



EDDI

Electronic Design  
Development Institute

---

# 에디로봇아카데미

## 임베디드 마스터 Lv1 과정

제 4기

2022. 10. 27

진동민

# 학습목표 & 8회차 날짜

## 학습목표

- Anaconda 설치하는 방법
- 파이썬 기초 문법과 자료구조 사용법

## 수업 날짜

2022-10-22 (토) 오후 6시~9시

# 목차

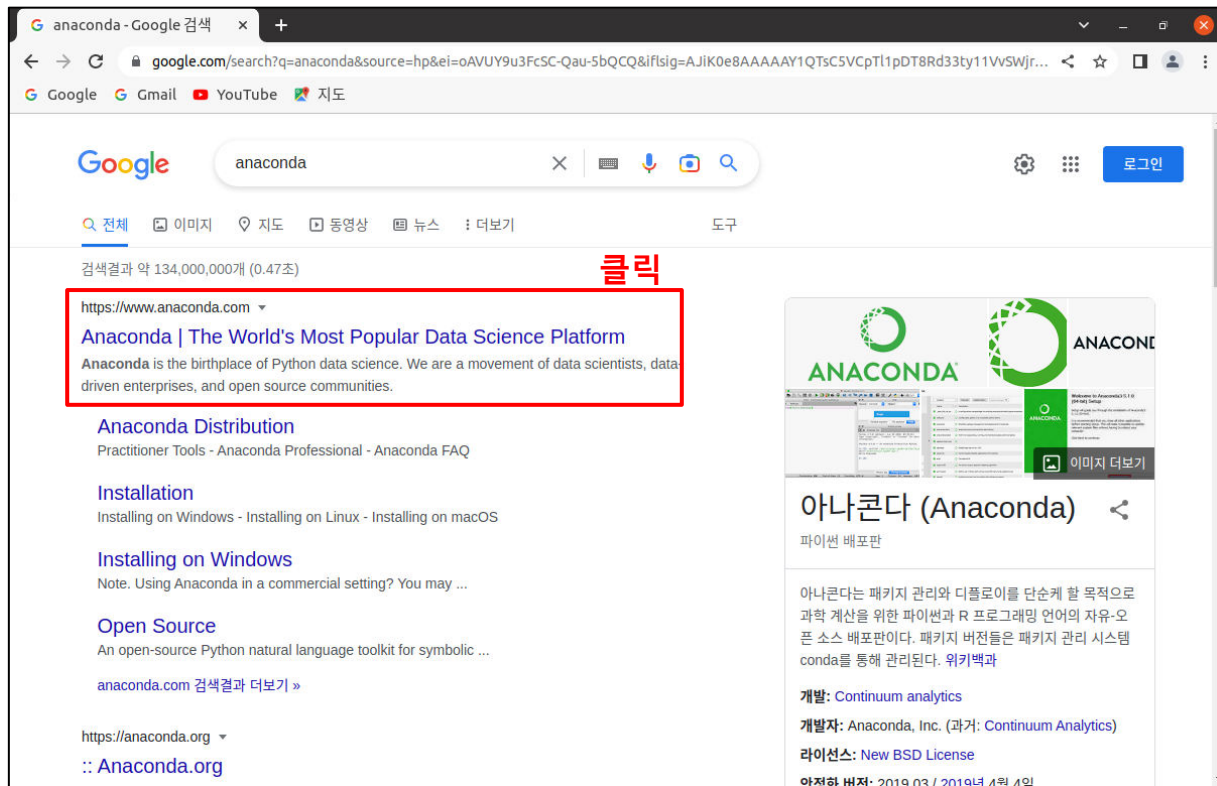
- 1) Anaconda 설치하기
- 2) 주피터 노트북 실행하기
- 3) 파이썬 노트북 만들기
- 4) 파이썬 실습
- 5) 주피터 노트북 종료하기
- 6) FPGA vs GPU
- 7) Windows vs Linux
- 8) 엔지니어링 위키 (노션)
- 9) 정리
- 10) 여담
- 11) 수업내용 사진

# Anaconda 설치하기

## 설치 환경

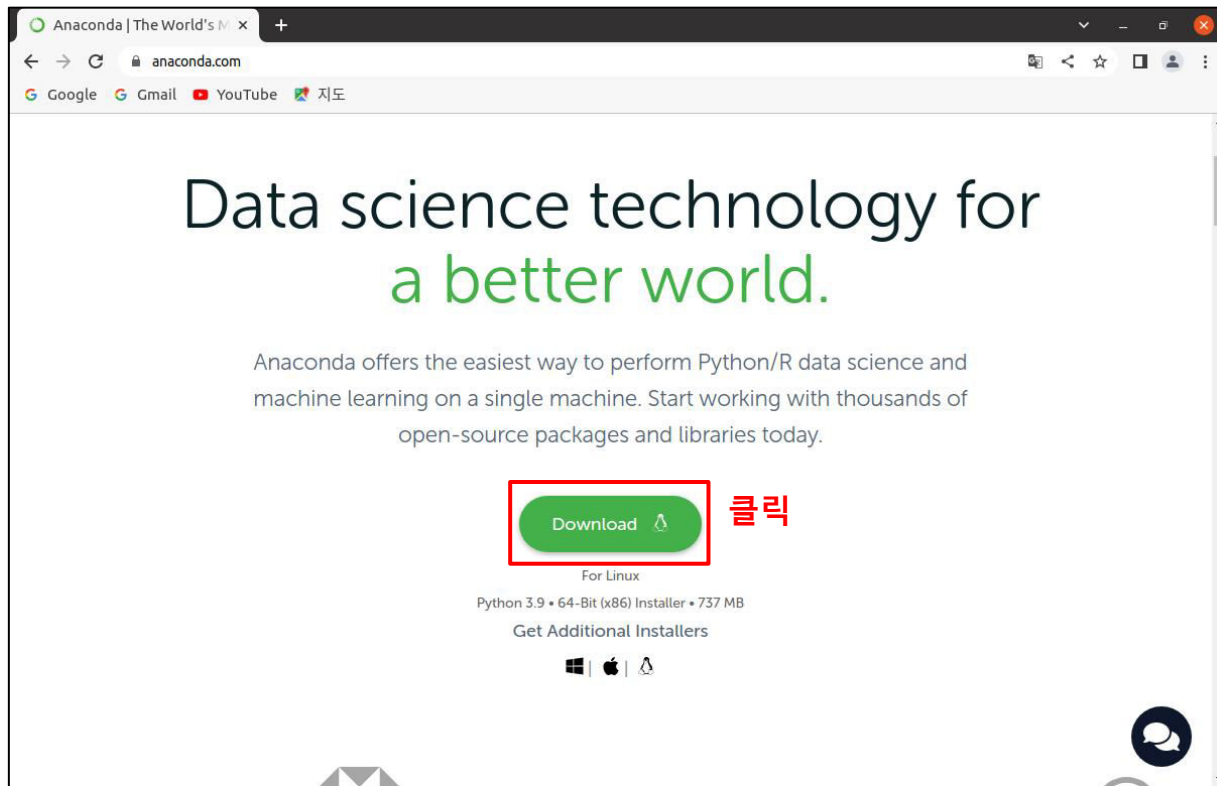
- VMware Workstation 16 Player (Anaconda를 설치하는 장면을 캡처하기 위해 가상환경에서 진행함)

