Setup Walkthrough

The purpose of this notebook is simply to outline some of the miscellaneous steps and best-practices when starting and maintaining an analytics project.

Virtual Environments

Virtual environments are contained instances that house the static libraries necessary to run a project. These are useful for maintaining version control of libraries that may deprecate certain features your project relies on. This is standard practice to create one for each project you work.

Jupyter notebooks/labs

If you're simply doing a project with some jupyter notebooks, you'll have to make your own virtual environment. Do so by entering the following into the command line:

```
python3 -m venv /path/to/new/virtual/environment/venv
```

Once setup, you should see a folder called "venv" in your project folder. Activate the venv, cd into your directory and enter:

```
venv\Scripts\activate
```

The command prompt should show `(venv)` before your current directory now. Now any pack you install with pip will be locally install and maintained.

To use a virtual environment in conjunction with Jupyter notebooks or Jupyter lab, you'll need to install ipykernal to the environment by:

```
pip install ipykernel
```

Then you'll need add the venv to jupyter lab by using the following:

```
python -m ipykernel install --user --name=myenv
```

This will return the following when done sucessfully:

```
Installed kernelspec myenv in /home/user/.local/share/jupyter/kernels/myenv
```

If and when you'd like to remove a venv from Jupyter labs, you can do so by the following:
`jupyter kernelspec uninstall myenv`

IDE's

If you're working in a full IDE like PyCharm or VSCode, the IDE will take care of this step for you when create the project.

Git and Github

As a crash course on Git and Github, Git is a version control system that is practically ubiquitous in software development. Git just synchronizes changes made to code when multiple individuals are working on the same project. Github comes in to house the projects (called repositories or repo's) online. These two technologies are the standard practice to house code and share code between systems. The following information will assume you have Git and Github already setup.

Starting a Project

To create a project, login onto Github and create a new repository. To link your new repo to your local project, cd into your project's directory and input the following commands in to command line:

```
 git init
git remote add origin https://github.com/your_username_here/repo_name.git
```

It's best practice to have README.md with every project. This serves as a summary of your poject, outlining what the project seeks to accomplish and what tech you will need to actually run the thing. You can create one by simply creating a .txt file and renaming the extension. As for formatting, see below for the Markdown overview.

Adding Files and Making Changes

Once you have made changes to a file or have created a new file, you can the push these changes to repo you've created. Started by adding all necessary files by using following command (Repeat as necessary):

```
 git add file.extension
```

Once you've added all the necessary

```
 git commit -m "description of what changes you've made"
git branch -M main
git push -u origin main
```

Updating the local Repository

To sychronize the files you have locally with what is in the repo, simply pass

```
 git pull
```

Markdown

The purpose of markdown is to format text files. This is relevant in the formatting you're seeing in this notebook of non-code cells as well as README files.