What is Version Control?

Do you find yourself storing copies of your code in different locations as a backup or naming them project1.py then project2.py etc just to keep copies? If so you are implementing a crude form of **version control**.

☣This approach works very well for small projects but can get muddled as you work on larger projects or in a team.

Version control is a formal approach to keeping versions of your code backed up and safe as you develop your code. That way if you wish to revert to old versions of the code, it is very easy to do so. For more about VC you may wish to view the presentation Download presentation .

Git

Git is the most widely used version control system. It allows you to track the code changes you make over time and revert to specific versions if you wish.

Git tracks and saves the history of the code in a **repository.** Git repositories can be created in the cloud using services such as GitHub or BitBucket. But for this module, we will create the repository on our PCs, in a hidden folder created by Git called .git.

Git and your project

You will be awarded marks for your version control in your project.

You will create a repo for your assignment.

Each time you add a function or a chunk of code to your project, you should **commit** this new version of the code to Git, along with an appropriate comment.

This creates a history of your project's development - a **log** - which you will share with us when submitting your project as evidence of the development of your project. Just take a screenshot of this log as evidence of the progression of your project.

In industry the comments often include emojis and Git even has its own meanings for emojis https://gist.github.com/parmentf/035de27d6ed1dce0b36a.Links to a

What do you need to do with Git for the project?

You must commit regularly

- every time you write a new function, verify it works then commit it to the repository i.e. create a version of the project that includes this function
- every time your write a chunk of code, verify it works then commit it e.g. let's say you write code to read data from a file into lists and committed that function; you might then go on to add an if statement to verify the file exists before you try to read from it: this is extra functionality and should be a commit.
- if you tidy the code (add hints/docstrings/comments, add named constants, refactor variable names, improve spacing, move functions into a module etc) commit these cosmetic changes

These commits would be spread over weeks so there might be 3 commits during a lab and then maybe no commits for a few days, then another few and so on as you build upon your code.

We are often asked how many commits and while there is no absolute number we can give, we can give an indication: when the project was coded there were 19 commits. You may have slightly more or less but this might serve to give you some idea.

We need a screenshot(s) of your Git log (history) which includes the dates and times of each commit (version). This can be achieved in line commands or PyCharm.

Download & Configure Git

1. Go to https://git-scm.com/downloads to download Git.
2. Install Git, making sure to add it to your environment variables
3. Set up your identity in the console:

```
git config --global user.name "John Doe"
git config --global user.email "johndoe@example.com"
```

Git in PyCharm

Git works well with most IDE's including PyCharm.

A file is included to describe how to use Git with PyCharm.

1 . Check in Settings that Git is accessible - Test
2. Choose GIT as your Version Control
3. When you create a new .py file add it to Git
4. Write a test function and run then commit these changes to Git
5. Make more changes , run and test and then commit the updated file
6. Use Git window to look at history and show difference between two different files.

Also useful :
https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-SetupLinks to an external site.

Git in Line Commands

If you don't use PyCharm then you can use Git line commands

==========