

기초빅데이터프로그래밍

Anaconda



Anaconda

Build machine learning models

Build and train machine learning models using the best Python packages built by the open-source community, including scikit-learn, TensorFlow, and PyTorch.



환경설정

- Anaconda
 - 파이썬 통합 개발환경
 - jupyter notebook 지원
 - 라이브러리 패키지 관리 및 환경 설정 쉽게 해줌
 - <https://www.anaconda.com/distribution>

Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

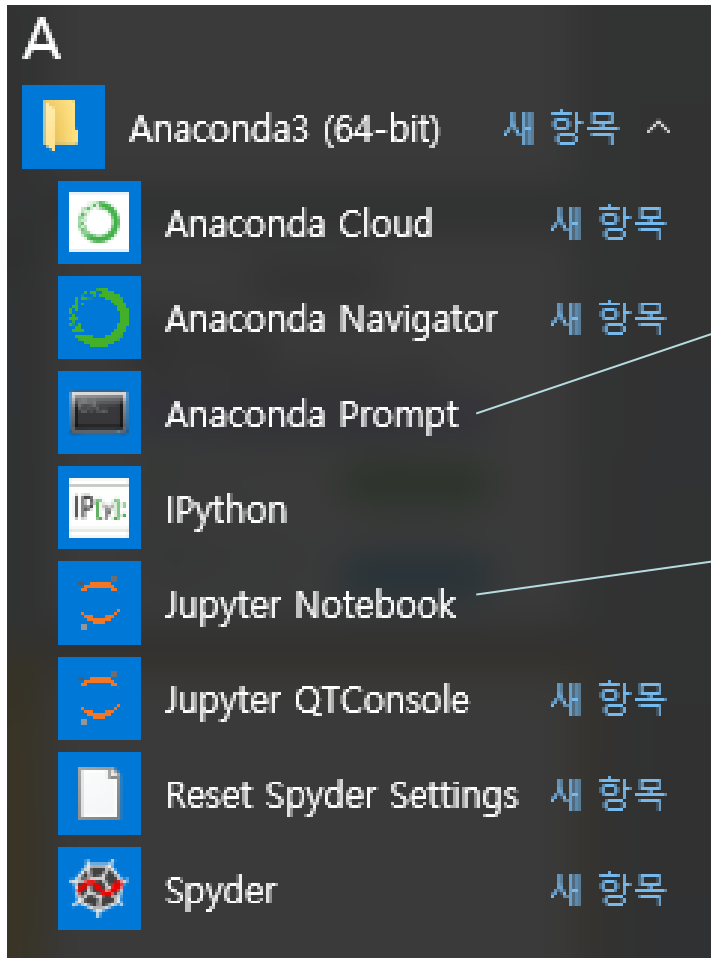
64-Bit Command Line Installer (454 MB)

Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

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

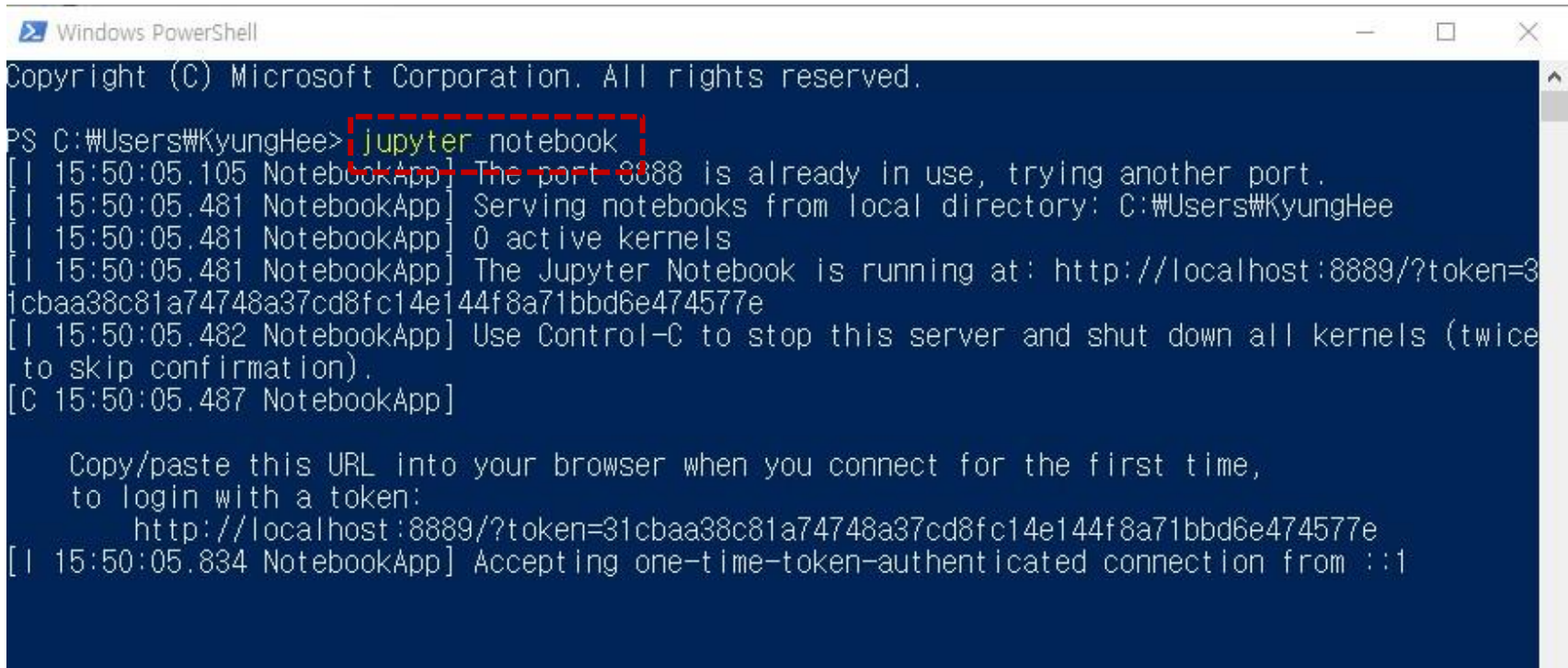


Anaconda 관리 터미널
패키지 업데이트 및 관리

Jupyter notebook
웹 환경에서의 개발

Jupyter notebook 실행하기

- Jupyter notebook 실행
 - Windows PowerShell 이용한 실행
 - 파워셸 실행 후 **jupyter notebook**

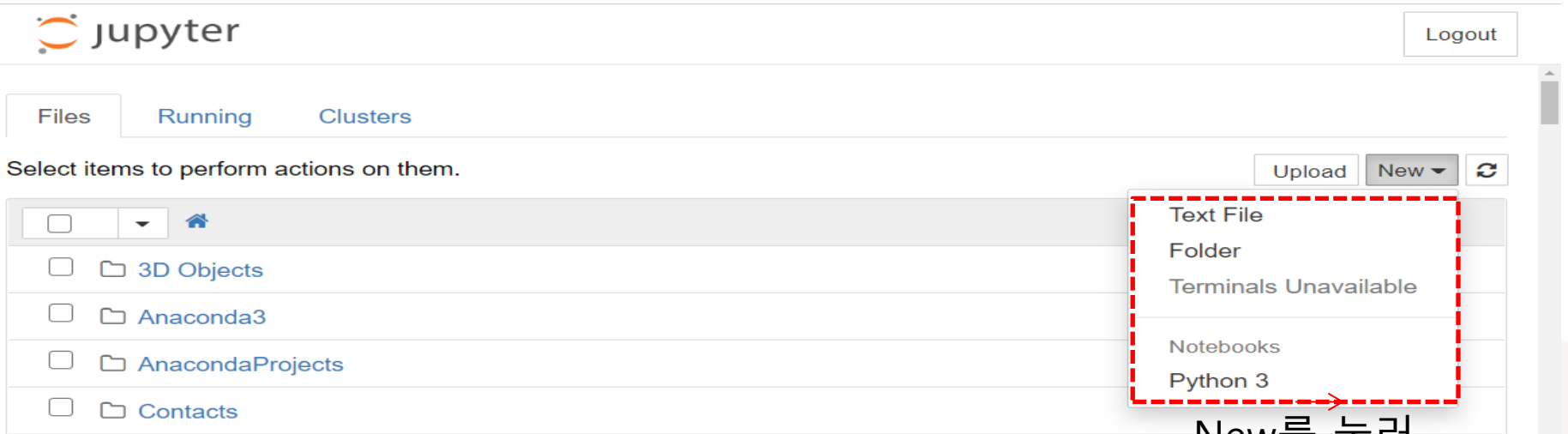


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

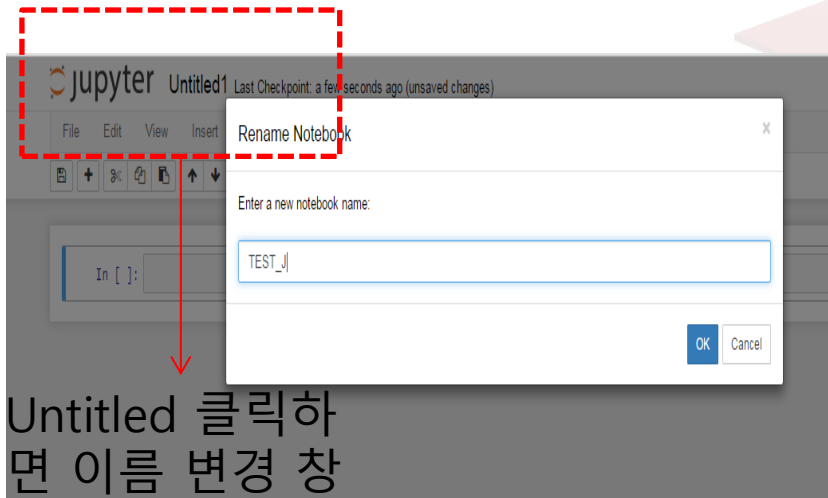
PS C:\Users\KyungHee> jupyter notebook
[I 15:50:05.105 NotebookApp] The port 8888 is already in use, trying another port.
[I 15:50:05.481 NotebookApp] Serving notebooks from local directory: C:\Users\KyungHee
[I 15:50:05.481 NotebookApp] 0 active kernels
[I 15:50:05.481 NotebookApp] The Jupyter Notebook is running at: http://localhost:8889/?token=31cbaa38c81a74748a37cd8fc14e144f8a71bbd6e474577e
[I 15:50:05.482 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 15:50:05.487 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8889/?token=31cbaa38c81a74748a37cd8fc14e144f8a71bbd6e474577e
[I 15:50:05.834 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

