**Lab 15: GitHub version Control**

# INTRO TO GITHUB FOR VERSION CONTROL

### KEEPING TRACK OF YOUR CODE AND ITS MANY VERSIONS

GitHub version control allows for easy collaboration and tracking of changes within a coding project. HTML format is a useful tool for presenting information on the web in a user-friendly and visually appealing way. By using GitHub version control and HTML format together, developers can efficiently manage and showcase their code in an organized and accessible manner.

# 1. Get familiar with version control, Git and GitHub

## What is version control?

Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. You are probably already doing some type of version control, if you save multiple files, such as Dissertation\_script\_25thFeb.R, Dissertation\_script\_26thFeb.R, etc. This approach will leave you with tens or hundreds of similar files, making it rather cumbersome to directly compare different versions, and is not easy to share among collaborators. With version control software such as Git, version control is much smoother and easier to implement. Using an online platform like Github to store your files means that you have an online back up of your work, which is beneficial for both you and your collaborators.

Git uses the command line to perform more advanced actions and we encourage you to look through the extra resources we have added at the end of the tutorial later, to get more comfortable with Git. But until then, here we offer a gentle introduction to syncing RStudio and Github, so you can start using version control in minutes.

## What are the benefits of using version control?

Having a GitHub repo makes it easy for you to keep track of collaborative and personal projects - all files necessary for certain analyses can be held together and people can add in their code, graphs, etc. as the projects develop. Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points. You can review other people’s code, add comments to certain lines or the overall document, and suggest changes. For collaborative projects, GitHub allows you to assign tasks to different users, making it clear who is responsible for which part of the analysis. You can also ask certain users to review your code. For personal projects, version control allows you to keep track of your work and easily navigate among the many versions of the files you create, whilst also maintaining an online backup.

## How to get started

**Please register on the**Github website**.**

On your computer, you need to install Git first. The process will depend on your operating system: please follow the instructions below by clicking the relevant button.

Linux

Windows

macOS

The files you put on GitHub will be public (i.e. everyone can see them & suggest changes, but only the people with access to the repository can directly edit and add/remove files). You can also have private repositories on GitHub, which means that only you can see the files. GitHub now offers free private repositories as standard with up to three collaborators per repository. They also offer a free education package,

## How does version control work?

### What is a repository?

You can think of a repository (aka a repo) as a “main folder”, everything associated with a specific project should be kept in a repo for that project. Repos can have folders within them, or just be separate files.

You will have a local copy (on your computer) and an online copy (on GitHub) of all the files in the repository.

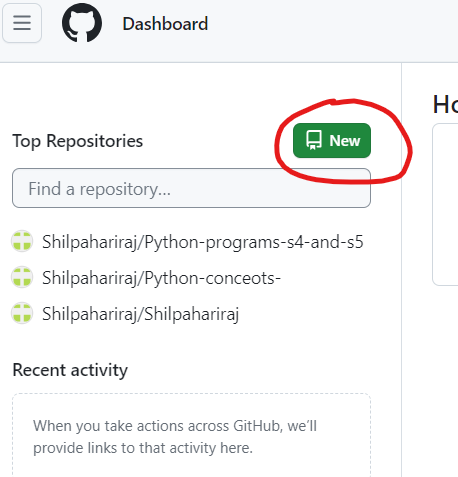
### The workflow

The GitHub workflow can be summarised by the “commit-pull-push” mantra.

