Activities 1: Working with Python projects

Table of contents

Activity 1: Python project structure

Theme: Working with code for applications

Before you start make sure you have created a GitHub account and have installed Python, git and a Python IDE e.g. Visual Studio Code or PyCharm Professional. If you have not, go to computer setup (1-06-opt-computer_setup.md).

What is a Python project?

A Python project is a structured collection of Python files and resources that work together to solve a specific problem or build an application. It typically includes:

- Python scripts or modules (.py files) containing the logic
- A main entry point (like main.py or app.py)
- Dependencies, other packages that your code needs to run
- Configuration files
- Optional assets such as data files, templates, or documentation

Using an IDE to work with Python projects

An IDE (Integrated Development Environment) is a software application that provides tools to help programmers write, test, and debug code. It combines features that are intended to make writing code more efficient.

There are numerous IDEs for writing Python code, online and standalone.

The course materials refer to VS Code and PyCharm, though neither is mandatory for the course. Notebooks e.g. Jupyter, are not suitable for this course as these differ from an IDE and don't provide the support for the range of tools and techniques that you will need to use in this minor. Notebooks are explicitly excluded for the coursework.

Project file structure: directories, packages, modules and files

There is no single correct structure for a Python project. A common layout is one referred to as the 'src' layout which for a project with data files might look like this:

Various configuration files for the project code and tools used in the project are placed in the root. An explanation of the more common files found here is given in a later activity in this document.

The Python code for the application is in a Python package in the **src** directory. "src" is short for "source code".

The application code is within a Python package called **my_package**. A package is a directory that contains an `__init__.py` file (even if it's empty). A package can include multiple Python files (modules) and sub-packages. A package allows for organized, modular code and supports importing e.g. `from my_package import some_module`.

A module is a single Python file. That file contains code and the code can be structured within it in classes, functions or procedural code.

Test code is in a separate **tests** folder. This can also be a package, but it does not need to be.

The **data** folder may be at the root level, or within the `my_project` package. This has implications for how the data files are referenced.

The **.venv** directory is a structure with code to create a Python virtual environment. This is explained later in this week's activities.

```
my_project_name/
├── README.md                # Gives info to others on your project and
how to use the it
├── pyproject.toml           # Project metadata and dependencies
├── requirements.txt         # Project dependencies
├── .gitignore               # Files to exclude from Git
├── .env                     # Variables specific to this Python project
environment
├── data/                    # Data files
│   ├── my_data.csv
│   └── my_database.db
├── src/                     # Application code
│   ├── my_app/
│   │   ├── __init__.py
│   │   ├── main.py (or app.py)
│   │   └── some_module.py
├── tests/                    # Test code
│   ├── conftest.py
│   └── test_some_module.py
└── .venv/                    # Python virtual environment
```

Task: Create a project in your IDE

Good practice for the project name is to:

- keep the project name brief but descriptive
- use all lowercase
- do not have spaces in the name, use underscored if you need to separate words
- avoid non-ASCII characters
- avoid names that conflict with reserved words such as test, setup or python
- avoid names that conflict with existing packages (this may be more difficult and not strictly an issue for the coursework, you can check existing packages in PyPi (https://pypi.org))

How you create a new project will vary by IDE. For example:

- In VS Code you open folder (https://code.visualstudio.com/docs/getstarted/getting-started)
- In PyCharm you create a new python project (https://www.jetbrains.com/help/pycharm/creating-and-running-your-first-python-project.html)

Either of the above will create a new folder (directory) on your computer in the location you specify using a name you specify.

During the rest of this week's activities, you will learn to use many of the default files shown in the project structure.

Next activity (1-02-environments.md)

Further reading

There are examples of typical Python project structures on these sites:

- Python Packaging User Guide (https://packaging.python.org/en/latest/discussions/src-layout-vs-flat-layout/#src-layout-vs-flat-layout)
- The Hitchhiker's Guide to Python (https://docs.python-guide.org/writing/structure/#sample-repository)
- Real Python (https://realpython.com/python-application-layouts/)

Activity 2: Python virtual environments

What is a virtual environment

A Python virtual environment is like a personal workspace for your Python project. It helps you:

- Keep things clean: It isolates your project's packages from others on your computer.
- Control dependencies: You can install exactly the versions of libraries your project needs—no conflicts!

