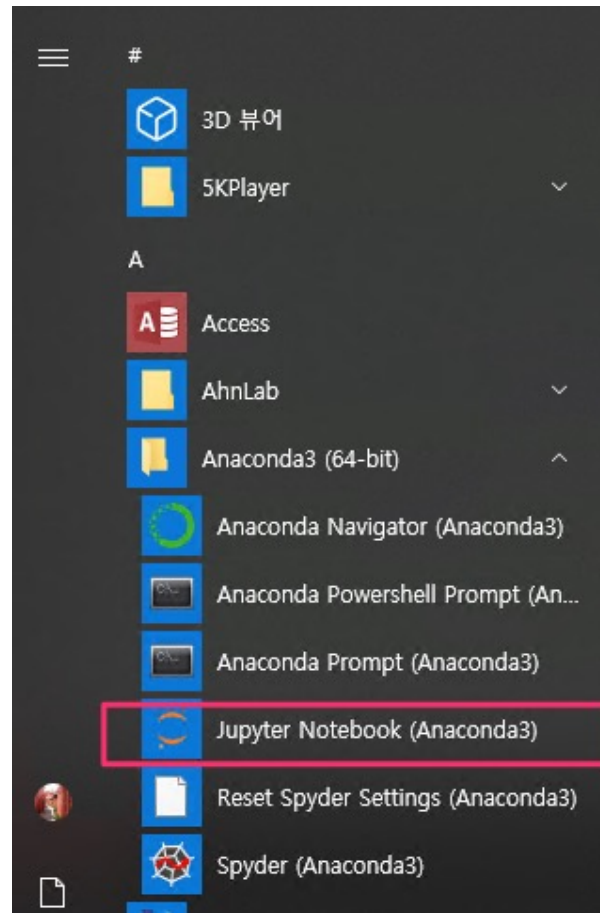


Jupyter Notebook 사용법

대화형 프로그래밍 기초

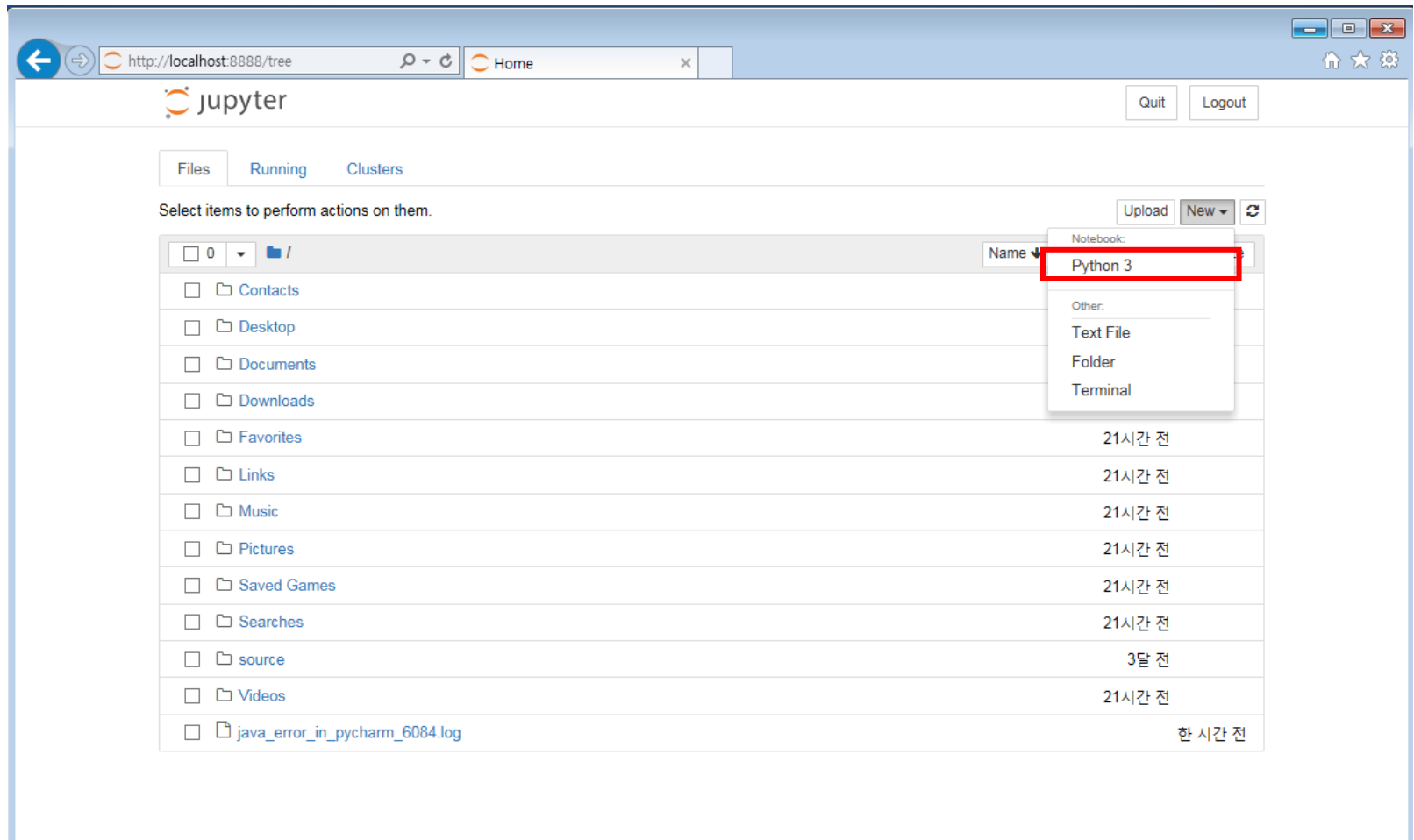
Jupyter Notebook 실행하기

- Anaconda 패키지 설치
 - Anaconda3 > Jupyter Notebook (Anaconda) 실행



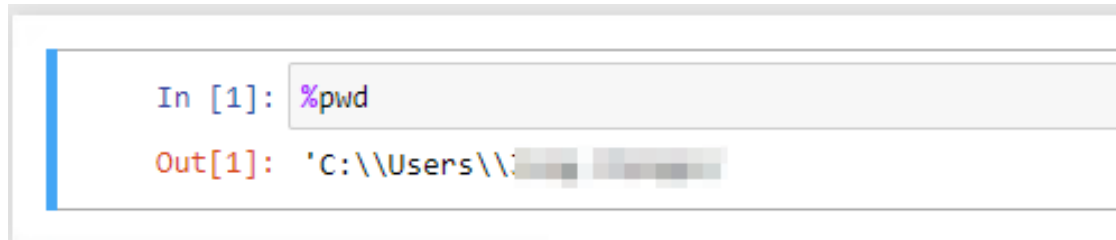
실행 화면 #1

- Jupyter notebook 실행
 - New > Python3 선택 > 새로운 노트북 생성됨



현재 경로 확인 (%pwd)

