 “Path” 검색 후 편집 클릭



# 환경 변수 설정

→ “새로만들기” 클릭 후 미니콘다 설치 주소인 아래의 4개 경로 추가

C:\Users\OWNER\miniconda3

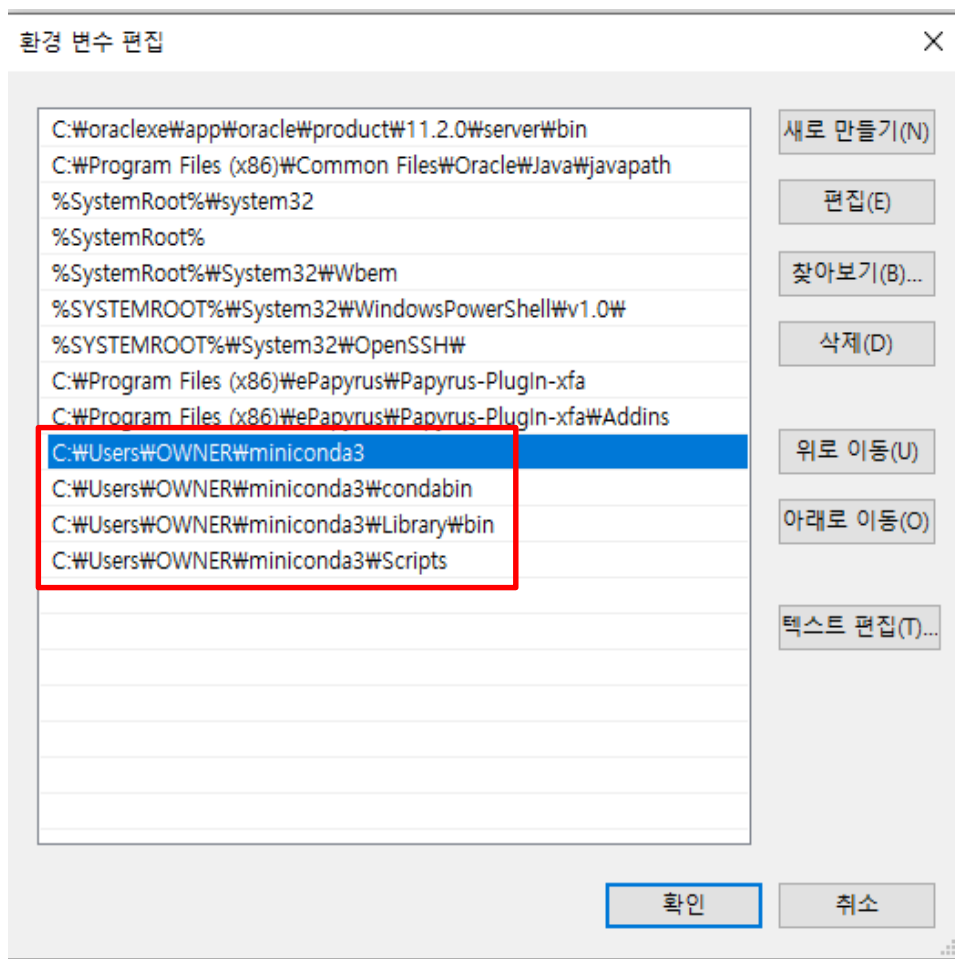
C:\Users\OWNER\miniconda3\condabin

C:\Users\OWNER\miniconda3\Library\bin

C:\Users\OWNER\miniconda3\Scripts

## <참고>

위에서 OWNER 부분은 실습 PC의  
어떤 계정으로 로그인 했는가에 따라  
다를 수 있음



# 환경 변수 설정

- 환경변수 설정이 제대로 되었다면 명령프롬프트 창에서 **python** 입력하면 아래와 같이 실행됨

```
명령 프롬프트 - python
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\OWNER>python
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

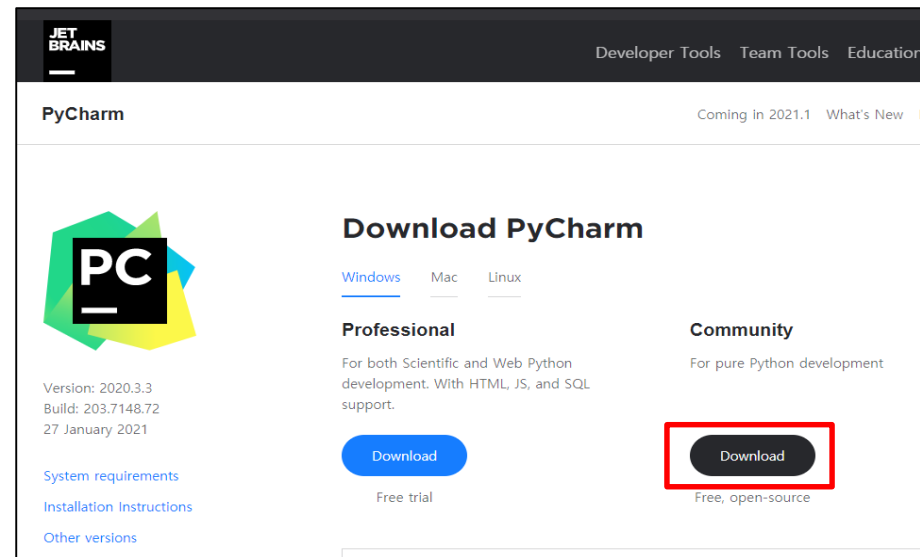
Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

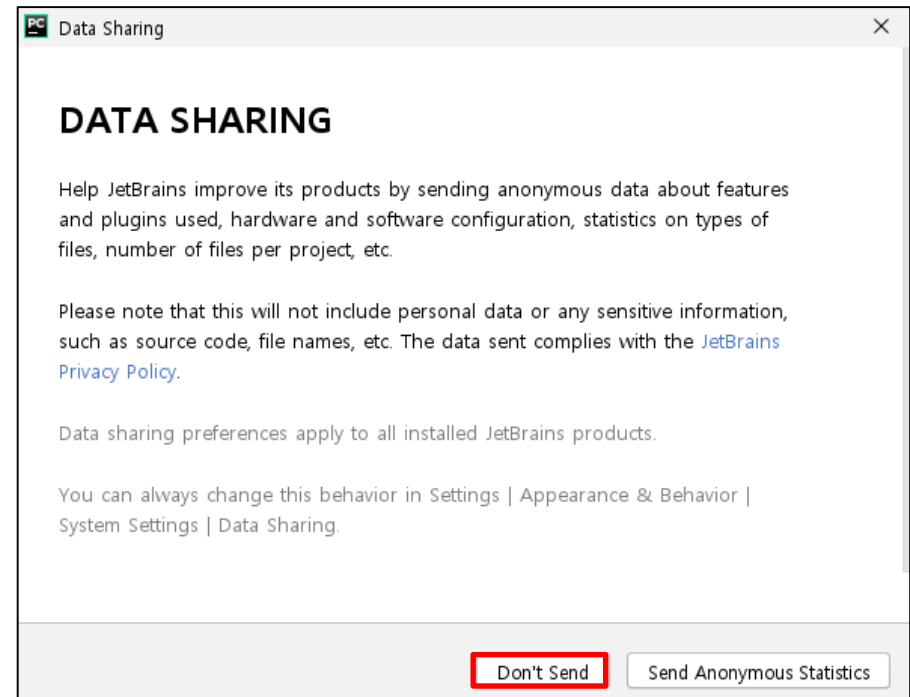
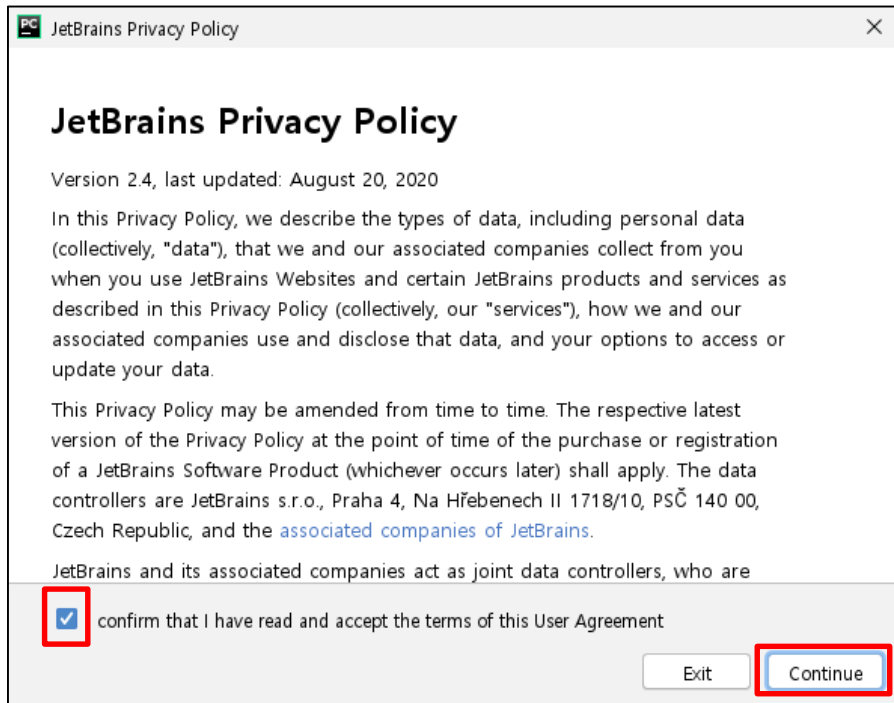


# 파이참(PyCharm)

- 파이썬 개발환경을 제공하는 **IDE**
  - ▣ 파이참, 스파이더 등
  - ▣ 파이참
    - 코드 편집기, 터미널, 디버거, 컴파일러, 인터프리터 등의 프로그래밍 툴과 **git** 인터페이스 제공
- 파이참 설치하기
  - ▣ <https://www.jetbrains.com/pycharm/download/#section=windows>
  - ▣ Community버전 다운로드
  - ▣ 더블클릭하여 실행



# 파일참



Projects

Customize

Plugins

Learn PyCharm

# Welcome to PyCharm

Create a new project to start from scratch.  
Open existing project from disk or version control.



