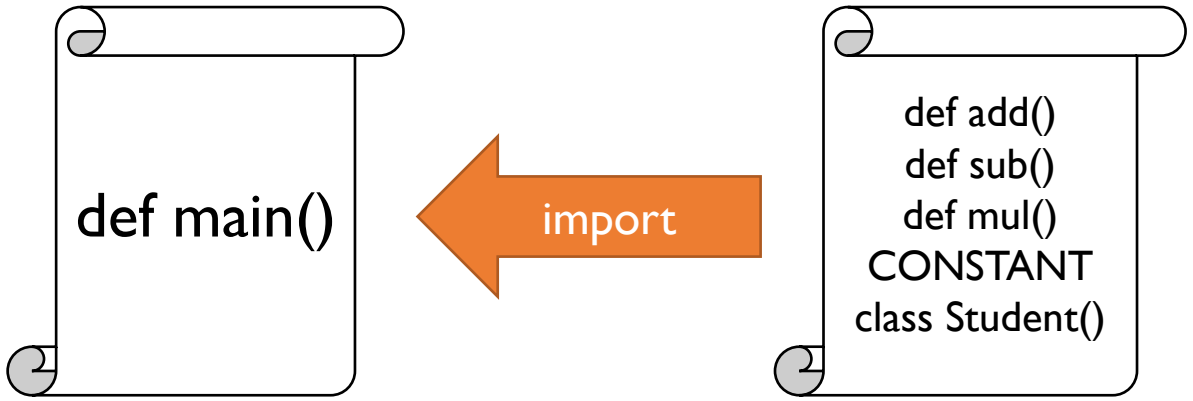
MODULE & PACKAGE

Importing from Other Files

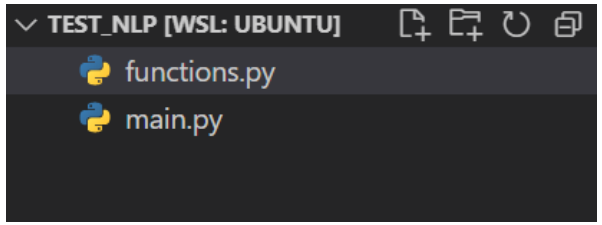


만약 다른 파일의 함수를 가지고 오고 싶다면?

- 다른 사람이 완성한 함수와 클래스 사용
- 내장 & 외부 라이브러리 사용

→ 모듈화가 필요

Import



./main.py

```python
import functions

print(functions.add(1, 2))
```

./functions.py

```python
def add(num1: int, num2: int) -> int:
    return num1 + num2
```

- 파이썬에선 모듈 == .py 파일
- Import 구문을 사용하여 모듈을 불러옴
  - 해당 파일 최상위에 선언된 모듈의 요소들을 불러오기 가능
  - module.element 식으로 사용
- . 혹은 .. 없이는 **절대 경로 기준** (Python이 실행되는 곳)

Notion for Importing

main.py

```python
import functions

print(functions.add(1, 2))
```

functions.py

```python
def add(num1: int, num2: int) -> int:
    return num1 + num2

print("Import 문은 global 코드 전체를 실행한다!")
```

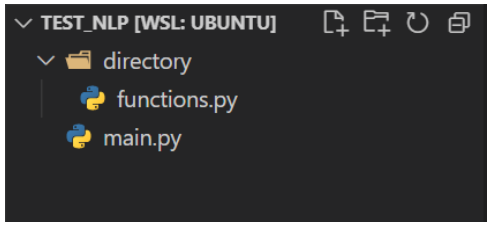
- Import 문은 Import된 .py 파일을 처음부터 끝까지 실행시킨다
- 만약 해당 모듈을 main으로 했을 때 특정 Block을 실행시키고 싶다면?
  - __name__ 기본 변수는 현재 모듈의 이름을 보여줌
  - Main으로 실행 중에는 "__main__"이라는 특수 이름을 가짐

```python
def add(num1: int, num2: int) -> int:
    return num1 + num2

print(__name__)
if __name__ == "__main__":
    print("이 코드는 functions 모듈이 메인일 때만 실행")
```

