3. PyCharm 가상 환경 튜토리얼

☑ 복습	☐
≔ 개념	
☑ 블로그	☐
≡ 페이지	

1. 튜토리얼 시작 전

이번 토픽에서는 많은 수강생분들에게 익숙할 PyCharm과 같은 IDE가 아니라 터미널에서 파이썬 환경을 다루는 방법에 대해서 공부했습니다.

토픽 앞쪽에서 말씀드렸다시피, 실제로 파이썬 환경이 돌아가는 원리를 이해하는 데에는 도움이 더 많이 되기 때문인데요. 원리를 파악하는 데 더 편하긴 해도 실제로 파이썬을 조금 더 전문적으로 활용할 때는 문서 편집기와 터미널을 쓰는 경우가 그렇게 많지는 않습니다.

2020년 PyCharm 개발자 설문조사에 의하면 다양한 문서 편집기 + 에디터 사용 비율은:

- **PyCharm: 33%**
- **VS Code: 29%**
- **Vim: 8%**
- **Sublime Text: 4%**
- **Jupyter Notebook: 4%**

이렇습니다.

그렇기 때문에 사실 이번에 배운 내용이 **원리를 파악하는 데** 도움이 돼도, 다른 도구에서 가상 환경을 사용하려고 어떻게 해야되는지 헷갈릴 수 있죠.

이 점을 보안하기 위해서 "파이썬 가상 환경 튜토리얼" 챕터에서는 파이썬 환경 토픽 선수강 토픽들에서 사용했던 PyCharm에서 파이썬 환경을 다루는 방법들을 다룹니다. 이번 토픽에서 배운 내용을 기반으로, 같은 기능들을 PyCharm에서 어떻게 사용할 수 있는지를 공부하는 건데요. 원리에 대한 설명보다는 PyCharm 기능 설명 위주로 진행됩니다.

관심 있으신 분들은 꼭 살펴보고 넘어가시는 걸 추천드립니다.

2. PyCharm 가상 환경 튜토리얼

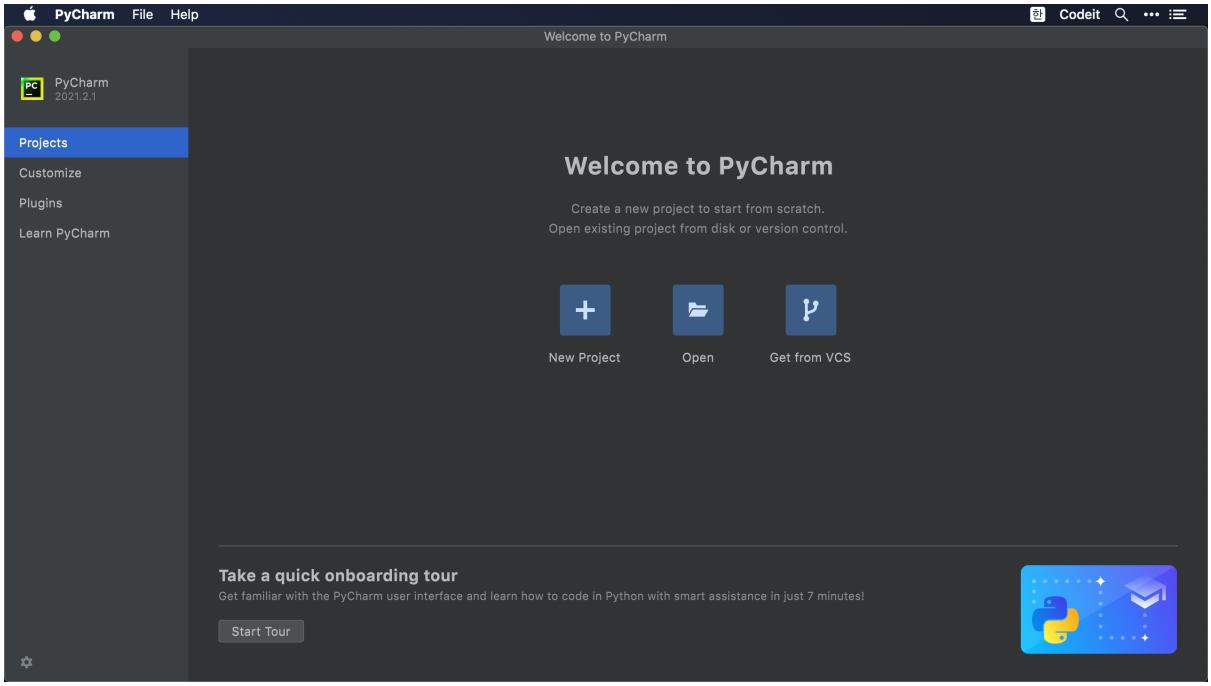
PyCharm

PyCharm은 다른 IDE나 문서 편집기와는 달리 온전히 파이썬 전용 도구입니다. 그렇기 때문에 다른 프로그램들보다 파이썬 자체를 쉽게 사용할 수 있도록 특화돼있는데요. 이번 튜토리얼에서는 여러 기능들 중 가상 환경을 쉽게 사용하는 법에 대해서 공부하겠습니다.