Using a virtual environment lets you build your Python project without affecting anything outside it. If you only use your main Python environment you can experience issues when different projects require libraries of different versions or that conflict.

A virtual environment, or **venv** is created in a directory, usually within your project, and includes:

- A copy (or symlink) of the Python interpreter you used to create it.
- A dedicated folder structure for installing packages (so they don't mix with system-wide ones).
- Scripts to activate/deactivate the environment.

The structure of the venv looks a little different on macOS and Windows:

```
.venv/
├── bin/                   # Contains executables like python, pip,
and the activate script
│   ├── activate
│   ├── python
│   └── pip
├── lib/
│   └── pythonX.Y/         # X.Y is your version of Python, e.g. 3.13
│       └── site-packages/  # Packages installed in this environment
├── pyvenv.cfg             # Configuration for this environment
```

and the Windows version:

```
.venv/
├── Scripts/             # Contains executables like python, pip, and
the activate script
│   ├── activate.bat
│   ├── python.exe
│   └── pip.exe
├── Lib/
│   └── site-packages/  # Packages installed in this environment
├── pyvenv.cfg
```

Tools for creating a virtual environment

There are various tools for creating virtual environments and some also manage the packages within it.

These include venv, virtualenv, pipenv, conda, poetry, uv.

`venv` is the simplest tool and is bundled with Python so this will be referenced in all the activities for this minor. However, if you are already familiar with another tool then continue to use it.

Naming the virtual environment folder

The venv folder can be named anything, however good practice is to name it clearly so common names are: `venv`, `.venv`, `env` or `.env`. The `.` denotes it as a hidden folder so depending on your operating system settings you may not see the folder. You don't need to manually edit the venv folder so it is common to make it hidden.

Task: Create and activate a virtual environment with venv

This is a 2-step process to create and then activate a virtual environment.

varies by operating system and by IDE.

- VSCode (https://code.visualstudio.com/docs/python/environments)
- PyCharm (https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html)

A method that also works for most IDEs is to open a Terminal window within the IDE and enter the command syntax (https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/#create-and-use-virtual-environments) for your operating system.

The following examples create a virtual environment in a folder called `.venv` using a Python tool named `venv`. The second command then activates that environment.

e.g for MaxOS and Unix:

```
python3 -m venv .venv
source .venv/bin/activate
```

e.g. for Windows:

```
py -m venv .venv
.venv\Scripts\activate
```

There is a visual clue in the Terminal that the environment is activiated as the command prompt should now start with the name of the venv in brackets, e.g. `(.venv)`. For example, mine looks like this:

```
(.venv) sarahsanders@ML-F37Y6FG64V COMP0035-tutorials-2025 %
```

NB For Windows users you may experience an issue activating a venv. It may be required to enable the Activate.ps1 script by setting the execution policy (https://go.microsoft.com/fwlink/?LinkID=135170) for the user. You can do this by issuing the following PowerShell command at the prompt:

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

Adding packages to the virtual environment

You will need to add other Python packages (libraries) to the environment, including your own project's application code.

There are various package management tools, however we will use `pip` for the same reason as we used `venv`, it is already bundled with Python and is simple and sufficient for our needs. If you are already using tools such as conda or poetry then continue to do so.

pip

`pip` can install one or more packages at once, upgrade packages and delete them. Read more on pip here (https://pip.pypa.io/en/stable/user_guide/).

You can find a list of published Python packages, and the `pip` command to install them on the Python Package Index, PyPi (https://pypi.org)

A few common pip commands:

```
pip install pandas            # Install the latest version of pandas
pip install pandas matplotlib # Install the latest version of pandas
and matplotlib
pip install pandas==1.19.5    # Install version 1.3.5 of pandas
pip install --upgrade pandas  # Upgrade to the latest version of pandas
pip uninstall pandas          # Uninstall, remove, pandas from the venv
pip list                      # List the current packages installed in
the venv
pip freeze > requirements.txt # Take all the current packages installed
in the venv and save the names and versions to a text file named
requirements.txt
```

Task: add pandas to the venv

In the Terminal where the venv is activated try installing one or more packages, e.g. pandas, matplotlib

Run `pip list` before and after you do so you can see what has been installed.

