Using **Pipenv** as environment manager

For more detailed infos visit https://pipenv-fork.readthedocs.io/en/latest/basics.html

INSTALL PIPENV
pip install pipenv

CREATING AN ENVIRONMENT
1. Open the folder for your project in your terminal
   - Navigate to your project folder using "cd path/to/folder"
   - Some apps (like the Windows Terminal) allow you to right-click on the desired folder and select "Open in Terminal" which automatically opens a new terminal directly on that folder path
2. run: pipenv install
   - If you already have a Pipfile in that folder, it will install all dependencies automatically.
   - If you only have a requirements.txt file available when running pipenv install, pipenv will automatically import the contents of this file and create a Pipfile for you. You can also specify $ pipenv install -r path/to/requirements.txt to import a requirements file from another folder.
   - If you don't have a Pipfile or a requirements.txt file, pipenv will automatically create those for you.

ACTIVATE ENVIRONMENT
cd to folder and then: **pipenv shell**

A simple and easy workflow to start working on an existing environment

Without activating the environment on your terminal:	*By first activating the environment:*
1. In your terminal, navigate to your folder 2. Simply enter pipenv run <IDE command> • pipenv run code . (for VSCode) • pipenv run jupyter notebook	1. In your terminal, navigate to your folder 2. Run pipenv shell 3. Run a command to open your desired IDE • code . (for VSCode) • jupyter notebook

SEE AND DELETE AVAILABLE ENVIRONMENTS
- All your environments are saved **in your user directory under the folder .virtualenvs**
- You can delete any environment by simply deleting the respective folders there.

INSTALL PACKAGES
First activate the environment

With saving changes to Pipfile	*Without saving changes to Pipfile*	*Install only to [dev-packages]*
pipenv install pandas	**pip install** pandas	pipenv install **--dev** pandas

If you want to force only a single specific version to be installed, use: pipenv install pandas**==1.0** (or any version)

INSTALLING FROM PIPFILE.LOCK
Pipfile.lock saves the **specific versions of packages** that were installed in the user's environment.

pipenv lock	This will create/update your Pipfile.lock with the currently installed package versions
pipenv install --ignore-pipfile	This tells Pipenv to ignore the Pipfile for installation and use what's in the Pipfile.lock. Given this Pipfile.lock, Pipenv will create the exact same environment you had when you ran pipenv lock, sub-dependencies and all.

For instance, if the pandas version in the Pipfile is pandas = "*", pip will install the latest available pandas version when creating the environment. If you use the commands above, instead it will install the exact pandas version the user had.

INSTALL FROM REQUIREMENTS
pipenv install -r requirements.txt
pipenv install --dev -r requirements_dev.txt

CREATE A REQUIREMENTS.TXT
pipenv lock
pipenv requirements > requirements.txt
pipenv requirements --dev-only > requirements_dev.txt