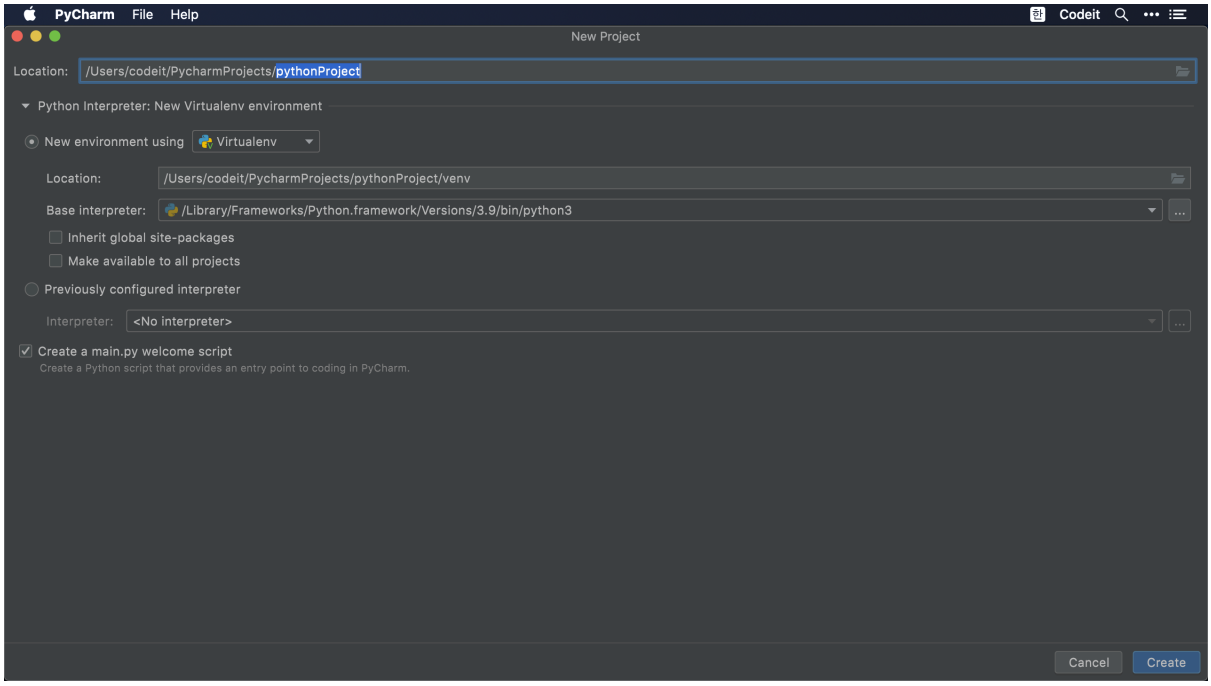
참고로 이 튜토리얼은 PyCharm 2021년 버전으로 진행이 되기 때문에 다른 버전을 사용하고 계시면 정확히 똑같이 작동하지 않을 수도 있습니다. 새롭게 설치를 하고 싶으신 분들은 이 링크를 사용하세요.

새로운 프로젝트에서 가상 환경 만들기

PyCharm을 키게 되면 처음에 이런 화면이 나옵니다.



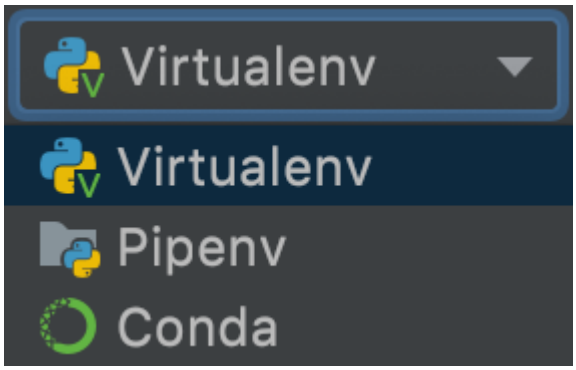
새로운 프로젝트를 만들 거나, 이미 존재하는 프로젝트를 열어서 파이참을 사용할 수 있는데요. 저희는 새로운 프로젝트를 생성하는 거부터 해볼게요. 위에서 "New Project"를 클릭하세요. 그럼 이런 화면이 나옵니다.



가장 위에 Location에서는 프로젝트 디렉토리 경로를 설정할 수 있습니다. 파이참은 기본적으로 모든 프로젝트를 ~/PycharmProjects/에 저장하기 때문에 특별히 다른 경로에 프로젝트를 만들고 싶지 않는 이상 그냥 하이라이트 된 부분의 이름에 프로젝트 이름을 입력하시면 됩니다.

Python Interpreter: 라는 섹션이 있는데요. 바로 여기서 프로젝트를 만들 때마다 그 프로젝트를 위한 가상 환경을 만들 수 있습니다. 프로젝트만의 새로운 가상 환경을 만들어서 사용하고 싶으면 New environment using을 선택하시면 되고요. 이미 컴퓨터에 있는 다른 가상 환경을 그냥 가지고 와서 사용하고 싶으시면 Previously configured interpreter를 선택하시면 됩니다.

저희는 그냥 새롭게 만들어서 사용할게요. 그리고 새로운 가상 환경을 만들 때도 어떤 도구를 사용할 건지 선택할 수 있는데요. Virtualenv라고 된 부분을 클릭하시면 이렇게 나옵니다:



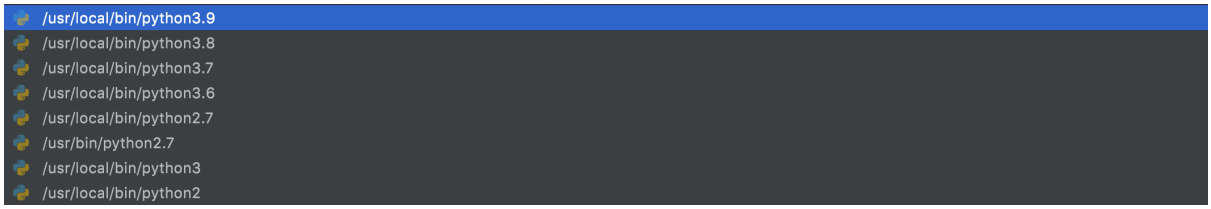
저희가 이번 토픽에서 배운 Virtualenv뿐만 아니라, Pipenv, 그리고 Conda를 사용해서 가상 환경을 만들 수 있습니다. 저희는 Virtualenv로 가상 환경을 다루는 방법밖에 안 봤으니까

이걸 선택할게요. 실제로 2020년 파이썬 개발자 설문 조사에서 답변한 개발자의 50%가 가상 환경을 다룰 때 virtualenv를 사용한다고 답했으니까 virtualenv가 가장 안전한 선택지라고 할 수 있겠죠?