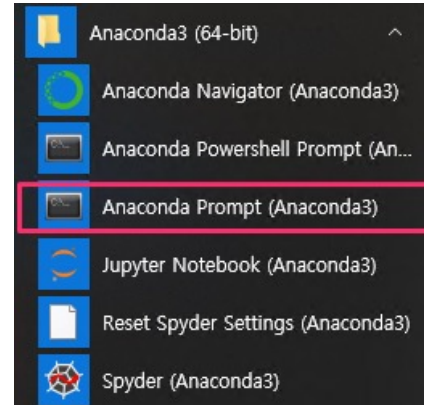
- jupyter notebook에서 `%pwd` 입력 후 `Ctrl + Enter`
 - Out[1]에 현재 실행 경로를 출력함
 - 현재 경로는 C:\Users\사용자 이름



The screenshot shows a Jupyter Notebook interface. The input area contains the text `In [1]: %pwd`. The output area below it shows `Out[1]: 'C:\\Users\\[redacted]'`, where the user name is redacted with grey boxes.

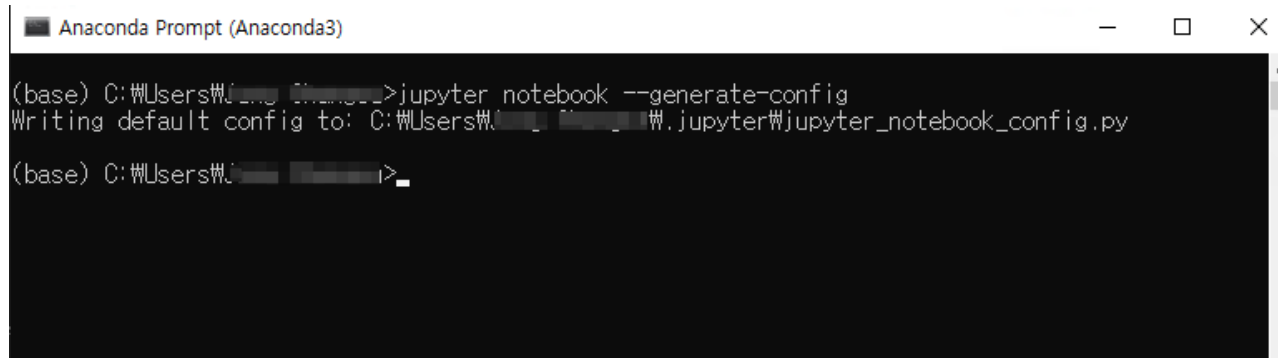
jupyter notebook 환경 설정 #1

- Anaconda prompt 실행



- 아래 명령어 입력하고 실행

```
C:\Users\사용자이름>jupyter notebook --generate-config
```

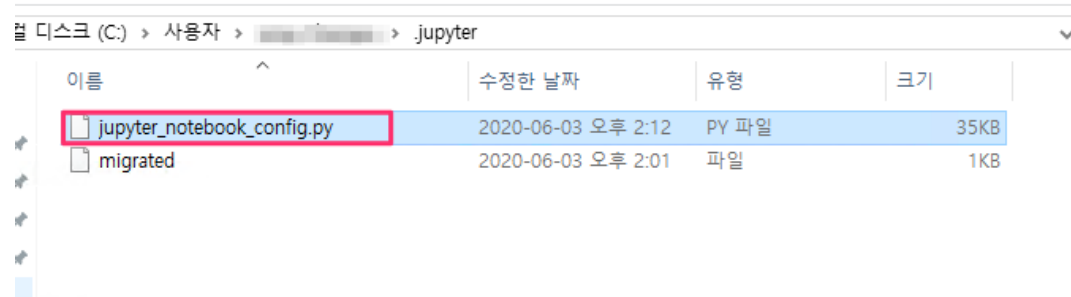
A screenshot of the Anaconda Prompt terminal window. The window title is 'Anaconda Prompt (Anaconda3)'. The terminal shows the command '(base) C:\Users\사용자이름>jupyter notebook --generate-config' being executed. The output is 'Writing default config to: C:\Users\사용자이름\AppData\Local\anaconda3\jupyter\notebook_config.py'. The prompt then shows '(base) C:\Users\사용자이름>_'.

```
(base) C:\Users\사용자이름>jupyter notebook --generate-config
Writing default config to: C:\Users\사용자이름\AppData\Local\anaconda3\jupyter\notebook_config.py
(base) C:\Users\사용자이름>_
```

- C:\Users\사용자이름\.jupyter\jupyter_notebook_config.py 생성됨
- 맥사용자: /Users/{사용자명}/.jupyter

jupyter notebook 환경 설정 #2

▪ jupyter_notebook_config.py 파일 편집



▪ Jupyter notebook을 저장할 폴더 생성

- ① `c:\jupyter_notebook` 폴더 생성
- ② `# c.NotebookApp.notebook_dir=' '항목 편집 (375라인)`
 - 주석 해제 및 시작 경로 추가 (해당 폴더 생성해야 됨)

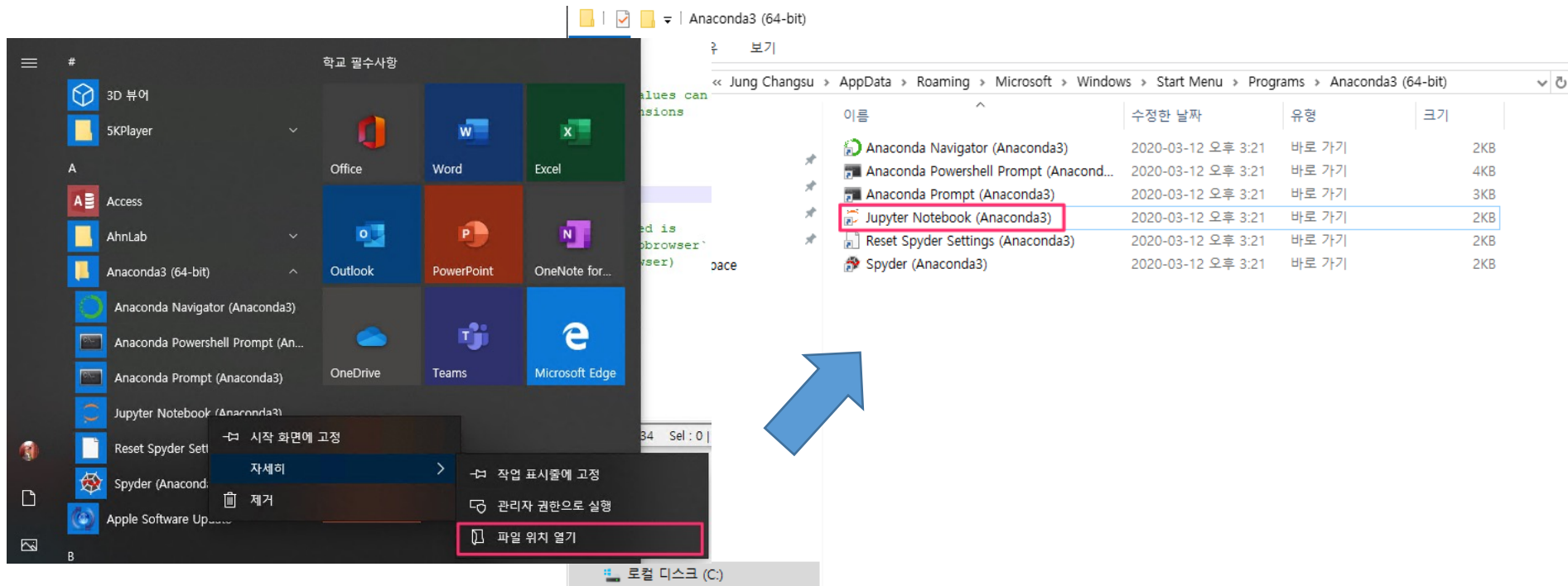
```
c.NotebookApp.notebook_dir = 'C:\jupyter_notebook'
```

