3b-Local.Basics.and.Package.Management

November 2, 2024

0.1 Local vs. Cloud

Local * Control * Customization * Git - Version Control * More files open * Persistent storage - no weird workarounds with connecting Google Drive

Colab * Easy to get started on * Preinstalled packages - could be good or bad * Google Drive integration * No installation needed * Consistent no matter what your OS is * Google's computing power

Other cloud options * There are other options for running in the cloud * Alternatives similar to Colab * Configure your own cloud-computing environment * Docker containers

0.2 Local Package management

0.2.1 General Ideas

- Create environments to install Python packages
  - Install only packages needed for a specific project
  - Control versions of packages (may need different versions for different projects)
  - Share project requirements with others
- General process
  - Setup:
  - Create a new environment
  - Activate the environment for your project
  - Install packages
  - Use
    - Activate environment
    - Code cool things
    - Deactivate
  - Share
    - Make a requirements file
    - Make an environment from a requirements file

  Package management options
  - conda/Anaconda
  - pip, venv, pipenv

Conda vs. pip vs. virtualenv commands

0.3 Conda - Anaconda

conda is a package manager
Anaconda is an installation that includes:
- conda - many other packages commonly used for Data Science - A GUI interface

0.3.1 Anaconda Navigator - GUI

- Download and Install

- Windows and Mac: launch Anaconda Navigator
- Linux- from command line: `conda activate` then `anaconda-navigator`
- Selecting 'Environments' from menu on left lets you create & manage environments
- From 'Home' you can pick an environment and then launch Jupyter Lab, or another program.

0.3.2 Conda - command line

- Install
- Documentation

Create and setup new environment `conda create --name $ENVIRONMENT_NAME python`

```
conda activate $ENVIRONMENT_NAME
```

```
conda install $PACKAGE_NAME
```

```
conda deactivate
```

Use a Conda environment `conda activate $ENVIRONMENT_NAME`

Do your work (for example run jupyter lab and work on a notebook)

```
jupyter lab
```

```
conda deactivate
```

Export/Reuse environment Sharing a Conda Environment Documentation

Activate environment

```
conda activate $ENVIRONMENT_NAME
```

Export requirements:

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

Create new environment using requirements

```
conda create --name $NEW_ENVIRONMENT_NAME --file environment.yml
```

0.4 Python venv

0.4.1 Initial setup

If using Debian or Debian derivative ...

```
export DEBIAN_FRONTEND=noninteractive
sudo apt-get update
sudo apt-get install -y python3-pip python3-venv tree
```

```
cd path/to/your/projects
```

```
python3 -m venv project_name
```

```
tree -L 2 project_name/
```

0.4.2 Activate venv

In your projects folder …

```
source project_name/bin/activate
```

0.4.3 Installing Packages

While the virtual environment is active, you can install packages using pip and pip will install packages within the folder hierarchy specific to the virtual environment.

```
pip install package_name
```

0.4.4 Listing installed python packages

```
pip freeze | tee requirements.txt
```

0.4.5 Installing packages from a requirements.txt file

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

0.4.6 Deactivating the Virtual Environment

```
deactivate
```

0.5 References

- Corey Schafer – Python Tutorial: VENV (Windows) - How to Use Virtual Environments with the Built-In venv Module