Multivariable Calculus Self-Learning Module

Setup & Usage Manual

Contents

1   Introduction

The website is currently hosted for free by *GitHub Pages*. In order to deploy a website using GitHub Pages you need a *GitHub* account.

What is GitHub?   GitHub is a very popular, free, cloud-based platform that allows users to store and share files (mainly but not limited to code). Think of it as a storage space just like Google Drive for anything you might want to share with others. Each project on GitHub is stored in its own directory, which is called a *repository*. GitHub uses *Git* to keep track of any changes that happen in a repository.

What is Git?   Git is an open-source version control software, which is widely used to keep track of any changes that happen in a file. Think of it as an alternative to making multiple files such as: "exam-v1", "exam-v2", "exam-v1-fixed", ..., "exam-final". Git keeps track of all the changes made, and you can revert back to any previous version of your file if you mess up, or you can test new stuff by branching out to a new path without ever losing your main path. Git will store the entire version tree and will let you jump on any branch at any moment. GitHub uses Git, but Git is independent of GitHub and can work with its own local repositories on your computer.

How does GitHub Pages work?   You simply deploy your website directly from your GitHub repository on GitHub Pages. After that point, your website is always running, and whenever you make a change to the repository, it will almost immediately be shown on the website.

What is the point of all this?   The idea is the following:

1. You keep your own version of the website on your computer, along with a local installation of Git.

2. Say you make some changes that you are happy with, e.g. you add an exercise or fix a mistake. You *commit* the changes to your local Git repository.

3. You then *push* the local Git repository to the online GitHub repository.

4. GitHub Pages will automatically detect the new changes from your GitHub repository and deploy a new version of your website.

The good part is that once setup this process is totally free, standardised, and should be familiar to anyone who has ever collaborated on a coding project.

2   Setup

2.1   Git

The latest version of Git for Windows can be downloaded from:

`https://git-scm.com/download/win`

In the vast majority of cases the 64-bit version (first link) is required. The default installation options are fine. After installation is done, check if Git has been installed correctly by opening a terminal (can be done by *Windows key + X > Terminal (Admin)*) and typing the command:

`git -v`

The response should be the latest version of Git (e.g. Figure 1).

If not, try restarting the computer so that Git is added to the Windows PATH variable, which lets Windows know what the *git* command means. If this also doesn't work, Git must be added to the PATH manually. This is annoying and should (and most likely will) not happen, and at that point it is probably best to trouble shoot by Google.

2.1.1   Initiating a local Git repository

Assuming that Git is already installed, there are two ways to initiate a local Git repository.

Figure 1: `git -v`

Option 1: Initiate a new Git repository for an existing local directory. Suppose you have the directory *mvc-self-learning-module* somewhere on your computer and it is not already associated with a local Git repository. In order to initiate one, you can open the directory in a terminal (can be done by navigating within the directory, and then on any empty space *Right click > Open in Terminal*) and typing the command:

`git init`

The output should be something like Figure 2.

Figure 2: `git init`

You can visually inspect the Git repository if you enable the option of seeing the hidden files on your computer (can be switched on within the File Explorer itself from *View > Show > Hidden items* (Figure 3)). In that case, a folder named *.git* should appear in the current directory, which shows that the directory is now a Git repository (first folder in Figure 3).