- ③ cmd 창에서 Ctrl + C 를 눌러서 jupyter server 종료
- Anaconda3 > Jupyter Notebook 다시 실행
- **Browser가 자동 실행 되지 않는 경우:** `c.NotebookApp.browser` 검색

```
c.NotebookApp.browser = 'C:/Program Files (x86)/Google/Chrome/Application/chrome.exe %s'
```

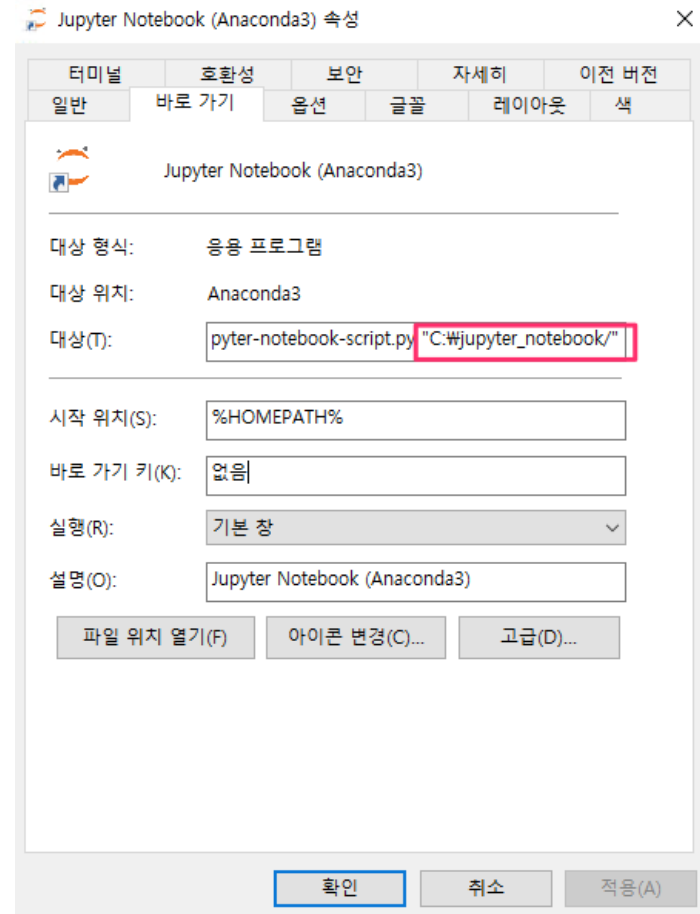
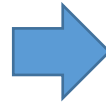
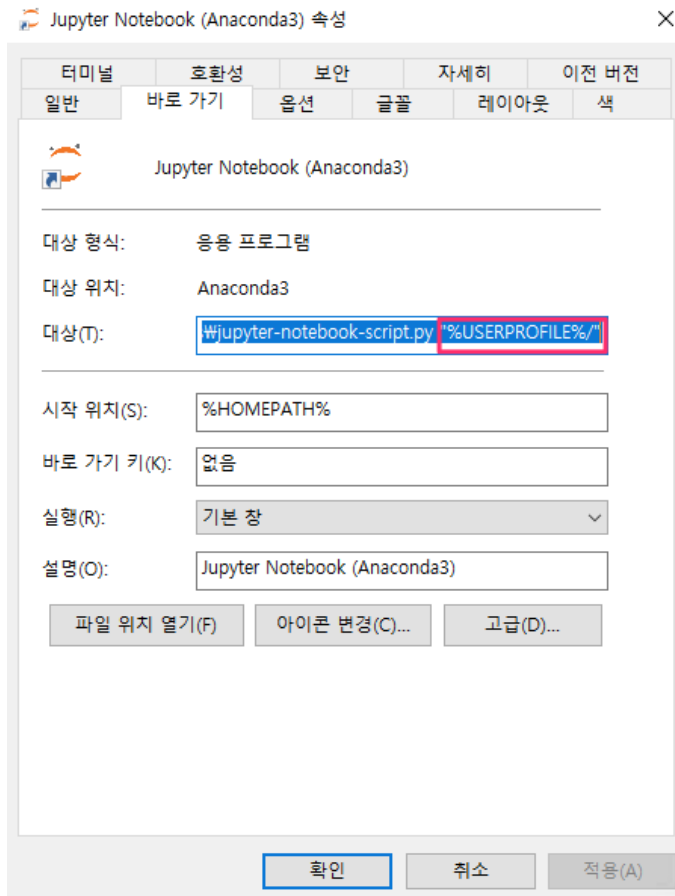
jupyter notebook 환경 설정 #3