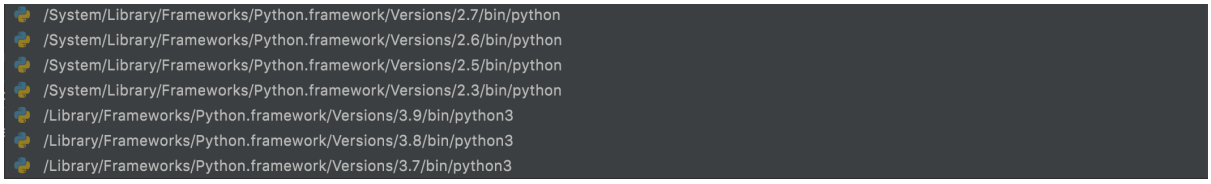
그리고 바로 밑을 살펴보면 Location이 있는데요. 여기서 가상 환경 디렉토리의 경로를 설정할 수 있습니다. 따로 설정하지 않으면 프로젝트 디렉토리 안에 venv라는 이름의 가상 환경이 생성됩니다. 이걸 바꾸고 싶으시면 클릭하신 후 이름 또는 경로를 수정할 수 있습니다. 저는 그냥 project_env로 바꿀게요.

이제 가상 환경 이름(경로)을 정했습니다. 다음은 바로 밑에 Base interpreter이라고 된 부분을 보세요. 가장 중요한 인터프리터를 정하는 부분입니다. 클릭해보면 다양한 옵션들이 나오는데요.

제 컴퓨터에서는 가장 위에는 이렇게 나오고:



스크롤을 밑까지 하면 이렇게 나옵니다.



파이참이 흔히 파이썬 인터프리터가 저장되는 경로들을 탐색해서 찾은 인터프리터들입니다. 버전만 같으면 이중에서 어떤 걸 골라도 상관 없습니다. 실제로 여기있는 모든 인터프리터가 진짜 인터프리터 프로그램이 아니라 symlink(symbolic link / 바로 가기의 개념)를 통해 다른 인터프리터를 가르키는 경우도 많습니다.

3.9 버전의 인터프리터를 사용하고 싶으면 /usr/local/bin/python3.9나 /Library/Frameworks/Python.Frameworks/Versions/3.9/bin/python3 둘 중 하나를 선택해주세요. 저는 이번 토픽에서 본 경로인 /Library/Frameworks/Python.Frameworks/Versions/3.9/bin/python3를 사용하겠습니다.

그 다음에 두 가지 옵션이 있는데요.

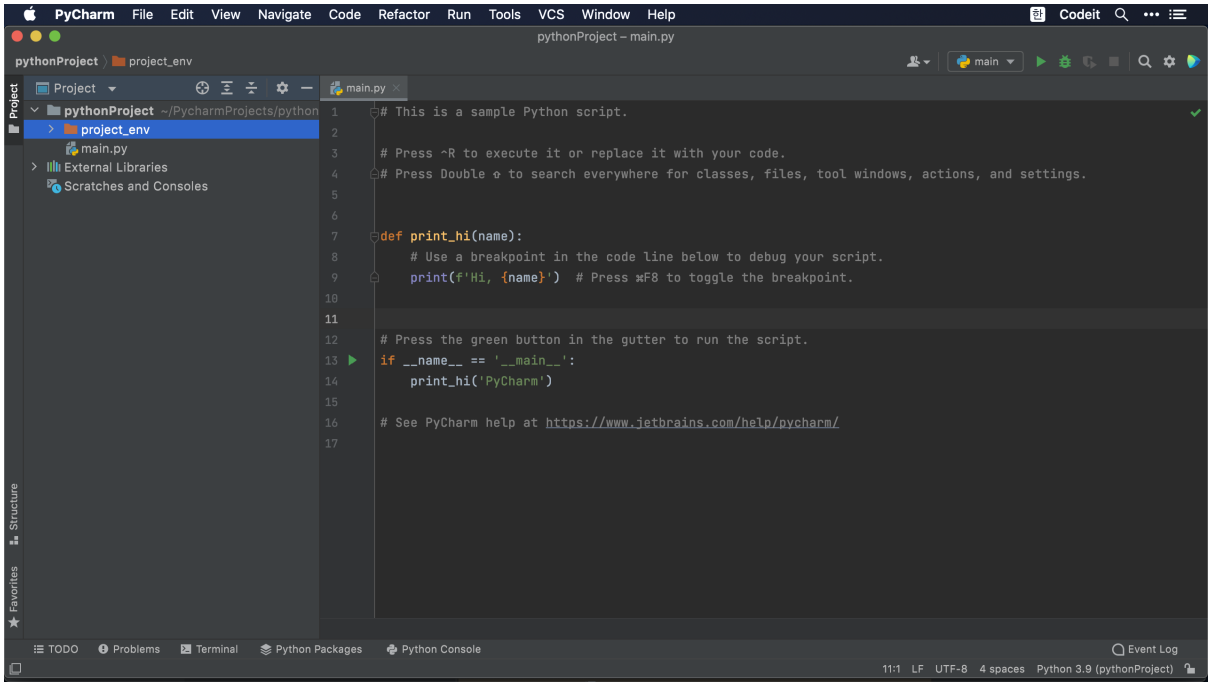
Inherhit global site-packages 옵션을 선택하면, 새로운 가상 환경에서 /Library/Frameworks/Python.Frameworks/Versions/3.9/bin/python3 인터프리터와 연결된 pip으로 설치된 패지키들을 모두 복사해서 새로운 가상 환경에 추가해줍니다.