You may find that other packages are also listed, this is because packages can have their own dependencies that need to be installed. For example, installing pandas also installs:

```
Package         Version
--------------- -----------
numpy           2.3.2
pandas          2.3.1
python-dateutil 2.9.0.post0
pytz            2025.2
six             1.17.0
tzdata          2025.2
```

requirements.txt

A common use case is to install multiple packages for a project using a file called `requirements.txt`. This is a file that lists the additional libraries that are required for the project code to run.

This is useful to allows anyone to create a virtual environment and install the same packages and versions to run the app code.

To install all the packages listed in a `requirements.txt` file:

```
pip install -r requirements.txt
```

You can update the requirements.txt file with new libraries or different versions by editing the file. You can specify just the package name, or you can also include information about the version to use. The requirements file format is documented here (https://pip.pypa.io/en/stable/reference/requirements-file-format/#requirements-file-format).

pyproject.toml

`pyproject.toml` is a file that is used to specify the build system requirements for the project to allow other developers to build and run your code. It has a number of uses, including an alternative for listing the packages to be installed.

More on pyproject.toml here (https://packaging.python.org/en/latest/guides/writing-pyproject-toml/).

There are many aspects that can be configured using pyproject.toml. Some of the tools, or packages, you use in the project can be configured using parameters in this pyproject.toml file. For this reason there is no single correct pyproject.toml, nor a single source of documentation. Typically, if a package can be configured using pyproject.toml then it will have a reference to that in its own documentation. For this course you are likely to use sections in pyproject.toml for setuptools (https://setuptools.pypa.io/en/latest/userguide/pyproject_config.html) and pytest (https://docs.pytest.org/en/6.2.x/customize.html#pyproject-toml)

For this course you will use pyproject.toml to install your own project application as a package in the venv.

To do this you can use the command `pip install -e .`

`-e` means install it in editable mode.

`.` means install it from a pyproject.toml in the current folder (usually the project root).

The minimum you will need to include in the pyproject.toml is:

```
# Basic project information
[project]
name = "hello-world-sample-project"
version = "2024.0.0"

# Most students will use setuptools, though poetry is also an option
[build-system]
requires = ["setuptools >= 61.0"]
build-backend = "setuptools.build_meta"

# Setuptools configuration see https://setuptools.pypa.io/en/latest/
        userguide/pyproject_config.html#setuptools-specific-
        configuration
[tool.setuptools.packages.find]
where = ["src"]  # list of folders that contain the packages (["."] by
        default)
include = ["my_package*"]  # package names should match these glob
        patterns (["*"] by default)
exclude = ["my_package.tests*"]
        # exclude packages matching these glob patterns (empty by
        default)
```

Task: Create a pyproject.toml and install your own code

1. Create a new file in the project root. This is a plain text file. Take care to name it exactly `pyproject.toml` (all lowercase).

    ```
    [project]
    name = "sample-project"
    version = "2025.0.0"

    [build-system]
    requires = ["setuptools >= 61.0"]
    build-backend = "setuptools.build_meta"
    ```

2. Enter the command `pip install -e .` in the Terminal window of your project where your venv is already activated.

You should see a new hidden folder (`.egg-info`) is created that has information about the installed project.

Next activity (1-03-source-code-control.md)

Further information

The Hitchhiker's Guide to Python: Structuring your project (https://docs.python-guide.org/writing/structure/)

Python Packaging User Guide: Packaging Python Projects (https://packaging.python.org/en/latest/tutorials/packaging-projects/)

freeCodeCamp: How to Build Your Very First Python Package (https://www.freecodecamp.org/news/build-your-first-python-package/)

Real Python: Python Application Layouts: A Reference (https://realpython.com/python-application-layouts/)

setuptools: Package Discovery and Namespace Packages (https://setuptools.pypa.io/en/latest/userguide/package_discovery.html)

setuptools: Data Files Support (https://setuptools.pypa.io/en/latest/userguide/datafiles.html)

Activity 3: Source code control with Git and GitHub

Setup

1. Make sure you installed have installed `git` on your computer (not in the project venv). See computer setup (1-06-opt-computer_setup.md).