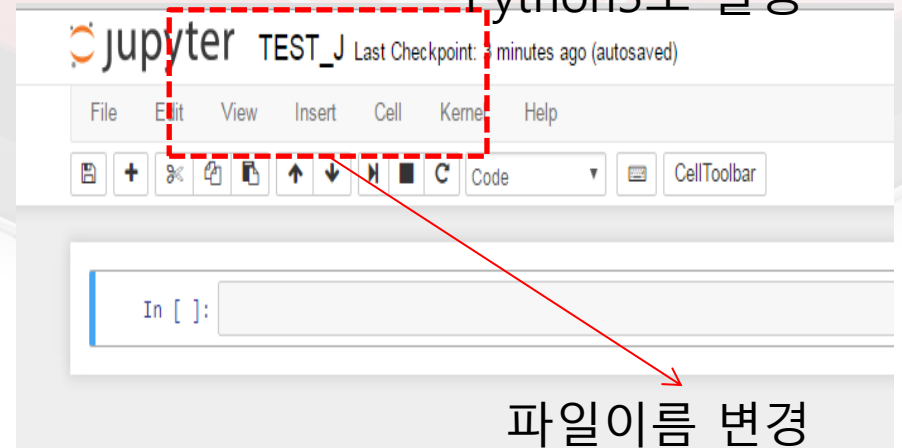
Jupyter notebook 커널 실행



New를 눌러
Python3로 실행

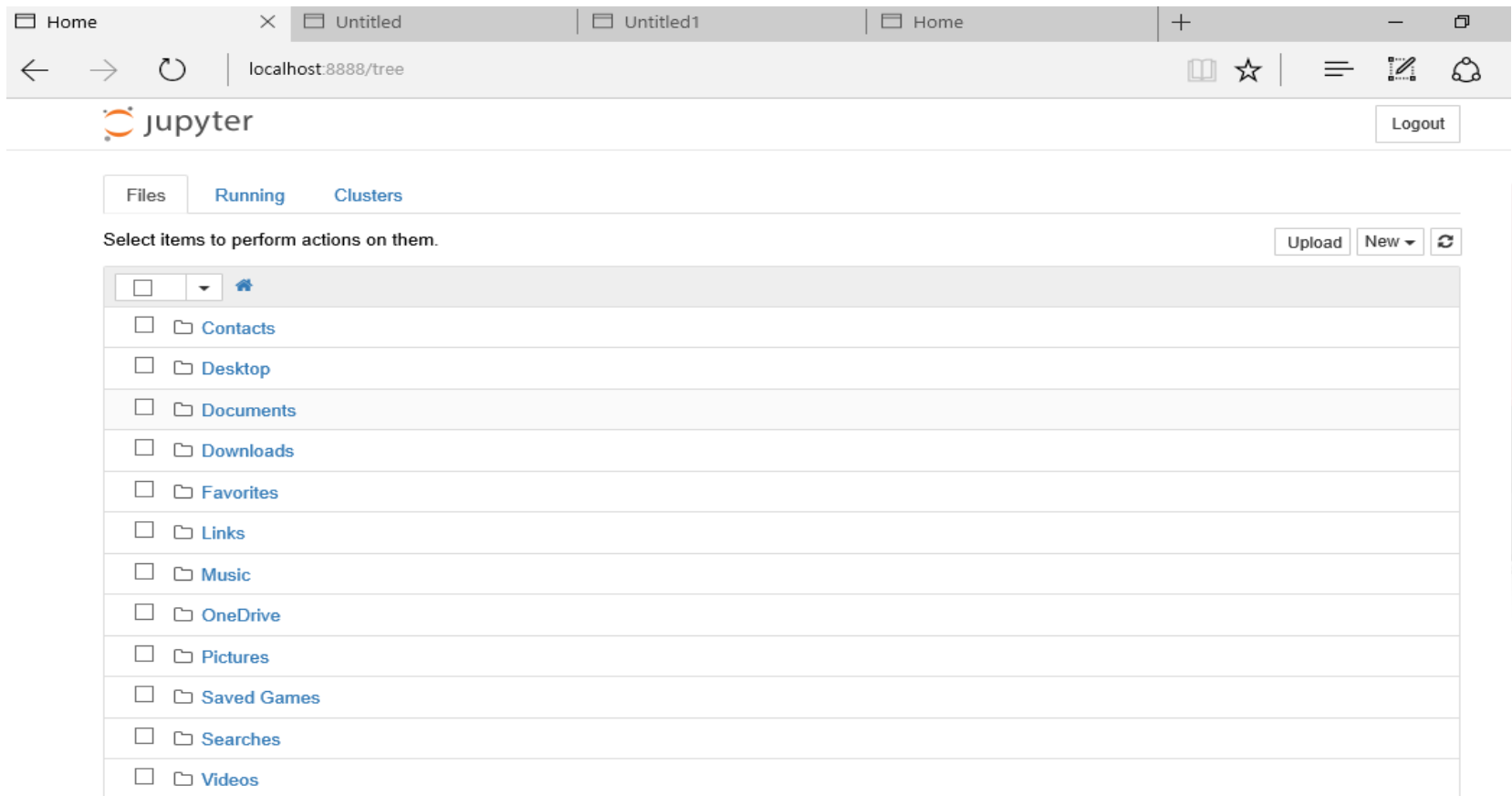


Untitled 클릭하
면 이름 변경 창
이 나오고 이름
변경