그리고 Make available to all projects를 선택하면 다른 프로젝트에서 이제 만들 새로운 가상 환경을 사용할 수 있게 만들어줍니다.

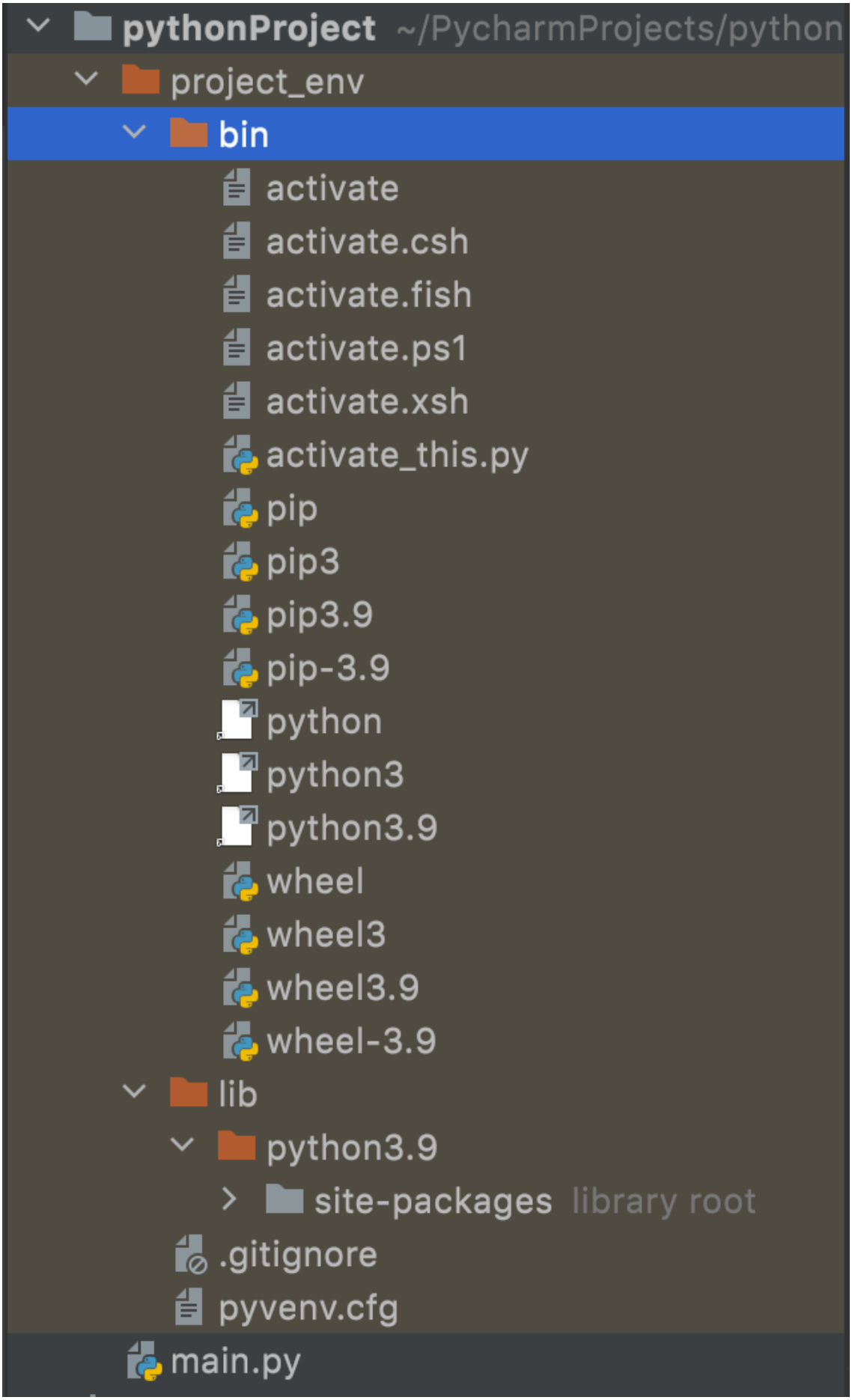
저는 둘 다 선택하지 않을게요.

마지막으로 Create a main.py welcome script 옵션을 선택하고 오른쪽 아래 있는 create 버튼을 누르겠습니다.

그럼 새로운 프로젝트와 함께 가상 환경이 만들어집니다.



새로운 프로젝트 안에는 자동으로 생성한 main.py와 project_env 가상 환경 디렉토리가 있는데요.

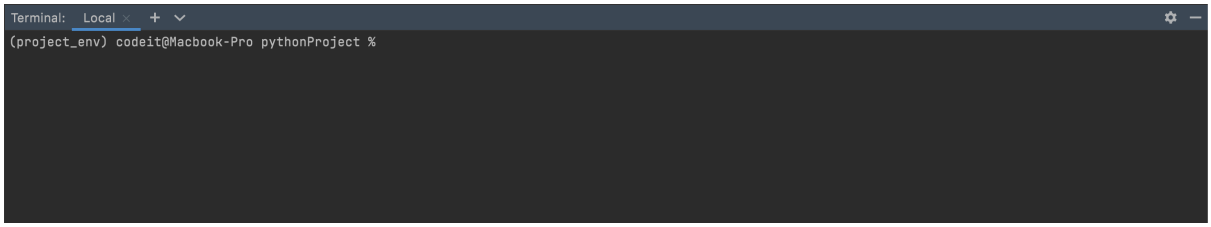

pythonProject ~/PycharmProjects/python
project_env
bin
activate
activate.csh
activate.fish
activate.ps1
activate.xsh
activate_this.py
pip
pip3
pip3.9
pip-3.9
python
python3
python3.9
wheel
wheel3
wheel3.9
wheel-3.9
lib
python3.9
site-packages library root
.gitignore
pyvenv.cfg
main.py

project_env 안을 살펴보면 저희가 봤던 파이썬 환경 디렉토리 구조 그대로 가상 환경이 만들어진 걸 확인할 수 있습니다. 이미 IDE는 이 안에 있는 인터프리터를 사용하도록 설정이 돼있는데요. 이 안에서 파이썬 파일에 오른쪽 클릭을 해서 실행시키면 이제 project_env 환경이 사용됩니다.