New Project



Open



Get from VCS

New Project

Location: C:\Users\OWNER\PycharmProjects\pythonProject

Python Interpreter: Previously configured interpreter

☐ New environment using Virtualenv

Location: C:\Users\OWNER\PycharmProjects\pythonProject\venv

Base interpreter: Python 3.8 <https://www.python.org/ftp/python/3.8.6/python-3.8.6-amd64.exe>☐ Inherit global site-packages☐ Make available to all projects☒ Previously configured interpreter

Interpreter: &lt;No interpreter&gt;

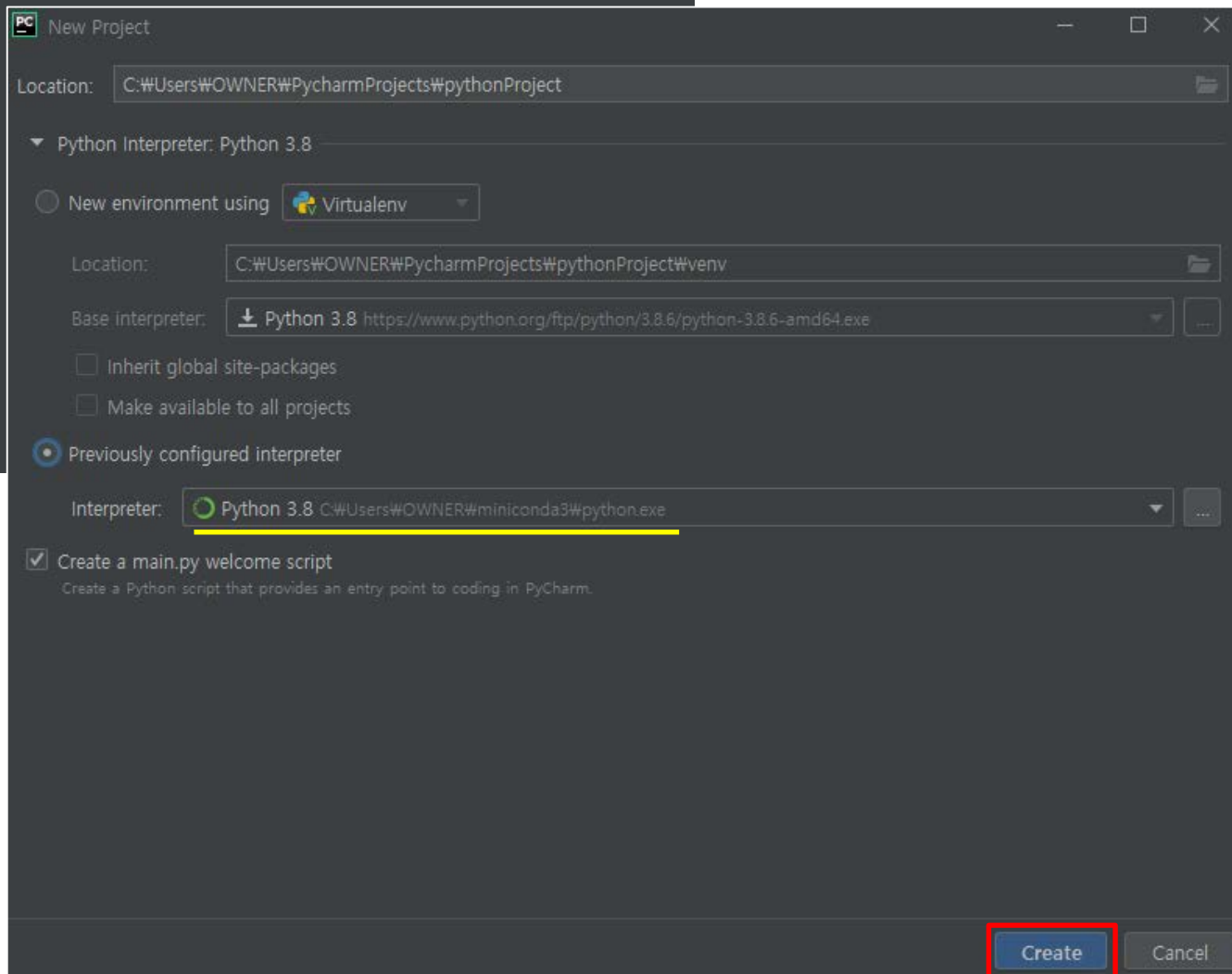
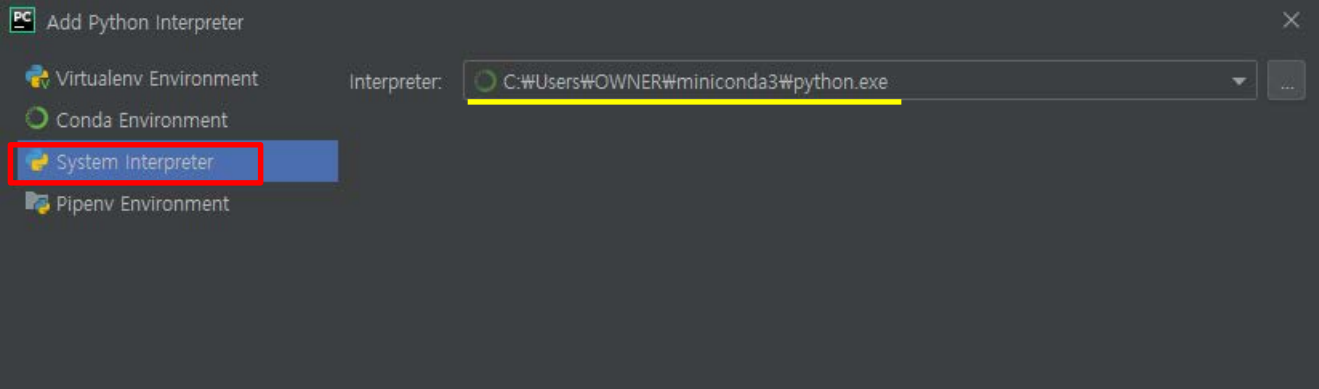
☒ Create a main.py welcome script

Create a Python script that provides an entry point to coding in PyCharm.

⚡ No Python interpreter selected

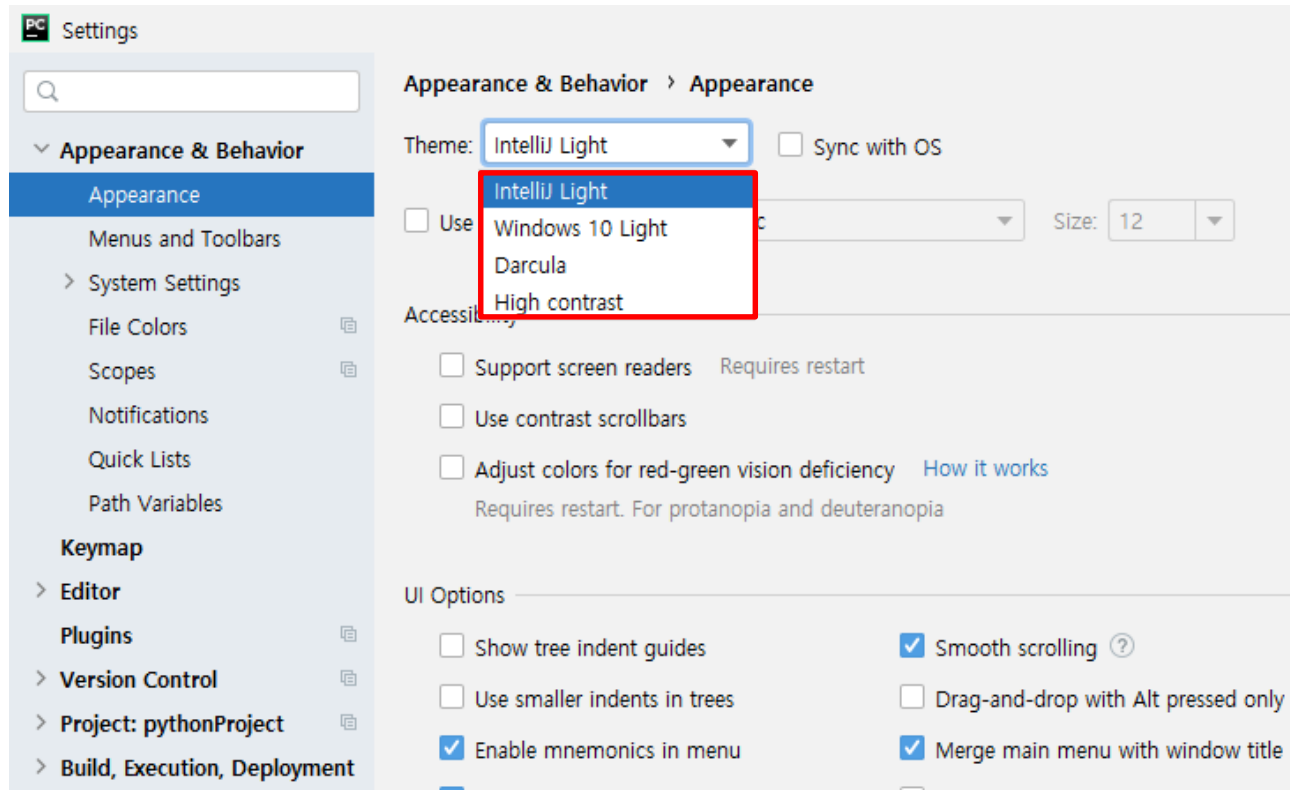
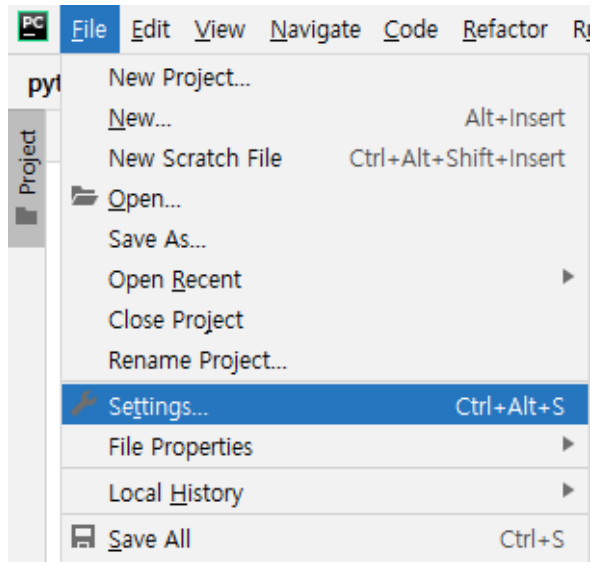
Create

Cancel

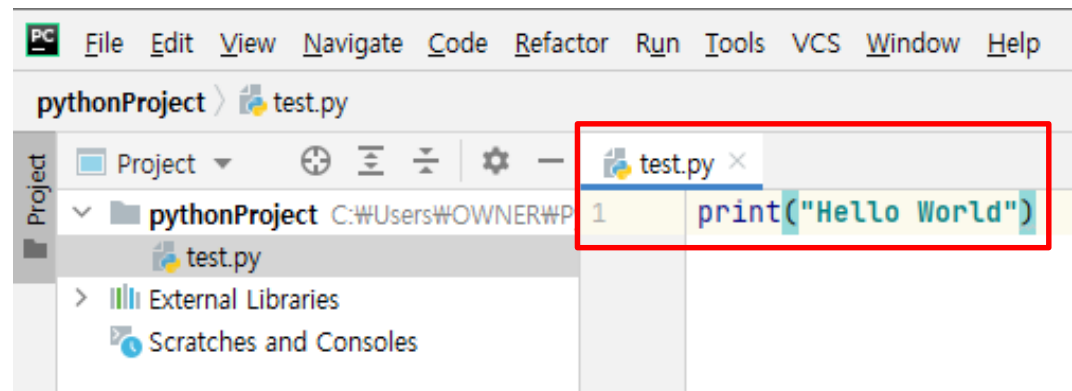
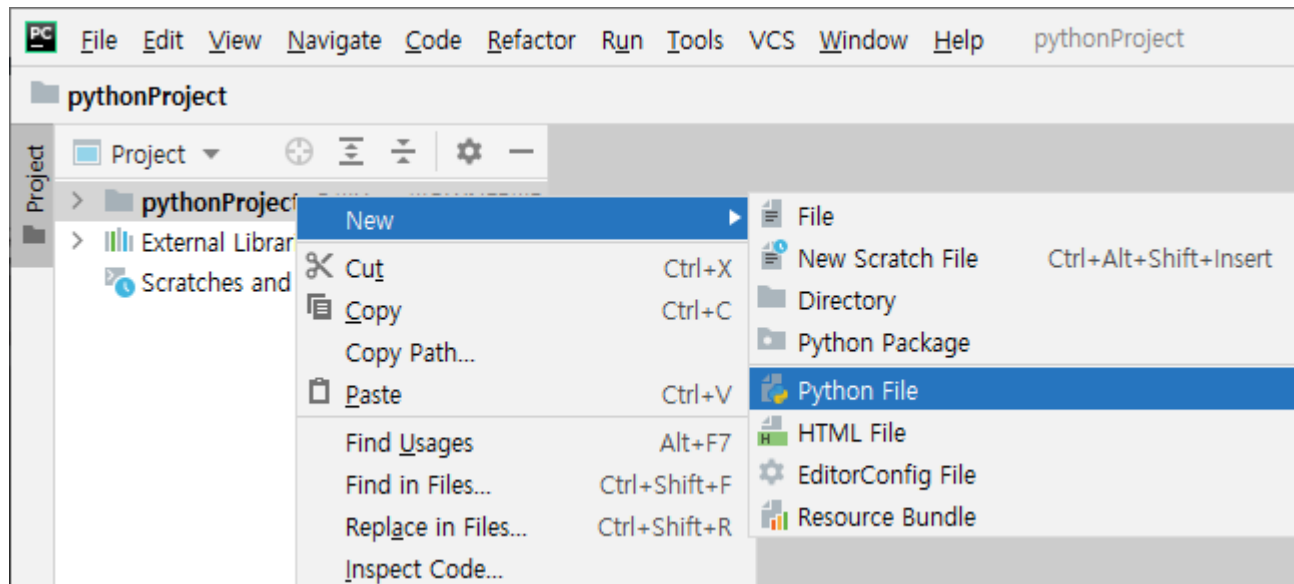


# 파이참 UI 변경하기

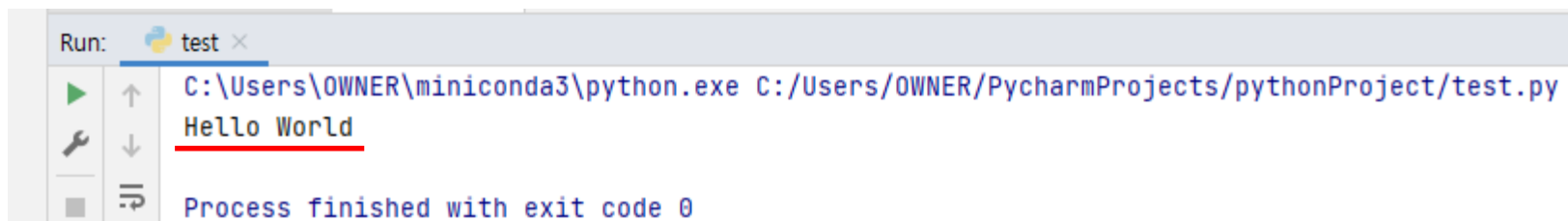
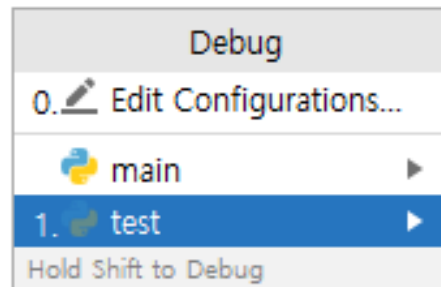
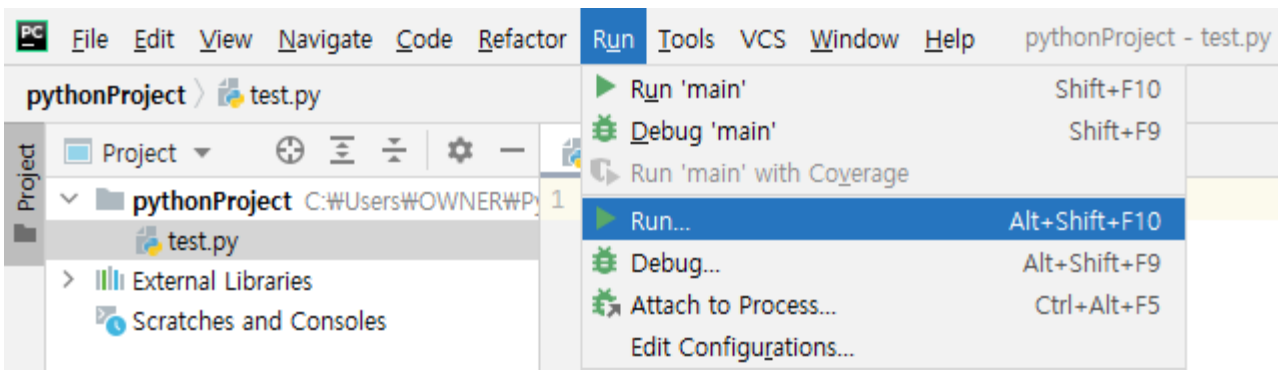
- File → Settings → Appearance & Behavior → Appearance → Theme 선택 → OK