2. Make sure you created a GitHub account. See computer setup (1-06-opt-computer_setup.md).

3. You may also want to install a gitignore plugin for your IDE, e.g.

- gitignore for VS Code (https://marketplace.visualstudio.com/items?itemName=codezombiech.gitignore)
- .ignore for PyCharm (https://plugins.jetbrains.com/plugin/7495--ignore)

1. Tool to work with GitHub

   You can work with GitHub directly from a Git command prompt. However, unless you feel comfortable using command line environments then this method isn't generally recommended for those just starting to learn Git and GitHub.

   PyCharm and VS Code have tools that integrate with GitHub. If you are using a python editor/ IDE that doesn't have Git support you may need to install a local Git desktop application. There are many Git clients that are freely available, for convenience you might want to use GitHub Desktop software.

Introduction

GitHub a tool that's become widely used in the world of software development and data science.

GitHub is like a social network for code—a platform where you can store your projects, track changes, collaborate with others, and showcase your work.

There are some key terms need to know:

- **Git**: Git is a version control system. It tracks changes to files over time, so you can revert to earlier versions, compare edits, and collaborate without overwriting each other's work. GitHub uses Git.
- **GitHub**: GitHub is a cloud-based platform that hosts Git repositories. It adds powerful collaboration features like issue tracking, pull requests, and project management tools. You can access your work from anywhere and invite others to contribute.
- **Repository** (or "**repo**"): A repository is a project folder managed by Git. It contains all your files, history of changes, and instructions for collaborators. You can have public repos (visible to everyone) or private ones (just for you or your team).

Overview of GitHub

GitHub holds a 'master' version of your project code, you can then have many copies (or 'clones') on one or more computers that you and others use to work with the code. GitHub uses workflows to manage these changes and allow them to be kept in sync.

There is no single way to work with GitHub though a minimal workflow would be:

1. Create repo on GitHub
2. Clone a repo to work locally
3. Commit changes as you work
4. Push one or more 'commit's (updates) to GitHub

If you are working with others, you would also need to 'pull' their changes, and make 'pull requests' to ask them to take your changes.

This introduced some further terms:

- **Branch**: A branch is a separate line of development within your project. It lets you work on new features, experiments, or fixes without affecting the main codebase (usually called the main or master branch). Once your changes are ready, you can merge your branch back into the main project.
- **Commit**: A commit is like saving a snapshot of your project. It records the changes you've made to your files and includes a message describing what you did (e.g., "Fixed login bug" or "Added new chart to report"). Each commit is part of your project's history, so you can always go back and see what changed, when, and why.
- **Push**: means you're sending your local commits to the remote repository on GitHub. You've made changes and committed them locally. Now you want to share those changes with others or back them up online.
- **Pull**: means you're downloading changes from GitHub to your local machine. Someone else may have updated the project on GitHub. You want to get the latest version before you start working. Pull will fetch and merge those changes into your local copy.

Additional files to create when using source code control

Typically, you add 2 further files to your project in the root folder when working with Git and GitHub:

1. `.gitignore`
2. `README.MD`

.gitignore

A `.gitignore` file is used to tell Git which files should not be tracked in the repository.

Create this before you use git for the first time in your project.

For example, you don't want to track configuration files that are specific to the code editor or IDE (e.g. VS Code), particularly when collaborating with others in a repo, as each person is likely to have a different local config (Windows v. Mac, file locations etc.). If you don't exclude these files from GitHub, then you may unintentionally cause problems for each other by overwriting each other's IDE configuration when you commit and push changes to the repo. You also don't want to track common python files such as the virtual environment for your project. These take up space and as you have seen the venv can be quickly recreated and packages added using `requirements.txt` or `pyproject.toml`.

You can create a typical .gitignore in your IDE using a text editor, though using a plugin in your IDE, PyCharm and VS Code can make it easier to create a .gitgnore file from templates.

