Development Setup

Before developing or modifying the code, we must get the code from the GitHub remote repository, set up the virtual environment, and know how to run the Python Script. This document will teach you how to perform all of these actions.

Using Git

Git is a software for distributed version control that keeps track of changes to files. If Git is not on the system, please use this link: https://git-scm.com/downloads Once installed, we are now able to use Git to push to GitHub, an online repository that will store files. When using Git, there are 3 stages for each file:

Untracked - Where changes are not yet saved or added to the staging area

Staged - Changes are added to the staging area and are read to be commited

Committed - Charges are now committed and is ready to be pushed to the remote repository

Git contains multiple important commands, the primary ones are:

- Git add
- Git commit
- Git push
- Git pull
- Git clone
- Git status
- Git init

Git Add

Git add allows untracked files to be put in the staging area. When added, files added can be committed in the next command.
Note: When added, data and changes are not yet saved.

An example of this code would be:

```
git add {filename}  # Adds the specific file named

git add .           # Adds all files to the staging area
```

Git Commit

Git commit allows changes from the staging area to be in the committed area. Once committed, data is now prepared be sent to the remote repository to be saved. When using git commit, a message must always be attached to the commit, else an error will occur. To add a message, us the `-m` arugment and then insert your

message between quotation marks. When making commits, it is important to be specific to what changes you made, just in case you need to find a specific version, the commit message can help ease that process of looking back.

An example of this code will be:

```
git commit -m "I am a very detailed message"
```

Git Push

Git push will push committed data to the remote repository. When pushing data, make sure you push to the correct repository, but also the correct branch. Branches are different snapshots that allows changes to be made to be separated until they need to be merged together. Usually, each deleoper or each feature is given a separate branch. The `main` or `master` branch should not be directly pushed into. When pushing, we use 2 arguments, `origin`, which specifies the location of where to push the data, and `{branch}`, which will be the branch to push this data to.

An example of a git push would be:

```
git push origin main         # Will push these changes to the main branch

git push origin my_branch    # Will push to a unique branch
```

Git Pull

Git pull will pull new changes from an existing repository that is already initialized in the folder. If the folder has not been linked or initialized with the remote repository, you must git clone an existing repository or initialize a new repository. Git pull allows the system to have the current and newest data in the repository.
An example of git pull would be:

```
git pull origin main        # Will pull the newest data from the main branch

git pull origin my_branch   # Will pull the newest data from my_branch
```

Git Clone

Git clone allows a remote repository to be cloned and initialized onto the local system. When working with a new repository, we must clone it. This links the local folder with the remote repository.
An example of git clone would be:

```

```

```
    git clone https://github.com/PenPen7531/ISSP-Triumf-435/tree/main   #This
command will clone all the data from the main branch of the current repository
```

Git init

Git init will initialize the repository you are currently in to allow connections to the remote repository. You should run git init when creating a new remote repository.
An example of git init would be:

```
    git init
```

GitHub

GitHub is an online repository that allows files and changes to be kept track of. In connection with Git, we are able to use Git commands to push our data to a repository on GitHub.

GitHub Connection

When working with GitHub, we must connect our local system with our account on GitHub. This helps us keep our data secure and allows an ecrypted SSH connection between our system and our respositories. Generally, GitHub will use an RSA public and private key to ensure that the connection is secure. We save one of these keys on our local system, and this allows an ecrypted path and connection. Please follow this documentation to perform this action:

Click Here

Once the connection is secure, GitHub will allow you to clone and push new repositories from your local computer to your remote repository.