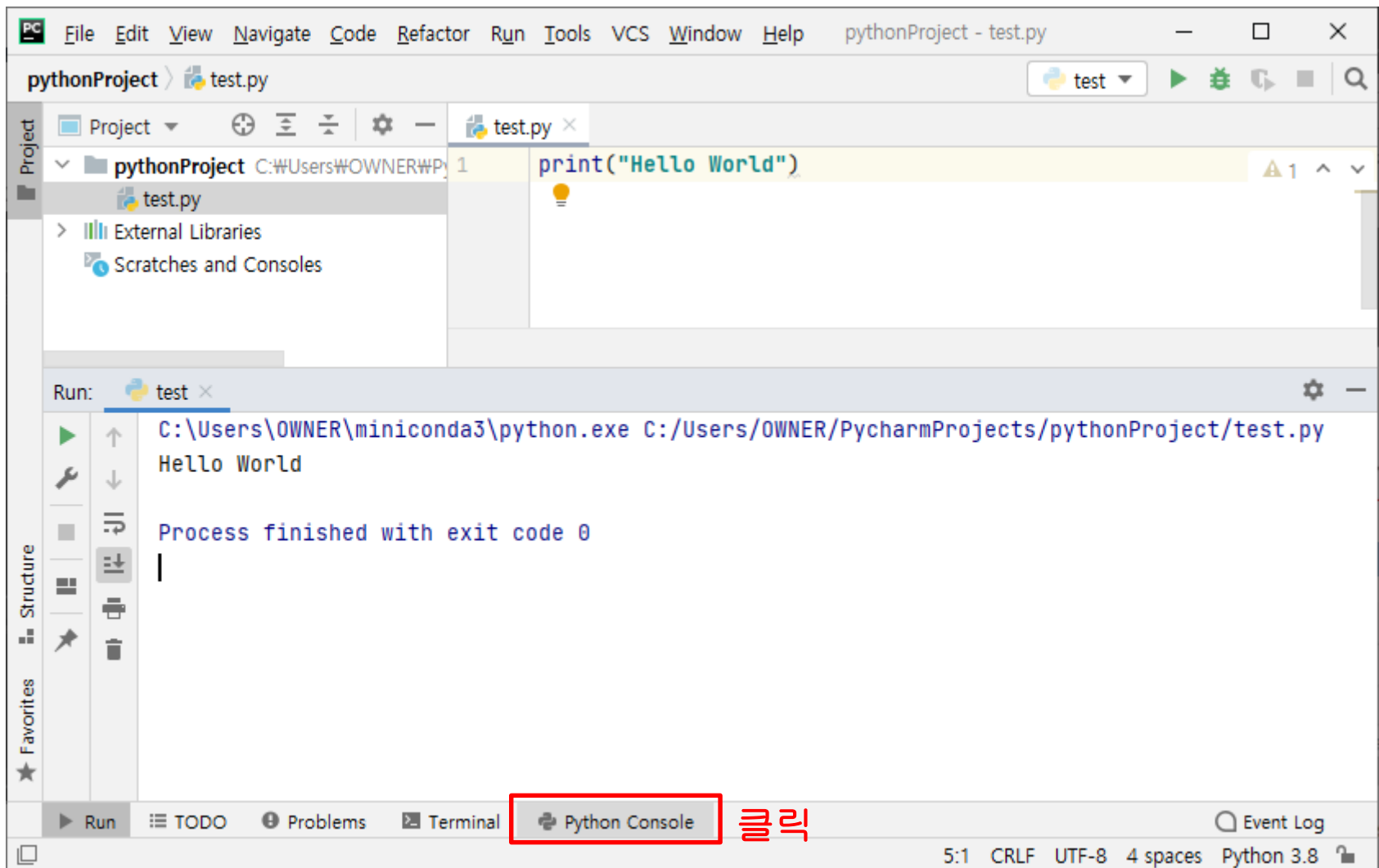
# 파일참 이용하여 코딩하기



# 파일참 이용하여 실행하기

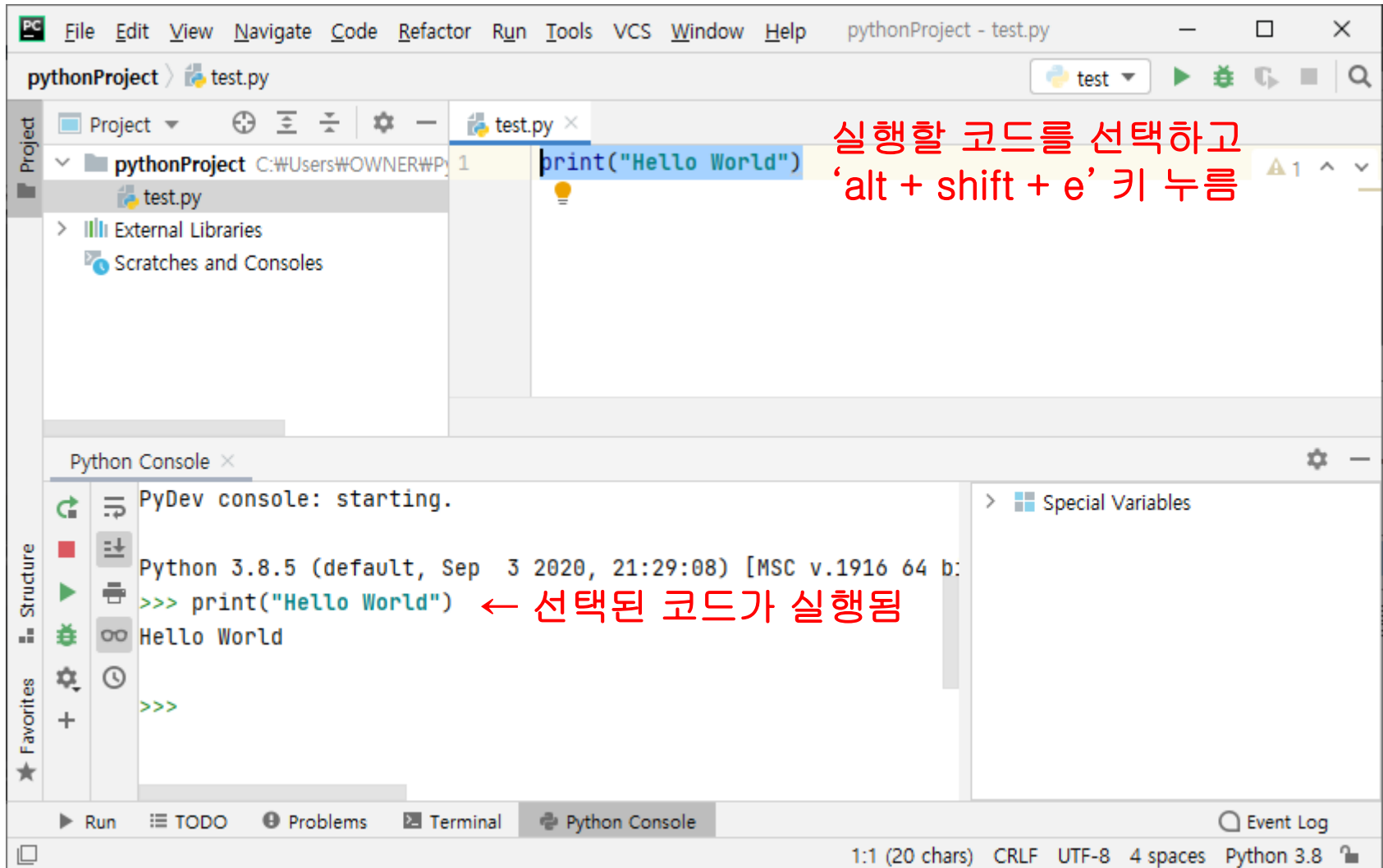


# 한 줄씩 실행하기

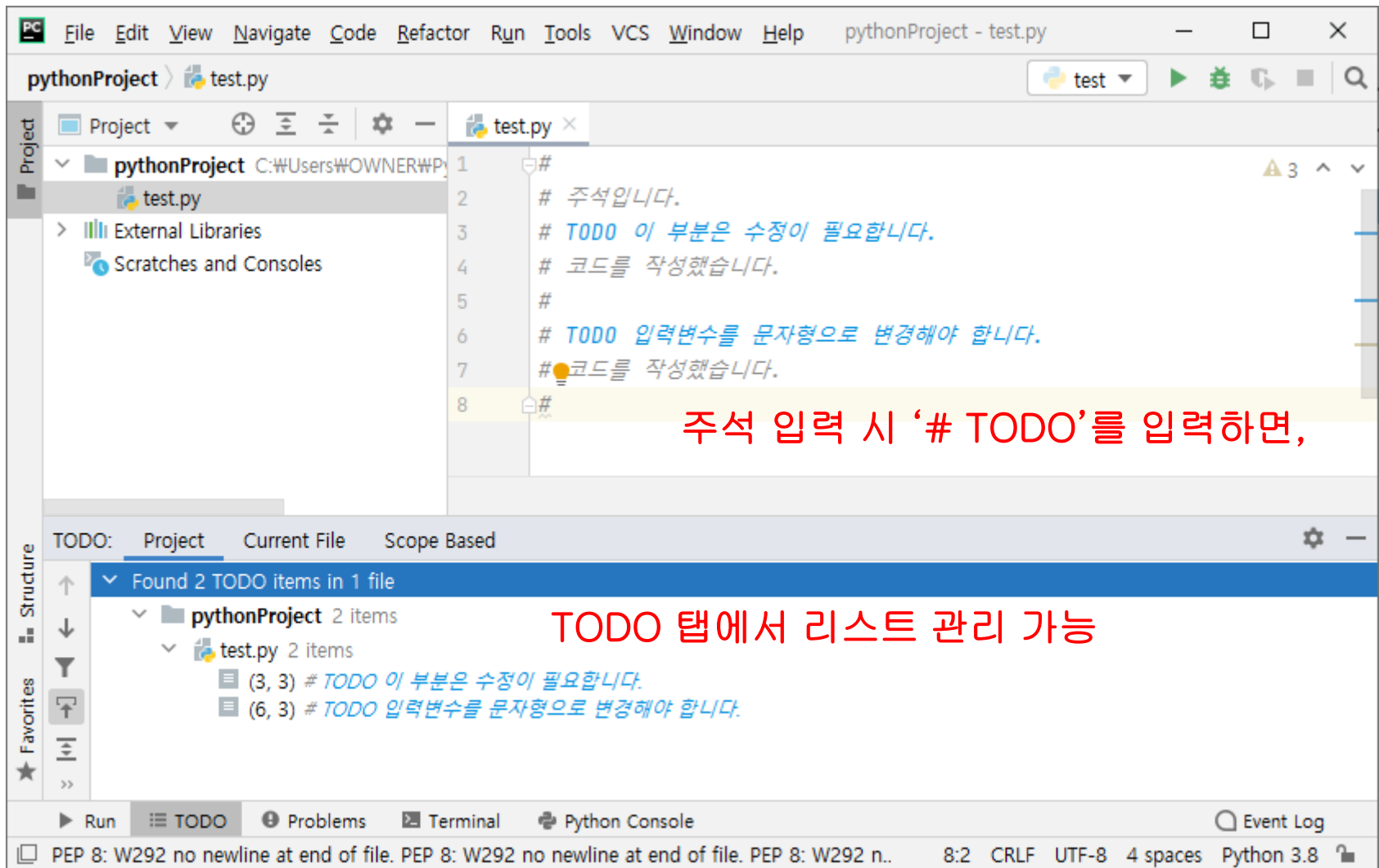




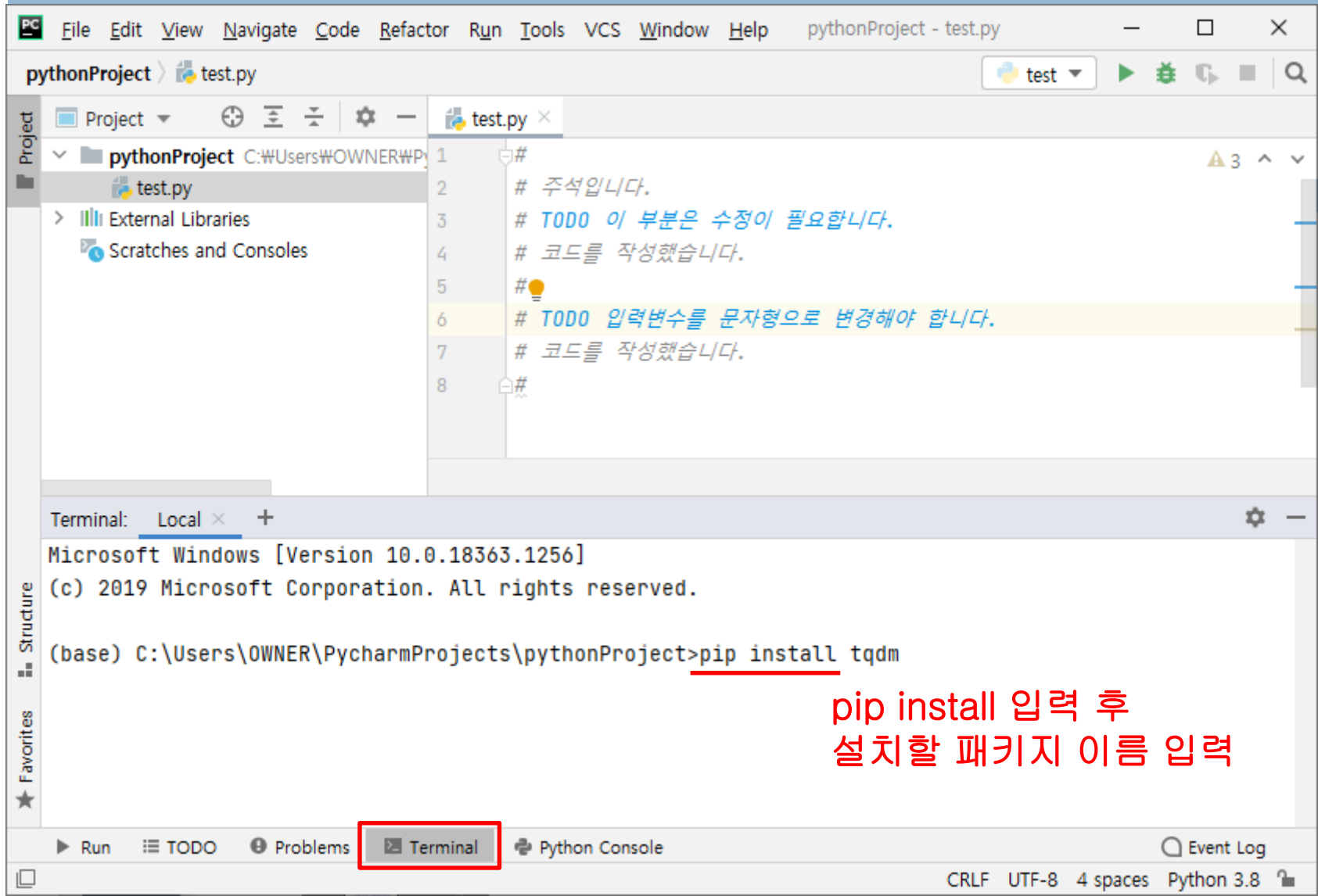
# 한 줄씩 실행하기



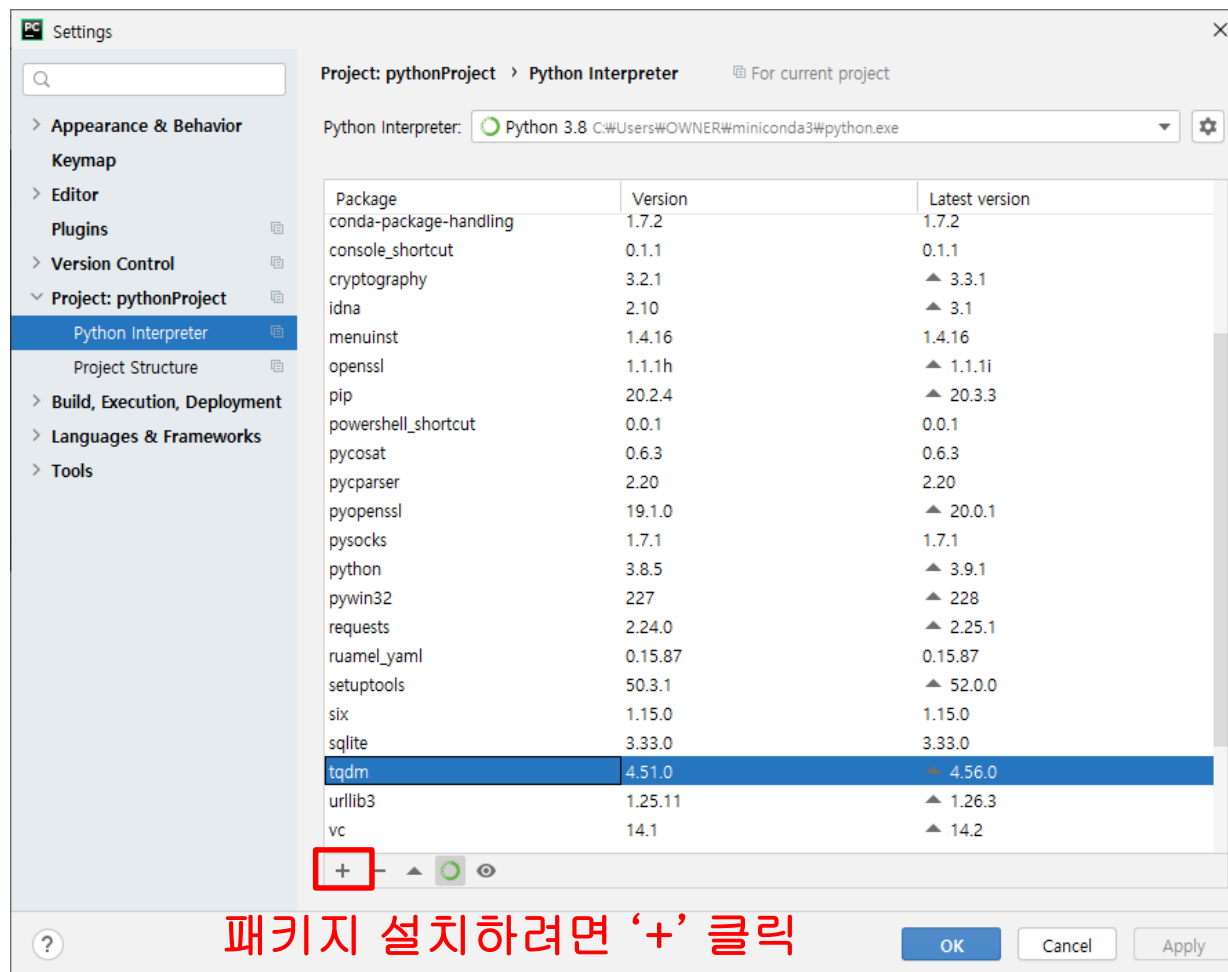
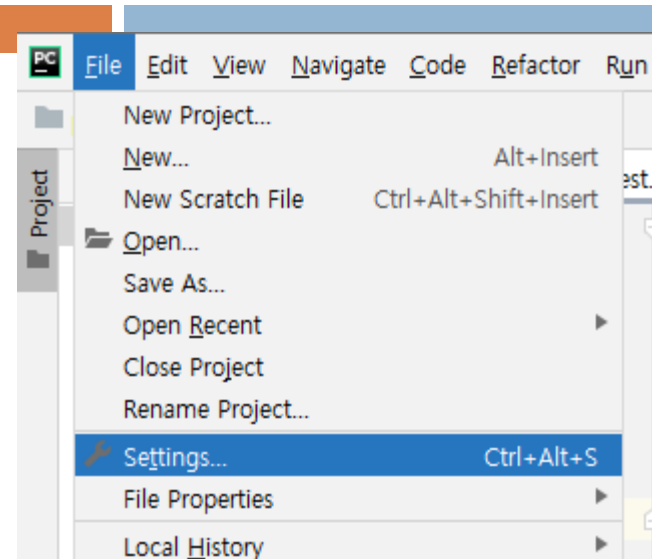
# 파이참에서 주석으로 'TODO' 관리하기



# 패키지 설치하기(1)



# 설치된 패키지 확인 및 설치하기(2)



Available Packages

Q jupyter

jupyter

hdijupyterutils

jupyter-packaging

jupyter\_client

jupyter\_console

jupyter\_core

jupyter\_dashboards\_bundlers

jupyter\_kernel\_gateway

jupyter\_telemetry

jupyterhub

jupyterlab

jupyterlab\_launcher

jupyterlab\_pygments

jupyterlab\_server

jupyterlab\_widgets

Description

Jupyter metapackage. Install all the Jupyter components in one go.  
Version 1.0.0  
Author Jupyter Development Team

Install Package

Manage Repositories

Settings

Project: pythonProject > Python Interpreter

Python Interpreter: Python 3.8 C:\Users\OWNER\miniconda3\python.exe

Package	Version	Latest version
importlib_metadata	2.0.0	2.0.0
ipykernel	5.3.4	5.3.4
ipython	7.20.0	7.20.0
ipython_genutils	0.2.0	0.2.0
