Programming for Data Science

Python Virtual Environments

Say OL

say_ol@yahoo.com

September 26, 2024

What is a Virtual Environment?

- A virtual environment is an isolated environment that allows you to manage dependencies for different projects.
- Each environment can have its own Python interpreter and packages.
- Helps avoid conflicts between package versions across projects.

Why Use Virtual Environments? I

- **Isolation of Dependencies:** Each project can have its own set of dependencies. A virtual environment prevents conflicts between package versions required by different projects.
- **Reproducibility:** Virtual environments allow you to recreate the same environment easily on different machines or at different times. This is crucial for collaborative work and maintaining consistent results.
- **Clean Global Environment:** Using virtual environments helps keep your global Python installation clean, avoiding issues with package installations or upgrades affecting unrelated projects.
- **Version Management:** You can have different versions of Python and packages for different projects without conflicts. This is especially important for data science, where different analyses might depend on specific versions of libraries.

Why Use Virtual Environments? II

- **Ease of Experimentation:** You can experiment with new packages or versions in a virtual environment without risking the stability of your main projects or system.
- **Deployment and Testing:** Virtual environments are useful for preparing projects for deployment. You can ensure that the production environment mirrors your development environment.
- **Documentation:** Virtual environments often come with a `requirements.txt` or similar files that document the packages and versions used, making it easier to share and manage dependencies.

How to Create Python Virtual Environment

- `venv`: A built-in module in Python 3.3 and later that allows you to create lightweight virtual environments.
- `virtualenv`: A third-party tool that provides more features and supports Python 2 and 3.
- `conda`: A package manager from Anaconda that also manages environments and is particularly popular in data science.
- Since we are using the latest version of Python, only creating Python virtual environments using the `venv` and conda will be discussed.

Creating a Virtual Environment (venv)

- Use the built-in ‘venv‘ module (Python 3.3+).
- Command to create a new virtual environment:

Command

```
python -m venv myenv
```

- **Windows:** Make sure that Python have been installed and the following paths have been added to the variable environment:

Command

```
C:\Users\<UserName>\AppData\Local\Programs\Python\Python312\Scripts
```

- and

Command

```
C:\Users\<UserName>\AppData\Local\Programs\Python\Python312
```

Activating a Virtual Environment (venv)

- **Windows:**

Command

```
myenv\Scripts\activate
```

- **macOS/Linux:**

Command

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

Installing Packages (venv)

- Use `pip` to install packages within the activated environment.

Command

```
pip install package_name
```

- To list installed packages:

Command

```
pip list
```

Deactivating a Virtual Environment (venv)

- To exit the virtual environment, simply run:

Command

```
deactivate
```

Using `requirements.txt`

- Keep track of dependencies using a `requirements.txt` file.
- Create the file:

Command

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

- Install dependencies from the file:

Command

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

Using Anaconda for Virtual Environments

- Anaconda is a popular distribution for data science.
- It provides an easy way to create and manage virtual environments with the ‘conda‘ package manager.
- Ideal for managing complex dependencies in data science projects.

Creating a Virtual Environment (conda)

- **Windows:** Make sure that you have already added the following paths to your ‘variable environment’

Command

```
C:\Users\<YourUsername>\Anaconda3\Scripts
```

- and

Command

```
C:\Users\<YourUsername>\Anaconda3
```

Creating a Virtual Environment (conda)

- Use the following command to create a new environment:

Command

```
conda create −name myenv
```

- You can specify the Python version:

Command

```
conda create −name myenv python=3.12.4
```

- We can list all Anaconda virtual environments by:

Command

```
conda env list
```

Activating a Virtual Environment (conda)

- To activate the created environment, use:

Command

```
conda activate myenv
```

- If it is your first time to use the Anaconda, you have to initialize it before activating the virtual environment:

Command (first time only)

```
conda init
```

Installing Packages (conda)

- Use `conda install` to install packages within the activated environment.

Command

```
conda install package_name
```

- To list installed packages:

Command

```
conda list
```

Exporting and Importing Environments (conda)

- Export the environment to a YAML file:

Command

```
conda env export > environment.yml
```

- Create an environment from a YAML file:

Command

```
conda env create -f environment.yml
```

Conclusion

- Virtual environments are essential for managing project dependencies.
- Both `venv` and `conda` provide effective solutions.
- Encourage students to choose the best tool for their needs!