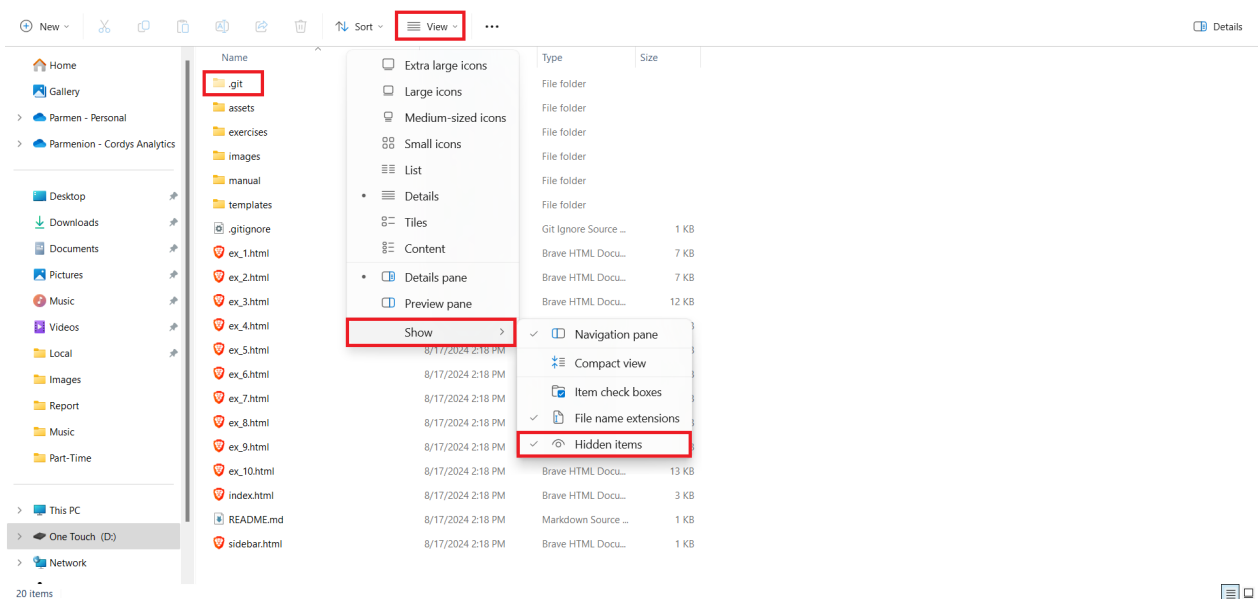You can now interact with this Git repository anytime by opening the directory in a terminal as before.

Figure 3: Enable hidden items. .git folder is shown.

Option 2: Clone an already existing Git repository from GitHub. This can be done as follows:

1. Copy the link to the repository from your browser, e.g.:

   `https://github.com/StefanMaubach/mvc-self-learning-module`

2. Navigate to the location on your computer where you would like to store the local Git repository and open a terminal within this location (*Right click > Open in Terminal*).

3. Type the command:

   `git clone https://github.com/StefanMaubach/mvc-self-learning-module`

The repository should then appear in a new directory named *mvc-self-learning-module* in the desired location.

In order to further interact with the Git repository you need to access this new directory with the terminal. You can enter the directory through the File Explorer and then open a new terminal in this directory in the same way as before, or if you still have the terminal open, you can access the new directory with the following command:

`cd mvc-self-learning-module`

(cd is the command used by Windows as well as Unix based systems to **c**hange **d**irectory).

2.1.2 Making the first commit

I now assume that you have a local Git repository. You now probably want Git to make a checkpoint out of the current state of your files by making a *commit*. This involves the following two steps:

- You tell Git what you want it to keep track of by using the command *git add*, followed by whatever filenames you want to add. In the vast majority of cases you probably want to add all of the files (except those from *.gitignore*, see Section 2.1.3) by using the dot (.) character. So the full command is:

  `git add .`

  You can ignore any warnings ending with "LF will be replaced by CRLF the next time Git touches it" or anything along those lines if they appear.

- Now it is time to finalise the commit by using the command *git commit*. Git by default asks you to label your commit by a message which briefly describes what the commit is about, e.g. "ex_1001 added" or "new sidebar layout" etc. For the first commit it is not uncommon to choose a less inspiring message, such as "first commit" or something. The messages are added using the *-m* flag. So the full command would look something like:

  `git commit -m "first commit"`

The terminal output should look something like Figure 4.

2.1.3 Adding .gitignore and README.md files (optional)

Since this Git repository is to be shared with others, it is a good idea to add a .gitignore file and possibly a README.md file. It is easiest to create and edit these files using VSCode (see Section 2.4 for setup). Alternatively, you can create these files as follows:

1. Enable file name extensions being shown in the File Explorer by *View > Show > File name extensions* (right above *Hidden items* in Figure 3).

2. Create a blank text file in the directory by *Right click > New > Text Document*.

3. A new text document with a name along the lines of *New Text Document.txt* should appear in the directory. Now you can simply replace the document's entire filename (including the extension .txt) by *.gitignore* or *README.md*.

You can edit these files in VSCode, or in Notepad by *Right click* (on the file) *> Open with*, and then either choosing Notepad if it appears as a suggestion, or searching for Notepad through *Choose another app*.

Figure 4: First commit.

The .gitignore file. The .gitignore file tells Git what to not keep track of, and therefore also not to share with others. You simply open the file and write row after row the names of any files/folders that you want Git to ignore. For example I have already included such a file which ignores the local LaTeX artefacts produced in my computer. Perhaps you would also like to add the entire *exercises* folder in the future so that nobody has access to the entire pdf (although someone who is motivated enough can still copy the entire exercise through the HTML files).