ipywidgets	7.6.3	7.6.3
jedi	0.17.0	0.18.0
jinja2	2.11.3	2.11.3
jpeg	9b	9b
jsonschema	3.2.0	3.2.0
jupyter	1.0.0	1.0.0
jupyter_client	6.1.7	6.1.7
jupyter_console	6.2.0	6.2.0
jupyter_core	4.7.1	4.7.1
jupyterlab_pygments	0.1.2	0.1.2
jupyterlab_widgets	1.0.0	1.0.0
libpng	1.6.37	1.6.37
libsodium	1.0.18	1.0.18
m2w64-gcc-libgfortran	5.3.0	5.3.0
m2w64-gcc-libs	5.3.0	5.3.0
m2w64-gcc-libs-core	5.3.0	5.3.0
m2w64-omp	6.1.0	6.1.0

Package 'jupyter' installed successfully

Appearance & Behavior

Keymap

Editor

Plugins

Version Control

Project: pythonProject

Python Interpreter

Project Structure

Build, Execution, Deployment

Languages & Frameworks

Tools

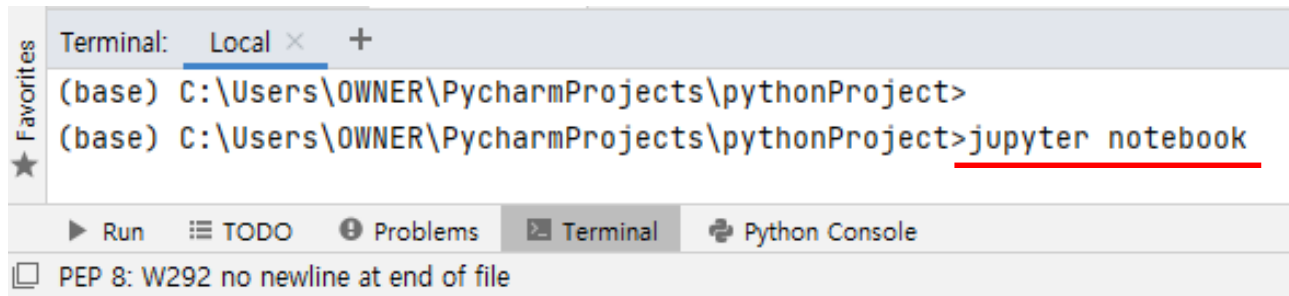
잠시후 jupyter 패키지가 설치된 것을 확인할 수 있음

# 주피터 노트북 실행하기

## □ 주피터 노트북(Jupyter Notebook)

- 세 가지 언어 Julia, Python, R 파일을 작성, 실행하는 개발 환경을 제공하는 웹 애플리케이션
- 셀 단위로 작성하여 실행할 수 있기에 큰 파이썬 파일도 셀 단위로 나누어 번역, 실행하면서 인터랙티브한 동작이 가능
- 데이터 분석을 위한 파이썬 파일 작성 후 실행하였을 때, 차트, 표 등의 결과값 출력도 바로바로 직관적으로 볼 수 있음
- Github에 주피터 노트북의 결과 출력 방식 그대로 올릴 수 있음

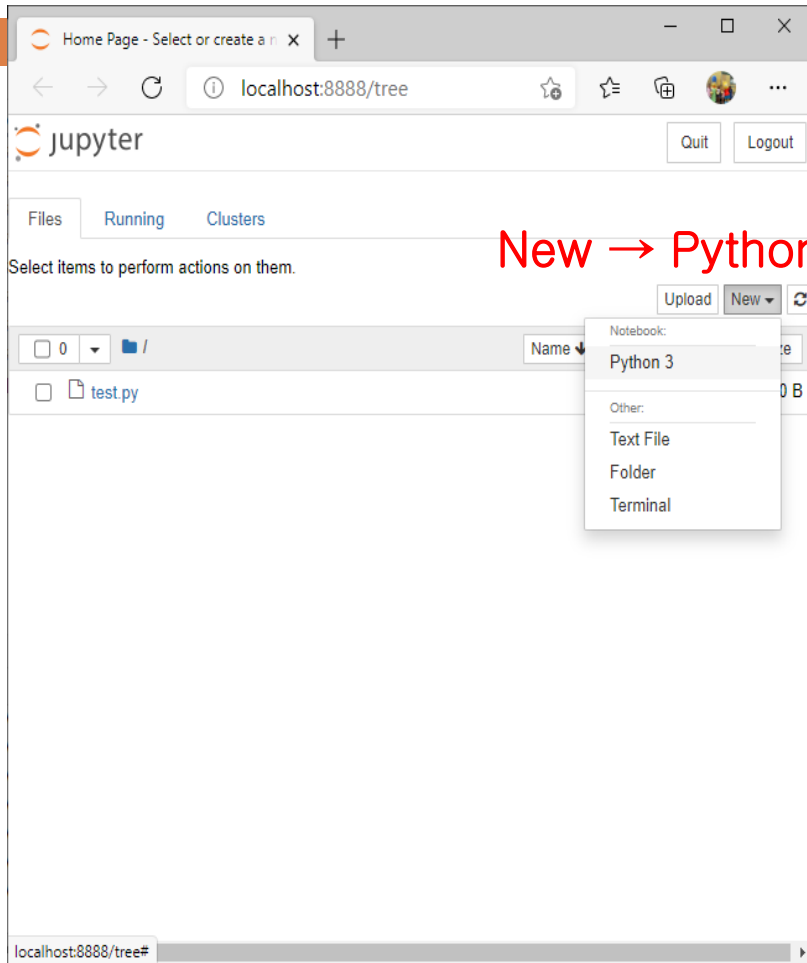
## □ 파이참 Terminal 탭에서 'jupyter notebook' 실행



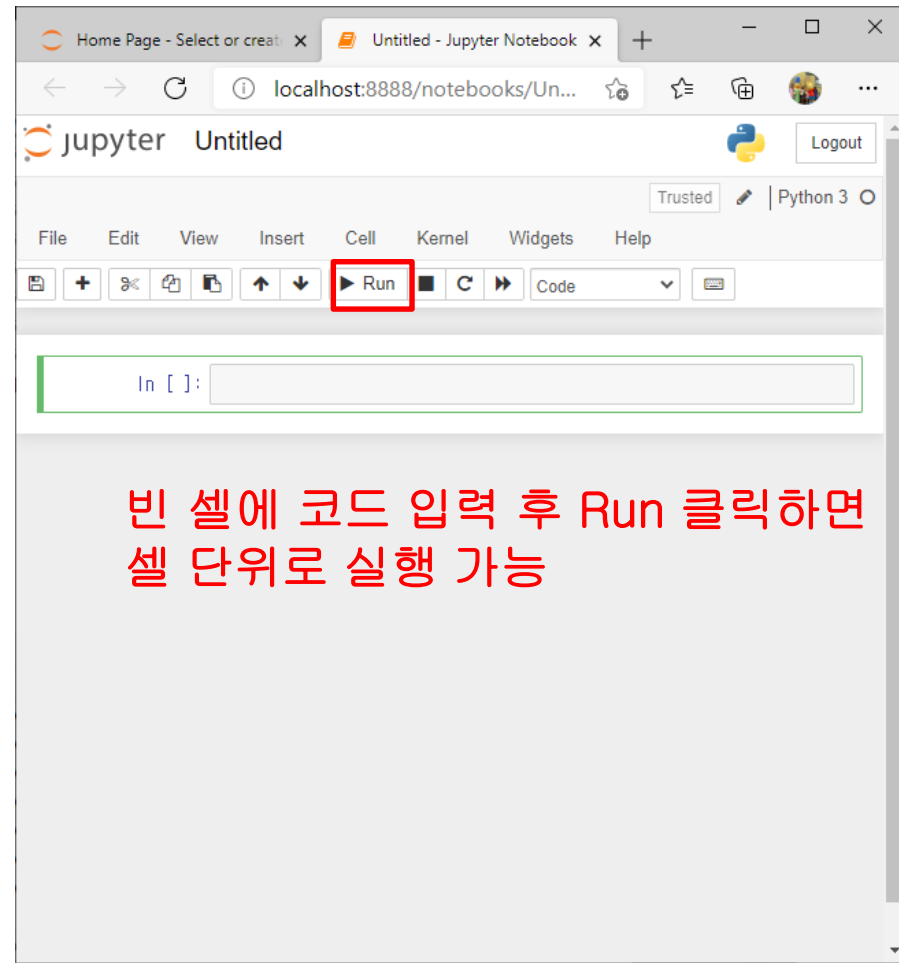
