Virtual Environments in Python with Venv

How does it work?

REQUIRED TOOL

```
venv (built-in in Python 3.x)
```

CREATE A VIRTUAL ENVIRONMENT

Open the terminal/command line and navigate your project directory
Create a Virtual Environment (replace “myenv” with your preferred name)

```
python3 -m venv myenv
```

ACTIVATE YOUR VIRTUAL ENVIRONMENT

Windows

```
myenv\Scripts\activate
```

macOS or Linux

```
source myenv/bin/activate
```

INSTALL REQUIRED LIBRARIES WITH PIP

```
pip install package-name
```

If you need to install a specific package version, run for example:

```
pip install numpy==1.21.2
```

Continue working on your project.

SAVE YOUR PROJECT DEPENDENCIES

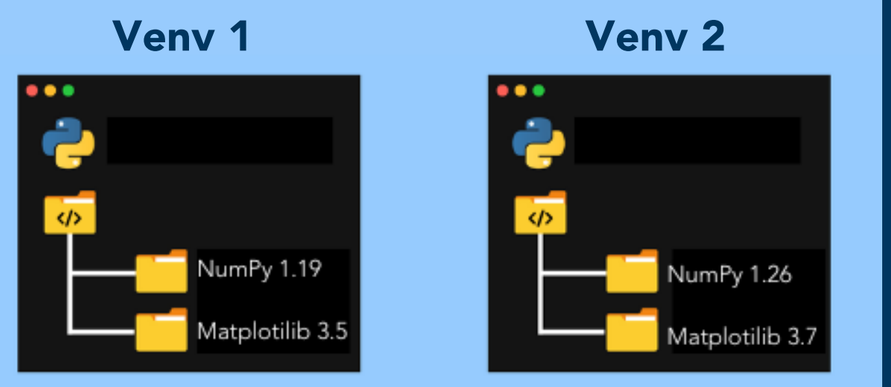
```
pip freeze > requirements.txt
```

EXIT THE VIRTUAL ENVIRONMENT

Once you are ready to move to another project, make sure to exit the virtual environment. This action returns you to the system's default Python environment.

```
deactivate
```

Why you should use it?

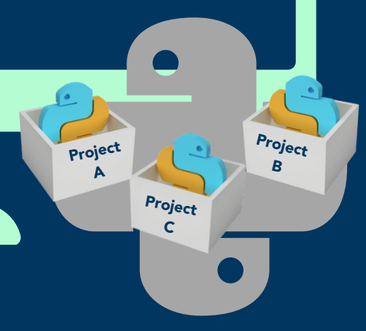


Venv operates independently, ensuring alterations to installed dependencies within one environment remain isolated from others and system-wide libraries. This isolation allows the creation of multiple virtual environments, each hosting its own Python versions and varying sets of libraries.

Sharing & Reproducing Virtual Environments

When sharing your projects, ensure to include the 'requirements.txt' file. This file enables others to effortlessly install all necessary dependencies by running the following command in their terminal or command prompt:

```
pip install -r requirements.txt
```



rds@library.ucsb.edu

www.library.ucsb.edu

UC SANTA BARBARA
Library