- Jupyter Notebook 파일 위치 열기
- 마우스 오른쪽 버튼 > 속성 메뉴 선택



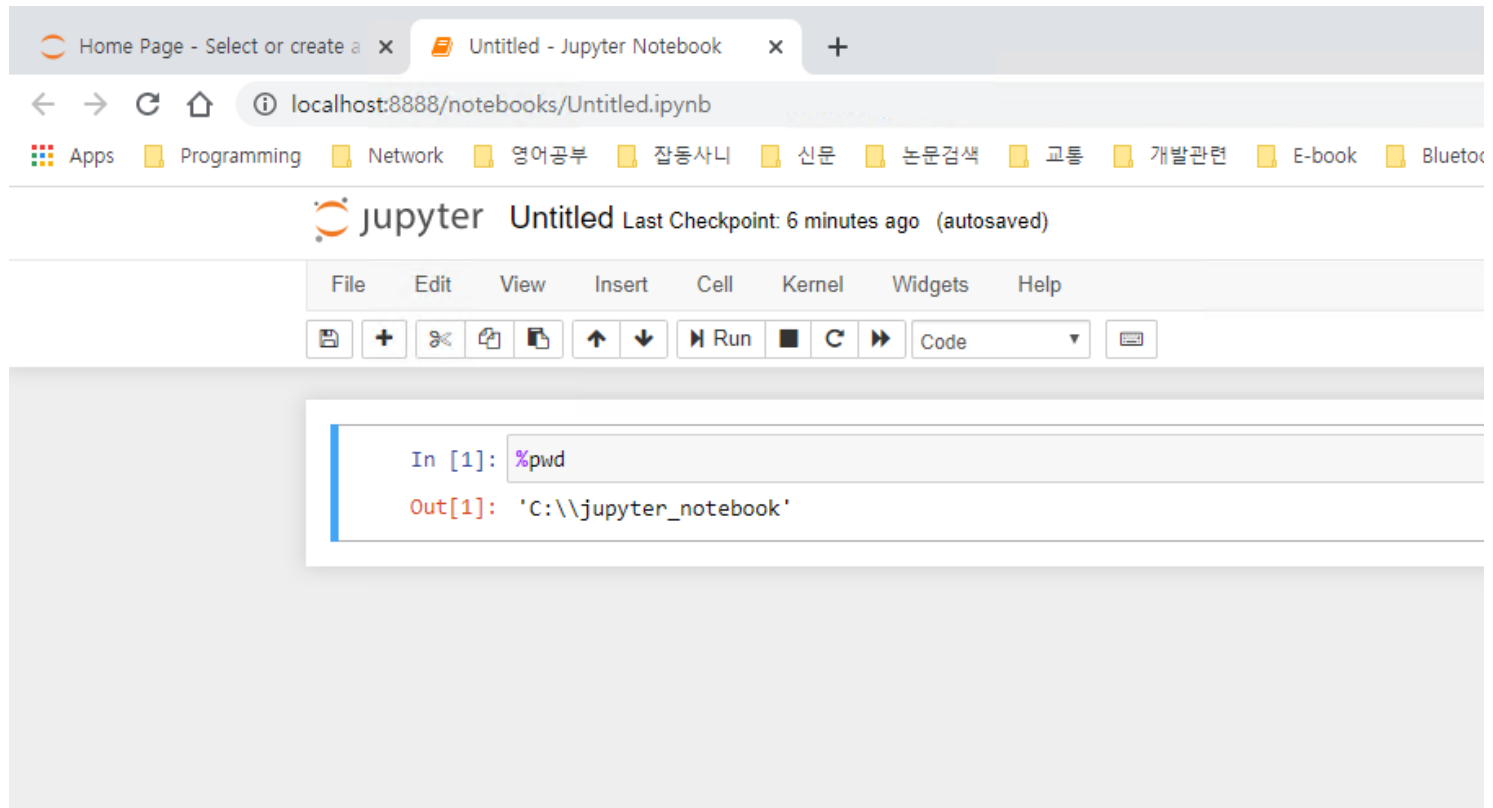
jupyter notebook 환경 설정 #4

- “%USERPROFILE%” 부분 수정
 - “C:\jupyter_notebook/”
- Jupyter Notebook 다시 실행



경로 변경 확인

- jupyter notebook 실행 후 %pwd 입력
 - 변경된 경로 확인



프로그래밍 폰트

■ 프로그래밍 폰트

- 고정폭 폰트(Monospace font)
- 아래 표와 같이 숫자(1, 0), 영문자(1, o)를 명확히 구분할 수 있음
- 코딩시 발생하는 오타 확인이 쉬움
- 소스 코드의 가독성이 증대
 - Consolas: Windows에 기본 설치되어 있음

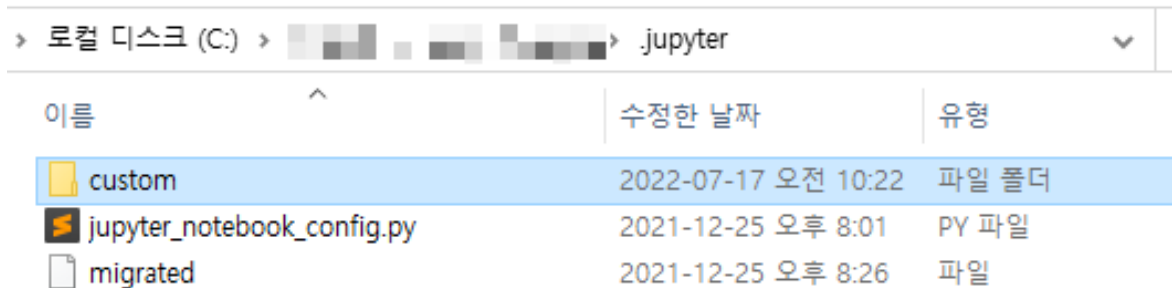
구분	폰트명	숫자	영문자	한글	특수기호
1자 형태	굴림체	1			
	Consolas	1	1		
	D2 Coding	1	l		
0자 형태	굴림체	0	0	o	
	Consolas	0	0	o	
	D2 Coding	0	0	o	
코드 기호	굴림체	() { } [] " ' : ; , . , - - + + = = _ _ /			
	Consolas	() { } [] " ' : ; , . , - - + + = = _ _ /			
	D2 Coding	() { } [] " ' : ; , . , - - + + = = _ _ /			

■ 프로그래밍 폰트 다운로드 및 설치

<https://futurecreator.github.io/2018/11/12/my-best-programming-font-top-3/>

Jupyter notebook 폰트 변경 #1

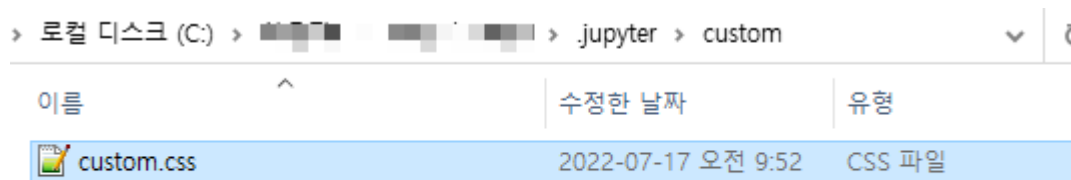
- .jupyter 폴더 아래에 **custom** 폴더 생성



로컬 디스크 (C:) > [숨김 폴더] > .jupyter

이름	수정한 날짜	유형
custom	2022-07-17 오전 10:22	파일 폴더
jupyter_notebook_config.py	2021-12-25 오후 8:01	PY 파일
migrated	2021-12-25 오후 8:26	파일

