General git configuration:

Make sure your git config (.gitconfig) file is set up so that you are committing with the correct username and email:

```
git config –global user.name="USERNAME_FOR_GIT"
git config –global user.email="EMAIL_FOR_GIT"
```

Note that your .gitconfig file may say that it's using a different email address than your main GitHub email if you have enabled email privacy. You can check that this other "no-reply" email in the git config file matches the anonymised email GitHub has linked to your account - it is listed in your (online) GitHub settings under Emails.

In order to be able to push to git, the config file also needs to contain the path of an SSH signing key (it's okay to direct this to your private key, but when adding a raw SSH key to your GitHub account, use the **PUBLIC KEY.** This will end in .pub - DO NOT USE THE PRIVATE ONE!!!!!!). The gpg format should also be SSH.

How to generate a new SSH key for GitHub:
https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
(Or see PowerPoint slides.)

How to add this SSH key to your GitHub account:
https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

The .gitconfig file is probably a hidden file in your main directory. Run

```
cd
ls –a
```

and see what settings are being used in the .gitconfig file. You can edit this file, and make sure that the path to the signing key file is for the private key half of the (private key, public key) being used by your GitHub account. **But only add the public key to the GitHub account!!!!!**

If you are asked to enter GitHub log-in credentials when trying to push to git, and you have set up SSH, try the following:

1) Is git trying to push to the right place?

Within your repository, run:

```
git remote –v
```

(The -v is for –verbose.)

If git returns http(s) URLs for the origin of fetch and push, for example…

```
>> origin  https://github.com/<user_name>/<repo> (fetch)
>> origin  https://github.com/<user_name/repo.git (push)
```

…then git is trying to push to the wrong place. To amend this, use:

```
git remote set–url origin git@github.com:<user_name>/<repo>
```

If you still cannot push to git without it asking for credentials (which is not supported, you would need to set up a personal access token to use as your password), check the following:

2) Is your general config file set up correctly for git?

Check this by running:

```
vim ~/.ssh/config
```

There should be a section of config file that looks like this:

```
Host github.com
     HostName github.com
     User git
     IdentityFile <path_to_your_private_git_key>
     IdentitiesOnly yes
```

If not, add or edit this section.

More troubleshooting advice can be found here:
https://docs.github.com/en/authentication/troubleshooting-ssh

Other Notes and Resources:

Recommended by David: https://learngitbranching.js.org/

Hattie's Workshop (8/12/23)

https://github.com/astro-group-bristol/python-template

Make a folder on your machine to store your github repos. Then clone the repository into your folder (under the code tab on github, USE THE SSH OPTION).

```
git clone git@github.com:astro-group-bristol/python-template.git
```

Follow Hattie's instructions in the readme to install and set up pipenv.

```
pipenv install --dev
```

Use `pipenv shell` to run the virtual environment.

Within the virtual environment, install pre-commit via:

```
pre-commit install
```

The pre-commit yaml file has been edited to use Black, which will make sure the code you are committing is formatted well (and give you sassy comments and cake emojis).

Made a new branch and switch to it using:

```
git branch <branch-name>
git checkout
```

or

```
git checkout -b <branch-name>
```

At any point, to see what branch you are on and what has been edited on that branch, use `git status`.

To switch to another branch (one that already exists), use:

```
git checkout <branch-name>
```

Make the changes to the file. Add them using `git add <file_name>` and then run `git commit` (to commit all changes that need to made) or run `git commit <file_name>`. **Black** will make changes to the file being committed to bring it into line with software development programming practices. If Black makes changes, you will **need to commit again after it has made these changes**, hence the name pre-commit.

When git committing (properly after pre-commit checks), you will be prompted to add a message describing the change you are making. If you just use `git commit` when committing, the a file summarising your changes will be opened in a text editor and you can add a message as a line to the top of the file. Alternatively, use `git commit -m "message about your commit"`.

To push to the branch you're interested in, use `git push -u origin <branch-name>`. To push to the main branch, we would use `main` for the branch name.

Use the -u (--upstream) tag if your repo has multiple branches. If you then try and merge/ git pull and you haven't used -u, git won't know which branch to pull from.