파일이름 변경

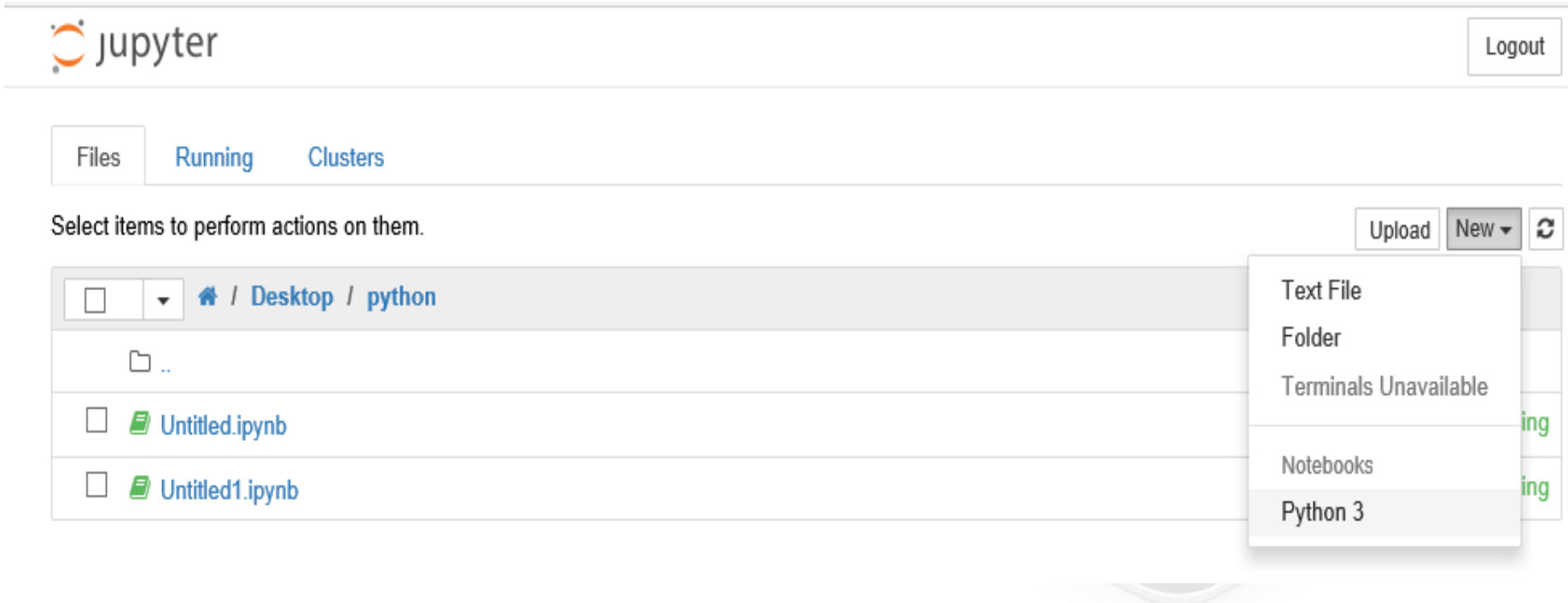
- 자동으로 웹브라우저가 실행되면서 로컬환경에서 notebook 실행

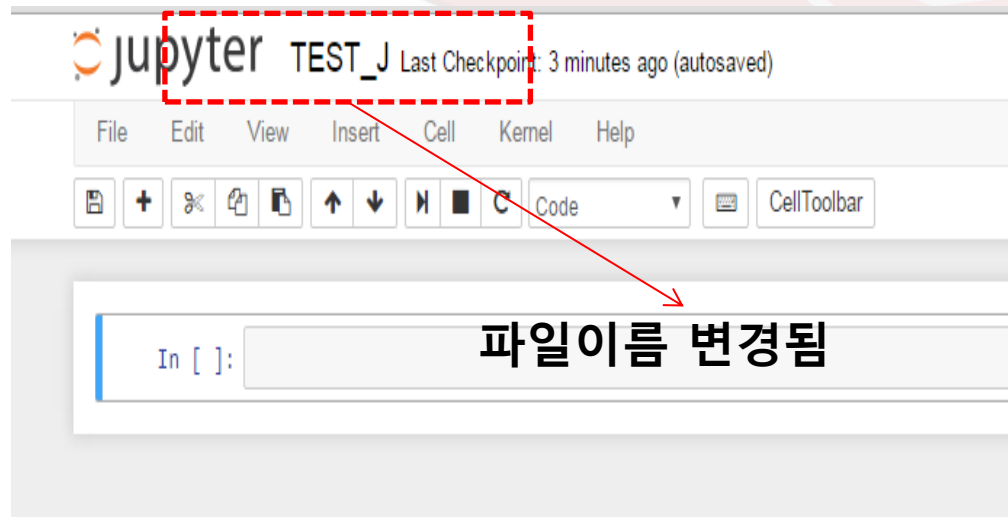
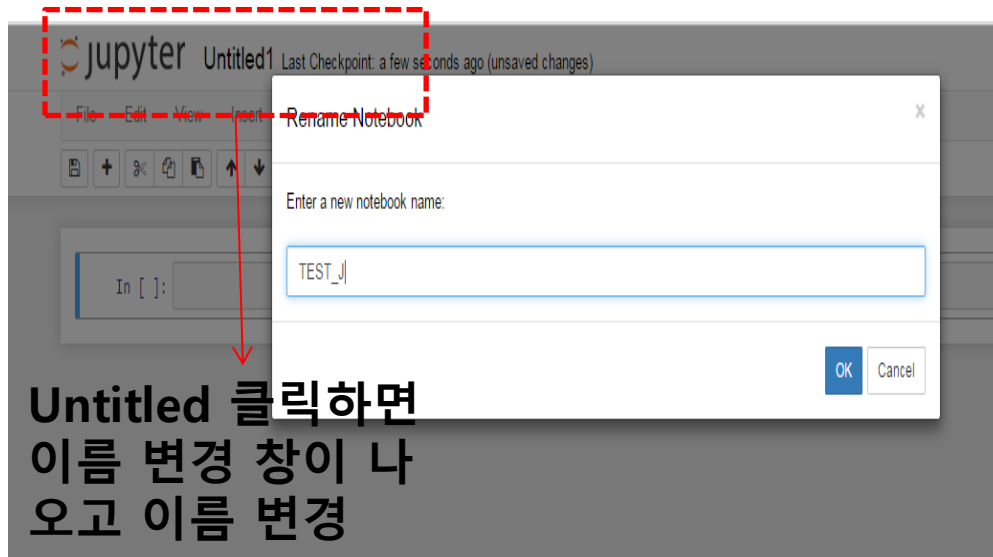