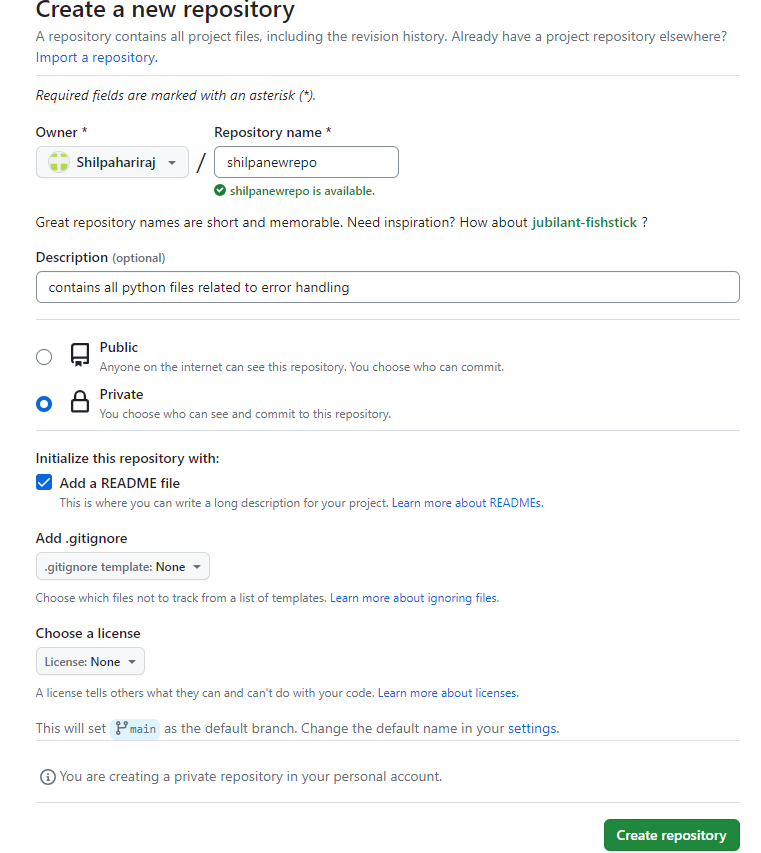
1. Commit
   * Once you’ve saved your files, you need to commit them - this means the changes you have made to files in your repo will be saved as a version of the repo, and your changes are now ready to go up on GitHub (the online copy of the repository).
2. Pull
   * Now, before you send your changes to Github, you need to pull, i.e. make sure you are completely up to date with the latest version of the online version of the files - other people could have been working on them even if you haven’t. You should always pull before you start editing and before you push.
3. Push
   * Once you are up to date, you can push your changes - at this point in time your local copy and the online copy of the files will be the same.

# 2. Create your own repository and project folder structure

To make a repository, go to Repositories/New repository - choose a concise and informative name that has no spaces or funky characters in it. This can be your master repo that holds together past and ongoing research, data, scripts, manuscripts. Later on you might want to have more repositories - e.g. a repository associated with a particular project that you want to make public or a project where you are actively seeking feedback from a wider audience. For now, we will focus on organising and using your main repository that holds the files for all your work. With a free GitHub account, you can use public or private respositories.



Let’s create a new private repository. You can call it whatever you like if the name is available.

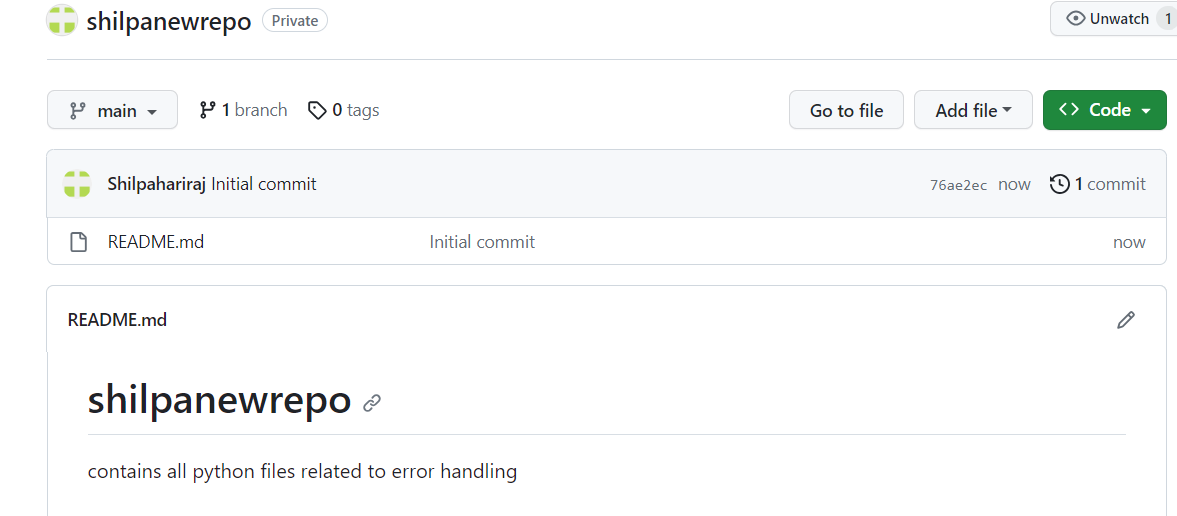


**Click on**Initialise repo with a README.md file. It’s common practice for each repository to have a README.md file, which contains information about the project, the purpose of the repository, as well as any comments on licensing and data sources. Github understands several text formats, including .txt and .md. .md stands for a file written in [Markdown](https://en.wikipedia.org/wiki/Markdown). You might have used Markdown before from within RStudio to create neatly organised reports of your code and its outputs (you can also check out our [Markdown tutorial](https://ourcodingclub.github.io/tutorials/rmarkdown/index.html). You can also use Markdown to write plain text files, for example the file you are reading now was written in Markdown.