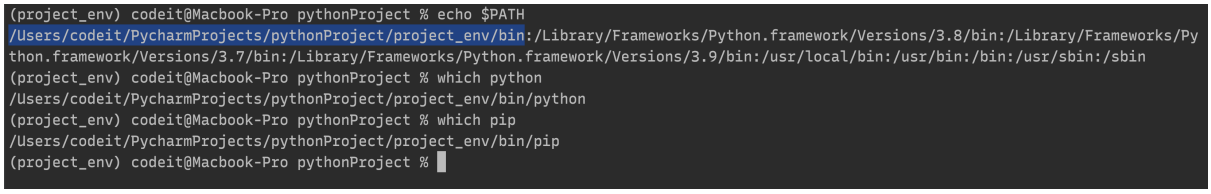
사실 파이참 안에서도 앱 GUI가 아니라 터미널을 사용할 수 있는데요.

화면 가장 아래 부분에 이렇게 나와 있는 부분에서

Terminal을 클릭해주세요. 그럼 이렇게 나옵니다:



이미 앞에 (project_env)라고 나와 있는 걸 보면 터미널에 자동을 가상 환경이 적용된 걸 확인할 수 있죠? 그리고 echo $PATH, which python , which pip을 실행시키면



PATH 변수가 바뀐 것과 가상 환경 안 인터프리터와 pip이 사용되고 있는 걸 확인할 수 있습니다.

그리고 main.py 파일을 실행해도:

```
(project_env) codeit@Macbook-Pro pythonProject % python main.py
Hi, PyCharm
```

run 버튼을 클릭할 때랑 똑같이 작동합니다.

PyCharm에서 가상 환경 관리하기

방금까지는 파이참에서 가상 환경을 만드는 법을 봤는데요. 사실 새로운 내용은 딱 여기까지가 끝입니다.

파이참에서 가상 환경을 관리하는 법, 그러니까 패키지를 다운받고, 삭제하고, requirements.txt를 생성하고 이런 건 이미 이번 토픽에서 어떻게 하는지 전부 다 배웠습니다.

그냥 방금 봤던 터미널 창에서 이번 토픽 내용을 그대로 적용하시면 됩니다. 어짜피 새로운 가상 환경을 Virtualenv로 만들었기 때문에 그냥 모든 커맨드들을 동일하게 사용할 수 있죠.

하나의 예시로 패키지를 설치해서 코드에서 사용하는 걸 볼게요.

다시 터미널 창을 열어보세요

먼저 pip을 업그레이드 하고 시작할게요:

```
pip install --upgrade pip
```

그 다음에 pip list 커맨드를 사용하면

```
(project_env) codeit@Macbook-Pro pythonProject % pip list
Package    Version
---------- -------
pip        21.2.4
setuptools 57.0.0
wheel      0.36.2
(project_env) codeit@Macbook-Pro pythonProject %
```

현재 프로젝트에 설치된 패키지들이 모두 나옵니다. 그리고

```
pip install numpy
```

로 numpy 패키지를 설치한 후, main.py를 이렇게 수정해주시고, 터미널이나 GUI를 사용해서 실행시키면, 결과물이 원하는대로 출력되는 걸 확인할 수 있죠.

main.py

```python
import numpy as np

array = np.array([1, 2, 3, 4, 5, 6])
print(array)

```

터미널에서 `python main.py` 로 출력시킨 내용:

```
(project_env) codeit@Macbook-Pro pythonProject % python main.py
[1 2 3 4 5 6]
```

gui로 run을 시켰을 때 출력 내용:

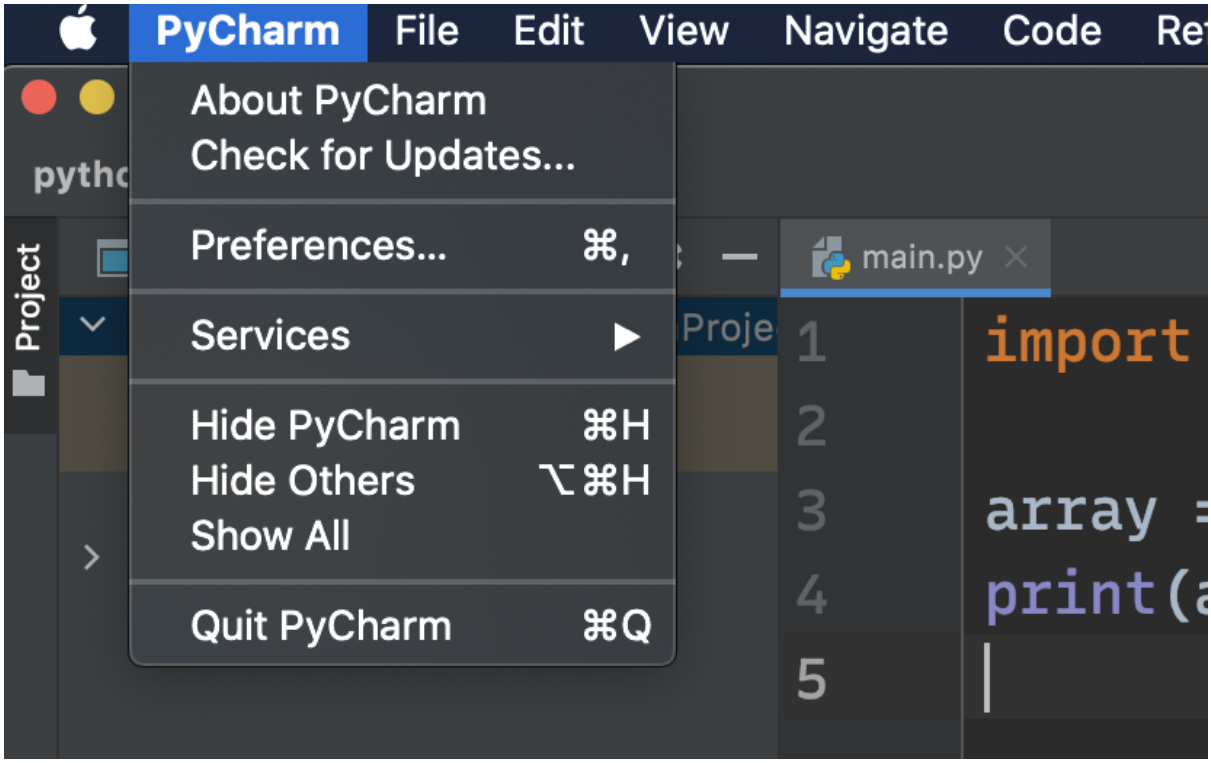
```
/Users/codeit/PycharmProjects/pythonProject/project_env/bin/python /Users/codeit/PycharmProjects/pythonProject/main.py
[1 2 3 4 5 6]

Process finished with exit code 0
```

