The `pipenv` Package

Pipenv provides an alternative command-line utility for installing third-party Python packages and managing Python versions and package dependencies.

Reference:

- https://pipenv-fork.readthedocs.io/en/latest/
- https://pipenv-fork.readthedocs.io/en/latest/install.html#installing-pipenv
- http://docs.python-guide.org/en/latest/dev/virtualenvs/#virtualenvironments-ref <-- an awesome guide to help you get started
- http://docs.python-guide.org/en/latest/starting/install3/win/
- https://github.com/pypa/pipenv

Installation

When installing Pipenv, one option is to install it via Pip, almost like you would any other Python package:

```
pip install --user pipenv
```

However, for Mac users, you can alternatively install Pipenv via Homebrew (recommended):

```
brew install pipenv
```

Usage

Setup

After installing Pipenv, you will mostly be using it from the root directory of some project repository to manage packages and versions. So navigate to your project directory:

```
cd path/to/my-project-repo/ # where path/to/my-project-repo/ is the actual path to your desired project directory
```

From your project's root directory, install a new virtual environment:

```
pipenv install
# or, for a specific version:
pipenv install --python 3.7
```

This should create two files in the root directory of your project repository: a `Pipfile` and a `Pipfile.lock`.

> NOTE: for some Windows users, you might not see these files, but the virtual environment is still created.

Installing Project-specific Packages

To install a specific package:

```
pipenv install my_package # where my_package is the name of the package to
install
```

This will add the package to the project's `Pipfile` and `Pipfile.lock`, and make it available for use by scripts run within the project's virtual environment.

Running a Virtual Environment

From your repository's root directory, enter into a virtual environment with all the specified packages installed and ready to use:

```
pipenv shell
```

From within the virtual environment, you should be able to examine its Python installation and execute scripts as usual:

```
--->> pipenv shell
# Loading .env environment variables…
# Spawning environment shell (/bin/bash). Use 'exit' to leave.
# bash-3.2$ . /Users/mjr/.local/share/virtualenvs/my-project-repo
# -app-JQd9Gsl3/bin/activate
(my-stocks-app-JQd9Gsl3) bash-3.2$ which python
#> /Users/mjr/.local/share/virtualenvs/my-project-repo-JQd9Gsl3/bin/python
(my-stocks-app-JQd9Gsl3) bash-3.2$ python --version
#> Python 3.6.5
(my-stocks-app-JQd9Gsl3) bash-3.2$ python my_script.py
#> (some output from my_script.py)
```

When you are done, type `exit` and press enter to leave the virtual environment and return to your normal command-line experience.