Import Examples 1

main.py

```python
import directory.functions                  # 폴더내 .py 파일
print(directory.functions.add(1, 2))

import directory.functions as func          # as로 별칭 만들기
Print(func.add(1, 2))

from directory import functions  # from으로 특정 부분 import
print(functions.add(1, 2))

from directory.functions import add         # 특정 함수 import
print(add(1, 2))

from directory.functions import *           # *로 모두 import
print(add(1, CONSTANT))                     # 권장하지 않음
```

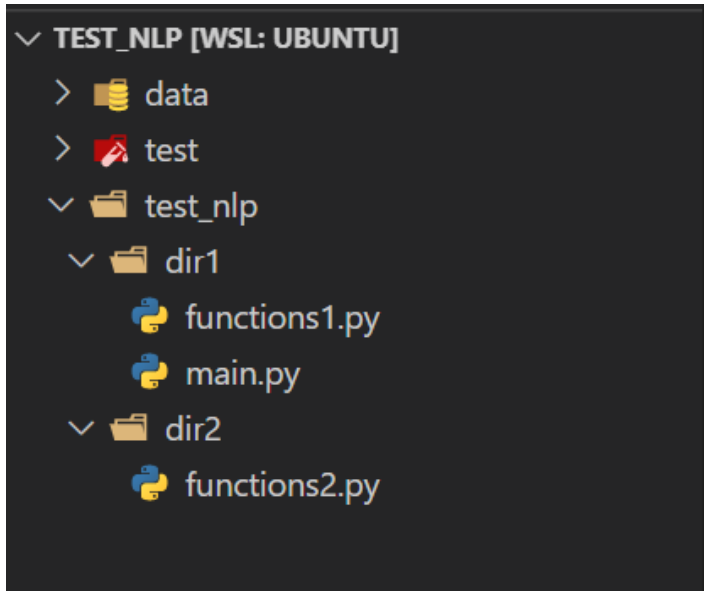


directory/functions.py

```python
CONSTANT = 10

def add(num1: int, num2: int) -> int:
    return num1 + num
```

- **from**: 특정 모듈/폴더내에서 import
- **as**: 별칭 만들기 (alias)

Import Examples 2



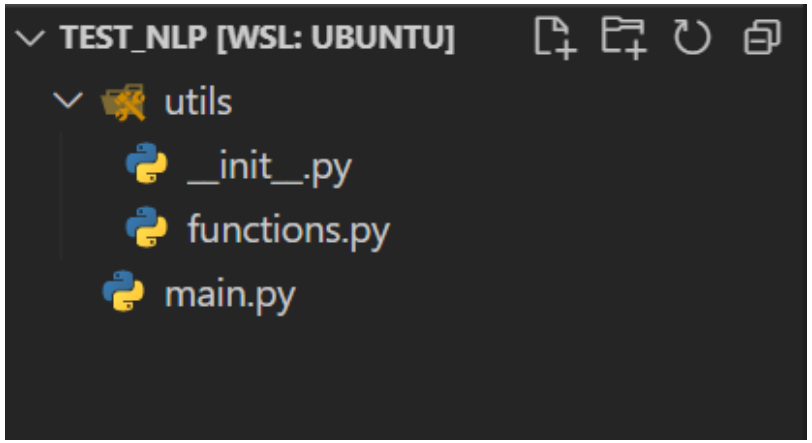
test_nlp/dir1/main.py

```python
from test_nlp.dir1.functions1 import add # 절대 경로
print(add(1, 2))

from .functions1 import add              # 현재 폴더
Print(add(1, 2))

from ..dir2.functions2 import sub        # 부모 폴더
Print(sub(1, 2))                  # 모듈 형태로 실행 필요
```

- 최상위에선 상대 경로가 작동되지 않음
  - 최상위를 거치는 경우 포함
  - 일반적으로 프로젝트 이름으로 폴더를 만들어 코드를 넣음
  - 부모 폴더 접근을 위해서는 모듈 형태로 실행 필요

```
(base) napping ~/test_nlp
python -m test_nlp.dir1.sub_main
```