- 파이썬 코드를 저장할 곳을 만들거나 선택



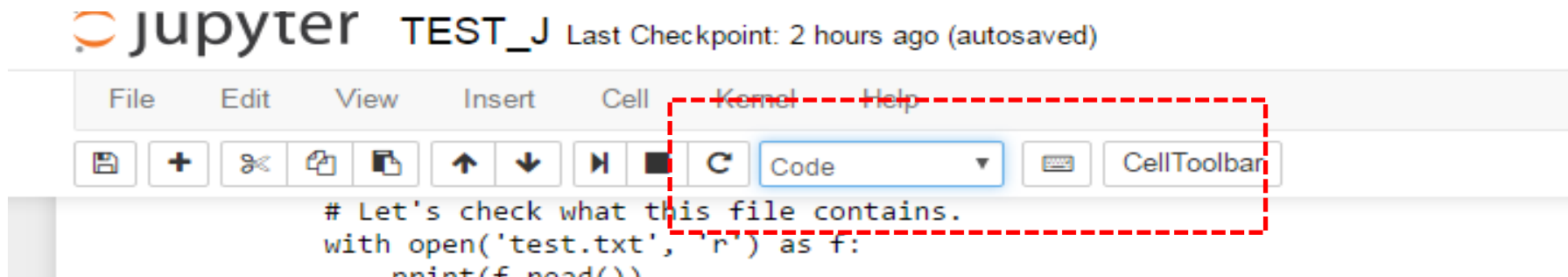
- New 버튼을 눌러 새로운 파이썬 파일을 생성할 수 있다





Cell type : code

Cell에 Python 코드가 입력되어 실행



- 코드는 셀 단위로 입력하고 실행시킬 수 있다.
- In [1] : print("hello world") 를 입력한 뒤 플레이버튼 혹은 단축키 **Shift+Enter**을 눌러 코드를 실행시켜보자

```
In [1]: print("hello world")
hello world
```

Magic command



Magic command

magic command에는 line(%)과 cell(%%)로 지정해서 처리할 수 있음

line

%magic command

Ex: %run foo.py 는 file foo.py를 실행

cell

%%magic command

Ex: %%latex 는 cell 전체에 대해 latex를 실행

Magic command 조회

%lsmagic을 이용해서 가지고 있는 command를 전체 조회

In [42]: %lsmagic

Out[42]: Available line magics:

```
%alias %alias_magic %autocall %automagic %autosave %bookmark %cd %clear %cls %colors %config %connect_info %copy %ddir %debug %dhist %dirs %doctest_mode %echo %ed %edit %env %gui %hist %history %install_default_config %install_ext %install_profiles %killbgscripts %ldir %less %load %load_ext %loadpy %logoff %logon %logstart %logstate %logstop %ls %lsmagic %macro %magic %matplotlib %mkdir %more %notebook %page %pastebin %pdb %pdef %pdoc %pfile %pinfo %pinfo2 %popd %pprint %precision %profile %prun %psearch %psource %pushd %pwd %pycat %pylab %qtconsole %quickref %recall %rehashx %reload_ext %ren %rep %rerun %reset %reset_selective %rmdir %run %save %sc %set_env %store %sx %system %tb %time %timeit %unalias %unload_ext %who %who_ls %whos %xdel %xmode
```

Available cell magics:

```
%%! %%HTML %%SVG %%bash %%capture %%cmd %%debug %%file %%html %%javascript %%latex %%perl %%prun %%pypy %%python %%python2 %%python3 %%ruby %%script %%sh %%svg %%sx %%system %%time %%timeit %%writefile
```

Automagic is ON, % prefix IS NOT needed for line magics.

주요 Magic command 1

%ismagic 내의 주요 명령어 설명

명령어	설명
%pwd, %cd	현재 위치 및 다른 디렉토리로 이동
%history	명령어 히스토리 출력
%reset	모든 정의된 변수 삭제
%whos	현재 정의된 변수 표시
%pdoc, %psource	Help 기능 실행
%timeit	평균 실행 시간을 출력
%bookmark	디렉토리에 대한 별칭을 저장하고 쉽게 이동할 수 있게 해줌
%%writefile	현재 디렉토리에 파일 생성
%load	디렉토리에 있는 파일을 셀에 로딩
%run	py 프로그램 파일을 실행
%matplotlib inline	matplotlib을 내부 셀에서 실행하기

주요 Magic command : 2

%lsmagic 내의 주요 명령어 설명

명령어	설명
%ls	현재 디렉토리에 파일들을 보기
%magic	모든 매직 함수에 대한 상세 도움말 출력
%pdb	예외가 발생하면 자동적으로 디버거 진입.(한번 입력시 ON, 다시 입력시 OFF)
%debug	작성된 코드에 대한 debug 처리

Magic command 내의 help

%magic command 뒤에 ?를 입력하면 설명이 나옴

```
In [45]: %edit?
```

```
In [ ]: |
```

Docstring:

Bring up an editor and execute the resulting code.