The README.md file. The README file is what everyone expects to see as a first introduction to your repository when you share it with them. GitHub by default displays the contents of the README.md file (if it exists) to anyone who accesses your repository through their browser. I have already included a README file in the repository, which only includes a title currently. The *.md* extension means that the file is written in *Markdown*, which is an easy markup language that essentially makes fancy text files. Pretty much all the Markdown formatting you will ever need to write a typical README file can be found here:

`https://www.markdownguide.org/cheat-sheet/`

2.2 GitHub

Opening a GitHub account should be as straightforward as opening any account in any other service if you select *Sign Up* from:

`https://github.com/`

2.2.1 Associating the local Git repository with a GitHub repository

I assume that Git is already installed and that you have a GitHub account. This step needs to be done only if you initiated a new local Git repository for your local directory (Option 1 in Section 2.1.1). If you cloned an already existing Git repository from GitHub, you can skip this section.

Creating a new GitHub repository. If you want to create a new GitHub repository for your project, you can do that by accessing your GitHub profile through your browser, at:

`https://github.com/StefanMaubach`

and then navigating to the *Repositories* tab and selecting *New* on the right (e.g. on my profile it looks like Figure 5).
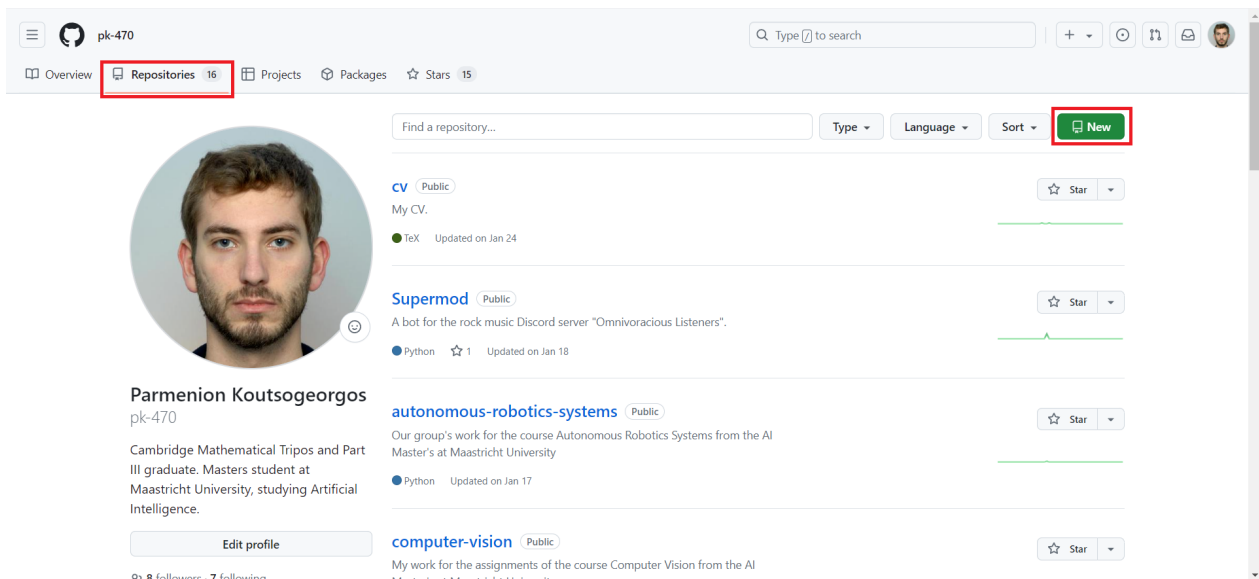
Figure 5: Start a new repository.

A menu as shown in Figure 6 should appear. The following are important:

1. Choose yourself as the repository owner.

2. Write down the local Git repository name as the GitHub repository name.

3. Make the repository public so that others will not need special permissions to view it, and more importantly so that you can deploy it on GitHub Pages for free. Note that even if the repository is public, nobody else will be able to edit it without permission.

You can also add a short description if you want. No other default options need to be changed. You can click *Create repository* after you are done. You should now be redirected to your new repository, at:

```
https://github.com/StefanMaubach/mvc-self-learning-module
```

Associating your local Git repository with an empty GitHub repository. I assume now that:

- You have a local Git repository for your project, preferably (but not necessarily) one at which at least one commit has been made.