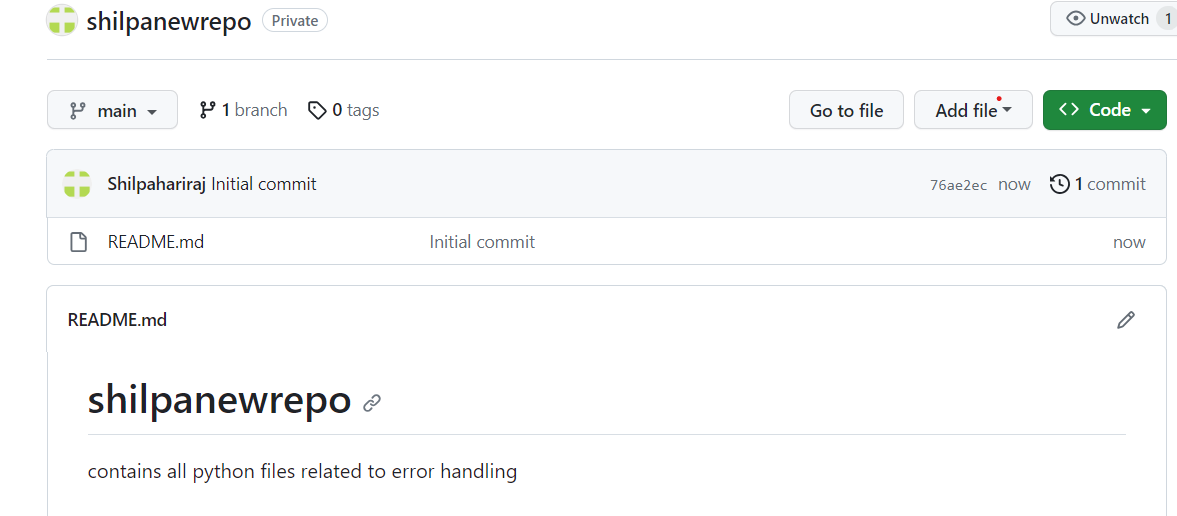
**We will also create a**.gitignore**file**. This file lets Git know what kind of files should not be included in the repository. We’ll have a look at that file in a bit. Tick the box, then search for **R** in the drop-down template (or whatever programming language you will be using for the project).

Once you are ready, click on **Create repository**.

Here is how the repository should look:



You can directly edit your README.md file on Github by clicking on the file and then selecting Edit this file.



## A few GitHub rules:

1.Keep file paths short and sensible.

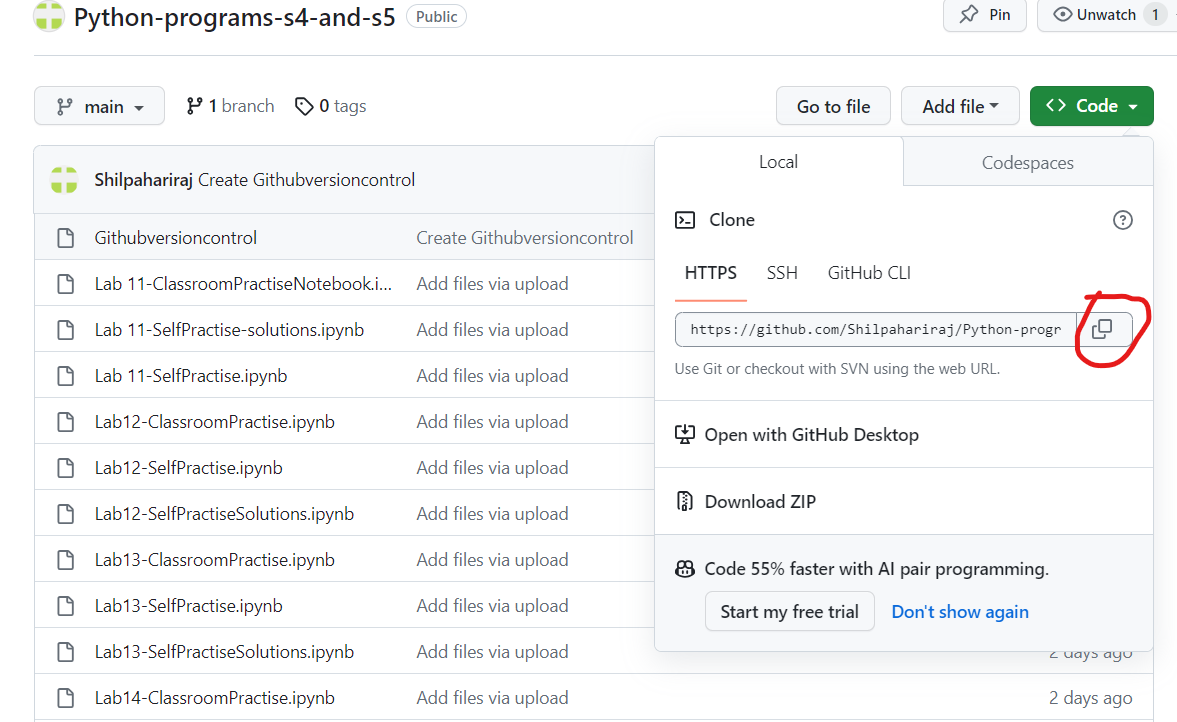
2.Don’t use funky characters and spaces in your file names, these cause trouble because of differences in Mac/Windows systems.

