Steps to Downloading Environment (follow default options for every software):

1. Download Python (Interpreter)
2. Download Git Interface
3. Download VS Code
4. Download VS Code Extensions (Python + Python Debugger) (Optional: Dracula Theme, Other Color themes, Install or Disable at any time)
5. (Optional) Change Background Settings for VS Code
    a. File -> Autosave
    b. File->New File->Python File
    c. File->Save->Select a folder to save to (or create a new folder);
6. Open Git BASH (select 'Launch Git Bash' on windows or simply terminal on MacOS)
7. Enter Folder to create Git repository
8. 'cd' to move forward, 'cd ..' to move backwards to folders. We cannot do this with files: 'cd [name folder]'. Video of navigating folders here
9. Once in the desired folder, use 'git clone https://github.com/jacksongong/nhs-ai-ml-club' to copy code
10. If you have multiple IDEs, right click on desired file, and click 'Open with VS Code'

11. **You will now have a created file that you can write on**
12. **Each time you want to copy changes from the shared Github, you can type ‘git pull’ in the folder that you created git repo in- the name will be ‘nhs-ai-ml-club’**

Creating Virtual Environment (on windows; linux is much easier) (Optional)

```
python -m venv [name]
.\nhs\Scripts\activate.bat
deactivate (to remove
```

Git Interface

The git command line (which is built into the Linux system) acts as a navigation to your files folder (Allows us to access code from other users and work together)

1. **Clone desired repository from GitHub:**
   ```
   git clone [url]
   ```
   This command is used to copy an existing Git repository from a remote server, typically GitHub, into a new directory on your local machine.
2. **Check the status of your repository:**
   ```
   git status
   ```
   Shows the current status of your working directory and staging area, allowing you to see which changes have been staged, which haven't, and which files aren't being tracked by Git.

3. **Add files to the staging area:**
   `git add [file]` or `git add .`
   Adds one or more files to the staging area. Using `git add .` adds all new and changed files to the staging area, preparing them for a commit.
4. **Commit changes to your local repository:**
   `git commit -m "[commit message]"`
   Commits your staged content as a new commit snapshot to your local repository, along with a brief description of the changes provided in the commit message.
5. **Pull updates from remote repository:**
   `git pull [alias] [branch]`
   Fetches and merges changes from the remote repository to your local repository. This is used to keep your local repository up-to-date with others' changes.
6. **Push changes to a remote repository:**
   `git push [alias] [branch]`
   Sends your commits from your local repository up to your remote repository on GitHub. It's commonly used to share your changes with others.
7. **List branches: (we will use github as a foundation for code, not as a collaboration, so this will not be as relevant)**
   `git branch`
   Lists all the branches in your repository. You can also create or delete branches using this command.
8. **git reset <file name>** will remove a file from the staging area
9. Removing Committed Files: **git reset --soft HEAD^1**
10. Removing Pushed Files to Github: **git rm --cached name_of_file**