Git & GitHub

Coding Collaboration

Paul Scott

Contents

Notes on Git

Why Git?

+ Smoother collaboration on tools
+ Ability to see and track changes
+ Cleaner tool structures
+ Cleaner Version control
+ Integration with VS Code

GitHub is a website that provides an architecture for creating and storing Git repositories. It is the recommended website for collaborating with Git, although others are available.

How does it work?

+ A tool or project is stored in what is known as a repository (equivalent to a folder)
+ The overall working tool is stored in the main 'branch'.
+ A branch is a copy of the repository files and all branches are stored within the repository
+ When a new feature/version is being created, a new branch can be made
+ Changes made in a separate branch do not affect the main branch, allowing edits without breaking the tool
+ Git allows changes made to be tracked, and when happy with the new edits, they can be merged to the main branch
+ Rinse and repeat for new features

Git can be used as an add-on to VS Code, where git commands are run in the terminal

Where should we use Git?

+ Any Python tool, to store them under one organisation
+ Particularly large tools with complex structures
+ Whenever multiple engineers will be coding

Basic Process

+ Tool repository is set up in GitHub
+ Engineer copies ('clones') repository to their local device
+ Engineer creates branch for changes
+ Engineer makes changes/additions
+ Engineer moves branch into online repository for checking
+ When checked, changes are merged onto main branch

Setting Up

Installing Git in VS Code

To install Git packages within your VS Code application

- Navigate to the Source Control tab on the left-hand side
- Follow the instructions given

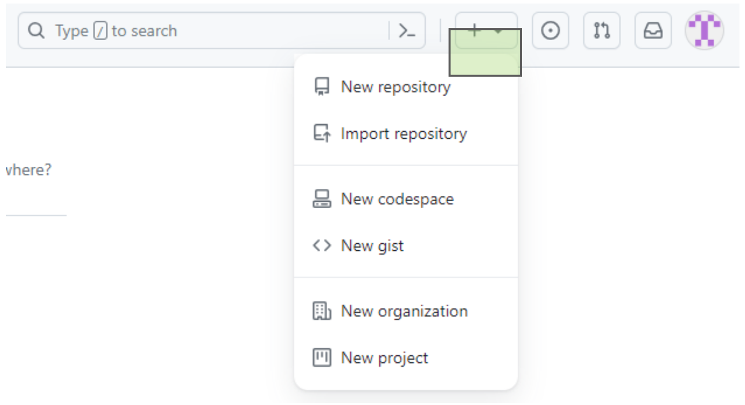
Creating a Repository

To create a repository on GitHub:

- Navigate to the + tab
- Create a new repository
- Follow the instructions given

The following link provides a tutorial on navigating repositories and completing a work-flow in GitHub:

https://docs.github.com/en/get-started/quickstart/hello-world



Creating a new repository in GitHub

Creating a GitHub Account

GitHub requires an account to use its features. Create your own account on GitHub through their website.

Note:

Remember that Git and GitHub are separate. You may wish to use a different website to store and access repositories using Git, such as BitBucket, but GitHub is the most popular and conventional.

Accessing another Repository

To access another user's repository on GitHub:

If Private:

- The user or organisation can invite you to join

If Public

- Search within GitHub or web search to find the repository you'd like to access

Version Control Process

This slide illustrates the process of Python Tool creation.

Tool's will have different working versions as they are developed. It is crucial to ensure the main tool is always working as changes are being made, and that old working versions of tools are stored.