```
Terminal: Local x +
(base) C:\Users\OWNER\PycharmProjects\pythonProject>
(base) C:\Users\OWNER\PycharmProjects\pythonProject>jupyter notebook
```

The screenshot shows the PyCharm IDE interface. The 'Terminal' tab is active, displaying the command prompt for the project directory. The command 'jupyter notebook' is entered and highlighted with a red underline. The bottom status bar shows a warning: 'PEP 8: W292 no newline at end of file'.

# 주피터 노트북 사용하기



New → Python 3 선택



빈 셀에 코드 입력 후 Run 클릭하면  
셀 단위로 실행 가능

# 스파이더 (주교재에서 소개)

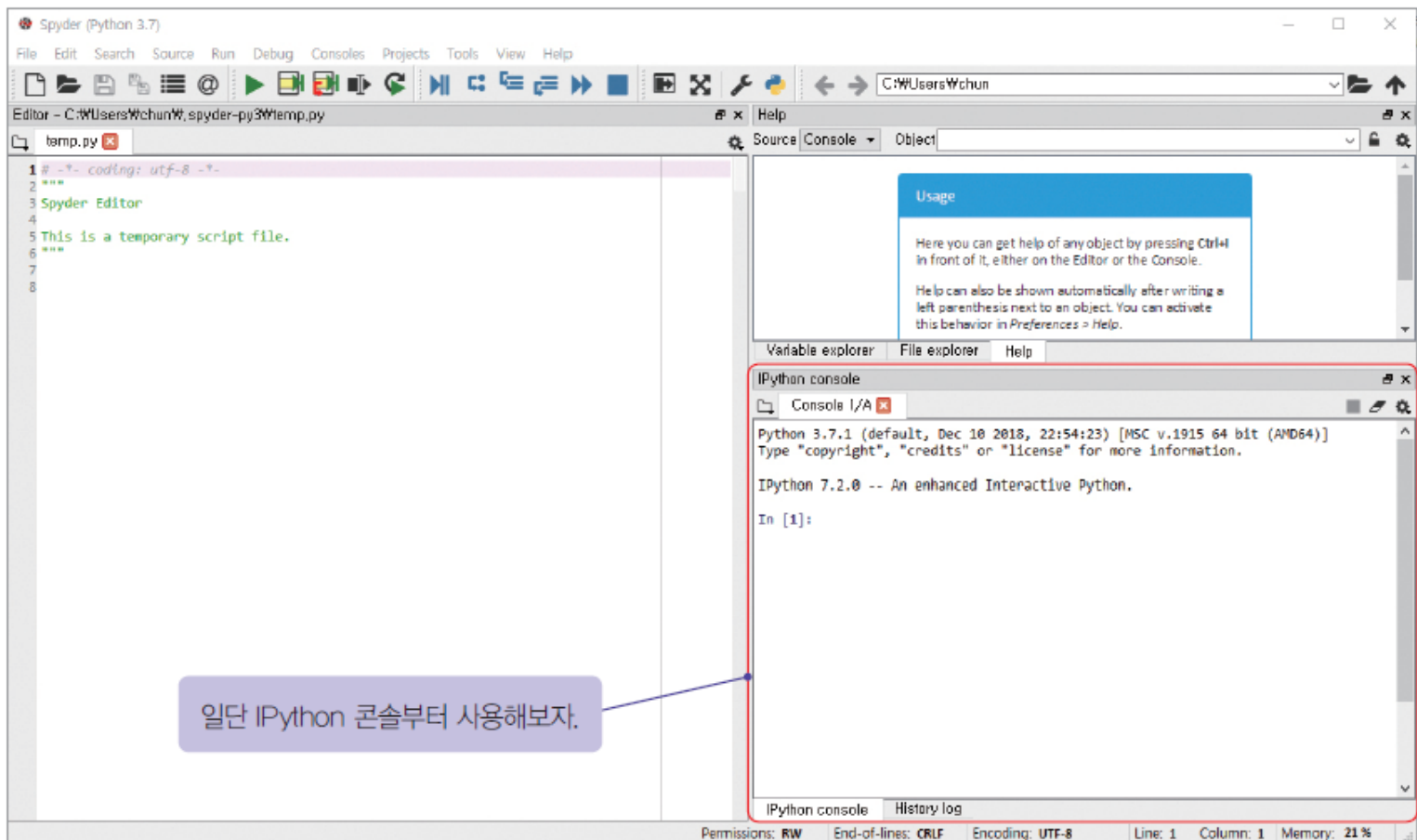
- 스파이더는 파이썬으로 작성된 파이썬 개발 도구

 Editor 함수 / 클래스 브라우저, 코드 분석 도구, 자동 코드 완성, 수평 / 수직 분할을 사용하여 효율적으로 작업이 가능하다.	 IPython Console GUI 인터페이스 안에서 코드를 라인별로 실행하거나 인라인으로 차트를 그릴 수 있다.	 Variable Explorer 변수와 상호작용하거나 변수를 변경할 수 있다. 히스토그램을 그리거나 데이터 프레임을 편집하고 컬렉션을 정렬할 수도 있다.
 Profiler 코드에서 가장 시간을 잡아먹는 부분을 바로 알아내서 제거할 수 있다.	 Debugger 디버거를 사용하여 코드를 한 줄씩 실행할 수 있다.	 Help 클래스에 관한 도움말을 즉시 볼 수 있다.

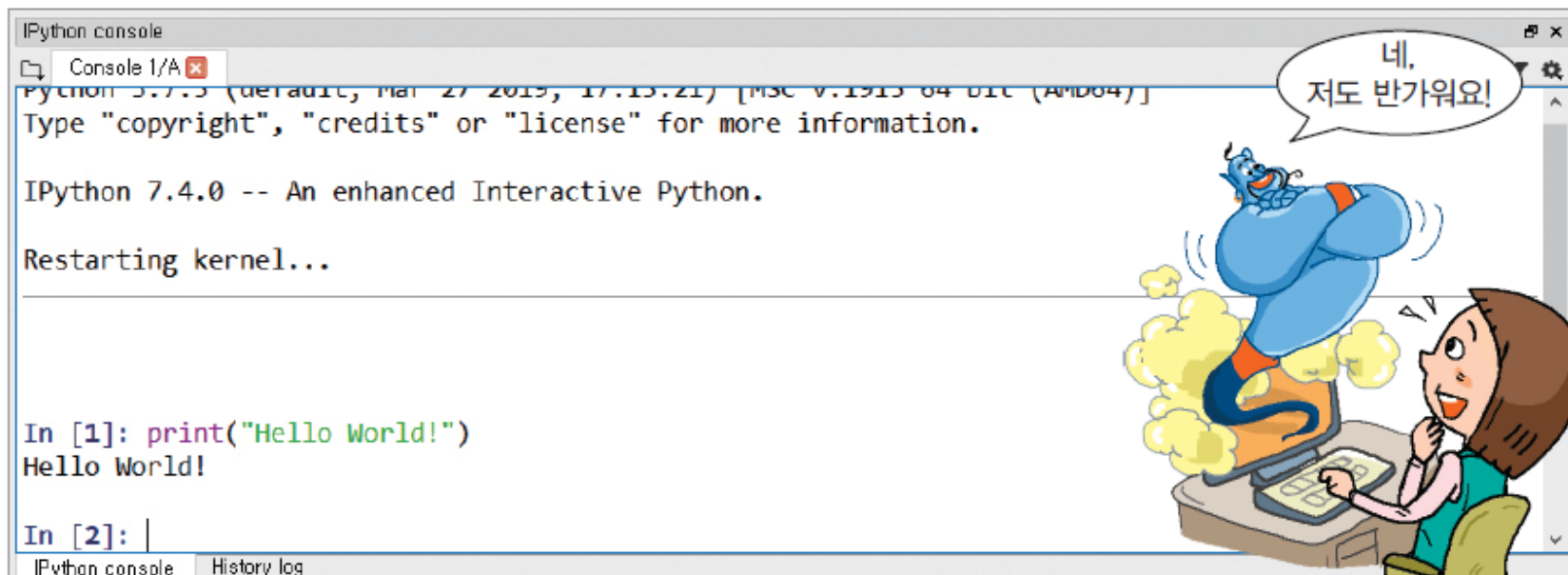


# Ipython 콘솔

- 파이참의 Python Console과 동일



# IPython 콘솔



# 계산하기 #1

- 반지름이 10cm인 피자 면적을 계산해보자.