__init__.py File



main.py

```python
import utils

print(utils.add(1, utils.CONSTANT))
```

utils/__init__.py

```python
from .functions import add

CONSTANT = 10
```

utils/functions.py

```python
def add(num1, num2):
    return num1 + num2
```

__init__.py 파일
폴더 import시 초기화 가능

The Python Standard Library

```
import random                                          # 난수 관련 라이브러리
print (random.randint(0, 100))                         # 0에서 100까지 정수 중 하나 반환
Print (random.uniform(0, 1))                           # 0에서 1까지 균등 분포에서 샘플링

import time                                            # 시간 관련 라이브러리
start = time.time()                                    # 현재 시각 (unix 시간 기준)
time.sleep(1)                                          # 1초 기다리기
print(time.time() - start)

import threading                                       # 쓰레드 관련 라이브러리

def print_function():
    print("사실, 파이썬은 GIL 때문에 사실상 싱글 쓰레드입니다.")
    time.sleep(1)

thread = threading.Thread(target=print_function) # 쓰레드 만들기
thread.start()                                         # 쓰레드 시작
thread.join()                                          # 쓰레드 수거
```

파이썬은 강력하고 다양한 표준 라이브러리를 가지고 있음

https://docs.python.org/3/library/

External Library & Package Managing

- **파이썬 표준 라이브러리로 해결할 수 없다면....?**
  - 인터넷 상의 오픈 소스 라이브러리 설치 필요
  - 수치 그래프 그리기 → matplotlib
  - 웹 서버 만들기 → flask
  - GPU 연산 사용하기 → cupy
  - 딥러닝 전용 라이브러리 → tensorflow & pytorch
- **웹 서버 프로젝트와 딥러닝 프로젝트가 따로 있다면...?**
  - 한 파이썬 위에 둘 다 설치한다 → 관리가 어려움
  - 각각 다른 환경 & 파이썬에서 돌리고 싶은데...

→ 패키지 관리자가 필요!

Python Package Manager

PIP + Virtual env

Python 기본 패키지 관리 프로그램

Anaconda3

https://www.anaconda.com/products/individual
기계학습 및 수치해석 특화 패키지 관리 프로그램
상용 프로그램이다

Notions for Colab Users

- 코랩은 개별 노트북 마다 개별 환경이 설정됨
  - **노트북이 꺼지면 환경이 삭제됨**
- 대부분의 라이브러리가 사전 설치되어 있음
  - matplotlib, tensorflow, pytorch, scipy, numpy, …
  - 만약 추가 설치가 필요한 경우 pip을 사용

→ 굳이 Anaconda를 설치할 필요가 없음

Anaconda & Minconda

- **Anaconda**
  - 기본 설치판
  - 200개 이상의 라이브러리가 사전 설치됨
  - 무거움
  - https://www.anaconda.com/products/individual
- **Minconda**
  - 최소 설치판
  - Anaconda의 기본 기능만을 설치
  - 가벼움
  - Miniconda — Conda documentation

Creating Virtual Environment

가상환경 만들기

환경 이름

가상환경 활성화

활성화된 가상환경

Managing Virtual Environment

```
(nlp)  napping  ~/test_nlp
 conda deactivate
```

가상환경 나가기

```
(base)  napping  ~/test_nlp
 conda install <패키지 이름> -c <설치 채널>
```

패키지 설치 설치 채널

```
(base)  napping  ~/test_nlp
 conda install python=3.9 pytorch -c pytorch
```