- You have an empty GitHub repository in which you want to push your local Git repository.

If the GitHub repository is indeed still empty, a helpful guide such as in Figure 7 should appear when you access the repository through a browser. Make sure to select the *HTTPS* option of the guide as the SSH option requires further setup (although nowadays the SSH option is often preferred because of security reasons). Assuming that you already have a local Git repository, simply open its directory in a terminal as before and follow the second block of instructions shown in the guide. A few words about each line:

1. The first line:

```
git remote add origin https://github.com/StefanMaubach/mvc-self-learning-module.git
```

is the one that actually associates the local Git repository with the remote GitHub repository.

2. Technically the second line:

```
git branch -M main
```

is not needed. It is there because the main Git branch used to be called *master* instead of *main*, so this line is there to simply ensure compatibility with older repositories. If you installed Git using the default options, you do not need this.

3. The third line:

Figure 6: New repository options.

```
git push -u origin main
```

is needed only if you have made at least one commit in your local Git repository. Otherwise this line will throw an error in the terminal (which you can ignore), as there is no commit to push. Whenever you want your files to appear on GitHub however, go back to Section 2.1.2 and make your first commit, and then use this line to push it to the main branch on the remote GitHub repository.

The terminal output should look something like Figure 8.
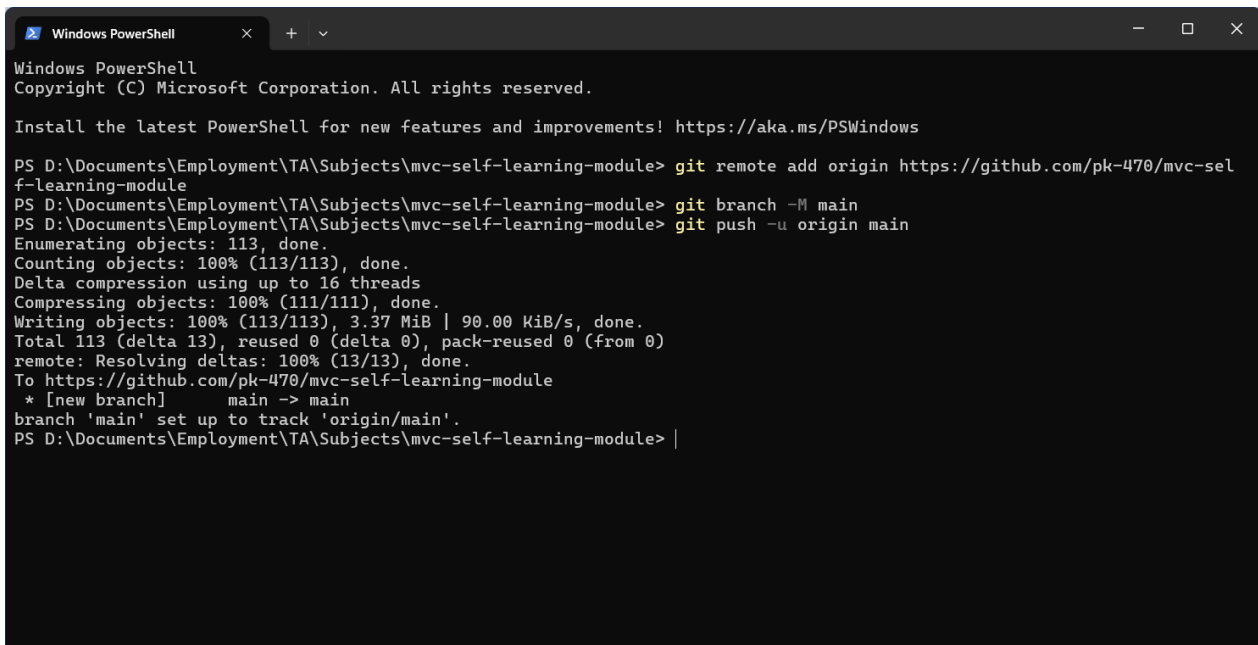

Figure 7: New repository setup.

Figure 8: Associate local Git repository to remote GitHub repository.

2.3   Deploying the website on GitHub pages

I now assume that:

- You have a local Git repository for your project at which at least one commit has been made.

- You have a GitHub repository associated with your local Git repository.