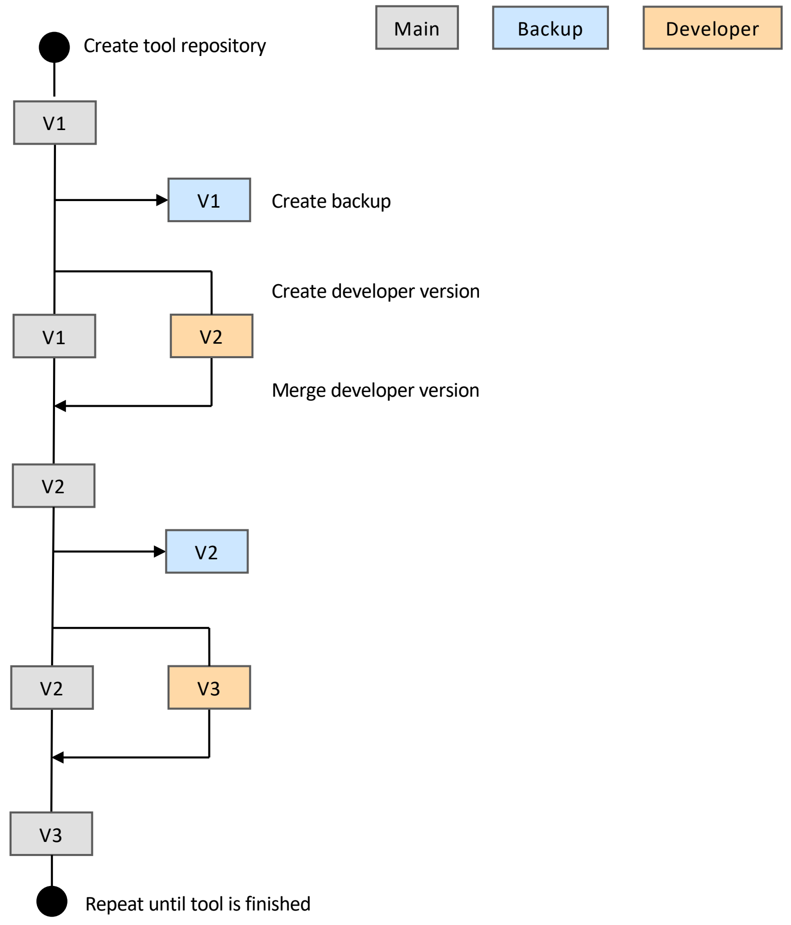
The Process

Create Tool (Version 1)

Make repository and store V1 as main branch

- + Copy main to a back-up branch
- + Copy main to a developer branch (Version 2)
- + Edit developer branch for changes required
- + When happy, merge to main branch
- + Repeat for next version

What it will look like



When should we create branches?

Branches are used to ensure that the current tool is not altered while improving or adding to it.

As such, create a branch when:

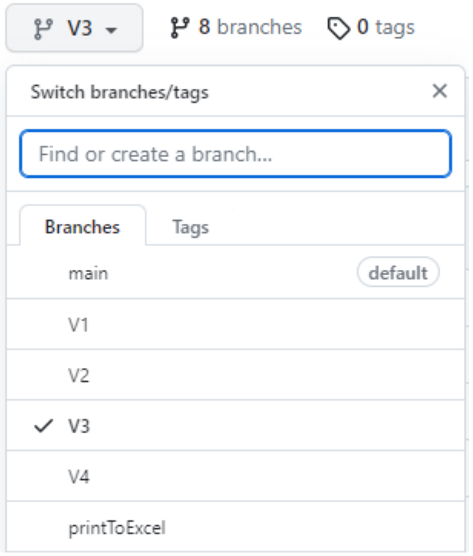
+ Developing a new feature
+ Creating a new version of a tool
+ Editing a section of a tool

Note that it is recommended that engineers working simultaneously on a feature work via the same branch. Information on this process is given later in this document.

See the image on the right for an example of how branches will look within GitHub, where 'main' is the working tool, V1-4 are the version branches, and 'printToExcel' is a feature in development to allow printing results to an Excel file.

Note:

For version overhauls, it is recommended to create a back-up branch that is unchanged. However, for simple feature changes/additions, the feature branch may be deleted once it has been integrated into the main.



Example of tool 'tree'

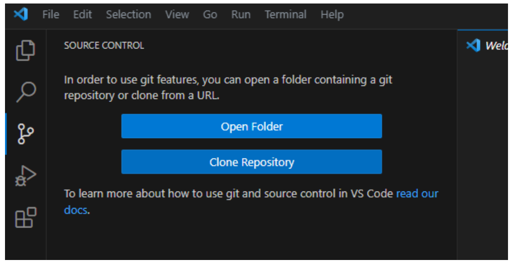
Using Git and VS Code

This slide describes the full development process, including terminal code inputs for working with branches.

Full development process / Terminal Code

+ Create remote repository for tool in GitHub

+ Clone repository to your local device in VS Code



SOURCE CONTROL

In order to use git features, you can open a folder containing a git repository or clone from a URL.

Open Folder

Clone Repository

To learn more about how to use git and source control in VS Code read our docs.

+ Create new branch

-b creates new branch

Checkout changes branch

name of branch

```
git checkout -b version5
```

+ Push branch to remote repository

origin refers to remote repository

```
git push -u origin version5
```

Changes will not show in the remote repository until they are 'pushed' from your local cloned repository

+ Make edits and additions locally

+ Add changed files to commit queue

. implies all changes are to be committed. You may also name specific files here

```
git add .
```

+ Commit to local repository

-m allows commit message to be entered. Describe briefly what's changed.

```
git commit -m "Changed once again"
```

Commit will make changes permanent in your local branch

+ Push changes to remote repository

```
git push
```

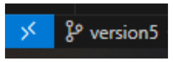
Copies changes to the remote repository

+ Switch local repository to main branch

```
git checkout main
```

Switches to working in main branch

Your current branch is indicated in the bottom left of VS Code.