`pip list`, `pip install numpy` 뿐만 아니라 `pip uninstall`, `pip freeze`, `pip freeze > requirements.txt` , `pip install -r requirements.txt` 등 모두 이번 토픽에서 봤던 거랑 정확히 똑같으니까요. PyCharm 내 터미널을 활용해서 배운 그대로 적용하시면 됩니다.

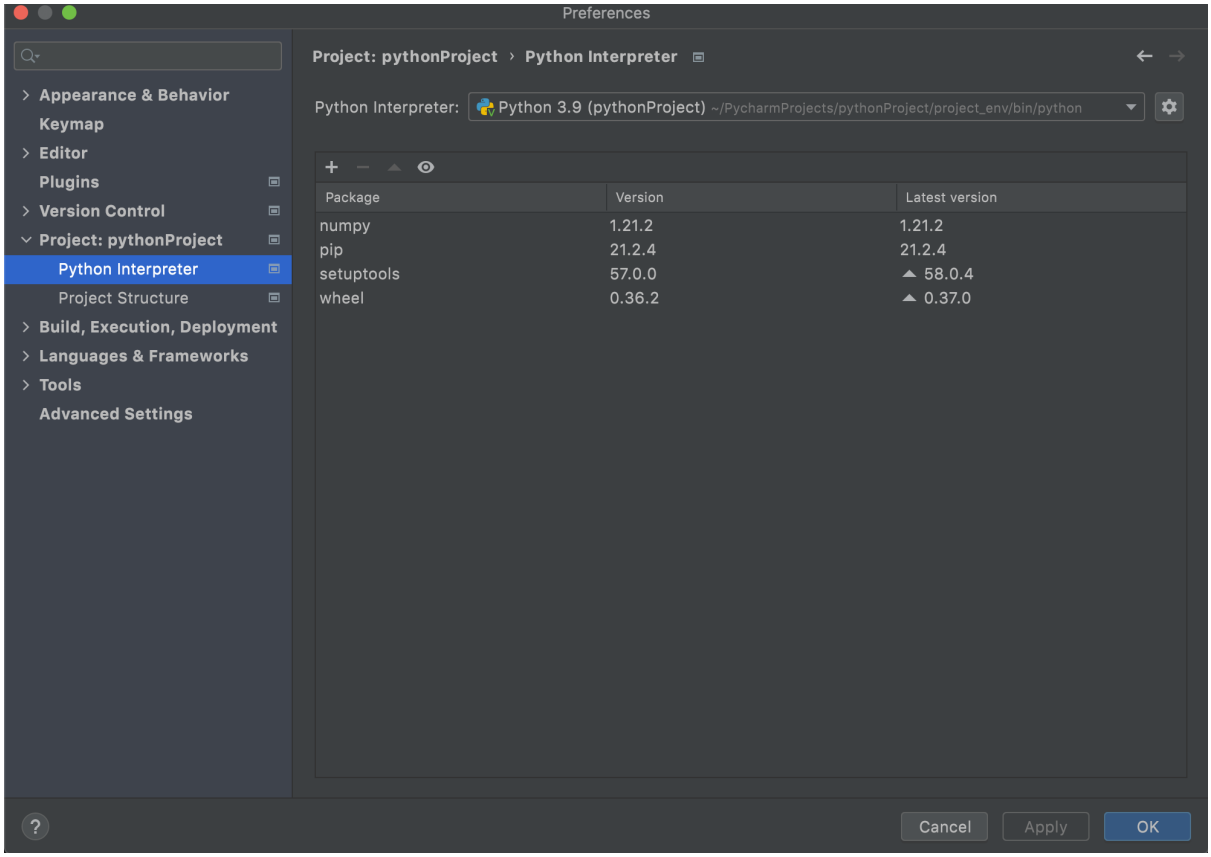
프로젝트 진행 도중에 가상 환경 만들기/바꾸기

물론 모든 프로젝트를 진행할 때 항상 처음부터 그 프로젝트만을 위한 **새로운** 가상 환경을 만들고 바꾸지 않고 계속 사용하면 좋을텐데요. 프로젝트를 진행할 때 프로젝트 단위로 파이썬 마이너 버전을 업그레이드하게 되거나 처음에 가상 환경을 만들 때부터 실수를 했으면 새로운 가상 환경을 만들어서 사용해야 되는 경우들이 종종 생깁니다. 이런 상황들을 대비해서 이미 만들어놓은 프로젝트에서 새로운 가상 환경을 만들고, PyCharm이 이 가상 환경을 사용하도록 설정하는 법에 대해서 알아보겠습니다.