- You have pushed your local commits to the remote GitHub repository.

If you access the GitHub repository through your browser, it should look something like Figure 9.
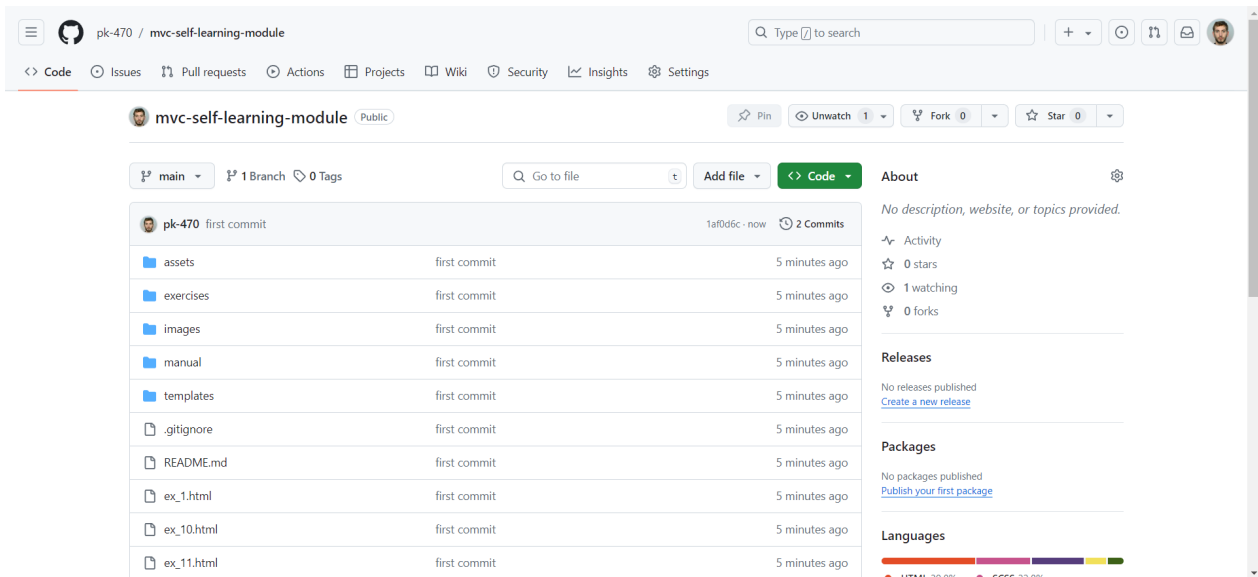

Figure 9: GitHub repository after setup and first push.

In order to deploy the repository as a website on GitHub pages, go to *Settings* (on the top) > *Pages* (on the left). Then at section *Build and deployment* and subsection *Branch* choose the *main* branch and click *Save* (Figure 10).
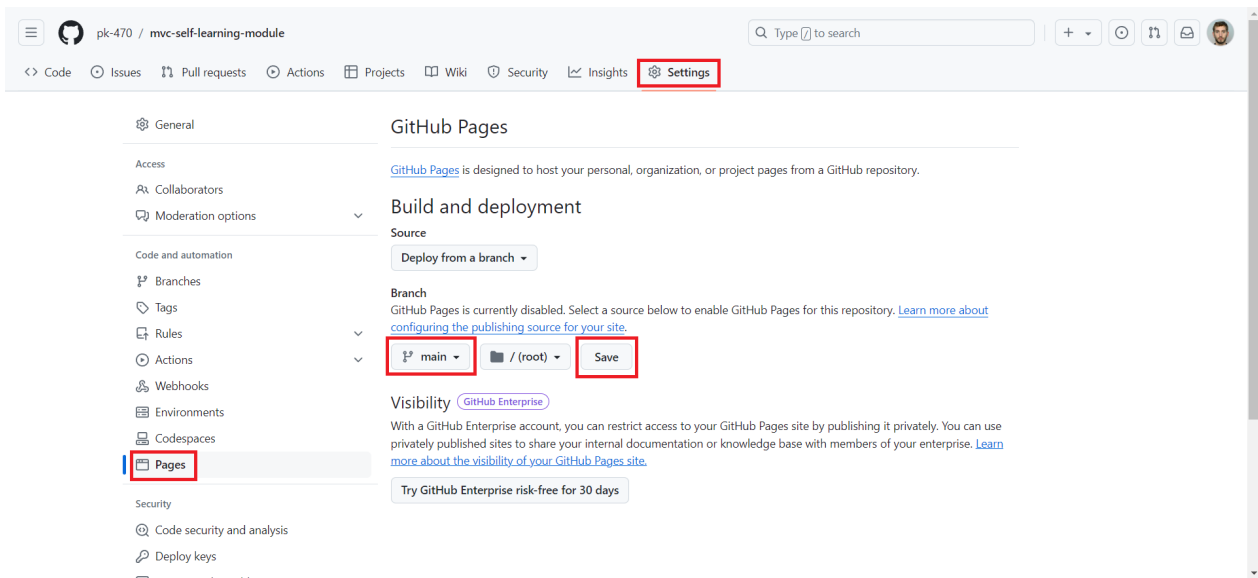