+ Merge new version to main

```
git merge version5
```

Adds branch changes to your main branch

+ Push merged version to repository

```
git push
```

Copies changes to remote repository for everyone to see

+ You have now created a new version of the tool

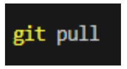
2 Developers, 1 Branch

When working on a feature or version in collaboration, it is recommended that developers work on the same branch, to ensure conflicts in their code are minimal when merging the new feature to the main branch.

Recommended process:

+ Create feature/version branch in the remote repository
+ Each engineer clones this branch to their local device
+ Engineers code in their local repository
+ Engineers update remote repository **as they work**
+ Engineers pull any changes from the remote repository to their local device as they work (more on this below)
+ When the updated feature/version has been checked and approved, engineers merge it with the main branch.

The primary issue to be aware of is your fellow engineer updating the remote repository without updating your local repository. Use the 'git pull' command to update your local repository with any changes made since you last worked on the script.



```
git pull
```

Git pull will update your local repository with any changes to the remote repository. This updates all the branches of that repository.

BE CAREFUL

Executing this command will **overwrite your files** in the local repository, so any changes that have been made that alter code you've worked on will be permanent. Ensure that you're happy with changes that have been made and **COMMUNICATE** with the engineer you are working with.

Temporarily copy and store files if you are unsure.

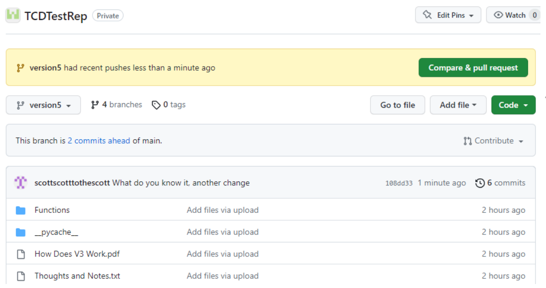
Notes:

Always push to remote repository **as you code** and develop working features. Don't script the entire new branch locally and push everything at once. Bit by bit will aid in simplifying the conflict control process and ensuring everyone works to the same files. Good practise is to think of it as refreshing your emails each morning.

Engineers are unlikely to be working on the exact same script at the same time but communicate thoroughly and plan who is working on which area and when. Working on the same code sections simultaneously will lead to many conflicts of code and is generally an inefficient method of working. Planning and communicating with your fellow engineers consistently is good practise regardless.

Additional Notes on Git and VS Code

This is what a repository in GitHub looks like



TCDTestRep Private

version5 had recent pushes less than a minute ago

This branch is 2 commits ahead of main.

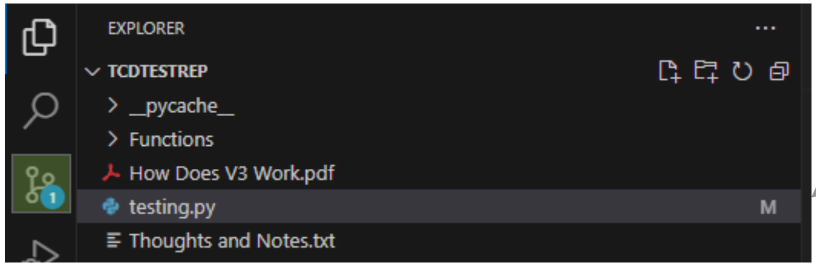
scottscotttothescott What do you know it, another change 108dd33 1 minute ago 6 commits

Functions	Add files via upload	2 hours ago
__pycache__	Add files via upload	2 hours ago
How Does V3 Work.pdf	Add files via upload	2 hours ago
Thoughts and Notes.txt	Add files via upload	2 hours ago

To clone a repository, get the url from the 'code' tab, under 'https'

A cloned repository will be saved on your local device as a folder, and files can be added and removed in the same way as any folder.

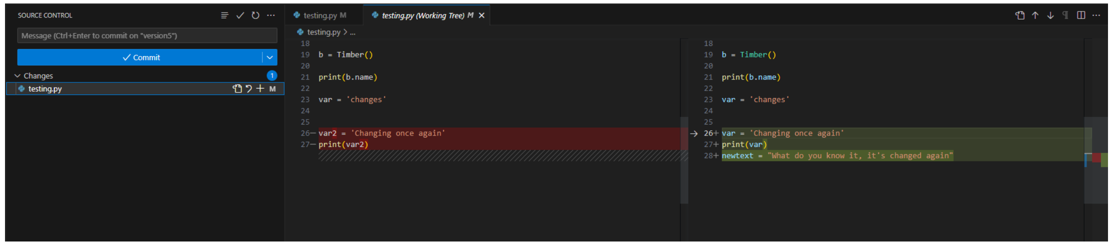
Git is accessed through the Source Control tab in VS Code



When a change is made, the file shows as 'M' for modified

Changes to code are highlighted in Source control so edits made can be easily seen. This shows before you commit changes.