# Anaconda 설치하기



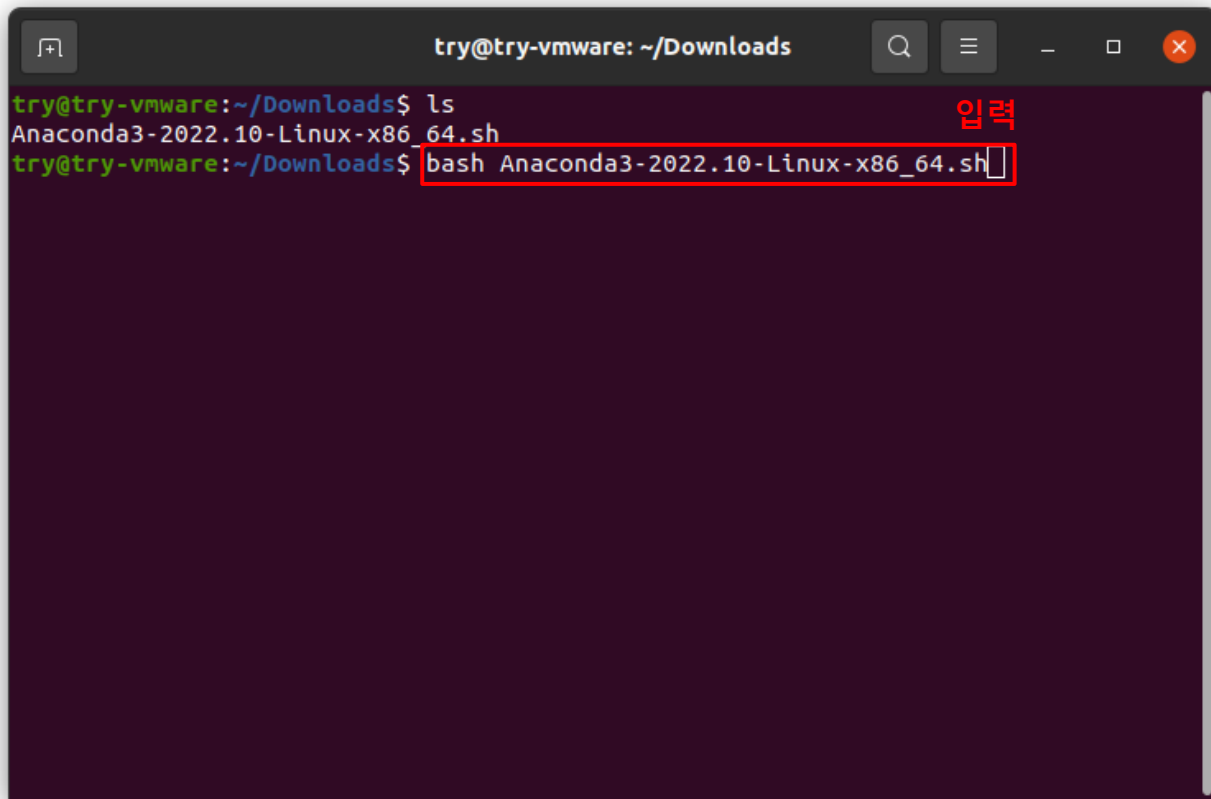
구글에 'anaconda' 검색하여 첫 번째 링크를 클릭

# Anaconda 설치하기



Download 버튼을 클릭

# Anaconda 설치하기

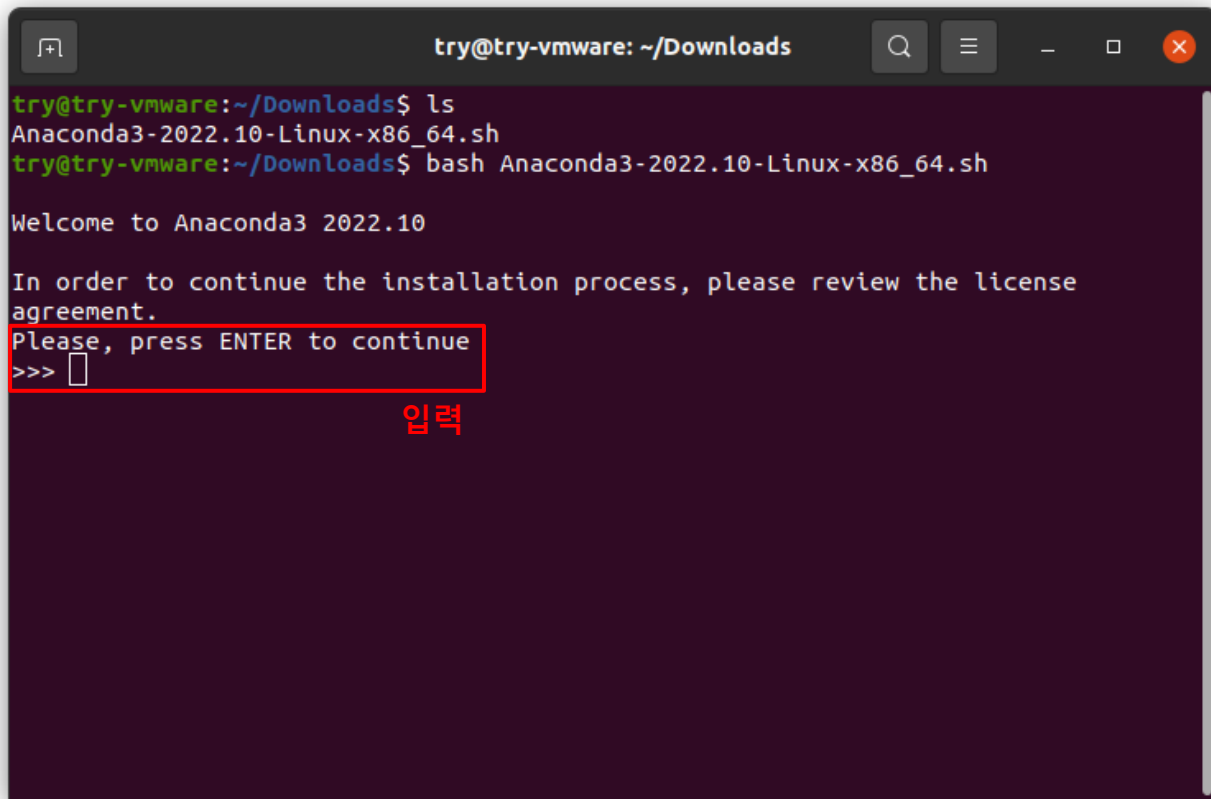


```
try@try-vmware: ~/Downloads
try@try-vmware:~/Downloads$ ls
Anaconda3-2022.10-Linux-x86_64.sh
try@try-vmware:~/Downloads$ bash Anaconda3-2022.10-Linux-x86_64.sh
```

The terminal window shows the user 'try' at 'try-vmware' in the directory '~/Downloads'. The user runs 'ls' and lists the file 'Anaconda3-2022.10-Linux-x86\_64.sh'. Then, the user runs 'bash Anaconda3-2022.10-Linux-x86\_64.sh', which is highlighted with a red box. A red Korean label '입력' (Input) is placed next to the second command.

다운로드 파일이 있는 경로에서 터미널을 실행, 그리고 'bash 파일명'을 입력

# Anaconda 설치하기



```
try@try-vmware: ~/Downloads
try@try-vmware:~/Downloads$ ls
Anaconda3-2022.10-Linux-x86_64.sh
try@try-vmware:~/Downloads$ bash Anaconda3-2022.10-Linux-x86_64.sh

Welcome to Anaconda3 2022.10

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>> 
```

입력

파일을 실행하면 위와 같이 뜨는데, Enter 입력



# Anaconda 설치하기

```
try@try-vmware: ~/Downloads
=====
End User License Agreement - Anaconda Distribution
=====

Copyright 2015-2022, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between y
ou and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Distribution
(which was formerly known as Anaconda Individual Edition).

Subject to the terms of this Agreement, Anaconda hereby grants you a non-exclusi
ve, non-transferable license to:

    * Install and use the Anaconda Distribution (which was formerly known as Anaco
nda Individual Edition),
    * Modify and create derivative works of sample source code delivered in Anacon
da Distribution from Anaconda's repository, and;
    * Redistribute code files in source (if provided to you by Anaconda as source)
and binary forms, with or without modification subject to the requirements set
forth below, and;

--More--
```

q 입력

# Anaconda 설치하기

```
try@try-vmware: ~/Downloads
ou and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Distributio
n
(which was formerly known as Anaconda Individual Edition).

Subject to the terms of this Agreement, Anaconda hereby grants you a non-exclus
i
ve, non-transferable license to:

    * Install and use the Anaconda Distribution (which was formerly known as Anac
o
nda Individual Edition),
    * Modify and create derivative works of sample source code delivered in Anaco
n
da Distribution from Anaconda's repository, and;
    * Redistribute code files in source (if provided to you by Anaconda as source
)
and binary forms, with or without modification subject to the requirements set
forth below, and;

Do you accept the license terms? [yes|no]
[no] >>> 
```

yes 입력

# Anaconda 설치하기

```
try@try-vmware: ~/Downloads
o
nda Individual Edition),
  * Modify and create derivative works of sample source code delivered in Anaconda Distribution from Anaconda's repository, and;
  * Redistribute code files in source (if provided to you by Anaconda as source) and binary forms, with or without modification subject to the requirements set forth below, and;

Do you accept the license terms? [yes|no]
[no] >>> yes

Anaconda3 will now be installed into this location:
/home/try/anaconda3

- Press ENTER to confirm the location
- Press CTRL-C to abort the installation
- Or specify a different location below

[/home/try/anaconda3] >>> 
```