```
In [ ]: 3.14 * 10 * 10 Enter↵  
Out[ ]: 314.0
```

```
>>> 3.14 * 10**2 Enter↵  
314.0
```

```
>>> 3.14 * 20**2 Enter↵  
1256.0
```

# 계산하기 #2

- 삼각함수, 로그함수 사용하기

```
>>> import math Enter↵  
>>> math.sin(math.radians(30)) Enter↵  
0.49999999999999994
```

```
>>> math.sqrt(9.0) Enter↵  
3.0
```

```
>>> math.log(1.0) Enter↵  
0.0
```

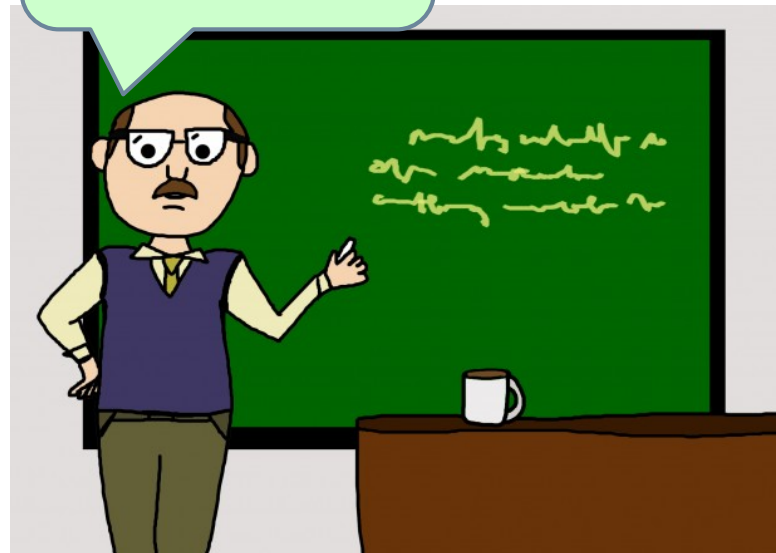
# 문자열 출력하기

```
>>> "나의 " + "고양이 "  
나의 고양이  
>>>
```

문자열은  
어떻게  
구별하나요?



따옴표("...")가  
붙으면  
문자열입니다.



# 문자열 반복하기

```
>>> "Hello" * 10
```

```
'HelloHelloHelloHelloHelloHelloHelloHelloHelloHello'
```

문자열이  
반복되네요!



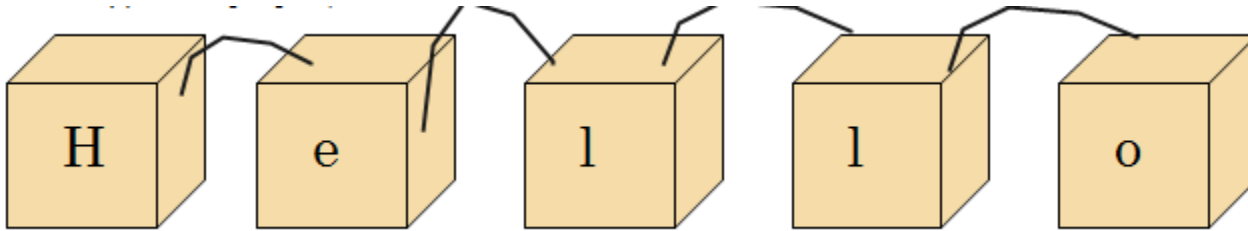
문자열에 \*을  
적용하면  
반복을 의미  
합니다.



# 문자열

- 문자열(string)

- ▣ 큰따옴표("...")나 작은따옴표('...') 안에 들어 있는 텍스트 데이터



- 반드시 따옴표가 있어야 한다.

```
>>> print(Hello World!)  
SyntaxError: invalid syntax
```

# 중간 점검

- 한글도 출력될까? 이번에도 따옴표를 올바르게 입력하여야 한다.
  - ▣ “안녕하세요?”를 화면에 출력하여 보자.
- “programming에 입문하신 것을 축하드립니다.”를 출력하여 보자.
- “생일 축하!!”를 10번 출력하는 명령문을 만들어보자. 문자열 반복을 사용한다.
- 다음과 같은 명령문을 실행하면 오류가 발생한다. 원인을 알아보자.  

```
>>> print("Hello)
```





# 대화형 모드와 스크립트 모드

- 대화형 모드(interactive mode)
  - ▣ 콘솔에서 문장을 한 줄씩 입력하여 실행
- 스크립트 모드(script mode)
  - ▣ 파일을 만들어서 저장한 후에 파이썬 인터프리터가 이 파일을 읽어서 한 번에 전부 실행

대화형 모드	스크립트 모드
<pre>&gt;&gt;&gt; print("Hello World!") Hello World! &gt;&gt;&gt; print(10+20) 30</pre>	<pre>print("Hello World!") print(10+20)</pre> <p>실행결과</p> <pre>Hello World! 30</pre>

문장이 즉시 실행되고  
실행결과가 바로 나타난다.

▶ 버튼을 눌러야 문장들이 실행  
되고 실행결과가 콘솔에 나타난다.

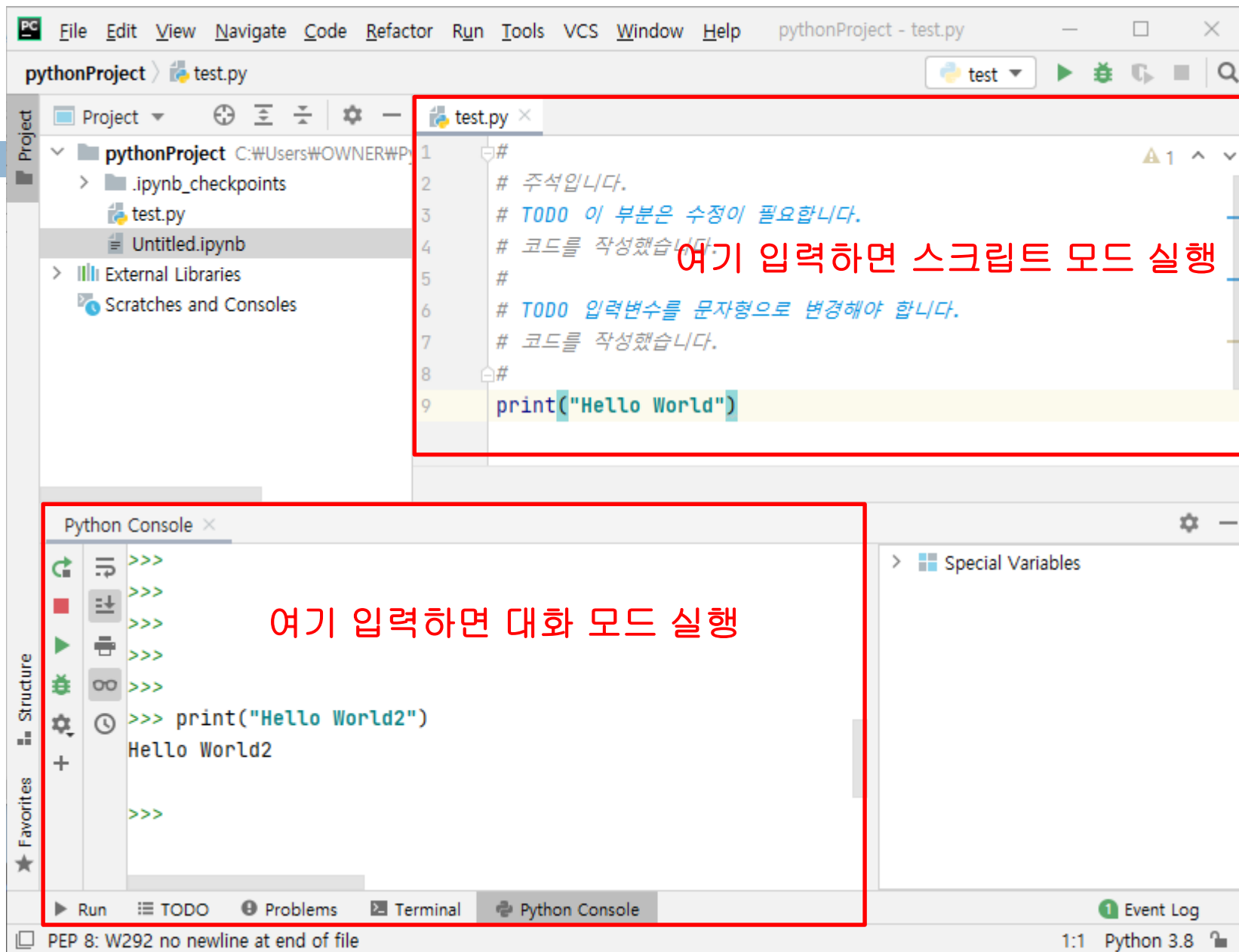
# 스크립트 모드

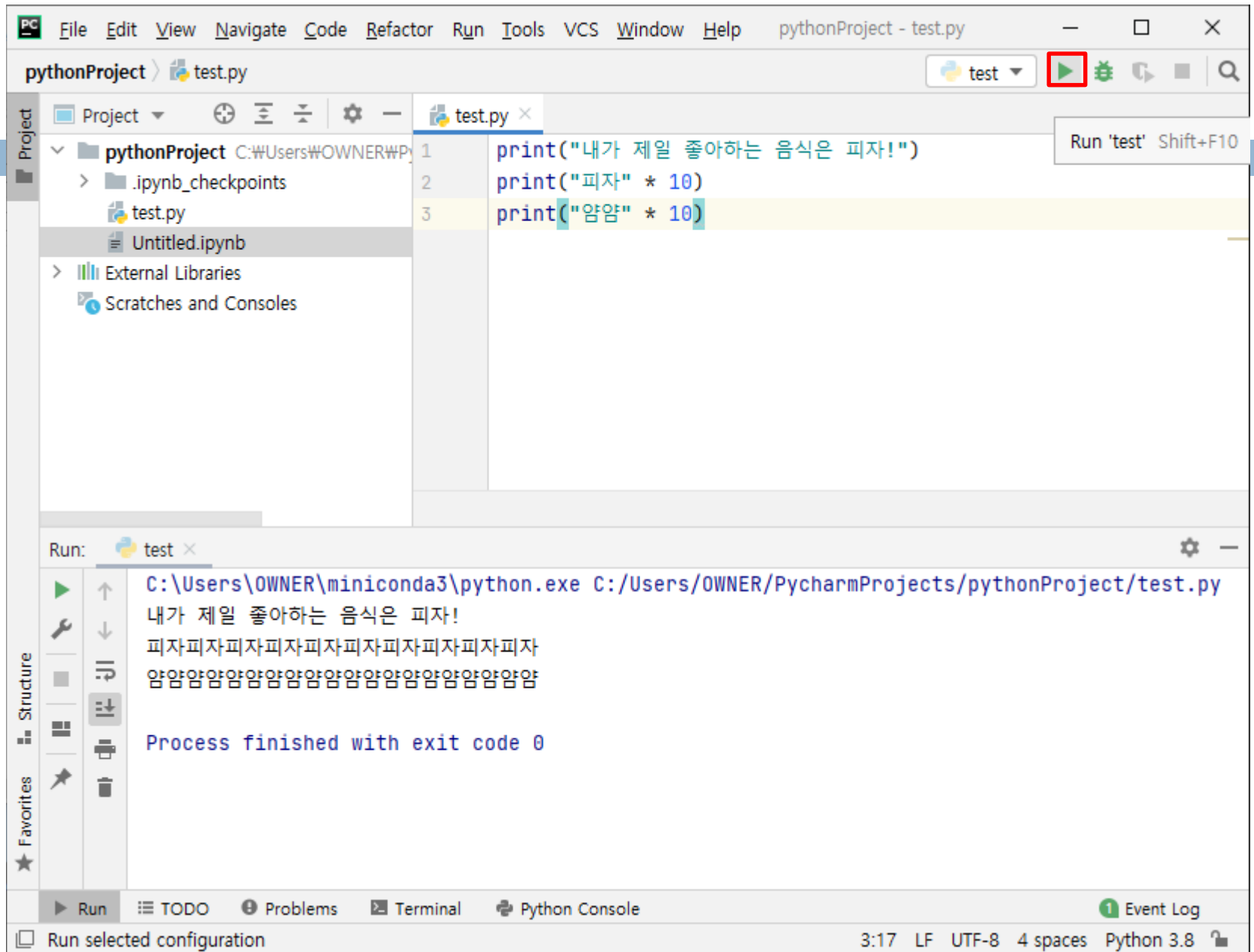
- 코드가 복잡해지면 인터프리트 모드는 번거롭다.

이걸 한 줄씩 입력하라고?