Usage:

```
%edit [options] [args]
```

%edit runs an external text editor. You will need to set the command for this editor via the ``TerminalInteractiveShell.editor`` option in your configuration file before it will work.

This command allows you to conveniently edit multi-line code right in your IPython session.

If called without arguments, %edit opens up an empty editor with a temporary file and will execute the contents of this file when you close it (don't forget to save it!).

Magic command 확인하기

magic command에 대한 설명 보기

```
In [118]: %magic
```

IPython's 'magic' functions

=====

The magic function system provides a series of functions which allow you to control the behavior of IPython itself, plus a lot of system-type features. There are two kinds of magics, line-oriented and cell-oriented.

Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. For example, this will time the given statement::

변수 관리



Notebook 내의 변수 삭제

`%reset`(매직 command)은 현재 실행되는 notebook 내의 모든 변수를 삭제함

```
In [77]: x
```

```
Out[77]: 10
```

```
In [78]: %reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

```
In [79]: x
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-79-401b30e3b8b5> in <module>()  
----> 1 x  
  
NameError: name 'x' is not defined
```

변수가 삭제되어 오류 메시지

Whos: 변수들 표시

현재 실행환경 내의 Variables을 표시, Matlab의 whos와 유사함

```
In [100]: %whos
```

Variable	Type	Data/Info

YouTubeVideo	type	<class 'IPython.lib.display.YouTubeVideo'>
add	function	<function add at 0x0643EF30>

```
In [101]: %reset
```

```
Once deleted, variables cannot be recovered. Proceed (y/[n])? y
```

```
In [102]: %whos
```

```
Interactive namespace is empty.
```

작업 위치 정하기



Directory 만들고 이동

%(매직 command)를 이용해서 현재 위치 및 디렉토리 생성 및 이동

```
In [4]: x=10
```

```
In [5]: %whos
```

Variable	Type	Data/Info
x	int	10

```
In [6]: %pwd
```

```
Out[6]: 'C:\\\\Users\\KyungHee'
```

```
In [7]: %mkdir KS  
%pwd
```

```
Out[7]: 'C:\\\\Users\\KyungHee'
```

```
In [8]: %cd KS  
%pwd
```

```
C:\\Users\\KyungHee\\KS
```

```
Out[8]: 'C:\\\\Users\\KyungHee\\KS'
```



파일 생성 및 실행



파이썬 파일 만들고 확인하기

%%writefile(매직 command)를 이용해서 현재 위치에 add.py 생성하고 조회

```
In [9]: %%writefile add.py
def add(x,y):
    return x+y

print(add(4,4))
```

Writing add.py

```
In [11]: %ls
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: F807-6E28

C:\Users\KyungHee\KS 디렉터리

2019-03-05	오후 09:43	<DIR>	.
2019-03-05	오후 09:43	<DIR>	..
2019-03-05	오후 09:43		48 add.py
		1개 파일	48 바이트
		2개 디렉터리	99,895,943,168 바이트 남음

```
In [12]: %run add.py
```

파이썬 파일 실행하기

%run(매직 command)을 이용해서 파이썬 모듈 실행

```
In [9]: %%writefile add.py
def add(x,y):
    return x+y

print(add(4,4))
```

Writing add.py

```
In [11]: %ls
```

C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: F807-6E28

C:\Users\KyungHee\KS 디렉터리

2019-03-05	오후 09:43	<DIR>	.
2019-03-05	오후 09:43	<DIR>	..
2019-03-05	오후 09:43		48 add.py
		1개 파일	48 바이트
		2개 디렉터리	99,895,943,168 바이트 남음

```
In [12]: %run add.py
```

함수에 대한 소스와 헤드 조회

"%pdef 객체", "%psource"를 입력해서 함수의 헤드와 소스를 조회

```
In [111]: def add(x,y) :  
          return x+y
```

```
In [112]: %pdef add  
          add(x, y)
```

```
In [113]: %psource add
```

```
In [ ]: |
```

```
def add(x,y) :  
    return x+y
```

history



Cell에 입력된 history 확인하기

Cell에 입력된 이력을 출력

```
In [15]: %history  
  
print("Hello world!")  
%lsmagic  
%magic  
x=10  
%whos  
%pwd  
%mkdir KS  
%pwd  
%cd KS  
%pwd  
%%writefile add.py  
def add(x,y):  
    return x+y  
  
print(add(4,4))  
%run add.py  
%ls  
%run add.py  
%pdef add  
%psource add  
%history
```



Cell에 입력된 일부 history 확인

현재 명령된 이전 명령 5개만 읽어오기

```
In [16]: %history -1 5
```

```
%ls  
%run add.py  
%pdef add  
%psource add  
%history
```

```
In [ ]:
```