Anaconda 설치 경로를 어떻게 할 것이냐 물어보는데, 여기서는 기본값으로 하기 위해 enter 입력

# Anaconda 설치하기

```
try@try-vmware: ~/Downloads
zstd pkgs/main/linux-64::zstd-1.5.2-ha4553b6_0

Preparing transaction: done
Executing transaction: \

    Installed package of scikit-learn can be accelerated using scikit-learn-intelex.
    More details are available here: https://intel.github.io/scikit-learn-intelex

    For example:

        $ conda install scikit-learn-intelex
        $ python -m sklearnex my_application.py

done
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[no] >>> 
```

yes 입력

# Anaconda 설치하기

```
try@try-vmware: ~/Downloads
no change      /home/try/anaconda3/shell/condabin/conda-hook.ps1
no change      /home/try/anaconda3/lib/python3.9/site-packages/xontrib/conda.xsh
no change      /home/try/anaconda3/etc/profile.d/conda.csh
modified       /home/try/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,
    set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!

=====

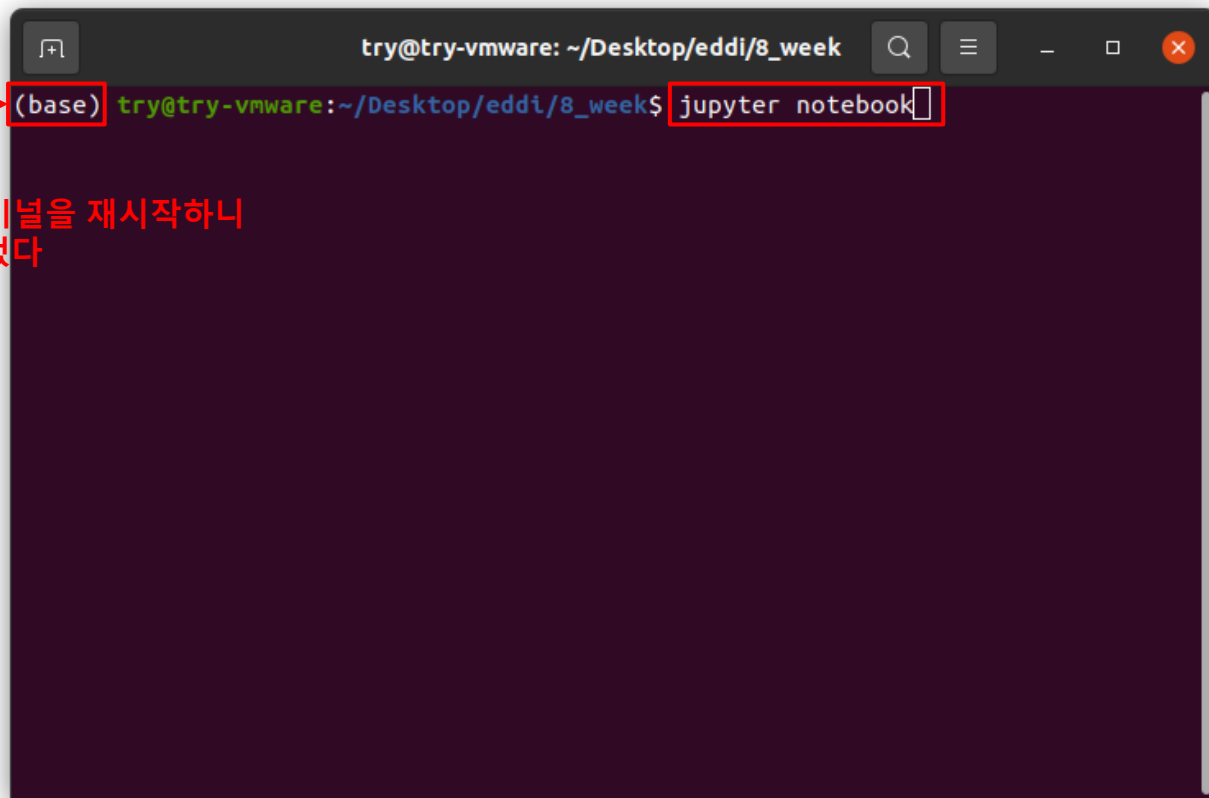
Working with Python and Jupyter is a breeze in DataSpell. It is an IDE
designed for exploratory data analysis and ML. Get better data insights
with DataSpell.

DataSpell for Anaconda is available at: https://www.anaconda.com/dataspell

try@try-vmware:~/Downloads$
```

설치 완료

# 주피터 노트북 실행하기

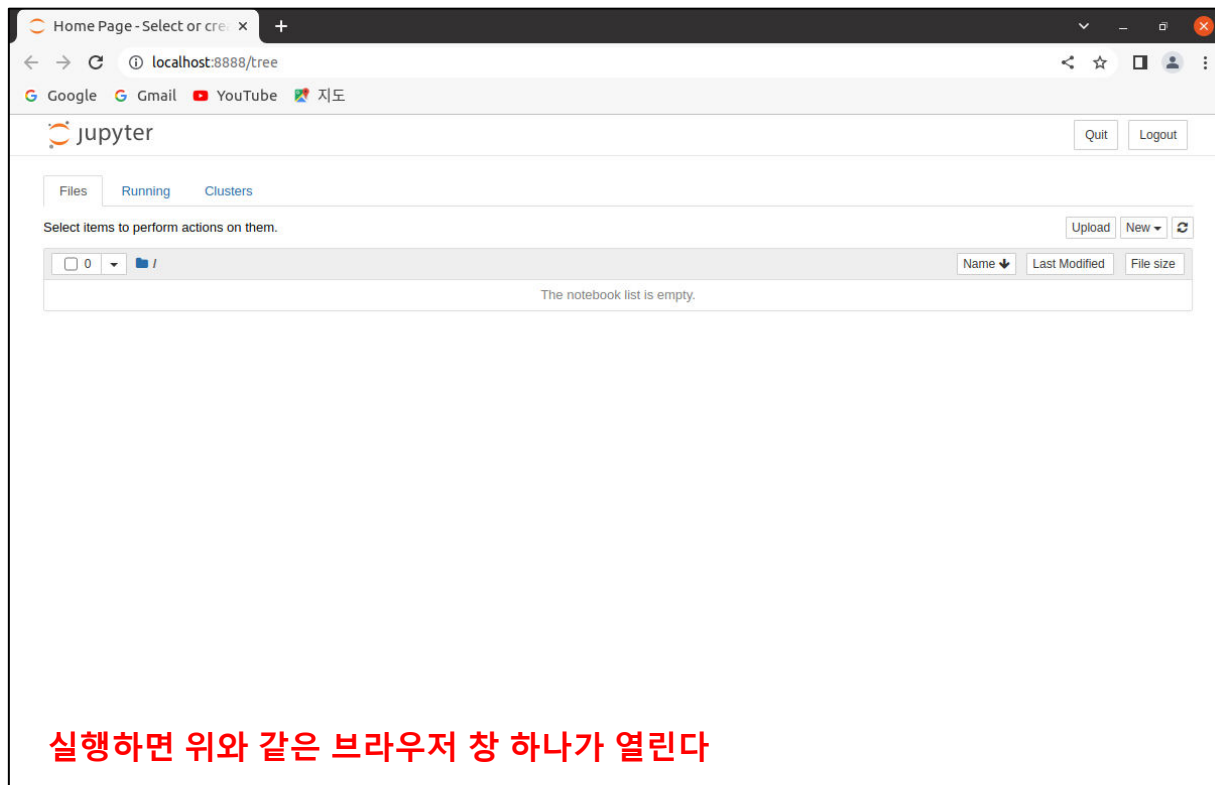


```
try@try-vmware: ~/Desktop/eddi/8_week
(base) try@try-vmware:~/Desktop/eddi/8_week$ jupyter notebook
```