- 채널이 명시되어 있지 않을 경우 default 채널 탐색
- 채널은 문서를 참고하거나 Anaconda에 검색

https://anaconda.org/

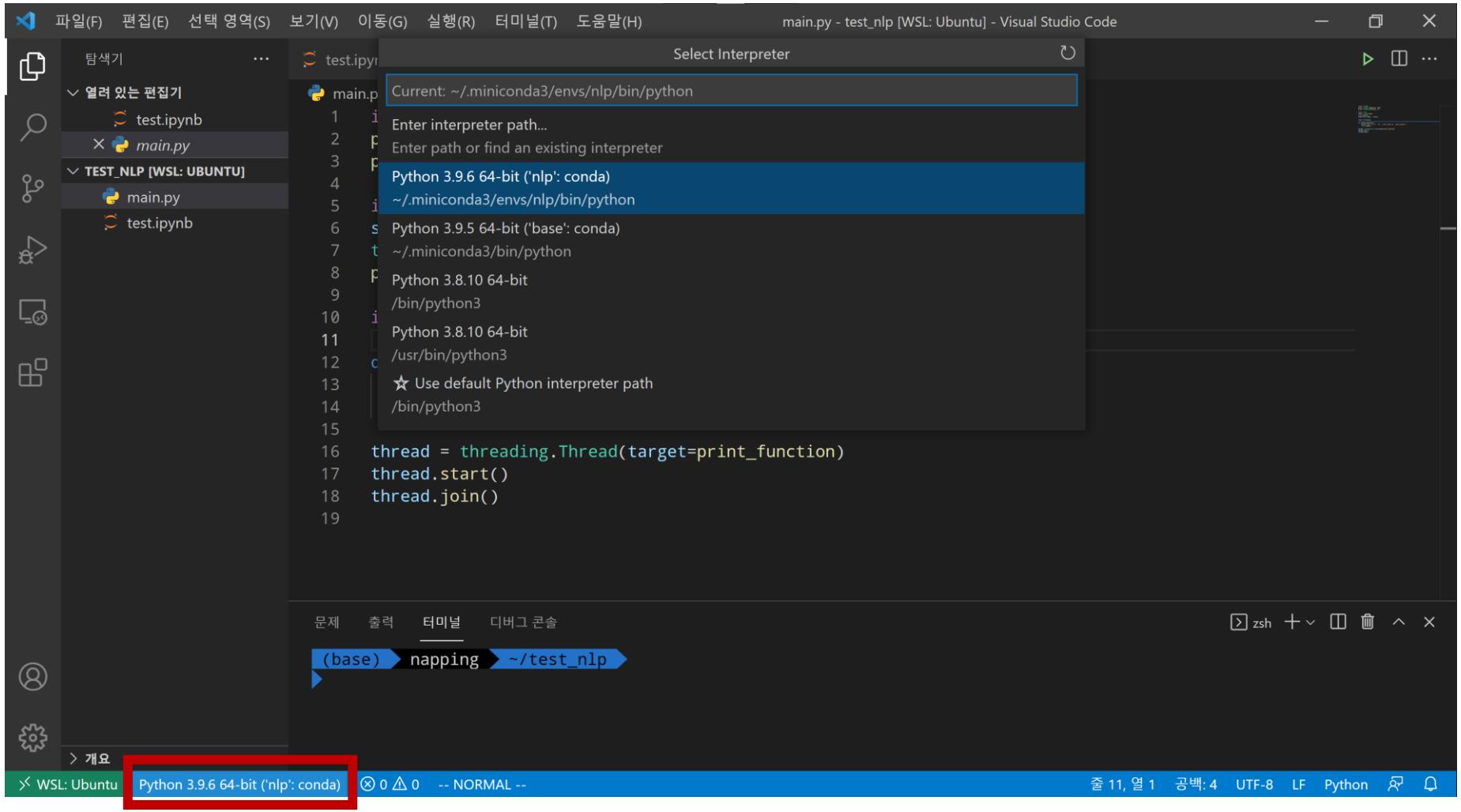
Managing Virtual Environment

```
(nlp) napping ~/test_nlp
 conda list
# packages in environment at /home/napping/.miniconda3/envs/nlp:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                        main
_openmp_mutex             4.5                       1_gnu
blas                      1.0                         mkl
ca-certificates           2021.7.5             h06a4308_1
certifi                   2021.5.30        py39h06a4308_0
```

```
(nlp) napping ~/test_nlp
 conda list | grep numpy
numpy                     1.20.3           py39hf144106_0
numpy-base                1.20.3           py39h74d4b33_0
```