```
def add5(x):  
    return x+5  
  
def dotwrite(ast):  
    nodename = getNodeName()  
    label=symbol.sym_name.get(int(ast[0]),ast[0])  
    print '    %s [label="%s' % (nodename, label)  
    if isinstance(ast[1], str):  
        if ast[1].strip():  
            print '= %s' % ast[1]  
        else:  
            print ''  
    else:  
        print ''  
        children = []  
        for n, child in enumerate(ast[1:]):  
            children.append(dotwrite(child))  
        print ', ' % ast[1] -> {' % nodename  
        for n, child in enumerate(children):  
            print '%s' % child,
```





The screenshot shows a Jupyter Notebook interface with the following content:

- Cell 1:**

```
In [2]: 3.14*10*10
```

```
Out [2]: 314.0
```
- Cell 2:**

```
In [3]: import math
         math.sin(math.radians(30))
```

```
Out [3]: 0.49999999999999994
```
- Cell 3:**

```
In [4]: print("내가 제일 좋아하는 음식은 피자!")
         print("피자" * 10)
         print("암암" * 10)
```

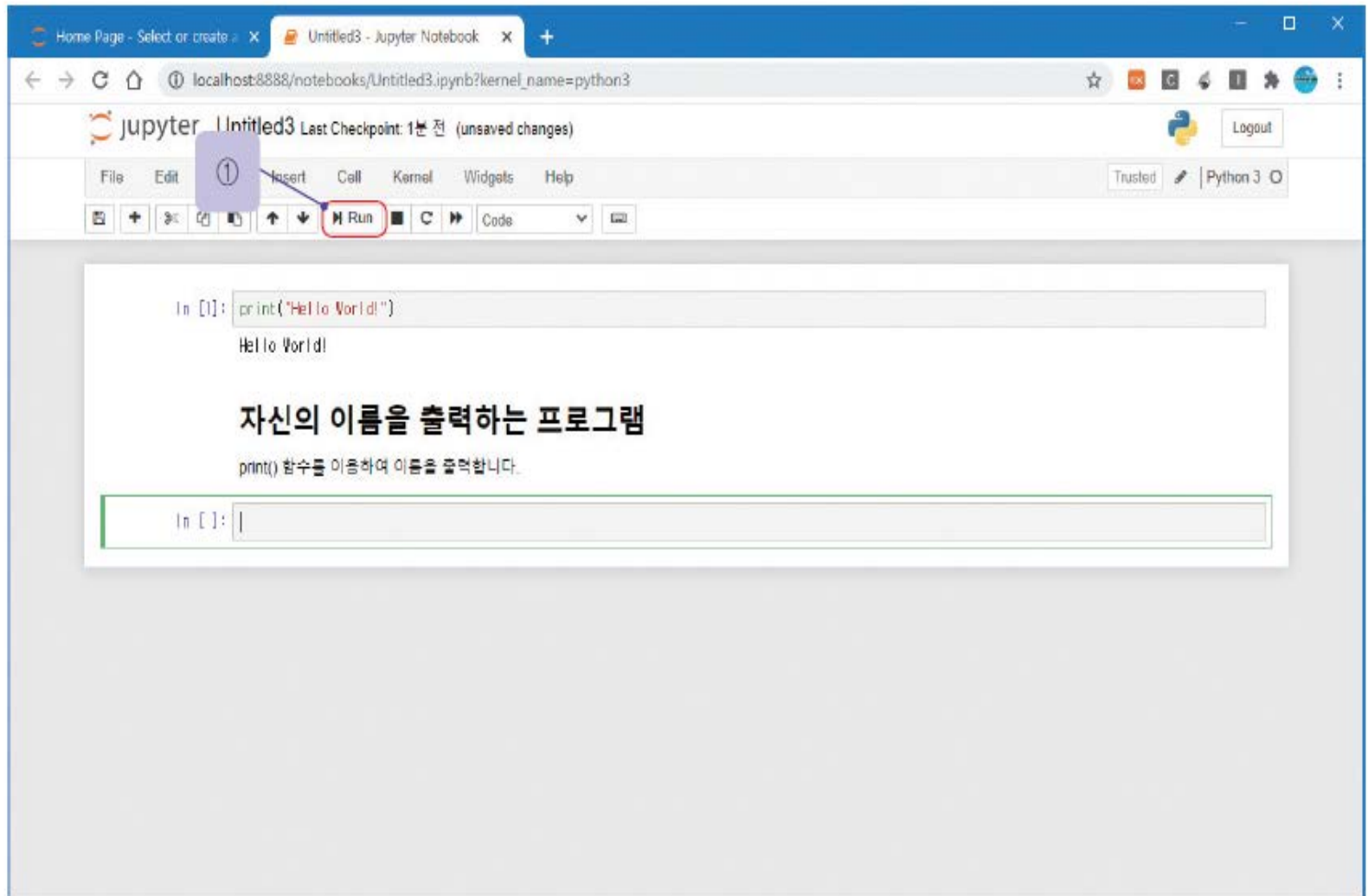
```
내가 제일 좋아하는 음식은 피자!
피자피자피자피자피자피자피자피자피자피자
암암암암암암암암암암암암암암암암암암
```

The 'Run' button in the toolbar is highlighted with a red box.

# 주피터 노트북에서 설명 추가하기

The screenshot shows a Jupyter Notebook interface in a web browser. The browser tab is titled "Untitled3 - Jupyter Notebook". The address bar shows the URL: `localhost:8888/notebooks/Untitled3.ipynb?kernel_name=python3`. The Jupyter logo and "Untitled3" are visible in the top left. The top right shows "Last Checkpoint: 1분 전 (unsaved changes)" and a "Logout" button. The menu bar includes File, Edit, View, Insert, Cell, Format, Widgets, and Help. The toolbar contains icons for adding new cells, undo, redo, and a dropdown menu currently set to "Markdown". A red box highlights the "Markdown" dropdown, with a callout bubble containing a circled "1" pointing to it. Below the toolbar, the first cell is a code cell containing `print("Hello World!")` and the output "Hello World!". Below that, a new markdown cell has been added, highlighted with a red box. It contains the text: **# 자신의 이름을 출력하는 프로그램** and `print()` 함수를 이용하여 이름을 출력합니다. |. A callout bubble containing a circled "2" points to this new markdown cell.

# 주피터 노트북에서 설명 추가하기



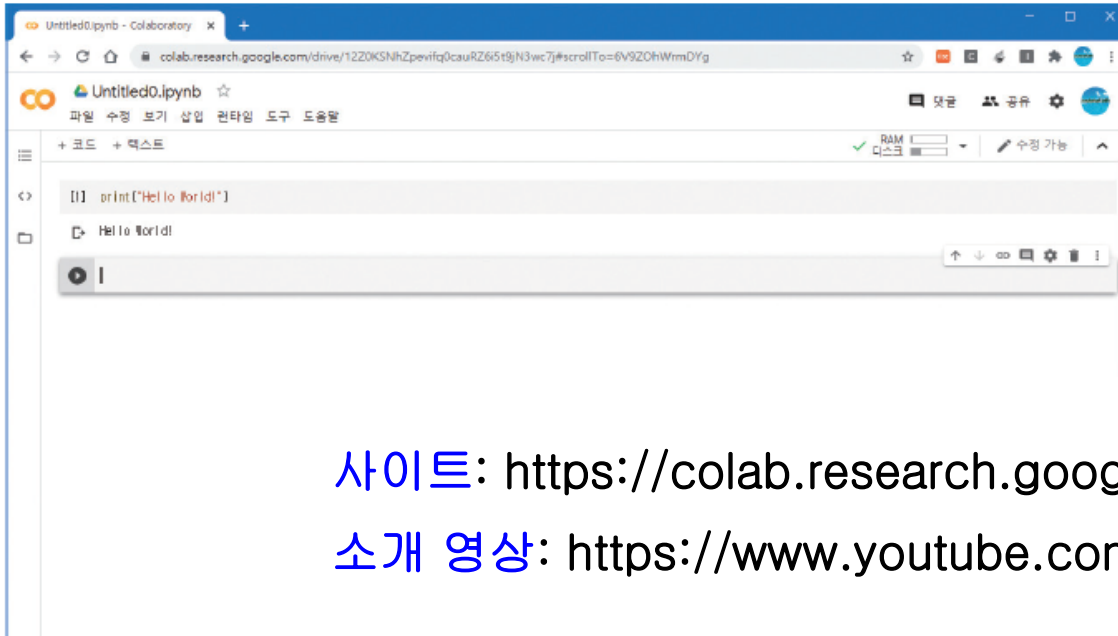
# 구글 Colab

## NOTE



### 구글 Colab

구글 Colab은 주피터 노트북 개념을 클라우드로 확장한 것이다. 구글 Colab을 이용하면 브라우저에서 파이썬 프로그램을 작성하고 실행할 수 있다. Colab 메모장을 사용하면 실행 코드와 서식 있는 텍스트를 이미지, HTML, LaTeX 등과 함께 하나의 문서로 통합할 수 있다. Colab 메모장을 만들면 Google 드라이브 계정에 저장된다. Colab 메모장을 간편하게 공유하여 동료나 친구들이 댓글을 달거나 수정하도록 할 수 있다. 자세한 설명은 유튜브 영상 "<https://www.youtube.com/watch?v=inN8seMm7UI>"을 참조하자. 주피터 노트북과도 호환된다.



사이트: <https://colab.research.google.com/notebooks/intro.ipynb>

소개 영상: <https://www.youtube.com/watch?v=inN8seMm7UI>



# print() 함수

- 파이썬 프로그램은 여러 줄의 명령문(statement)들로 이루어진다.
- 함수(function)는 특별한 작업을 수행하는 명령어들의 모임이다.

Syntax: print() 함수 #1

형식

print(문자열)

print() 함수는 텍스트를 화면에 출력한다.

예

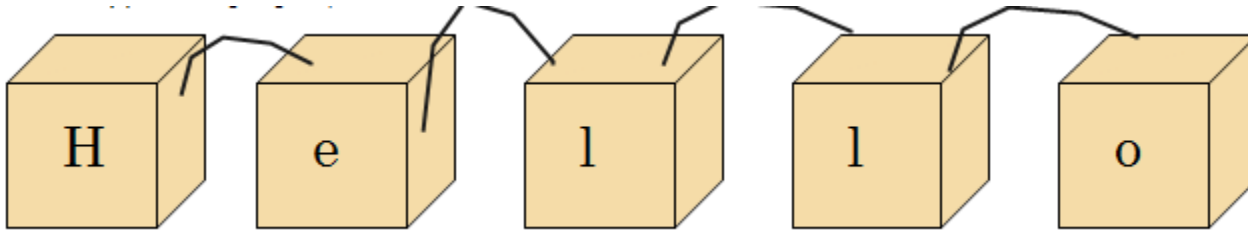
print("Hello World!")

"Hello World!"가 콘솔에 출력된다.

# 문자열

## □ 문자열(string)

- ▣ 큰따옴표("...")나 작은따옴표('...') 안에 들어 있는 텍스트 데이터



- 반드시 따옴표가 있어야 한다.