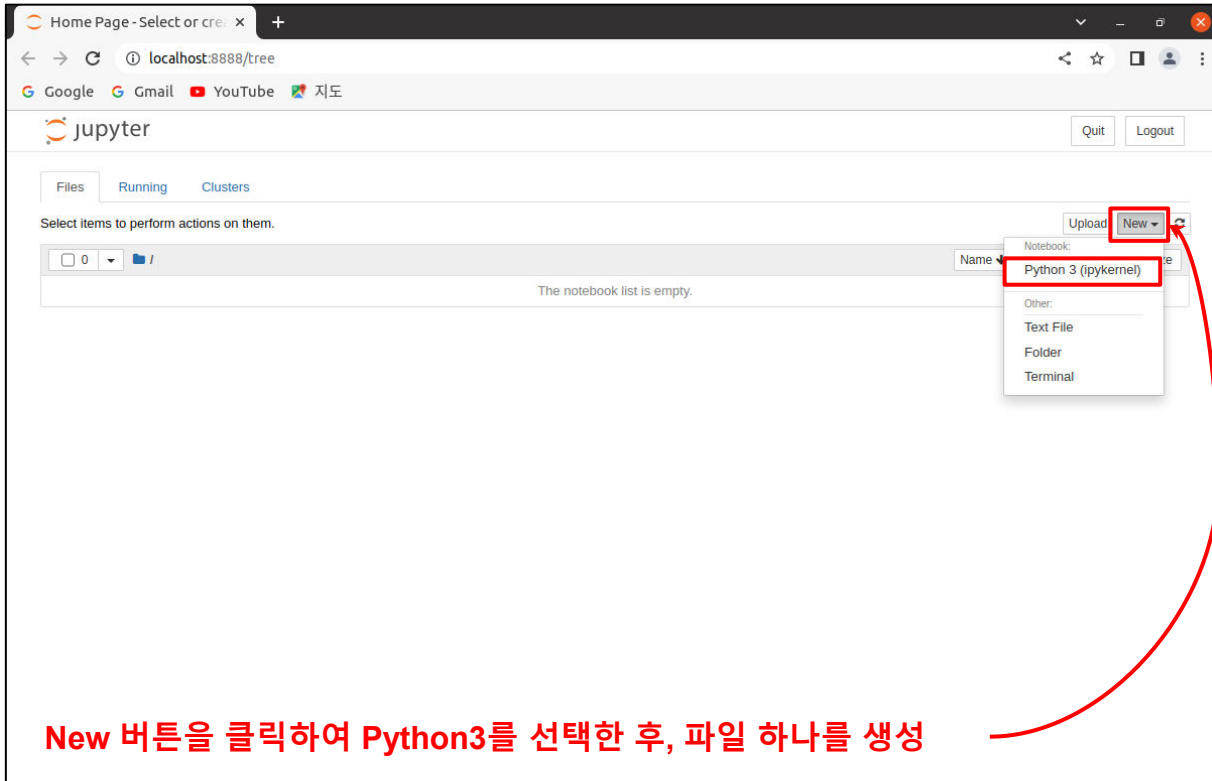
설치가 끝나고 터미널을 재시작하니  
앞에 (base)가 붙었다

파이썬을 실행할 디렉토리에서 'jupyter notebook' 입력하여 주피터 노트북을 실행

# 파이썬 노트북 만들기



# 파이썬 노트북 만들기

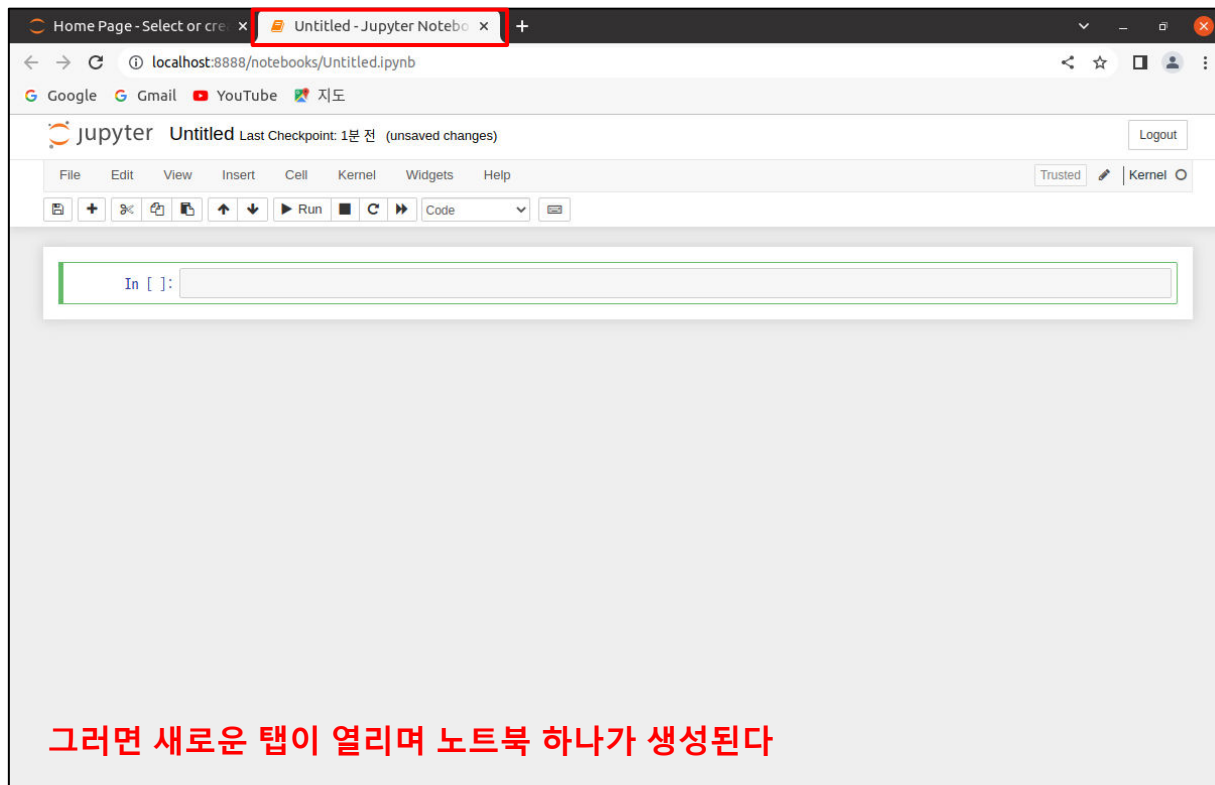


The screenshot shows the JupyterLab web interface in a browser window. The address bar shows 'localhost:8888/tree'. The JupyterLab logo is at the top left, and 'Quit' and 'Logout' buttons are at the top right. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters'. The main area shows a file browser with a message 'The notebook list is empty.' On the right side, there is a toolbar with 'Upload' and 'New' buttons. The 'New' button is highlighted with a red box, and a dropdown menu is open, showing 'Python 3 (ipykernel)' as the selected option, also highlighted with a red box. A red arrow points from the text below to the 'Python 3 (ipykernel)' option.