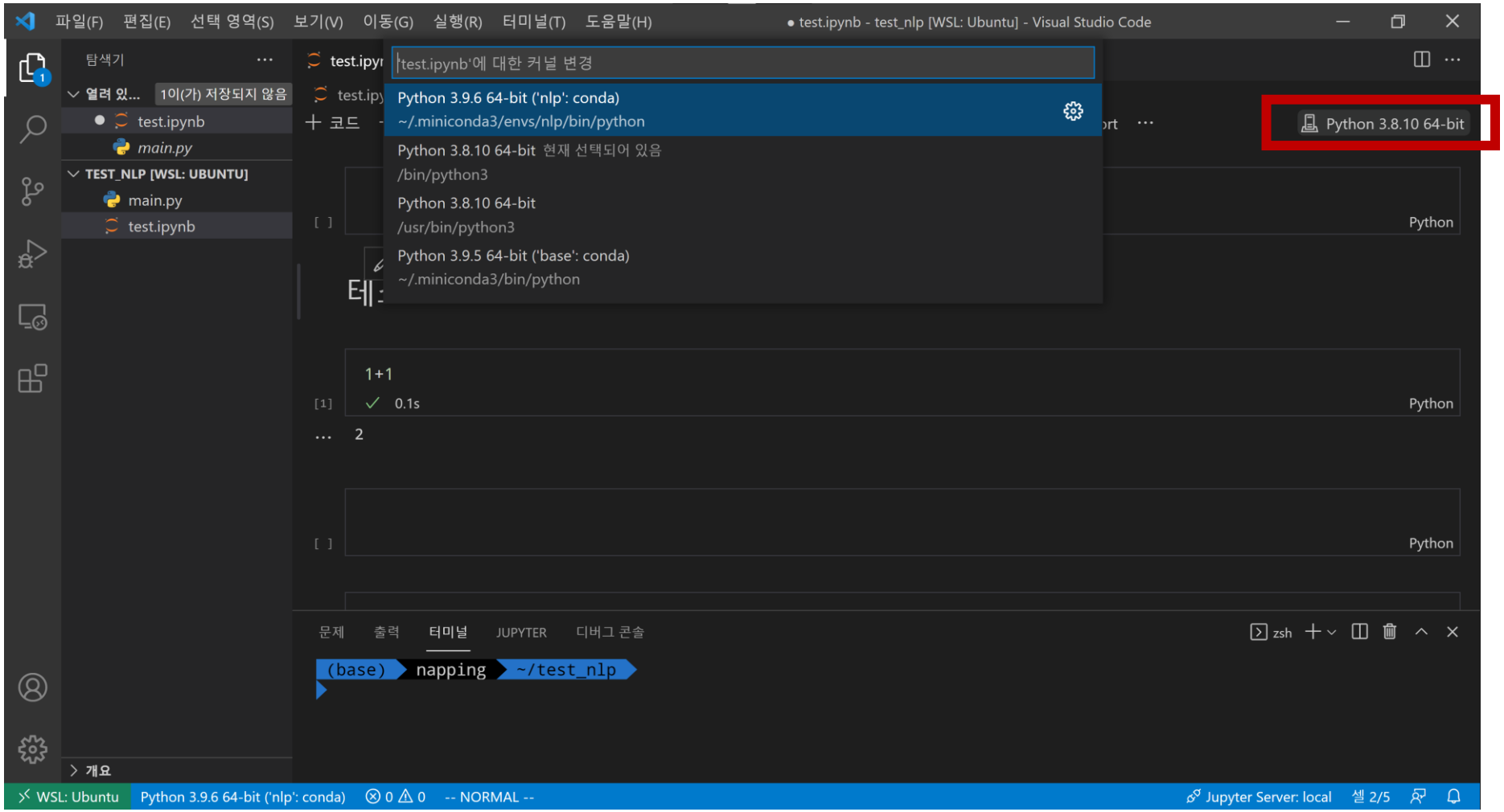
- conda list로 현재 환경에 설치된 패키지 확인
  - Linux grep 명령어랑 결합시 편리