- custom.css 파일 복사

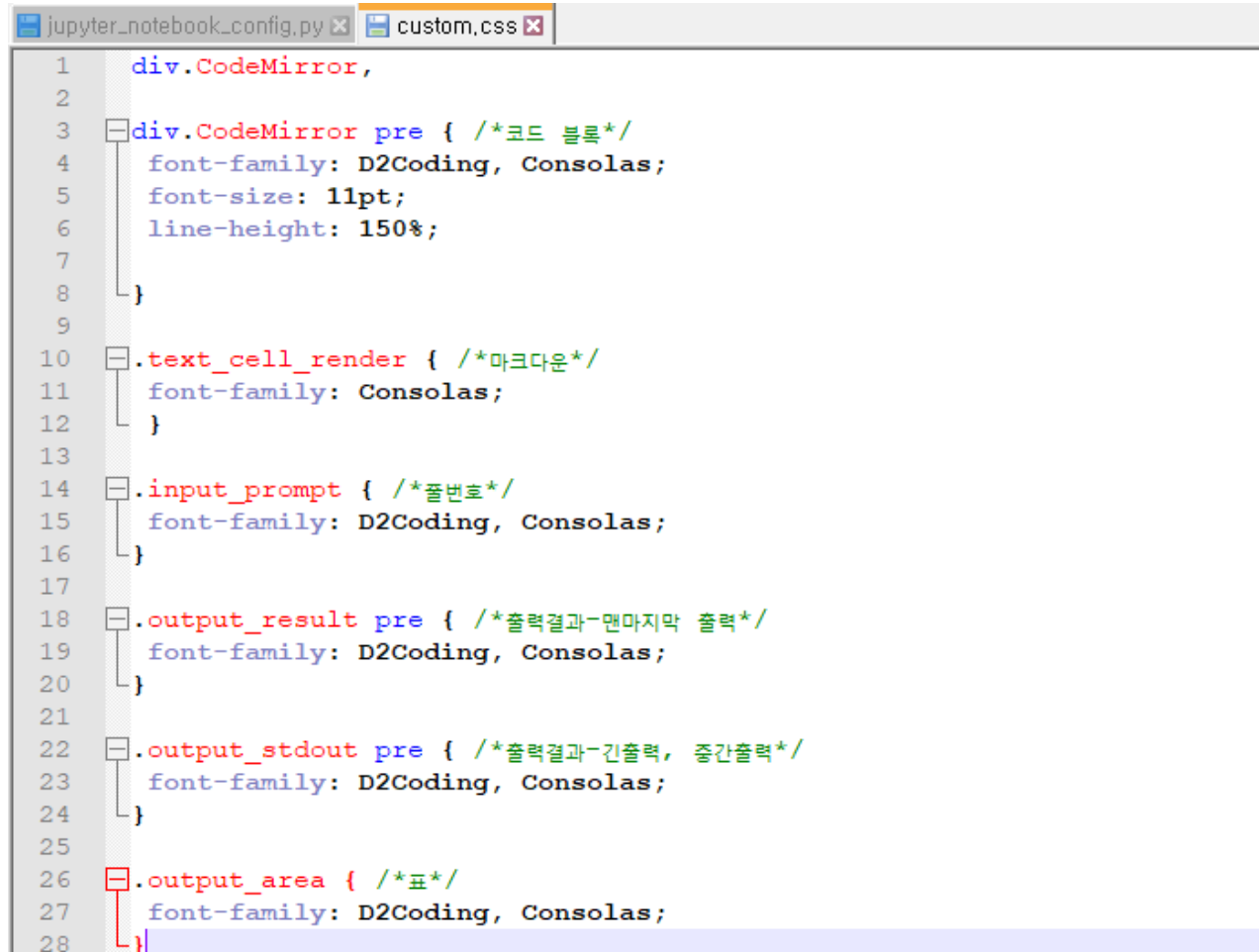


로컬 디스크 (C:) > [숨김 폴더] > .jupyter > custom

이름	수정한 날짜	유형
custom.css	2022-07-17 오전 9:52	CSS 파일

Jupyter notebook 폰트 변경 #2

custom.css 파일 내용

A screenshot of a Jupyter Notebook interface. The top bar shows two tabs: 'jupyter_notebook_config.py' and 'custom.css'. The 'custom.css' tab is active, displaying a CSS file with 28 lines of code. The code defines styles for various Jupyter Notebook components, including code mirrors, text cells, input prompts, and output areas. The styles specify the font family as 'D2Coding, Consolas' and the font size as '11pt'. The line height for code mirrors is set to '150%'. The code is color-coded with syntax highlighting.

```
1  div.CodeMirror,
2
3  div.CodeMirror pre { /*코드 블록*/
4      font-family: D2Coding, Consolas;
5      font-size: 11pt;
6      line-height: 150%;
7  }
8
9
10 .text_cell_render { /*마크다운*/
11     font-family: Consolas;
12 }
13
14 .input_prompt { /*줄번호*/
15     font-family: D2Coding, Consolas;
16 }
17
18 .output_result pre { /*출력결과-맨마지막 출력*/
19     font-family: D2Coding, Consolas;
20 }
21
22 .output_stdout pre { /*출력결과-긴출력, 중간출력*/
23     font-family: D2Coding, Consolas;
24 }
25
26 .output_area { /*표*/
27     font-family: D2Coding, Consolas;
28 }
```

Jupyter notebook 다시 실행

Jupyter notebook 폰트 변경

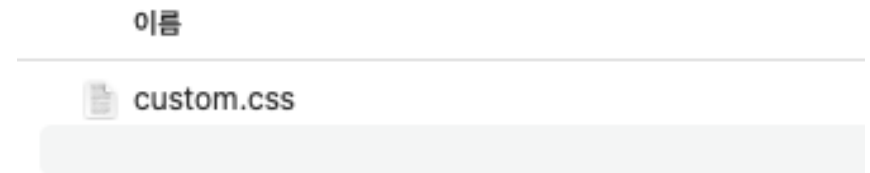
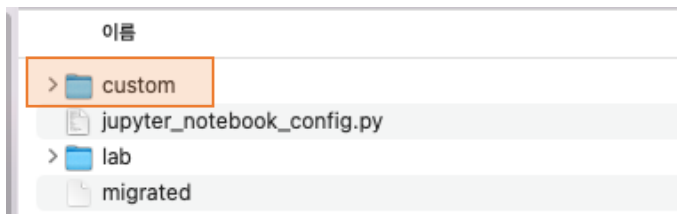
```
In [2]: import pandas as pd
import matplotlib.pyplot as plt

array_list = [0, 1, 2, 3, 4, 5]
for item in array_list:
    print(item, end=' ')
```

```
0 1 2 3 4 5
```

Mac: Jupyter notebook 폰트 변경

- 파일 경로
 - `/Users/{사용자명}/.jupyter` 폴더
- Finder에서 숨김 파일 보기
 - `Command + Shift + .`
- `/Users/{사용자명}/.jupyter/custom` 폴더 생성
 - `custom.css` 파일 복사



JupyterLab 사용

- Jupyter Notebook의 향상된 버전
 - jupyter notebook과 호환
- Anaconda Navigator 실행 > Jupyter Lab 실행

The image shows the Anaconda Navigator desktop application. On the left is a dark sidebar with a list of recently added applications. 'Anaconda Navigator (Anaconda3)' is highlighted with a red box. A large black arrow points from this box to the main window. The main window displays a grid of applications available on the 'base (root)' channel. 'JupyterLab' is the first application in the grid, with its 'Launch' button highlighted by a red rectangle. Other applications include Jupyter Notebook, PyCharm, Qt Console, RStudio, Spyder, VS Code, Glueviz, and Orange 3. The interface includes a top bar with the Anaconda Navigator logo, an 'Upgrade Now' button, and a 'Sign In to Anaconda.org' button. The bottom of the sidebar contains links to 'Documentation' and 'Developer Blog', along with social media icons for Twitter, YouTube, and GitHub.