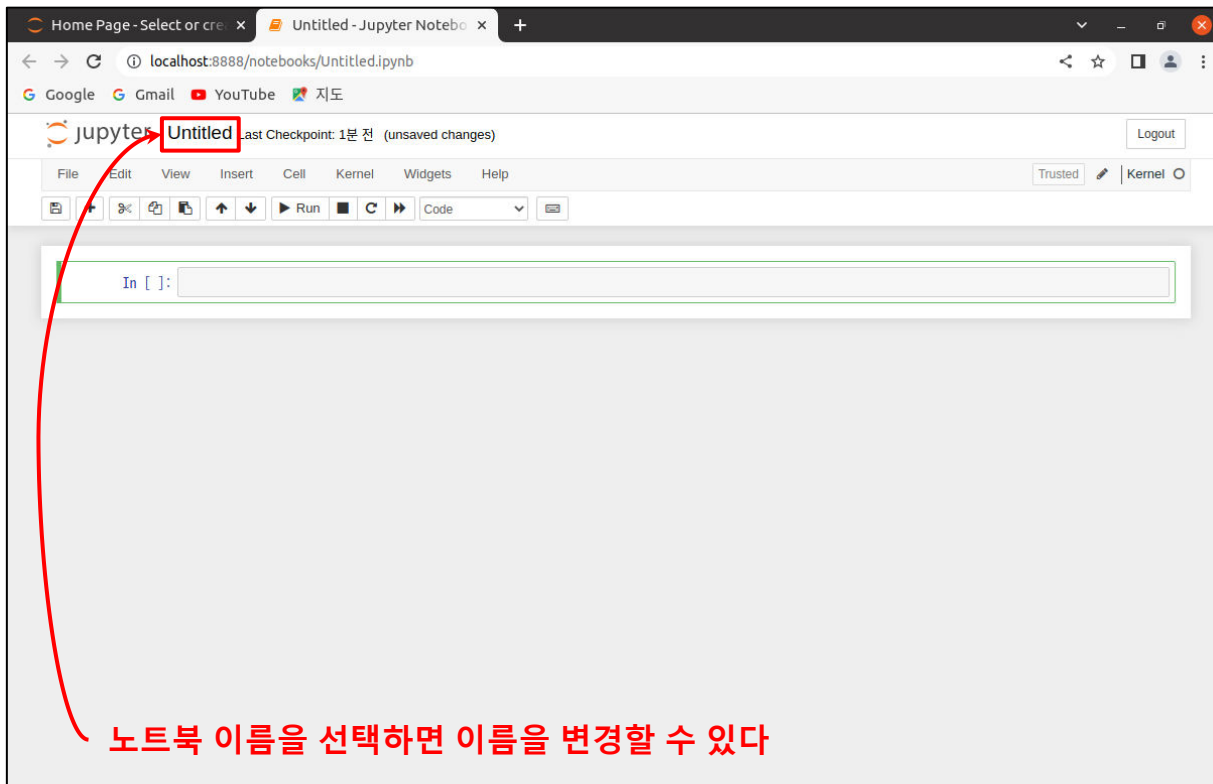
New 버튼을 클릭하여 Python3를 선택한 후, 파일 하나를 생성



# 파이썬 노트북 만들기



# 파이썬 노트북 만들기



Home Page - Select or cre x Untitled - Jupyter Notebo x +

localhost:8888/notebooks/Untitled.ipynb

Google Gmail YouTube 지도

jupyter Untitled Last Checkpoint: 1분 전 (unsaved changes) Logout

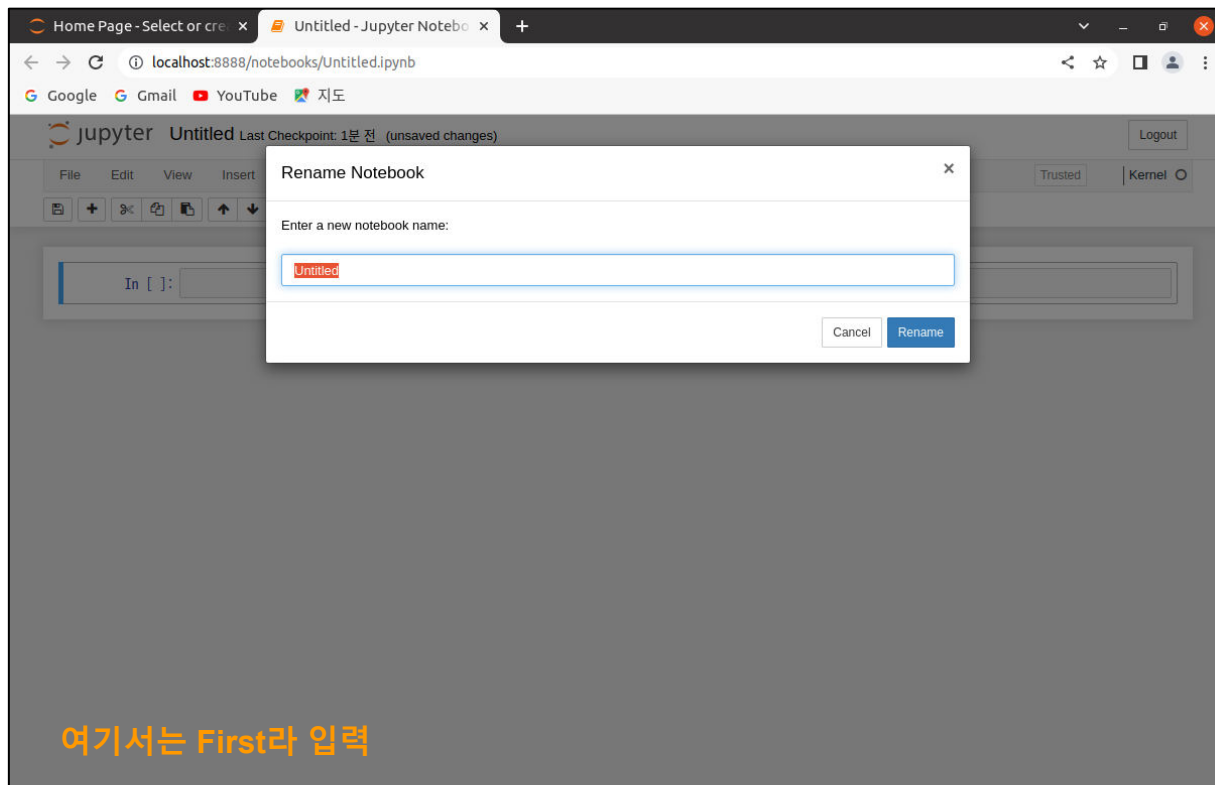
File Edit View Insert Cell Kernel Widgets Help

Trusted Kernel

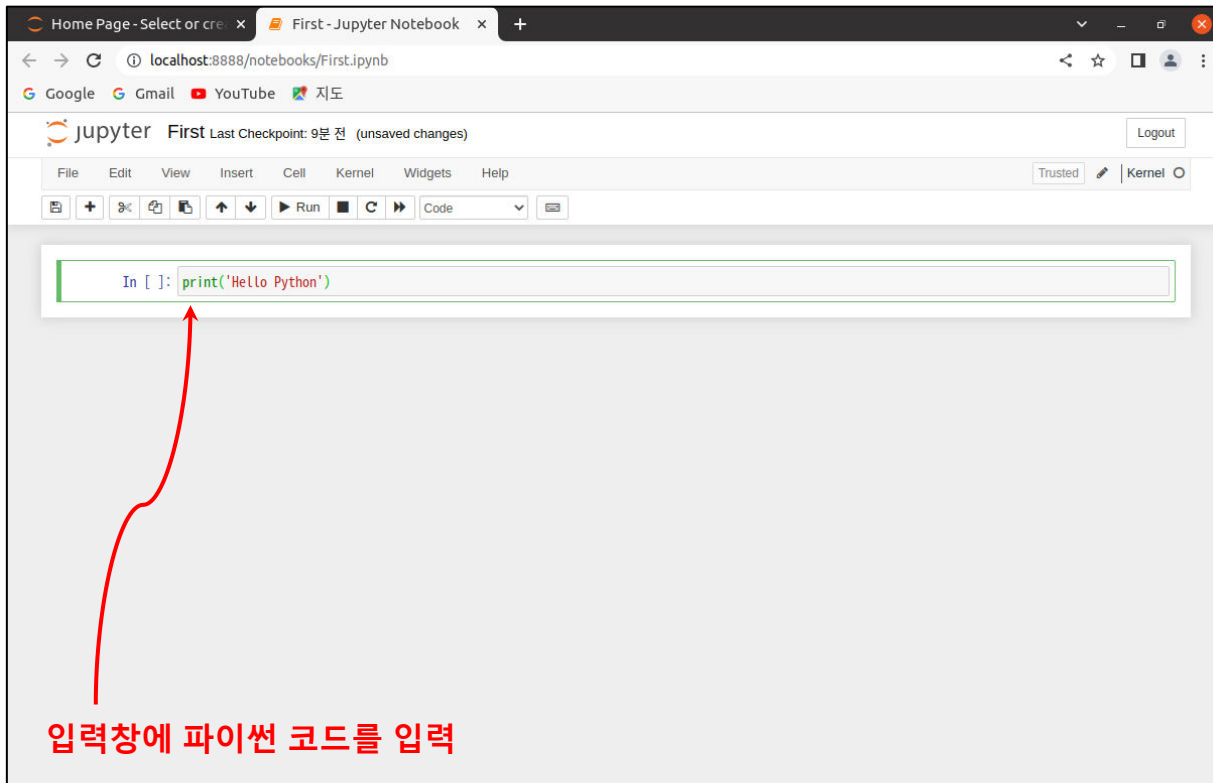
In [ ]:

노트북 이름을 선택하면 이름을 변경할 수 있다

# 파이썬 노트북 만들기



# 파이썬 실습



Home Page - Select or create a new notebook x First - Jupyter Notebook x +

localhost:8888/notebooks/First.ipynb

Google Gmail YouTube 지도

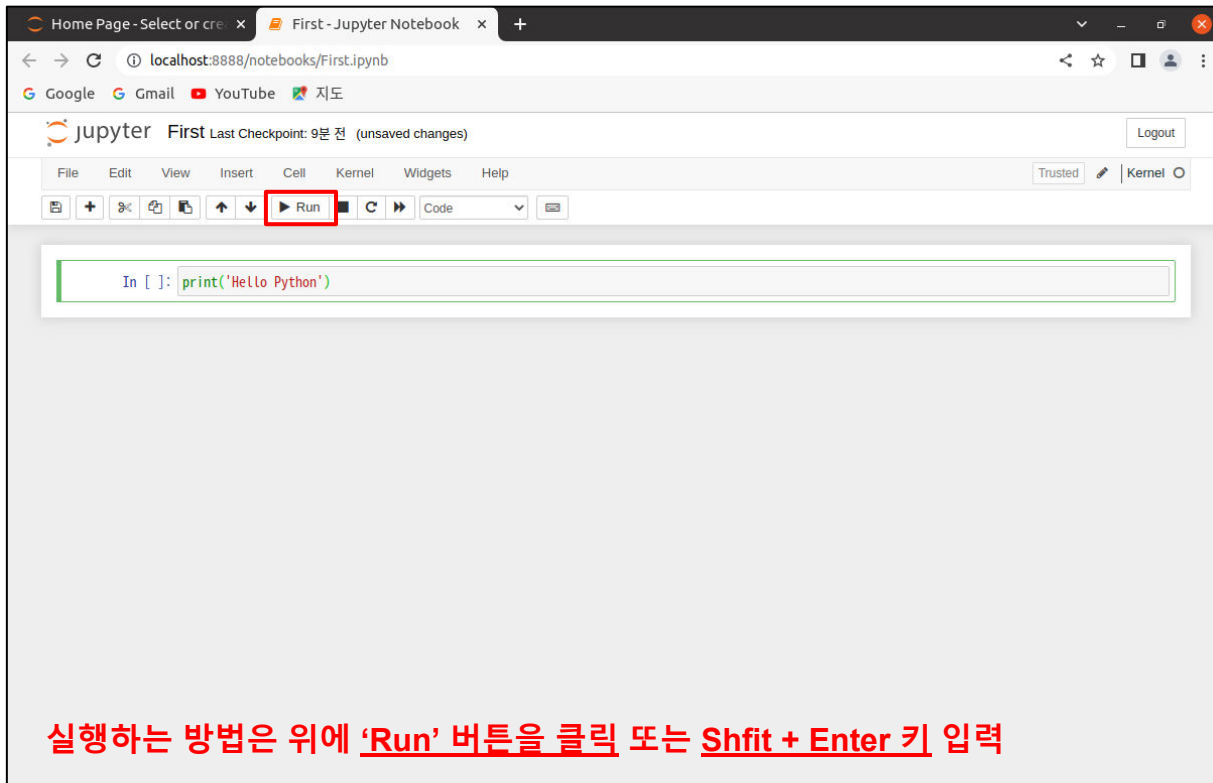
jupyter First Last Checkpoint: 9분 전 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Kernel