Figure 10: Deploy the repository on GitHub Pages.

If you go back to your main GitHub repository page now you should see a new section named *Deployments* on the right sidebar (Figure 11).

Clicking on *github-pages* should lead to a page similar to Figure 12.

Finally, following the first link should lead to the website (Figure 13). The website will remain in this link (unless you decide to change the domain, which can be done but is more complicated), and will update itself whenever you make a new commit and push it to the GitHub repository.

Figure 11: Deployment section added to the sidebar of the repository.

Figure 12: Deployment page for the repository.

Self-Learning Module: Multivariable Calculus

Welcome to the Multivariable Calculus self-learning module
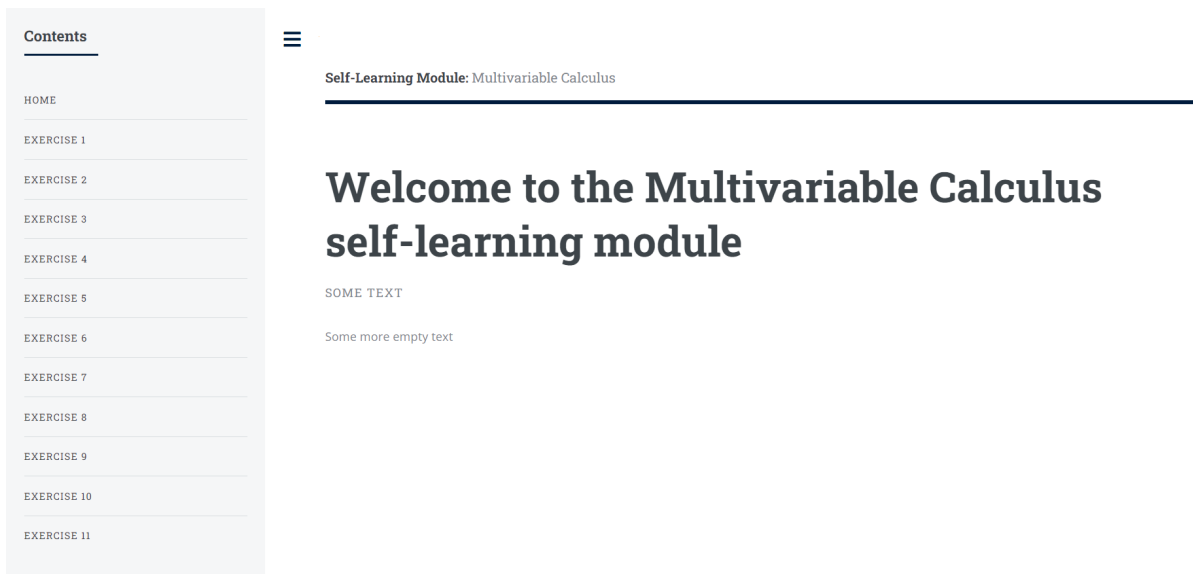
SOME TEXT

Some more empty text

Figure 13: Deployed website.

2.4   Setting up Visual Studio Code (optional)

I recommend using Visual Studio Code to edit HTML or any other files that you don't already know how to edit reliably. VSCode is a free and lightweight text editor that can be customised to work with almost every programming or markup language (including LaTeX) using extensions. You can download VSCode from:

`https://code.visualstudio.com/`

In order to add extensions to VSCode, you need to click on the block-looking icon on the left sidebar. This should open a small window next to the left sidebar, which contains a search bar and shows your installed extensions and possibly some recommended extensions. You can use the search bar to search for new extensions to install. For the purposes of this project I recommend the following three extensions:

3   Usage

Once everything is setup, things should be much easier! You can