- VS Code .gitignore extensions (https://code.visualstudio.com/docs/editor/versioncontrol#_scm-provider-extensions)
- Pycharm create .gitignore (https://www.jetbrains.com/help/pycharm/set-up-a-git-repository.html#ignore-files)

README.md

`README.md` tells other developers important information about your repo such as what the repo contains, any set-up instructions etc. You need to edit this for your coursework to explain to the markers anything they need to know to set up and run your code.

`.md` means it is a Markdown format file. Markdown is widely used for documentation of code. It is not difficult to learn, you just need a few basic syntax rules. Refer to the GitHub markdown guide (https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax).

1. Create a new file named `README.md` in your 'practice' project and open it.

2. Add a few lines e.g.:

   ◦ A first level heading "Practice".
   ◦ Add a second level heading e.g. "Introduction"
   ◦ Add a line of text e.g. "This repository is for learning to use GitHub"

```
# A level 1 heading

## A level 2 heading

Some text
```

3. Save the file.

Task: Add your project code to a GitHub repo

There are various ways to create a repository (your project code) in GitHub, including

- Create a new empty repo on GitHub
- Create a new repo by pushing code from your IDE to GitHub
- Create a new repo by 'fork'ing another person's repo
- Create a repo by accepting a GitHub Classroom assignment (uses fork)

For this activity you will create a new repo on GitHub from your IDE.

The steps you need to follow will vary depending on IDE so check their documentation.

First you need to link your IDE to your GitHub account:

- VS Code
- PyCharm (https://www.jetbrains.com/help/pycharm/github.html)

For PyCharm you share the code with GitHub (https://www.jetbrains.com/help/pycharm/manage-projects-hosted-on-github.html#share-on-GitHub). Enter the login when prompted.

The VS Code GitHub documentation (https://code.visualstudio.com/docs/editor/github) tells you to install the Pull requests extension which is not strictly necessary for this course. This is an alternative guide here that simplifies the guidance (https://graphite.dev/guides/how-to-push-code-from-vscode-to-github).

You can do also this with 'git' commands from the terminal in either IDE:

1. Go to GitHub (and log in)

2. Click New repository

3. Give it a name and choose visibility (public/private)

4. Do not initialize with README (you already have files locally)

5. Return to your computer. Use the IDE Terminal, or a command prompt tool where you are in the project root folder:

   ```
   git init
   git add .
   git commit -m "Initial commit"
   git remote add origin https://github.com/USERNAME/REPO-NAME.git #
           replace USERNAME and REPO-NAME with your GitHub username
           and repo name
   git push -u origin main
           # If your default branch is called master, use master
           instead of main.
   ```

6. Return to GitHub, you many need to refresh the page, but you should now see your files on GitHub.

You do not need to keep this practice repository. To delete:

- Close your IDE (close not minimise)
- Go to the folder structure on your computer and delete the folder.
- Go to GitHub and use their documentation (https://docs.github.com/en/repositories/creating-and-managing-repositories/deleting-a-repository) to delete the repo.

Next steps

For the rest of COMP0035 you should be able to manage with a minimum of two repositories:

1. Tutorial activities
2. Coursework (one repo for both)

The next two activities walk you through how to set these up. Each uses a different method.

You may also want to optionally integrate your IDE with GitHub (1-07-opt-integrate-IDE-GitHub.md) to make it easier to work between your IDE and GitHub.

Next activity (1-04-activities-repo.md)

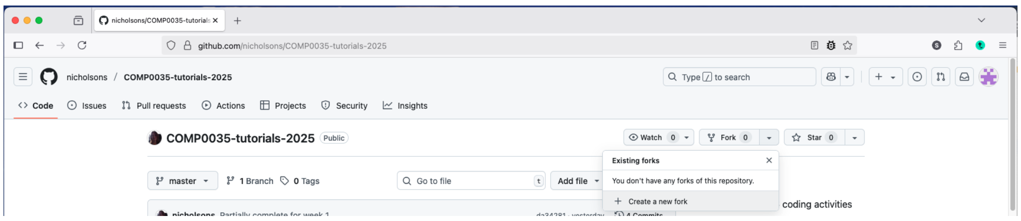
Activity 4: Set up the COMP0035 coding activities project

Create a copy of the repository using the ‘fork’ method

A ‘fork’ is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. You can fork a repository to propose changes to someone else’s project or to use someone else’s project as a starting point for your own idea.