In [ ]: `print('Hello Python')`

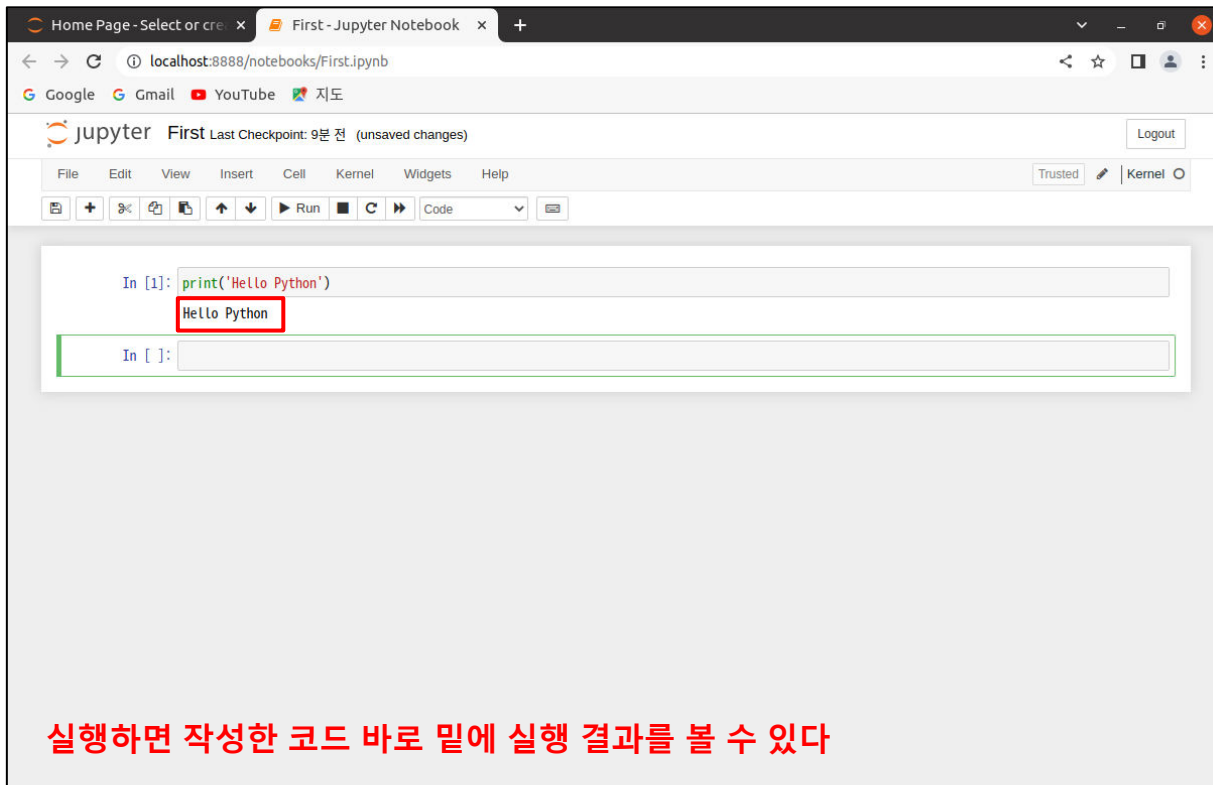
입력창에 파이썬 코드를 입력

# 파이썬 실습



실행하는 방법은 위에 'Run' 버튼을 클릭 또는 Shfit + Enter 키 입력

# 파이썬 실습



Home Page - Select or create a notebook x First - Jupyter Notebook x +

localhost:8888/notebooks/First.ipynb

Google Gmail YouTube 지도

jupyter First Last Checkpoint: 9분 전 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Kernel

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

Hello Python

In [ ]:

**실행하면 작성한 코드 바로 밑에 실행 결과를 볼 수 있다**

# 파이썬 실습

```
In [2]: num = 1
        Num = 10
        print(num, Num)
```

← C언어와 다르게 파이썬은 타입을 신경쓰지 않아도 됨

1 10

```
In [3]: string_value = "test"
        num = 3
        float_num = 7.7

        print(string_value, num, float_num)
```

print 함수를 사용하여 변수에 저장한 값을 확인할 수 있음

test 3 7.7

```
In [4]: print(type(string_value))
        print(type(num))
        print(type(float_num))
```

type 함수를 사용하여 변수의 데이터타입을 확인할 수 있음

```
<class 'str'>
<class 'int'>
<class 'float'>
```

print 함수에서 여러 값을 출력하려면 ,(coma)를 사용하면 된다.

# 파이썬 실습

```
In [5]: z1 = 3 + 4j
        z2 = 4 + 5j
        z = z1 + z2

        print(z)
        print(type(z))

        # 복소수의 허수부
        print(z.imag)

        # 복소수의 실수부
        print(z.real)

        # 켤레 복소수
        print(z.conjugate())

        (7+9j)
        <class 'complex'>
        9.0
        7.0
        (7-9j)
```

파이썬에서는 복소수를 저장할 수 있는데, 수학에서 허수를 표기하는 문자  $i$  대신  $j$ 를 사용하면 된다.

(참고)  $j$ 를 사용하는 이유

<https://stackoverflow.com/questions/24812444/why-are-complex-numbers-in-python-denoted-with-j-instead-of-i>



# 파이썬 실습

```
In [6]: big = 1.2e30
        small = 1.57e-20
        test = 2 ** 10

        print(big)
        print(small)
        print(test)

        print(type(big))
        print(type(small))
        print(type(test))

1.2e+30
1.57e-20
1024
<class 'float'>
<class 'float'>
<class 'int'>
```

e표기법도 지원한다.

변수 big에 대입한 값은  $1.2 * 10^{30}$   
변수 small에 대입한 값은  $1.57 * 10^{-20}$

그리고 지수를 구할 수 있는 \*\* 연산자도 있다.  
변수 test에 대입한 값은  $2^{10}$

# 파이썬 실습

```
In [7]: # 파이썬 주식
print('주식 처리')

# 파이썬은 큰따옴표와 작은따옴표 구분을 하지 않음

주식 처리
```

```
In [8]: # // 연산자는 배열의 크기를 계산할 때 사용
num1 = 3 // 7
num2 = 3333 // 10
num3 = 3 / 7

print(num1)
print(num2)
print(num3)

0
333
0.42857142857142855
```

주식을 다는 법은 #을 붙이면 해당 줄의 #이 붙은 문자에서 끝까지 주식처리 된다.

파이썬은 큰따옴표와 작은 따옴표를 구분하지 않는다. (나는 작은 따옴표를 선호함)

// 연산자는 몫을 구하고, / 연산자는 나눈 값을 구한다.

# 파이썬 실습

```
In [9]: # format 스트링, DB 연산을 할 때 많이 사용한다고 한다.  
format_string = """  
    이렇게도  
        뿌릴 수가  
            있습니다.  
    """  
  
print(format_string)
```

```
    이렇게도  
        뿌릴 수가  
            있습니다.
```

```
In [10]: testStr = "C" + " Python"  
print(testStr)
```

```
C Python
```

SQL을 사용할 때 여러 줄을 작성해야하는 경우가 있는데, 이럴 때는 문자열을 만드는 방법 중 하나인 따옴표 3개를 사용하면 된다.

또한, 문자열끼리 더할 수 있다

# 파이썬 실습

In [11]: `str1 = "pointer"`

```
print(str1)
print(str1[0])
print(str1[3])
```

문자열을 배열처럼 인덱스로 접근

```
pointer
p
n
```

In [12]: `# 0 ~ 1 까지가 아닌 0 부터 1 이전까지`

```
print(str1[0:1])
print(str1[1:4])
```

슬라이싱

```
p
oin
```

In [13]: `# 0 생략`

```
print(str1[:2])
```

```
po
```

In [14]: `print(str1[-2:])`

```
er
```

In [15]: `# 전체 복사 (new)`

```
print(str1[:])
```

(주의) 전체를 복사

```
pointer
```

In [16]: `print(str1[::2])`

두 칸씩 건너뛰

```
pitr
```

# 파이썬 실습

```
In [17]: colors = ['red', 'green', 'blue']
```

```
print(colors)
print(type(colors))
```

```
['red', 'green', 'blue']
<class 'list'>
```

```
In [18]: colors.append('gold')
```

**append 메소드를 사용하면 값을 리스트 끝에 추가**

```
print(colors)
```

```
['red', 'green', 'blue', 'gold']
```

```
In [19]: colors.insert(1, 'black')
```

**insert 메소드를 사용하여 인자로 전달한 인덱스에 값을 추가**

```
print(colors)
```

```
['red', 'black', 'green', 'blue', 'gold']
```

```
In [20]: colors.extend(['white', 'grey'])
```

**extend 메소드를 사용하여 여러 값을 추가**

```
print(colors)
```

```
['red', 'black', 'green', 'blue', 'gold', 'white', 'grey']
```

대괄호([])를 사용하여 리스트를 생성할 수 있다.  
변수 colors는 여러 개의 문자열을 담은 리스트이다.