최근에 추가한 앱

- Jupyter Notebook (Anaconda3)
- 3D 뷰어
- 5KPlayer
- Access
- Acrobat Reader DC
- AhnLab
- AhnLab V3 Internet Security 9.0
- Anaconda3 (64-bit)
- Anaconda Navigator (Anaconda3)**
- Anaconda Powershell Prompt (An...
- Anaconda Prompt (Anaconda3)
- Jupyter Notebook (Anaconda3)
새로 설치됨
- Reset Spyder Settings (Anaconda3)
- Spyder (Anaconda3)
- Android Studio

ANACONDA NAVIGATOR

Home

Environments

Learning

Community

Applications on base (root) Channels Refresh

JupyterLab 2.2.6
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Jupyter Notebook 6.1.4
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

PyCharm 2021.1.1
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific enhancements for web development and data science.
Launch

Qt Console 5.0.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

RStudio 1.1.456
A set of integrated tools designed to help you be more productive with R, includes R essentials and notebooks.
Launch

Spyder 4.2.1
Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.
Launch

VS Code 1.56.2
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch

Glueviz 1.0.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install

Orange 3 3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert: interactive workflows with a large toolbox.
Install

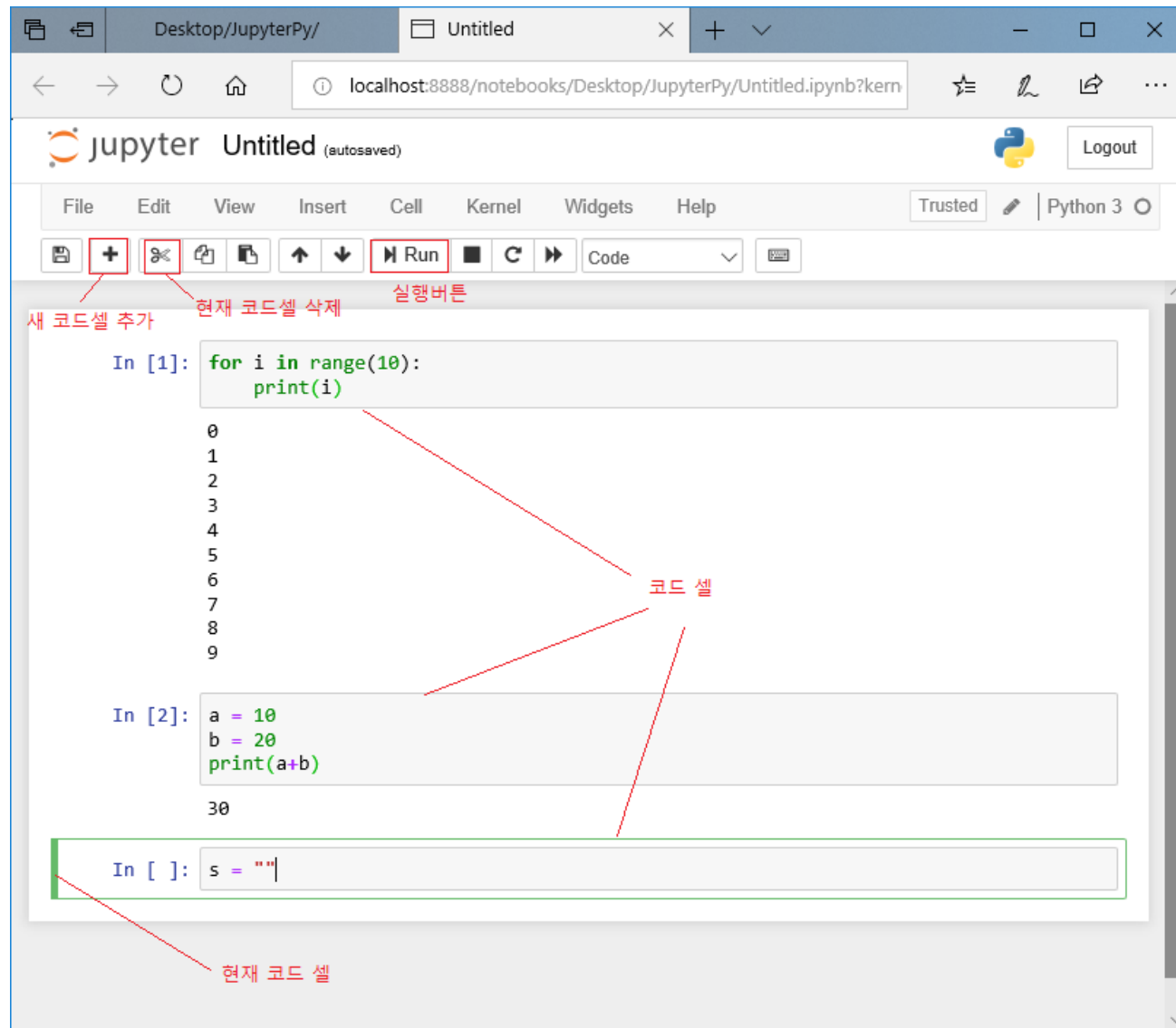
Documentation

Developer Blog

Twitter YouTube GitHub

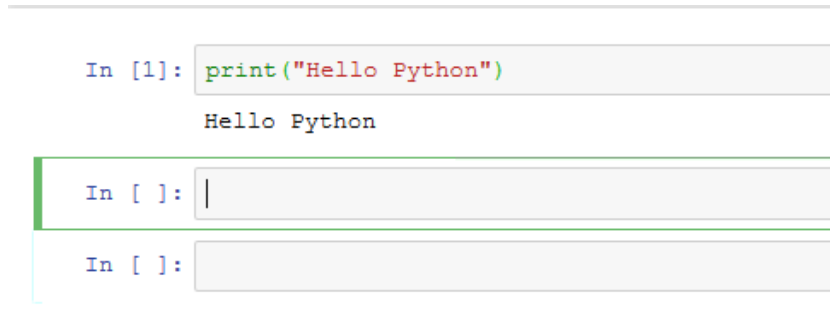
15

Jupyter notebook 화면 구성



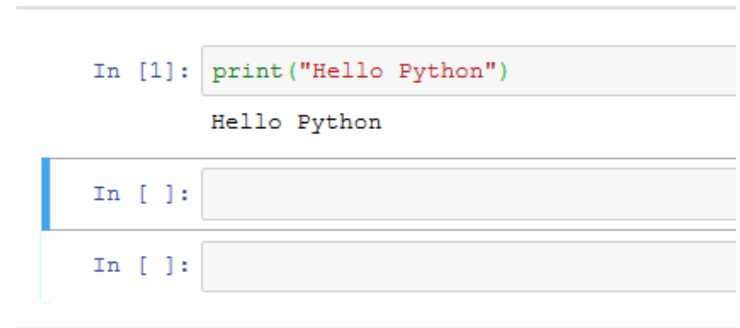
Jupyter notebook 인터페이스

[Edit Mode] $\xrightarrow{\text{ESC Key}}$ [Command Mode]