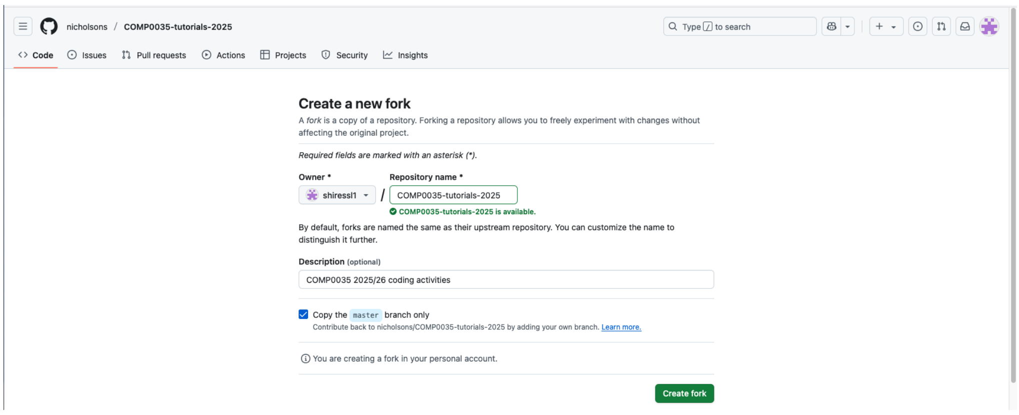
Go to the https://github.com/nicholsons/COMP0035-tutorials-2025 (https://github.com/nicholsons/COMP0035-tutorials-2025) repository.

1. Click on the ‘Fork’ button in the top right of the page.



0-1. Fork button

2. Select ‘Create a new fork’ then ‘Create fork’.



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * shiressl1 / Repository name * COMP0035-tutorials-2025

COMP0035-tutorials-2025 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional) COMP0035 2025/26 coding activities

✓ Copy the master branch only

Contribute back to nicholsons/COMP0035-tutorials-2025 by adding your own branch. Learn more.

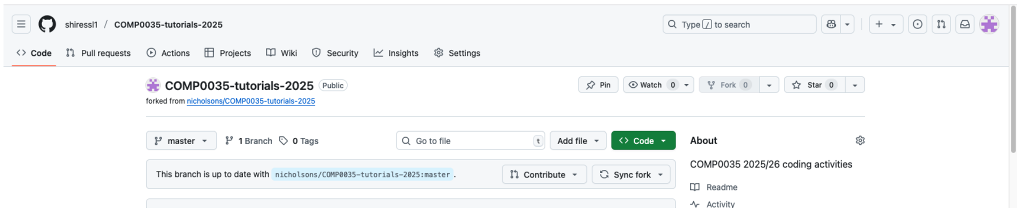
You are creating a fork in your personal account.

Create fork

0-1. Fork repository

You may see a temporary screen while the fork is created.

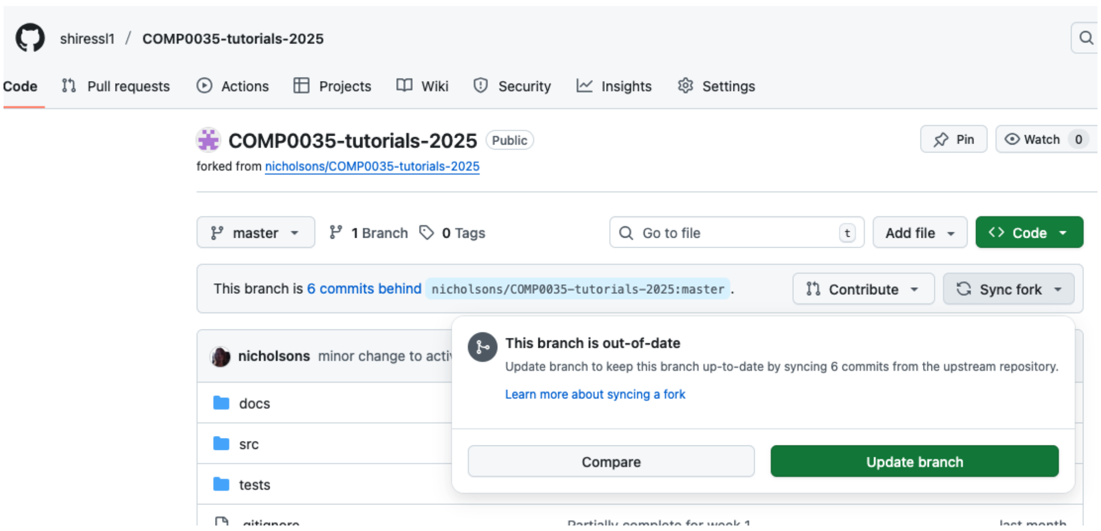
3. You will be taken to your new forked repository once created. The URL will be in the format `yourusername/comp0035-tutorials-2025`.