# 파이썬 실습

In [21]: # ㅇㅇ 이런 것도 되는구나

```
colors += ['purple']
```

```
colors += ['red']
```

복합 연산자로 값을 추가

```
print(colors)
```

```
['red', 'black', 'green', 'blue', 'gold', 'white', 'grey', 'purple', 'red']
```

In [22]: `print(colors.index('purple'))`

리스트에 저장된 purple의 인덱스를 반환

```
7
```

In [23]: # 한 번 더 추가

```
colors += ['purple']
```

```
colors += ['red']
```

In [24]: # 8번 인덱스부터 찾아

```
print(colors.index('purple', 8))
```

두 번째 인자로 인덱스를 지정하여 그 이후의 purple의 인덱스를 반환

```
9
```

In [25]: `print(colors.count('red'))`

값의 개수를 반환

```
3
```

In [26]: # pop이라는 용어는 스택에서 사용

```
print(colors.pop())  
print(colors.pop())  
print(colors.pop())  
  
print(colors)
```

**pop** 메소드를 사용하면 리스트의 맨 마지막 요소를 삭제하고,  
그 값을 반환한다

```
red  
purple  
red  
['red', 'black', 'green', 'blue', 'gold', 'white', 'grey', 'purple']
```

In [27]: # remove는 리턴값이 없으므로 None이 출력됨

```
print(colors.remove('blue'))  
  
print(colors)
```

**remove** 메소드를 사용하여 리스트의 요소 제거

```
None  
['red', 'black', 'green', 'gold', 'white', 'grey', 'purple']
```

In [28]: print(len(colors))

**len 함수**를 사용하여 리스트의 개수를 확인

```
7
```

In [29]: colors.sort()

```
print(colors)
```

**sort** 메소드를 사용하여 알파벳 순으로 정렬

```
['black', 'gold', 'green', 'grey', 'purple', 'red', 'white']
```

# 주피터 노트북 종료하기

종료 방법 1 - 주피터 노트북을 실행한 터미널에서 Ctrl + C 키 입력

```
try@try-vmware: ~/Desktop/eddi/8_week
(base) try@try-vmware:~/Desktop/eddi/8_week$ jupyter notebook
[I 2022-10-27 18:55:23.330 LabApp] JupyterLab extension loaded from /home/try/anaconda3/lib/python3.9/site-packages/jupyterlab
[I 2022-10-27 18:55:23.330 LabApp] JupyterLab application directory is /home/try/anaconda3/share/jupyter/lab
[I 18:55:23.338 NotebookApp] Serving notebooks from local directory: /home/try/Desktop/eddi/8_week
[I 18:55:23.338 NotebookApp] Jupyter Notebook 6.4.12 is running at:
[I 18:55:23.338 NotebookApp] http://localhost:8888/?token=5e1abe27c6dde9b22438e7120b72550c46aba68c4ffc69a
[I 18:55:23.338 NotebookApp] or http://127.0.0.1:8888/?token=5e1abe27c6dde9b22438e7120b72550c46aba68c4ffc69a
[I 18:55:23.338 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 18:55:23.372 NotebookApp]

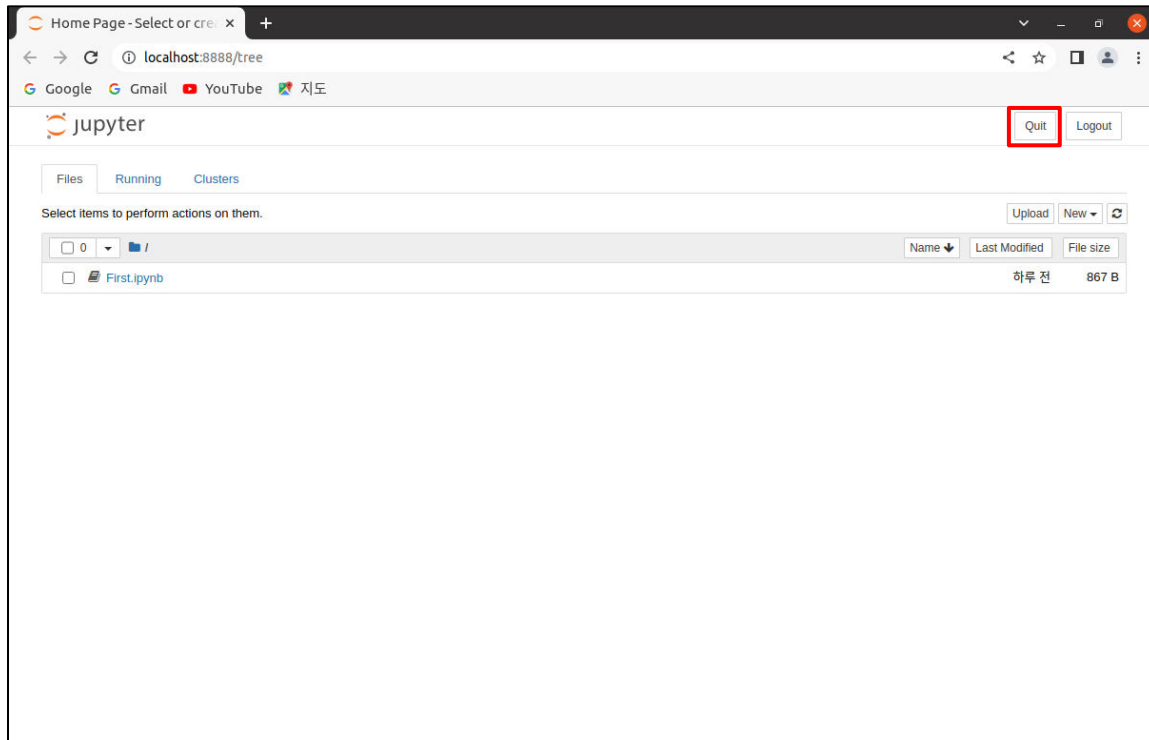
To access the notebook, open this file in a browser:
    file:///home/try/.local/share/jupyter/runtime/nbserver-3274-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=5e1abe27c6dde9b22438e7120b72550c46aba68c4ffc69a
    or http://127.0.0.1:8888/?token=5e1abe27c6dde9b22438e7120b72550c46aba68c4ffc69a

```



# 주피터 노트북 종료하기

## 종료 방법 2 - Quit 버튼을 클릭



# FPGA vs GPU

쉬는 시간에 강사님께 인공지능을 훈련할 때 FPGA와 GPU 중에서 어떤 걸로 사용하는지 질문했는데, 답변으로 둘 다 사용한다고 말하셨다.

그러면서 FPGA와 GPU를 비교하면서 설명해주셨다.

## FPGA

- 속도 ↑↑↑
- 생산성 ↓↓↓
- 가격 ↑

## GPU

- 속도 ↑
- 생산성 ↓↓↓
- 가격 ↓

FPGA는 커스터마이징을 할 수 있기 때문에 속도를 최고로 끌어올릴 수 있다는 장점이 있지만, 생산성과 가격 측면에서는 GPU가 압승이다.

(참고) FPGA의 사용 분야는 군사, 우주 등에 쓰이며, 또한 FPGA의 생산성을 향상시키기 위해 파이썬의 PYNQ가 있음

# Windows vs Linux

한 수강생이 리눅스의 장점이 무엇인지 질문했다.

그러자 강사님이 Windows와 Linux를 비교하면서 설명해주셨다.

## Windows

- 소스코드 비공개
- 유료

## Linux

- 소스코드 공개
- 무료

리눅스를 쓰는 곳은 **대규모 데이터 처리**가 필요한 곳... 예를 들어 영상처리, CCTV 처리 이런 곳에 쓰임

(참고)

- Windows CE를 쓸거면 VxWorks를 써라
- 한국에서 임베디드는 윈도우에 C#을 올린다고 한다... (다음 장 참고)

# Windows vs Linux



예전에 롯데리아가 마감하기 전에 키오스크 화면에서 윈도우 기본 바탕화면을 본 적이 있다.

사진을 가져오기 위해서 구글에 검색했는데 마침 누가 찍어놓은 사진이 있다.

아니 근데 이건 임베디드가 아니고 윈도우즈프로그래밍 아닌가...

출처: <https://www.clien.net/service/board/park/14638516>

# 엔지니어링 위키 (노션)

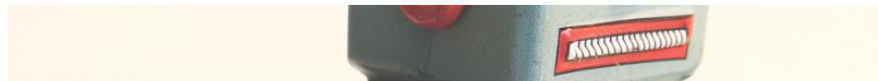
자유 게시판 >

## 엔지니어링 위키 개편하였습니다.

 링크뎀 카페매니저  1:1 채팅  
2022.10.18. 13:16 조회 81

 댓글 5  URL 복사

프로세스 전달이 어렵고 문서들이 너무 여기저기 흩어져 있다는 생각이 들어서  
전반적으로 한 번에 전부를 볼 수 있도록 구성을 개편하였습니다.



## 이러링 위키

[ 실무, 설정 가이드 등 엔지니어링의 모든 것을 확인하세요! ]

### 프로세스

니어링 지침

니어링 지침

주기

...