방문한 모든 디렉토리 history 확인

%dhist로 현재까지 방문한 모든 디렉토리 이력을 읽어오기

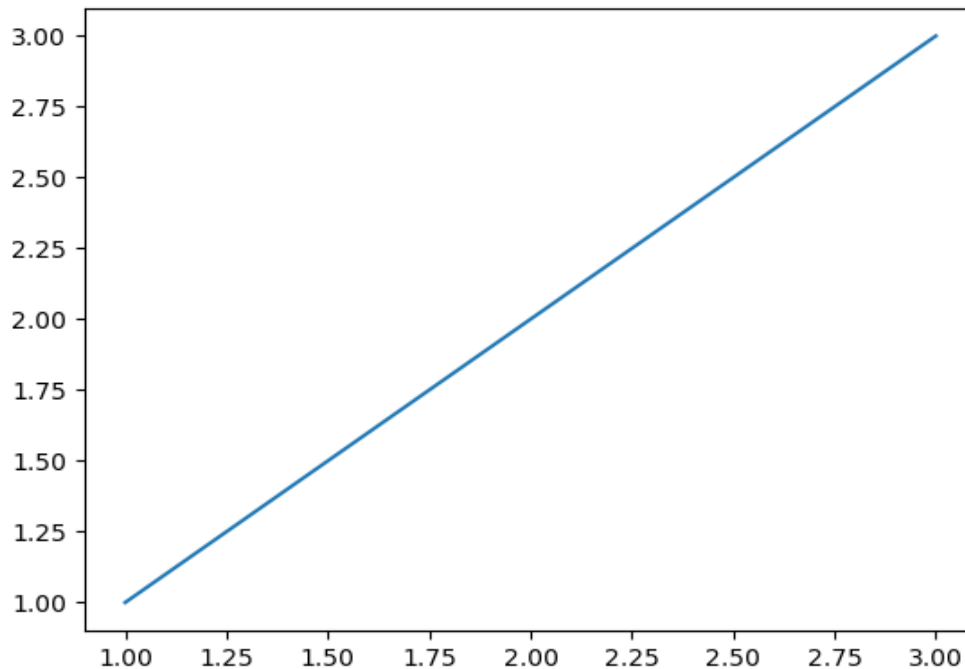
```
In [17]: %dhist  
Directory history (kept in _dh)  
0: C:\Users\KyungHee  
1: C:\Users\KyungHee\KS
```



Matplotlib 라이브러리로 그래프를 그려보자

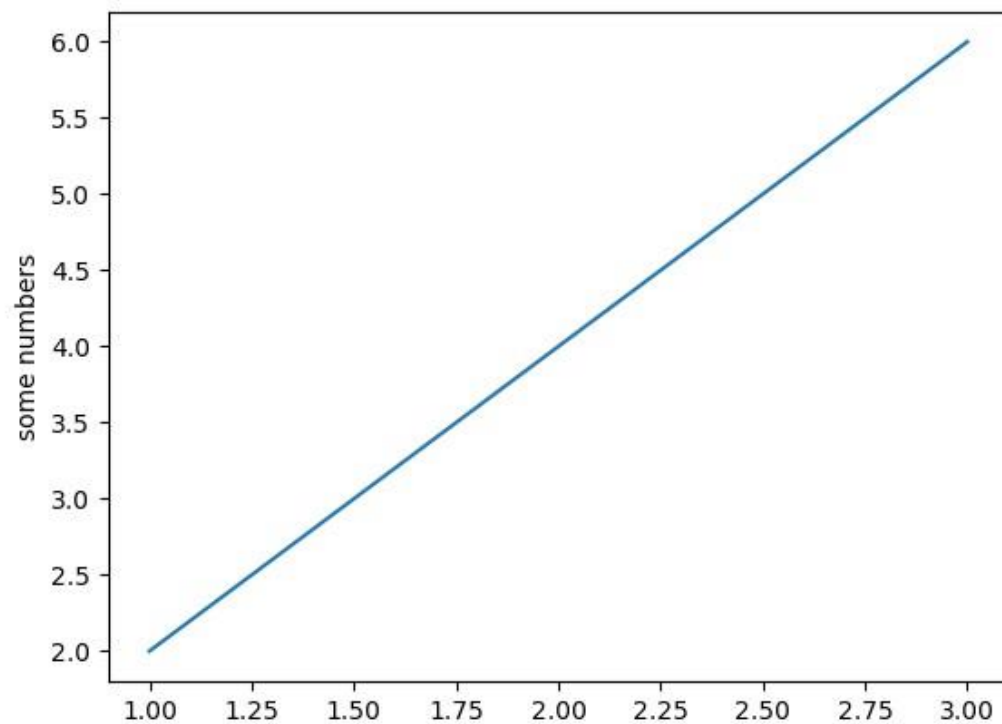
```
In [2]: import matplotlib.pyplot as plt  
%matplotlib notebook  
plt.figure()  
plt.plot([1, 2, 3],[1, 2, 3])  
plt.show()
```

Figure 1




```
In [2]: import matplotlib.pyplot as plt
%matplotlib notebook
plt.figure()
plt.plot([1,2,3],[2,4,6])
plt.ylabel('some numbers')
plt.show()
```