Conda in VsCode



왼쪽 아래 버튼을 눌러 가상환경 설정가능

Conda in VsCode with Jupyter

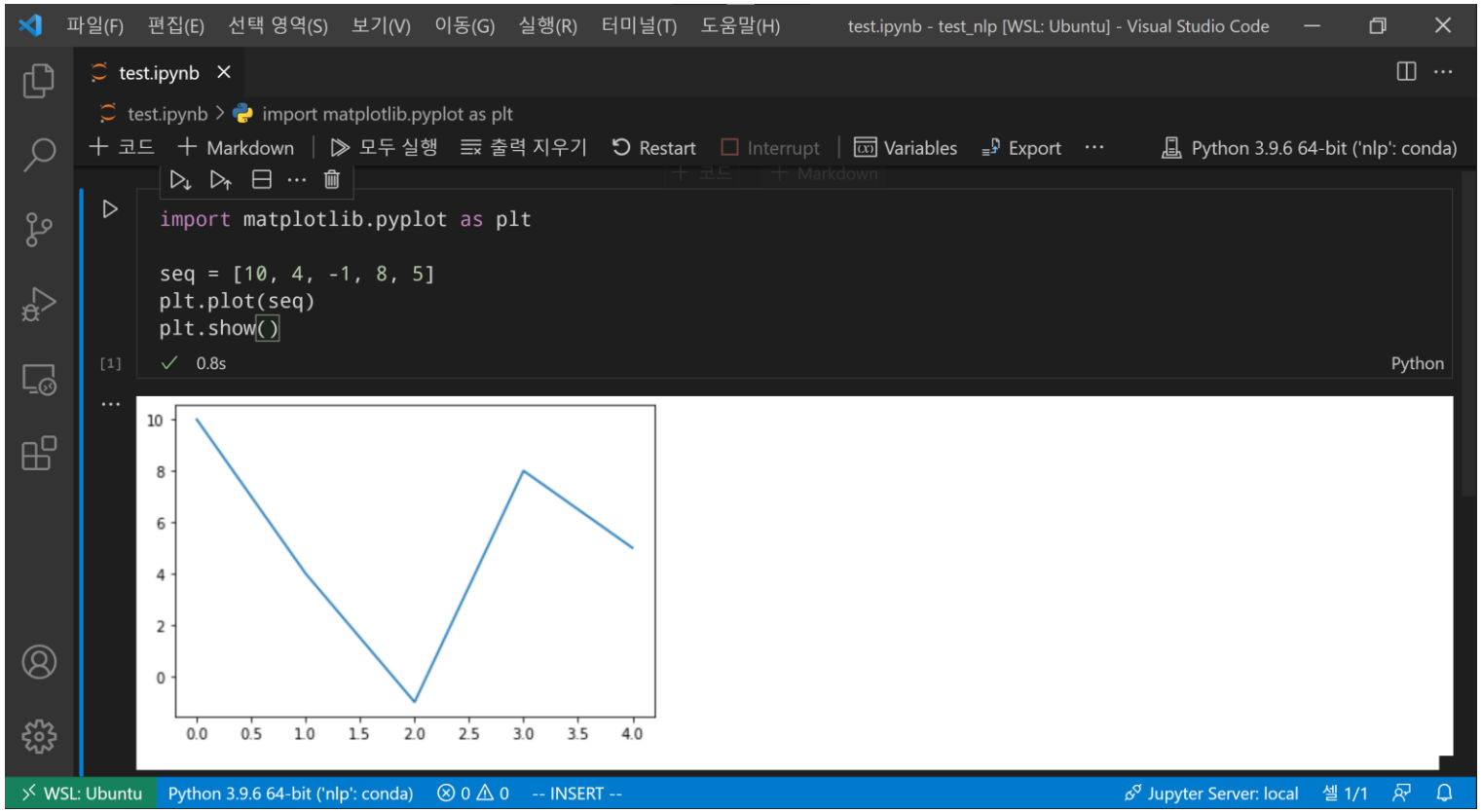


.ipynb의 경우 쥬피터 서버의 가상환경 설정 필요

Using External Libraries: matplotlib

matplotlib: 그래프 그리기 라이브러리

```
(nlp) napping ~/test_nlp
> conda install matplotlib
```



Using External Libraries: tqdm

tqdm: 진행 바 (Progress Bar) 만들기

```
(nlp) napping ~/test_nlp
 conda install tqdm
```

```
Python 3.9.6 (default, Jul 30 2021, 16:35:19)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import time
>>> from tqdm.auto import trange
>>> for _ in trange(10):
...     time.sleep(1)
...
 30%|                                                  | 3/10 [00:03<00:07,  1.00s/it]
```