영어

장 인터뷰

ed Master Lv.2 수업 관련 ...

관련 학습

### 코드베이스

✓ 코드 검토

 React

 Vue

 Python

 C

 Rust

 Flutter

 System Verilog

 Java

### 질문 게시판

 리눅스 프로그래밍

 자료구조

 펌웨어 프로그래밍

 Rust 프로그래밍

 C 프로그래밍

 Git 형상 관리

 AVR 프로그래밍

 AVR 프로그래밍

 AVR 프로그래밍

### 재작자 :

 자유주

 자유주

 자유주

 자유주

 자유주

 자유주

 자유주

 자유주

 자유주

이번에 강사님이 임베디드 Lv1과 Lv2 그리고 웹  
노션을 전부 합치셨다.

아직 공사 중이지만 임베디드와 관련된 페이지만  
보더라도 배워야 할 게 수두룩 한 것 같다.. ㅋㅋ

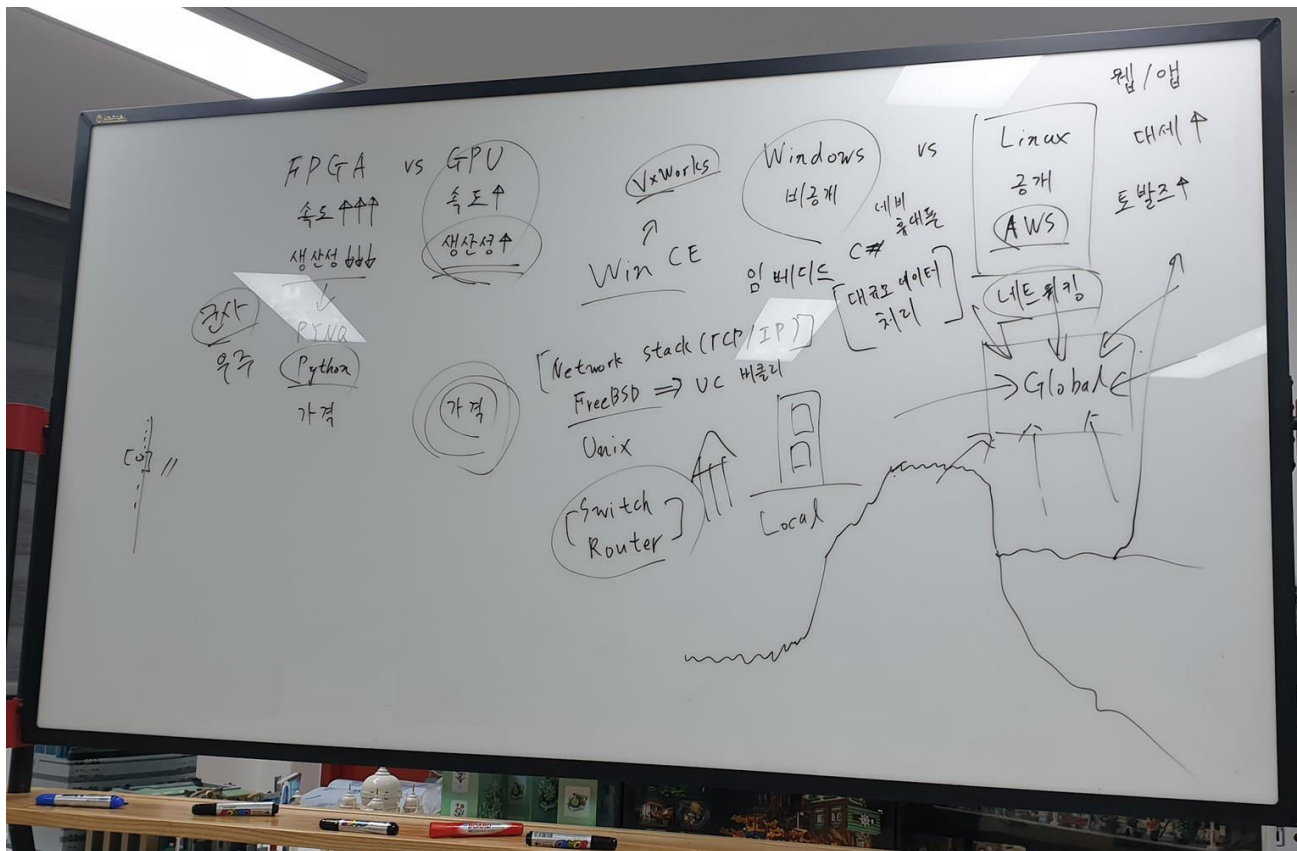
## sh 파일 실행방법

터미널 창에서 **sh** '파일명' 입력

오랜만에 수업하다보니까 수강생들과 잡담을 좀 많이 해버렸다. 그러다보니까 수업 끝나기 직전에 이번 수업에서 뭘 했는지 생각해보니 Anaconda 설치하고 파이썬 실습한 거 밖에 없었는데 솔직히 한 시간 동안 뽀 집중하면 했을 내용이긴하지만 시간이 아깝다는 생각이 들었다. 근데 잡담이 은근히 재밌었다 ㅋㅋㅋㅋ (하루 날 잡아서 회식 한 번 더해야 할 듯..?)

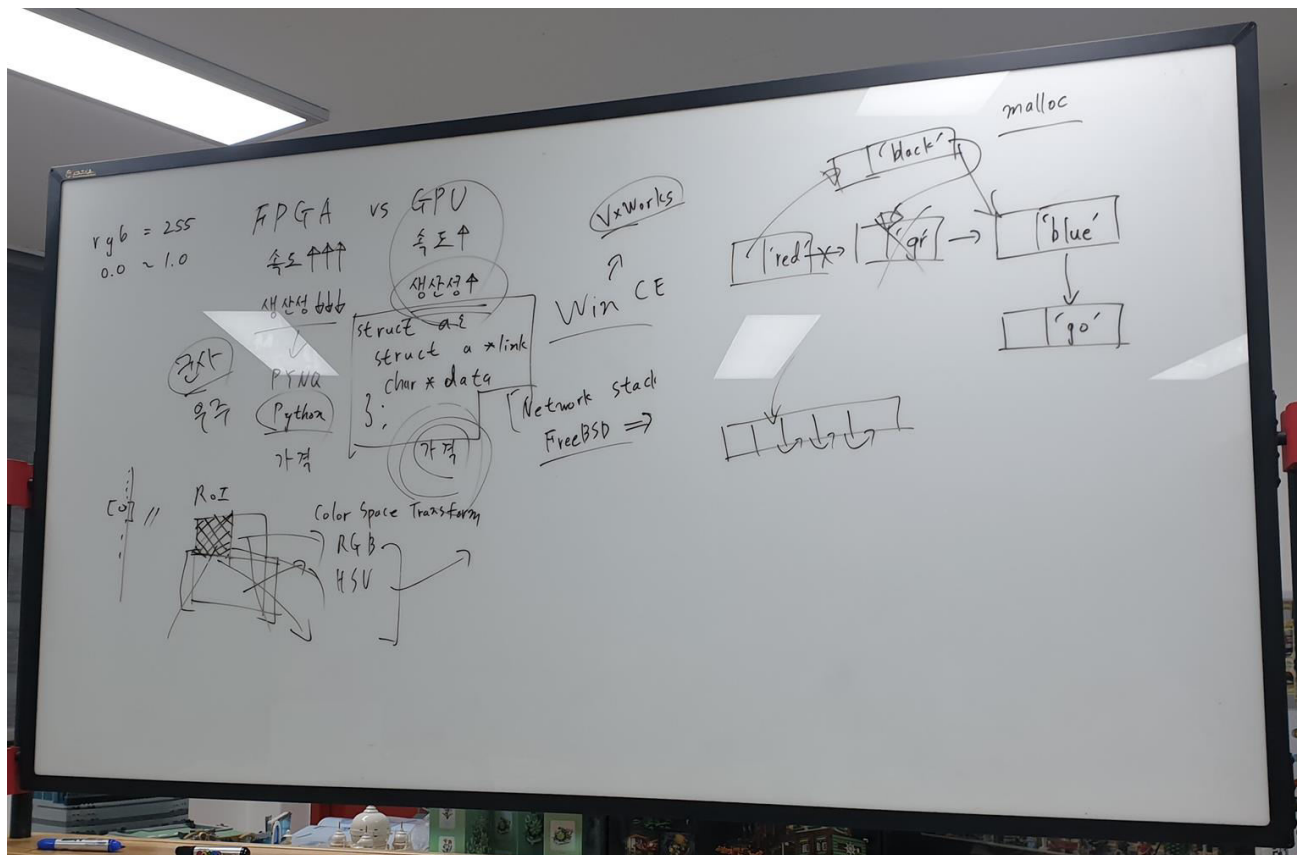
다음에는 잡담 관리하면서 시간을 금 같이 써야겠다 ㅋㅋ

# 수업내용 사진





# 수업내용 사진



# 수업내용 사진