3.Always **pull** before you start working on your project and **before you push** in case someone has done any work since the last time you pulled - you wouldn’t want anyone’s work to get lost or to have to resolve many coding conflicts.

3. Sync and interact with your repository through

**Log into your Github account and navigate to the repository you created earlier**

Click Code and copy the HTTPS link.

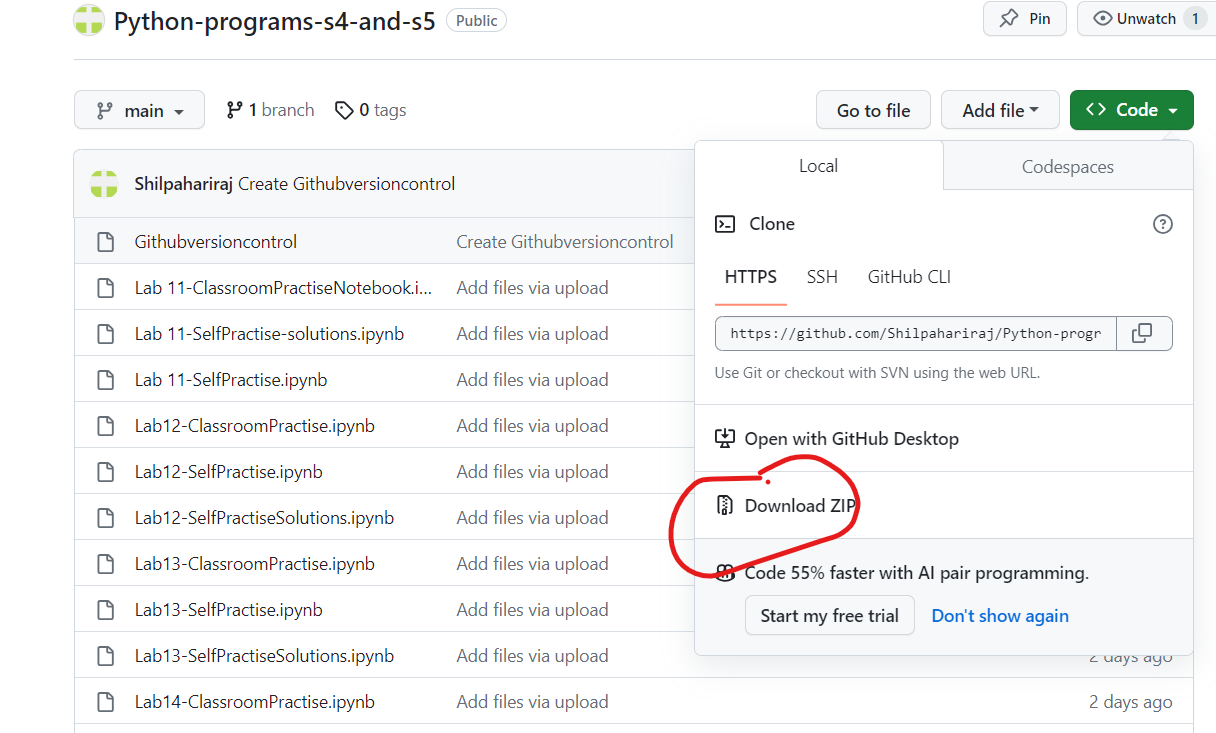


Now open Repository URL: field. Select a folder on your computer - that is where the “local” copy of your repository will be (the online one being on Github).

Sign in via Personal Access Token (All platforms, recommended)

Sign in via internet browser (Windows, fast)

**afterwards you can commit-pull-push at your convenience!**



f you click on the Git tab you will see that now your README.md file is listed there. Add a tick next to it. Now it has an M - this means you have modified the file. If there’s an A, that’s an added file, and a D is a deleted file.

If you select the README.md file and click on Diff, you will see the changes you have made. Once the file is selected, it is staged, ready to be commited to Github.

Click on Commit and add in your commit message - aim to be concise and informative - what did you do? Once you have clicked on Commit, you will get a message about what changes you have made.

You will see a message saying that your branch is now one commit ahead of the origin/main branch - that is the branch that is on Github - we now need to let Github know about the changes we have made.

