On-boarding tech stack for Python development.

The following is a highly suggested tools you need to install before the start of our workshop. The intent of this list is to keep a consistent development environment that we can grown with.

If you are using a Mac, most of the tools are pre-installed. The following steps are to help Windows users to reach feature parity with the Mac environment on a clean computer.

- Git (Windows)
  - https://git-scm.com/download/win
    - If your OS is 64-bit then download the 64-bit version of the setup.
  - Remember to select "Use Git and optional Unix tools from the Command Prompt", under the "Adjusting your PATH environment" dialog.
    - You will have access to most of the Unix utilities in your Command Prompt.

- Install Python (Mac or Windows)
  - https://www.python.org/downloads/
  - Minimum version to use is 3.7.6
  - If you have a previous Python installed from a different distribution, you may encounter issue. Check with other team members to resolve this.

- Install Python the virtual environment, `pipenv` (Mac or Windows).
  - From your command prompt type the following:
    ```
    pip install pipenv
    ```
  - If you are prompted to upgrade your pip version, go ahead and do it.

- Install VS Code (Mac or Windows)
  - https://code.visualstudio.com/download
  - Once installed, launch VS Code and **click** on the extensions icon (4 little squares) on the the left to **search** and **install** the extensions. Type in just first few character of the below extensions to search and click install.
    - **Python** by Microsoft
    - **Live Share** by Microsoft
    - **SQL Server** by Microsoft
    - **SQLite Explorer** by AlexCvzz
    - **Bracket Pair Colorizer** by Coenraads
    - **Excel Viewer** by GrapeCity
    - **Json Editor** by Nick DeMayo
    - **REST Client** by Huachao Mao
    - **GitLens** by Eric Amodio

- Slack (Mac or Windows)
  - https://slack.com/downloads/windows (My browser keeps pointing me to the windows download)
  - The workspace is: **cstuworkspace** .slack Get your login invite from Glen Qin.
  - This shall be the centralized communication platform. I suggets install it on your phone.

- Install C/C++ compiler (Windows)
  - https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=BuildTools&rel=16 (should auto download)
    - Need it to compile the **`ujson`** package.
  - https://www.itechtics.com/microsoft-visual-c-redistributable-versions-direct-download-links/#Microsoft_Visual_C_Redistributable_2019

If you want to get deeper understanding beyond this document, you can always get from here: http://letmegooglethat.com/?q=Google+is+your+friend+..+do+abuse+it+with+your+questions

Version control with Git

Once you have installed the above tools, you are ready to work with the code in my personal repository. All the following shall be performed from the command line terminal. The following are done once.

Create a sub-directory on your computer that will be easy for you to find your work area. On my computer, I have create my master folder ws follows:

```
mkdir \DataRoot\Projects\GitHub
cd    \DataRoot\Projects\GitHub
```

Configure your commit profile.

```
git  config  --global user.name  "First Last"
git  config  --global user.email "email@example.com"
```

Clone my repo down to you computer.

```
git  clone  https://github.com/elau1004/ETLite.git
```

Change into the newly cloned sub-directory and install all the dependent libraries for the cloned project.

```
cd    \DataRoot\Projects\GitHub\ETLite
pipenv sync
```

The following are the basic commands that you will use on a regular basis.

Create your working branch. Never work on the **master** branch!

```
git checkout -b "branch name" #FirstName-Task as branch name.
```

Display the branches on your local repository.

```
git branch
```

Switch between working branches.

```
git checkout master
git checkout "branch name"
```

Get the status of the current branch.

```
git status
```

Pull the latest code down from the remote repository. **Do this as often as you can!**

```
git checkout master
git pull
```

Merge the code from a different branch onto your current branch. Do this beginning of day, especially after you have done "git pull".

```
git checkout "branch name"
git merge master
```

Review differences between current and original file.

```
git diff "path to your file"
```

Stage your changes for commit into your local repo. Don't use wildcard or entire folder!

```
git add "path to your file"
```

Commit your changes. Do provide a good description!

```
git commit -m "Be descriptive"
```

Push your local repo back up to the remote repo.

```
git push
```

When things are working and you want to contribute to the project, request a pull of your contribution branch back into the master branch. You perform this at the Git website:

https://github.com/elau1004/ETLite/pulls

Your submission shall be reviewed. Do NOT be offended if your pull request is rejected.

Review Process

Dos:

1. Look for Pythonic improvement.
2. Look for design flaws.
    3. Security
    4. Maintainability
    5. Performance
    6. Funtionality
7. Review within 24 hours of submission.
8. Slack the person(s) who you want your PR to be reviewed by.
9. Initial discussions shall be private and direct between reviewer and submitter.
10. Comment in the PR, if only your want to escalate the issue up a level for the community to discuss.
11. Understand the context of the review and **not** just the code differences between revisions.

Donts:

- Nitpick the style and spelling. PEP-8 is only a starting point. I have loosen pep-8 with the following over-writes:
    - Max line length is around 160 but discouraged.
    - Spaces are encouraged if they improve readability.
        - Don't munge/butt your characters too tightly.
        - Vertical alignment is allowed.
        - Space before comma is allowed.
- Let a review hang for more than 48 hours. Do follow up with reviewer.
- Hack together a solution. There need to be balance between quality and delivering and err on the side of quality.