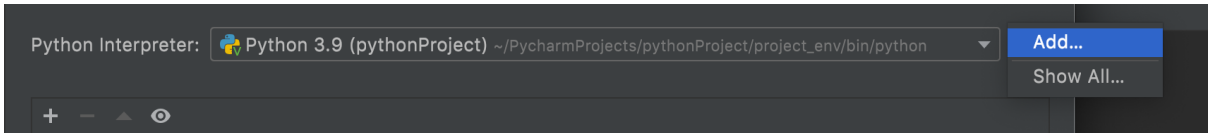
먼저 프로젝트 진행 도중 새로운 가상 환경을 만드는 방법에 대해서 알아보겠습니다.



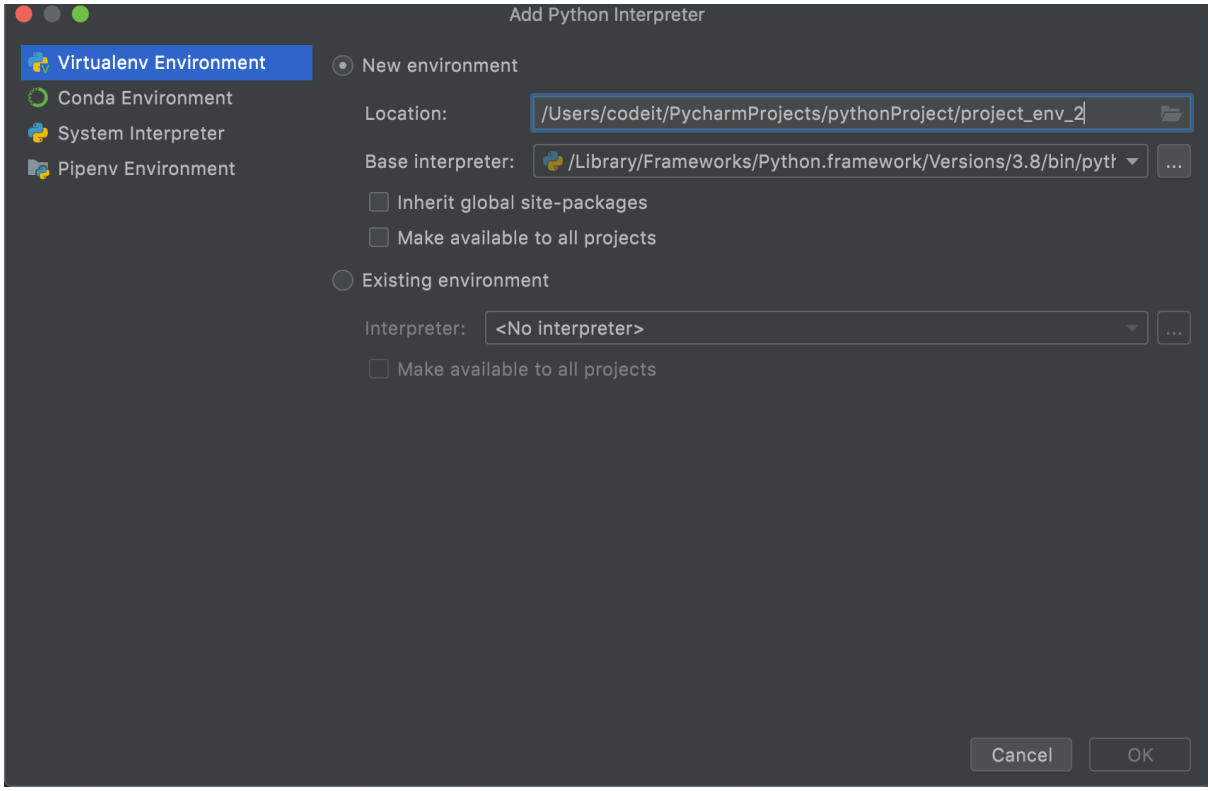
왼쪽 상단 PyCharm을 누른 , Preferences를 클릭해서 열어주세요. 그리고 Project: projectName로 들어가면 이렇게 된 걸 확인할 수 있는데요. 여기서 Python Interpreter에 해당하는 부분이 바로 현재 프로젝트에서 사용하고 있는 인터프리터입니다. (경우에 따라서 이 부분이 비어 있을 수 있습니다)



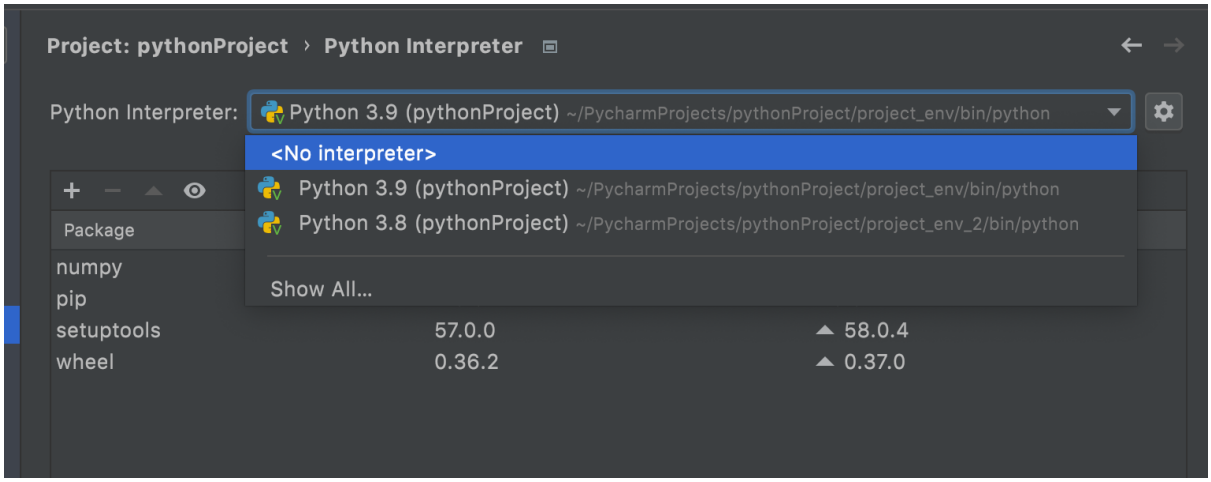
여기서 새로운 파이썬 인터프리터(가상 환경)를 사용하고 싶으면 Python Interpreter 옆 설정 아이콘을 클릭하시고 add를 누르세요.