Figure 1

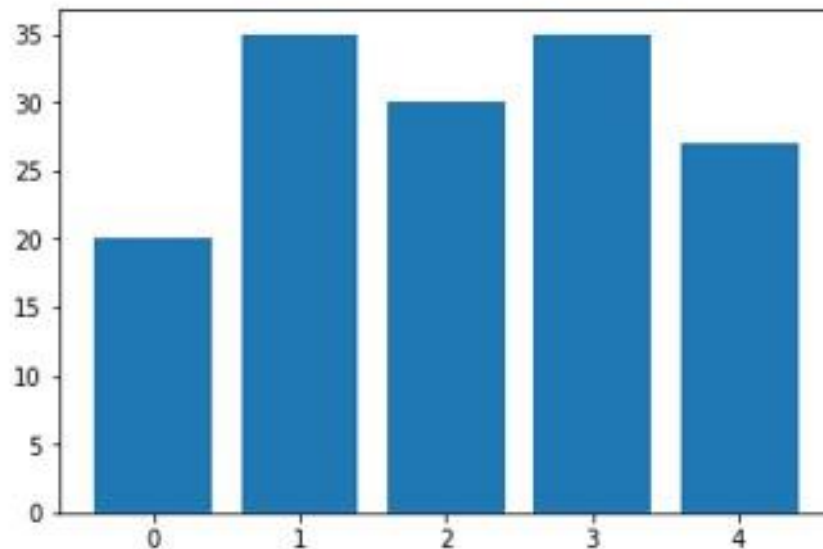


Matplotlib의 bar함수 : 기본

bar함수는 폭 0.8, 파란색 막대가 기본으로 처리

```
In [15]: import numpy as np
import matplotlib.pyplot as plt
N=5
menMeans =(20, 35, 30, 35, 27)
ind = np.arange(N)
print(ind)
plt.bar(ind, menMeans)
plt.show()
```

[0 1 2 3 4]

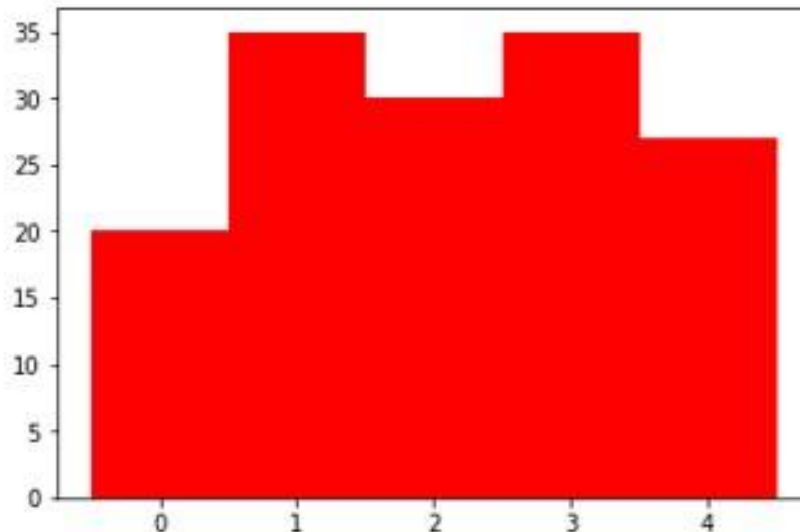


Matplotlib의 bar함수 : 폭 늘리기

bar함수는 위치와 값을 막대그래프로 표시

```
import numpy as np
import matplotlib.pyplot as plt
N=5
menMeans =(20, 35, 30, 35, 27)
ind = np.arange(N)
print(ind)
plt.bar(ind, menMeans, width=1.0, color="r")
plt.show()
```

[0 1 2 3 4]

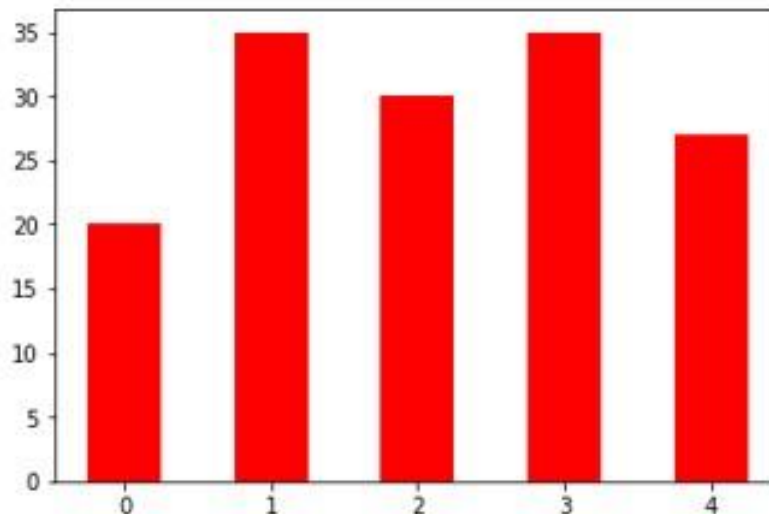


Matplotlib의 bar함수 : 폭 줄이기

bar함수는 막대 그래프의 폭을 0.5로 처리

```
import numpy as np
import matplotlib.pyplot as plt
N=5
menMeans =(20, 35, 30, 35, 27)
ind = np.arange(N)
print(ind)
plt.bar(ind, menMeans, width=0.5, color="r")
plt.show()
```

[0 1 2 3 4]



실습 1

- 입력 받은 문장을 아래와 같이 거꾸로 출력하는 파이썬 프로그램 inverseIn.py 를 magic command를 이용해서 만들어 실행시키시오.

```
%run inverseIn.py
```

Input :

```
In [26]: %run inverseIn.py
```

Input :Sogang University

```
S
oS
goS
agoS
nagoS
gnagoS
  gnagoS
U gnagoS
nU gnagoS
inU gnagoS
vinU gnagoS
evinU gnagoS
revinU gnagoS
srevinU gnagoS
isrevinU gnagoS
tisrevinU gnagoS
ytisrevinU gnagoS
```

