Virtual Environment

The use of a Virtual Environment is to test python code in encapsulated environments, and to also avoid filling the base Python installation with libraries we might use for only one project.

virtualenv

1. Install virtualenv

   ```
   pip install virtualenv
   ```

2. Install virtualenvwrapper-win (Windows)

   ```
   pip install virtualenvwrapper-win
   ```

Usage:

1. Make a Virtual Environment named `HelloWold`

   ```
   mkvirtualenv HelloWold
   ```

   Anything we install now will be specific to this project. And available to the projects we connect to this environment.

2. Set Project Directory

   To bind our virtualenv with our current working directory we simply enter:

   ```
   setprojectdir .
   ```

3. Deactivate

   To move onto something else in the command line type `deactivate` to deactivate your environment.

   ```
   deactivate
   ```

   Notice how the parenthesis disappear.

4. Workon

   Open up the command prompt and type `workon HelloWold` to activate the environment and move into your root project folder

   ```
   workon HelloWold
   ```

Poetry

1. Install Poetry

   ```
   pip install --user poetry
   ```

2. Create a new project

```
poetry new my-project
```

This will create a my-project directory:

```
my-project
├── pyproject.toml
├── README.rst
├── poetry_demo
│   └── __init__.py
└── tests
    ├── __init__.py
    └── test_poetry_demo.py
```

The pyproject.toml file will orchestrate your project and its dependencies:

```
[tool.poetry]
name = "my-project"
version = "0.1.0"
description = ""
authors = ["your name <your@mail.com>"]

[tool.poetry.dependencies]
python = "*"

[tool.poetry.dev-dependencies]
pytest = "^3.4"
```

3. Packages

To add dependencies to your project, you can specify them in the tool.poetry.dependencies section:

```
[tool.poetry.dependencies]
pendulum = "^1.4"
```

Also, instead of modifying the pyproject.toml file by hand, you can use the add command and it will automatically find a suitable version constraint.

```
$ poetry add pendulum
```

To install the dependencies listed in the pyproject.toml:

```
poetry install
```

To remove dependencies:

```
poetry remove pendulum
```

For more information, check the documentation or read here:

Pipenv

1. Install pipenv

```
pip install pipenv
```

2. Enter your Project directory and install the Packages for your project

```
cd my_project
pipenv install <package>
```

Pipenv will install your package and create a Pipfile for you in your project's directory. The Pipfile is used to track which dependencies your project needs in case you need to re-install them.

3. Uninstall Packages

```
pipenv uninstall <package>
```

4. Activate the Virtual Environment associated with your Python project

```
pipenv shell
```

5. Exit the Virtual Environment

```
exit
```

Find more information and a video in docs.pipenv.org.

Anaconda

Usage:

1. Make a Virtual Environment

```
conda create -n HelloWorld
```

2. To use the Virtual Environment, activate it by:

```
conda activate HelloWorld
```

Anything installed now will be specific to the project HelloWorld

3. Exit the Virtual Environment

```
conda deactivate
```