```
>>> print(Hello World!)  
SyntaxError: invalid syntax
```

# print() 함수

- 여러 개의 값들을 화면에 차례대로 출력할 수 있다.

```
>>> print("결과값은", 10, "입니다.")
```

```
결과값은 10 입니다.
```

# 중간 점검

1. 다음과 같이 3단 구구단의 일부를 출력하는 프로그램을 작성해보자.  
3\*1, 3\*2, 3\*3의 수식을 계산하여서 결과를 출력한다.

$$3 * 1 = 3$$

$$3 * 2 = 6$$

$$3 * 3 = 9$$



# Lab: print() 함수 실습

파이썬에 오신 것을 환영합니다.

파이썬은 쉽습니다.

파이썬으로 빅데이터, 인공지능 프로그램을 작성할 수 있습니다.

# Sol: print() 함수 실습

```
print("파이썬에 오신 것을 환영합니다.")
```

```
print("파이썬은 쉽습니다.")
```

```
print("파이썬으로 빅데이터, 인공지능 프로그램을 작성할 수 있습니다.")
```

# Lab: 간단한 계산

$$2+3= 5$$

$$2-3= -1$$

$$2*3= 6$$

$$2/3= 0.6666666666666666$$

# Sol: 간단한 계산

```
print("2+3=", 2+3)
```

```
print("2-3=", 2-3)
```

```
print("2*3=", 2*3)
```

```
print("2/3=", 2/3)          # 결과값에 유의
```



# Lab: 오류를 처리해보자.

```
print(안녕하세요?)
```

```
Print("이번 코드에는 많은 오류가 있다네요")
```

```
print("제가 다 고쳐 보겠습니다.")
```

```
File "D:/모두의 파이썬/sources/chap01/hello.py", line 1
```

```
    print(안녕하세요?)
```

```
        ^
```

```
SyntaxError: invalid syntax
```

# Sol: 오류를 처리해보자.

bug.py

```
1 print(안녕하세요?)  
2 Print("이번 코드에는 많은 오류가 있다네요")  
3 print("제가 다 고쳐 보겠습니다.")
```

문자열인데 따옴표가 없다.

P가 대문자이다.

끝나는 따옴표가 없다.

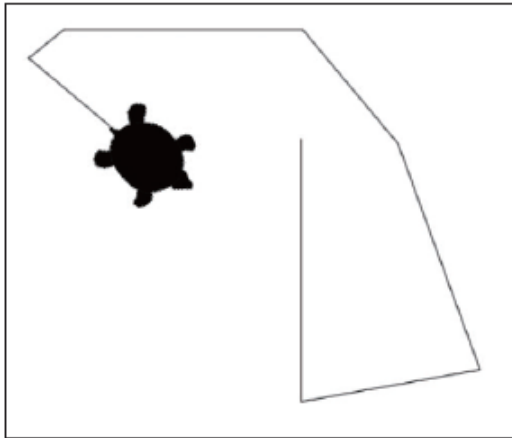
```
print("안녕하세요?")
```

```
print("이번 코드에는 많은 오류가 있다네요")
```

```
print("제가 다 고쳐 보겠습니다.")
```

# 터틀 그래픽

- **터틀 그래픽**은 화면에서 거북이를 이용하여서 그림을 그리는 기능이다.



# 터틀 그래픽 시작

```
import turtle      # (1)
```

```
t = turtle.Turtle() # (2)
```

```
t.shape("turtle")  # (3)
```

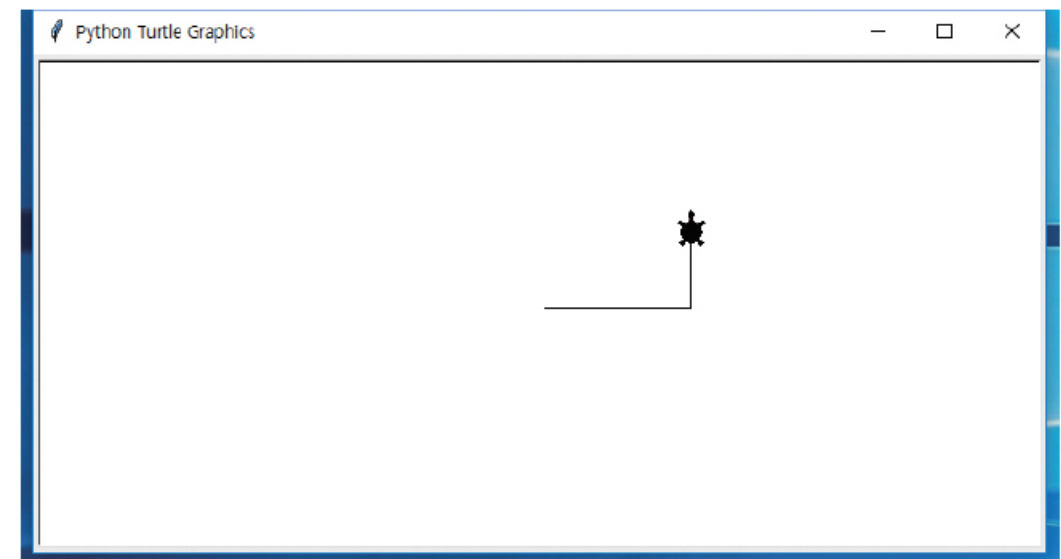
```
t.forward(100)      # (4)
```

```
t.left(90) # (5)
```

```
t.forward(50)
```

```
turtle.mainloop()  # (6)
```

```
turtle.bye()
```

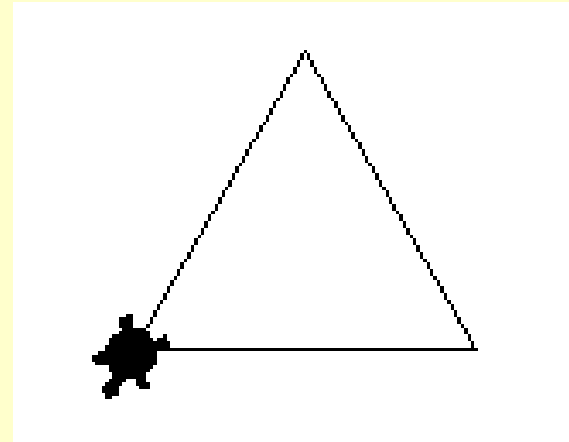


# Lab: 삼각형 그리기

```
import turtle  
t = turtle.Turtle()
```

```
t.shape("turtle")  
t.forward(100)  
t.left(120)  
t.forward(100)  
t.left(120)  
t.forward(100)
```

```
turtle.mainloop()  
turtle.bye()
```



# 파이썬으로 무엇을 만들 수 있을까?

```
import turtle

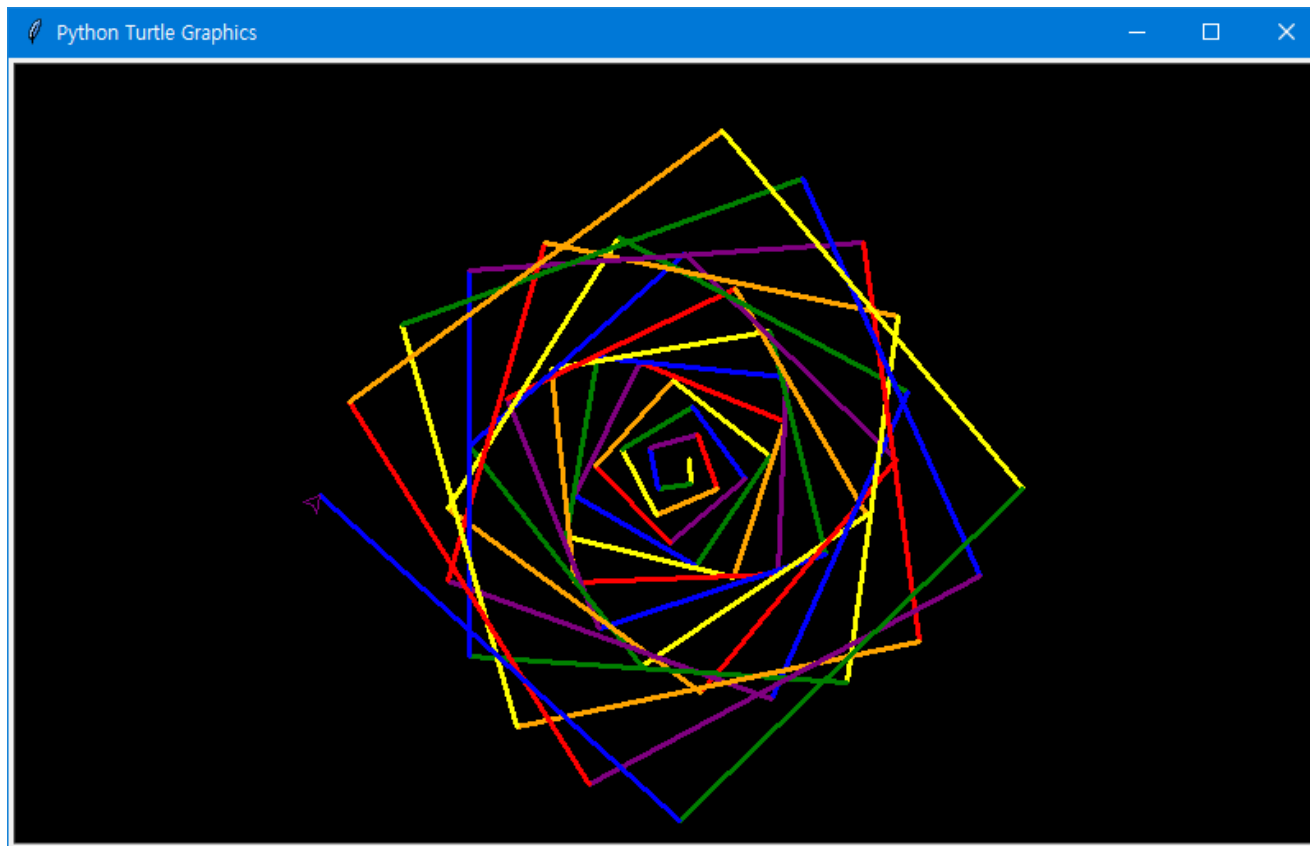
colors = ["red", "purple", "blue", "green", "yellow", "orange"]
t = turtle.Turtle()

turtle.bgcolor("black")
t.speed(0)
t.width(3)
length = 10

while length < 300:
    t.forward(length)
    t.pencolor(colors[length%6])
    t.right (86)
    length += 5

turtle.mainloop()
turtle.bye()
```

# 실행 결과는?



# Lab: 파일 안의 단어 분석하기

```
from collections import Counter
```

```
f = open("D:\WWWtemp\WWWmobydick.txt", encoding="utf-8")  
count = Counter(f.read().split())  
print("단어 출현 횟수 :", count)
```

```
단어 출현 횟수 : Counter({'the': 13851, 'of': 6638, 'and': 6000, 'a': 4549, 'to':  
4529, 'in': 3904, 'that': 2692, 'his': 2428, 'I': 1723, 'with': 1695, 'as': 1600,  
'is': 1588, 'was': 1567, 'it': 1516, 'he': 1495, 'for': 1385, 'all': 1314, 'at':  
1231, 'this': 1169, 'by': 1121, 'from': 1072, 'not': 1043, 'but': 1034, 'be': 991,  
'on': 926, 'so': 785, 'you': 784, 'or': 763, 'one': 755, 'have': 752, 'had': 751,  
'were': 645, 'But': 637, 'The': 635, 'their': 613, 'are': 586, 'an': 579, 'some':  
571, ...})
```