Once you commit, the message you place with the commit will be displayed next to any lines altered.



SOURCE CONTROL
Message (Ctrl+Enter to commit on "version5")
Commit
Changes 1
testing.py M
testing.py M
testing.py (Working Tree) M
testing.py > ...

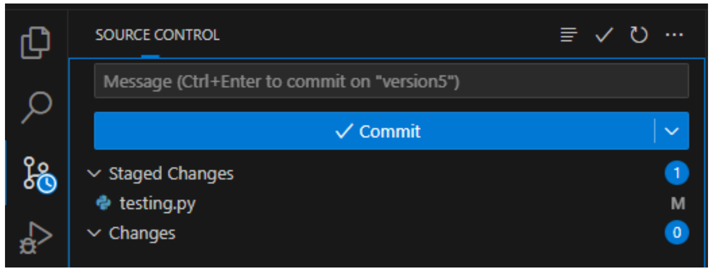
```
b = Timber()

print(b.name)

var = 'changes'


- var2 = 'Changing once again'
- print(var2)
```

```
b = Timber()

print(b.name)

var = 'changes'


+ var = 'Changing once again'
+ print(var)
+ newtext = "What do you know it, it's changed again"
```

Additional Notes on Git and VS Code

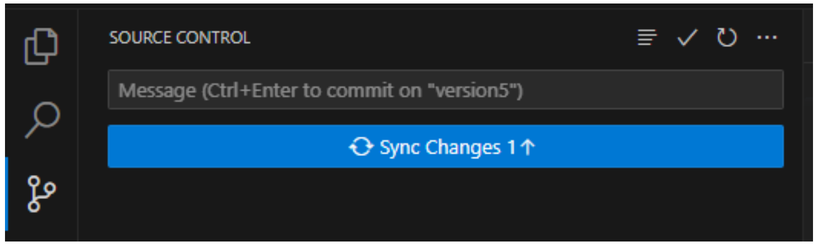
Changes made need to be staged in order to be committed. This allows individually edited files to be changed, instead of having to commit all changes at once.

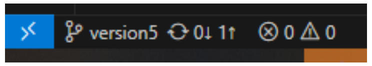
This is done with the 'add' command



When a change is committed, it will only be added to the local device the engineer is working on. To add to the remote repository, it must be synced.

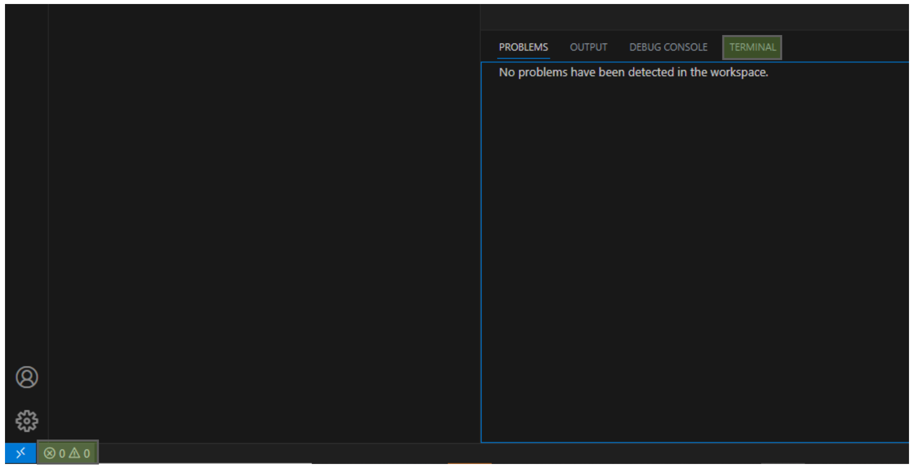
This is done with the 'push' command





This indicates that 1 commit has been made to the local branch but is yet to be pushed to the main branch.

Most git commands can be executed in either Source Control or typing directly into the terminal, however it is recommended to use terminal commands as it is much quicker and easier once learnt.



No problems have been detected in the workspace.

The commands shown in the 'using Git and VS Code' page are likely all you'll need.

If the terminal isn't showing when you open VS Code, the easiest way to access is clicking the errors icon in the bottom left.