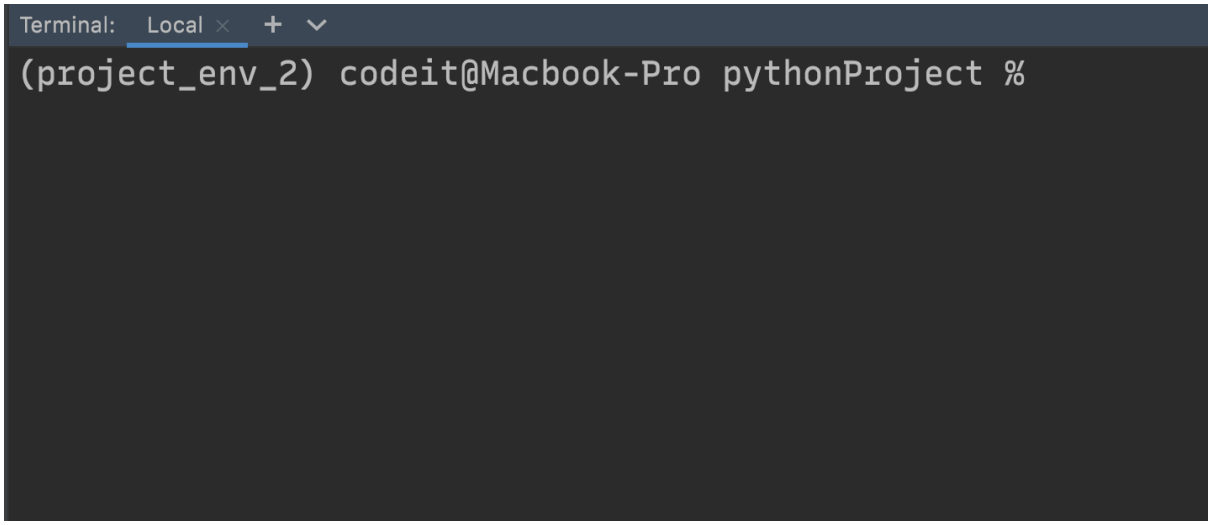
그럼 처음 프로젝트를 생성할 때와 똑같은 화면이 나오는데요. 여기서 그냥 새로운 가상 환경을 만들어주면 됩니다. 다만 Location은 이미 사용하고 있는 가상 환경과는 다른 경로로 만들어주세요. 저는 그냥 project_env_2라는 이름으로 만들게요. 그리고 이번에는 base interpreter은 3.8 버전으로 바꿔주겠습니다. 그리고 ok를 누르세요.



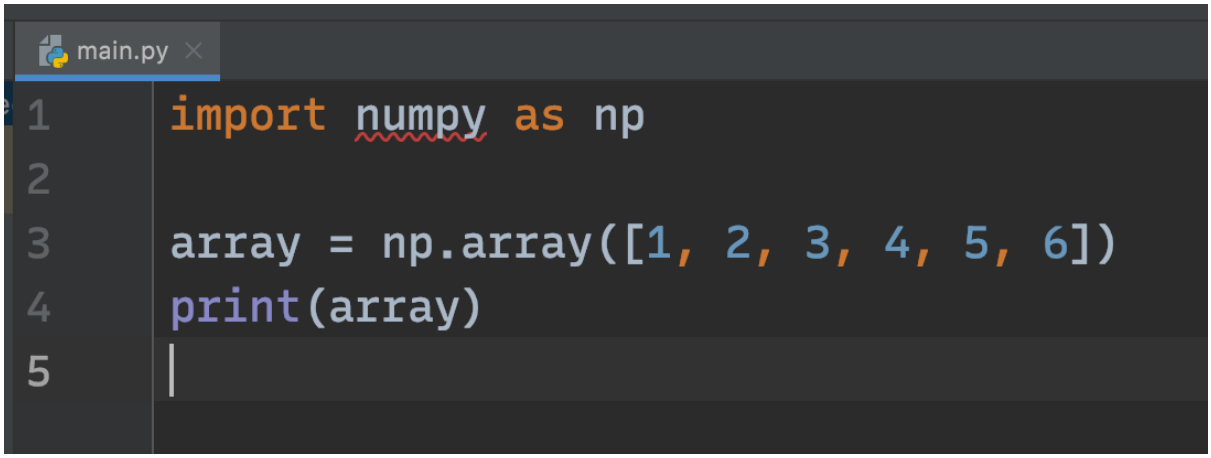
그 다음에 위에서 봤던 Preference —> Project: projectName —> Python Interpreter에서 사용하는 인터프리터를 project_env_2 가상 환경의 인터프리터로 바꿔줍니다.



그리고 하단에 ok를 클릭하면 프로젝트 내에서 사용하는 가상 환경이 바뀝니다. 터미널 창을 켜봐도 새로운 가상 환경이 적용된 걸 볼 수 있고요.



이 환경에서는 numpy 패키지를 설치한 적이 없기 때문에 더 이상 가지고 올 수 없는 것도 확인할 수 있습니다. 프로젝트에서 다시 numpy를 사용하고 싶으면 새롭게 다운받아야 되죠.



마지막으로 필수는 아니지만 더 이상 전에 사용하던 project_env 가상 환경을 사용하지 않을 거기 때문에 지워주시는 게 좋습니다.