The image shows a Jupyter notebook cell in Edit Mode. The cell contains the code `print("Hello Python")` and the output `Hello Python`. Below the cell, there are two empty input fields for new code, each preceded by `In []:`. The first input field has a cursor. The cell is highlighted with a green border.

Python 코드 입력 또는
Markdown 작성하는 모드



The image shows a Jupyter notebook cell in Command Mode. The cell contains the code `print("Hello Python")` and the output `Hello Python`. Below the cell, there are two empty input fields for new code, each preceded by `In []:`. The cell is highlighted with a blue border.

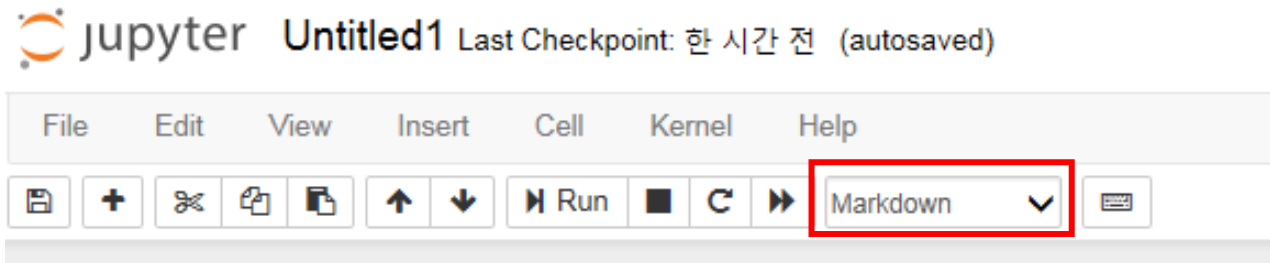
명령어를 입력하기 위한 모드

- a: 위칸에 셀 추가
- dd: 셀 삭제

Ctrl + Enter: 해당 셀 실행

Markdown 사용

- Jupyter notebook에서 **Markdown** 선택



```
# Markdown 설명
## 중제목
### 소제목
* 구분점
  * 구분점 1
  * **볼드체**
1. 하나
2. 둘
3. Link: [Google] http://www.google.com
4. `simplify()` 함수
```

실행



Markdown 설명

중제목

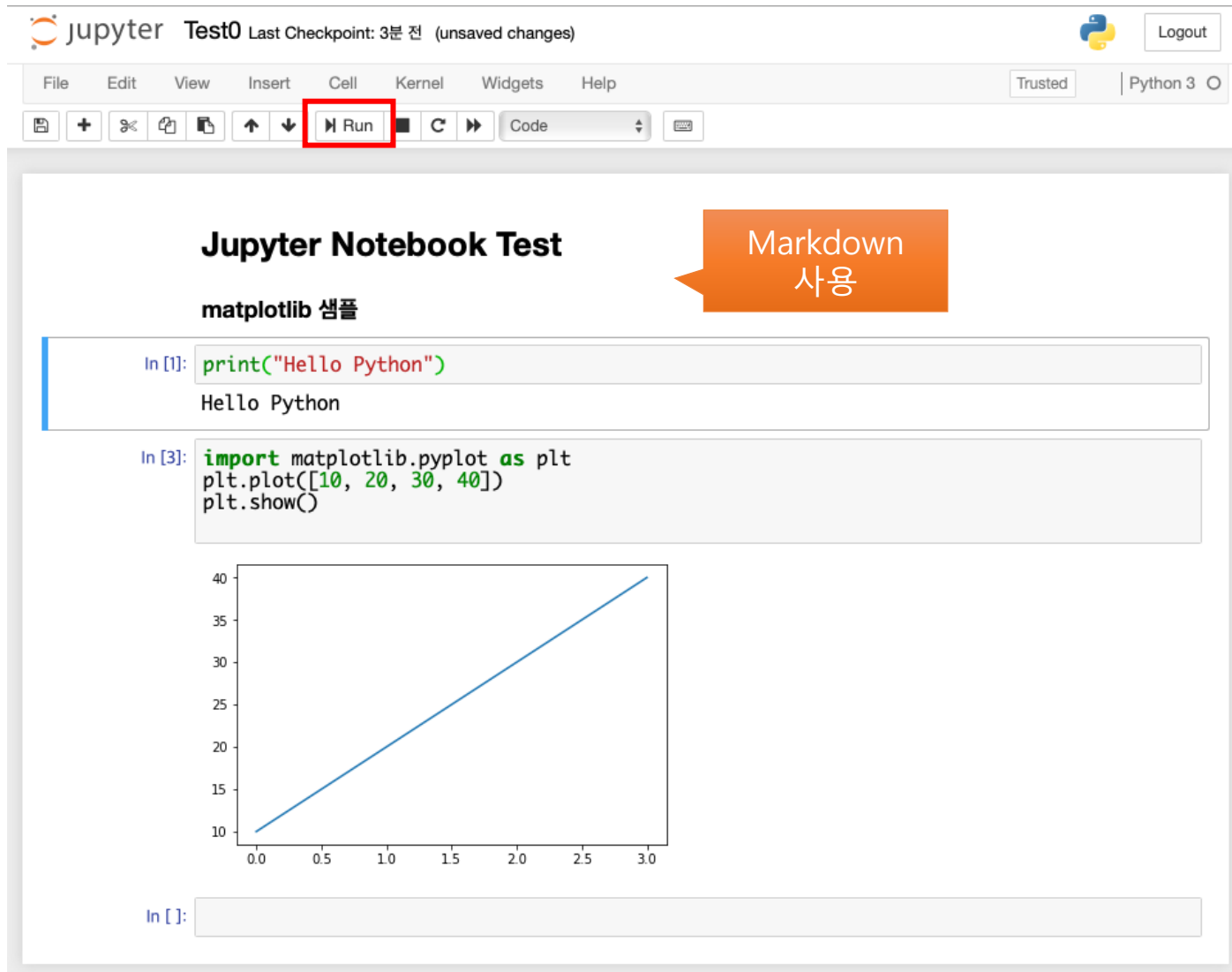
소제목

- 구분점a
 - 구분점 1
 - 볼드체
- 하나
- 둘
- Link: [Google] <http://www.google.com>
- `simplify()` 함수

Markdown 작성 후 실행 (Control + Enter)

Jupyter notebook 실행

■ 소스 코드 입력 > Run 실행



The screenshot displays the Jupyter Notebook interface. At the top, the header shows 'jupyter Test0' with a 'Last Checkpoint: 3분 전 (unsaved changes)' status and a 'Logout' button. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar contains various icons, with the 'Run' button (a play icon) highlighted by a red rectangle. The main content area is titled 'Jupyter Notebook Test' and includes a section for 'matplotlib 샘플'. An orange callout bubble points to the 'Markdown 사용' (Use Markdown) option. The notebook contains two code cells. The first cell, labeled 'In [1]:', contains the code `print("Hello Python")` and has executed, showing the output 'Hello Python'. The second cell, labeled 'In [3]:', contains the code `import matplotlib.pyplot as plt`, `plt.plot([10, 20, 30, 40])`, and `plt.show()`. Below the code, a line plot is displayed with a blue line. The x-axis ranges from 0.0 to 3.0, and the y-axis ranges from 10 to 40. The plot shows a linear increase from (0.0, 10) to (3.0, 40). At the bottom, there is an empty code cell labeled 'In []:'.

jupyter Test0 Last Checkpoint: 3분 전 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run

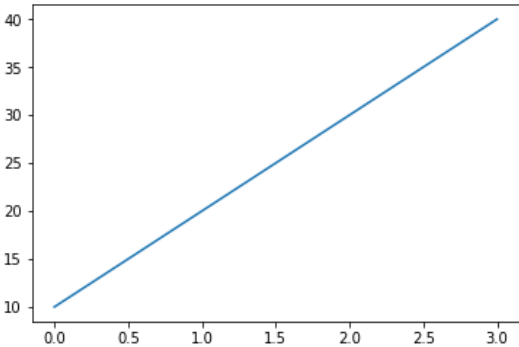
Jupyter Notebook Test

Markdown 사용

matplotlib 샘플

In [1]: `print("Hello Python")`
Hello Python

In [3]: `import matplotlib.pyplot as plt`
`plt.plot([10, 20, 30, 40])`
`plt.show()`



In []:

Jupyter notebook 단축키

■ 단축키 공통

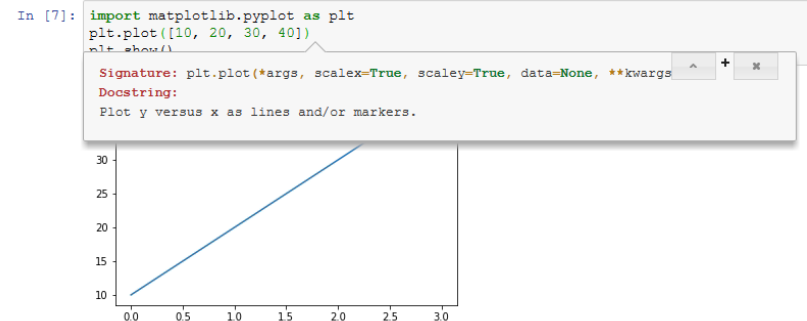
- **Shift + Enter**: 셀 실행 및 아래 셀 추가
- **Control + Enter**: 셀 실행
- **ESC**: Command mode 진입
- **Enter**: 에디터 모드
- **Control + Shift + P**: show command palette

■ Command mode (press ESC to enable)

- **A**: 위쪽에 셀 추가
- **B**: 아래쪽에 셀 추가
- **DD**: 셀 삭제
- **Shift + M**: 셀 아래 병합

■ Edit mode (press Enter to enable)

- **Control + Shift + -**: 셀 분할
- **Control +]** (Tab 키): 들여 쓰기
- **Control + [**: 내어 쓰기
- **Shift + Tab**: 툴팁 표시



매직 명령어(%)

■ 매직 명령어

- 맨 앞에 '%'를 붙여서 특정 명령어 수행
- 파이썬 문법에는 포함되지 않은 Jupyter notebook만의 기능임
- %: line
- %%: cell 지정

매직 명령어	설 명
%pwd	현재 디렉토리 경로 출력
%time 코드	코드의 실행 시간을 측정하여 표시
%timeit 코드	코드를 여러 번 실행한 결과를 요약하여 표시
%history -l 3	최근 3개이 코드 실행 이력 취득
%ls	윈도우의 dir, Linux의 ls 명령어와 같음
%autosave 숫자	자동 저장 주기를 설정 (초 단위, 0이면 무효로 함)
%matplotlib inline	코드 셀의 바로 아래에 그래프를 그림
%run script.py	파이썬 모듈 실행
%prun 함수명	함수를 실행 시킬때 어느 부분에서 시간이 가장 많이 소요되는지 출력

매직 명령어 예제 #1

▪ timeit 명령어

- ex) `timeit -n 500 -r 10`: (-n: 회당 수행 회수, -r: 몇 회)

```
%%timeit -n 500 -r 10
def test_func(s):
    return sum([i for i in range(0, s)])
test_func(100)
```

4.09 μ s \pm 331 ns per loop (mean \pm std. dev. of 10 runs, 500 loops each)

▪ time 명령어

```
%time sum(range(10000))
```

CPU times: user 192 μ s, sys: 4 μ s, total: 196 μ s Wall time: 205 μ s

49995000

매직 명령어 예제 #2

▪ %prun

- 시간 측정 및 함수 실행시 시간이 많이 소요되는 부분 출력

```
from time import sleep
def wait_one_second():
    sleep(1)
def wait_half_second():
    sleep(0.5)
def prun_example():
    wait_one_second()
    wait_one_second()
    wait_half_second()

%prun prun_example()
```

- ncalls: 함수가 호출된 회수
- tottime: 함수 내에서 소요된 시간
- percall: tottime / ncalls
- cumtime: 다른 함수를 호출할 때 그 함수내에서 소요된 시간 포함
- percall: cumtime / ncalls

10 function calls in 2.506 seconds Ordered by: internal time

```
ncalls tottime percall cumtime percall filename:lineno(function)
3 2.506 0.835 2.506 0.835 {built-in method time.sleep}
1 0.000 0.000 2.506 2.506 {built-in method builtins.exec}
1 0.000 0.000 2.506 2.506 <ipython-input-8-b2d5635b5daf>:8(prun_example)
2 0.000 0.000 2.004 1.002 <ipython-input-8-b2d5635b5daf>:3(wait_one_second)
1 0.000 0.000 0.503 0.503 <ipython-input-8-b2d5635b5daf>:5(wait_half_second)
1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects}
1 0.000 0.000 2.506 2.506 <string>:1(<module>)
```