This branch is up to date with nicholsons/COMP0035-tutorials-2025:master .

0-1. Forked repository

IMPORTANT: This repository will be updated weekly with new activities. If there have been changes, you will see a message showing the branch is out of date e.g. “This branch is 1 commit behind...”. Select the green ‘Update branch’ button to pull the latest updates to your copy of the repo.



This branch is 6 commits behind nicholsons/COMP0035-tutorials-2025:master .

This branch is out-of-date

Update branch to keep this branch up-to-date by syncing 6 commits from the upstream repository.

Learn more about syncing a fork

Compare

Update branch

0-1. Update branch

Create a new project in your IDE

You will create a local copy, called a **clone**, of the repo you just created in your IDE. Make sure you use your forked copy and not the course master as you won't then be able to synch your changes.

VS Code

In VS code you have options to open remotely on GitHub, open in codespaces (online IDE), or clone. The following summarises the last approach.

Use the VS Code documentation (https://code.visualstudio.com/docs/sourcecontrol/github#_cloning-a-repository) which has the latest screenshots and guidance.

NB If you have integrated your IDE with GitHub (1-07-opt-integrate-IDE-GitHub.md) then it may give you a list of your repos to choose from. If not then you may be prompted to log in.

- Select the source code control icon in the left nav bar (3 circles connected with 2 lines)
- Select 'clone repository'
- Enter your GitHub repo URL e.g. `yourusername/comp0035-tutorials-2025` replacing yourusername with your actual GitHub username
- When prompted, choose the location where you want the repo to be saved on your computer then press "Select as Repository Destination".
- VS Code will create a subfolder within the selected destination for the project you are cloning.
- At the prompt to 'would you like to open the cloned repo' select Open.

PyCharm

Use the PyCharm documentation (https://www.jetbrains.com/help/pycharm/manage-projects-hosted-on-github.html#clone-from-GitHub) which has the latest screenshots and guidance.

NB If you have integrated your IDE with GitHub (1-07-opt-integrate-IDE-GitHub.md) then it may give you a list of your repos to choose from. If not then you may be prompted to log in.

Summary of steps:

- Menu option File > Project from version control
- Enter your GitHub repo URL e.g. `yourusername/comp0035-tutorials-2025` replacing yourusername with your actual GitHub username and select the folder you want this to be saved to on your computer then select Clone

- When asked if you want to create a venv from requirements.txt say yes (otherwise you will have to create it yourself)!

Setup the virtual environment

If using VS Code then refer to activity 2 and complete these steps:

- Create and activate a venv
- Install dependencies from requirements.txt
- Install the project code

If you used PyCharm as above then the venv will have been created already and the packages installed from requirements.txt so you only need to install the project code.

Summary of commands if using the Terminal:

MacOS

```
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
pip install -e .
```

Windows

```
py -m venv .venv
.venv\Scripts\activate
pip install -r requirements.txt
pip install -e .
```

Next activity (1-05-coursework-repo.md)

Activity 5: Set up the coursework project

You can use the same project for coursework 1 and 2 so you only need to do this once.

GitHub Classroom provides a way for you to create repository so that it can then be accessed by yourself, the course tutor and PGTAs.

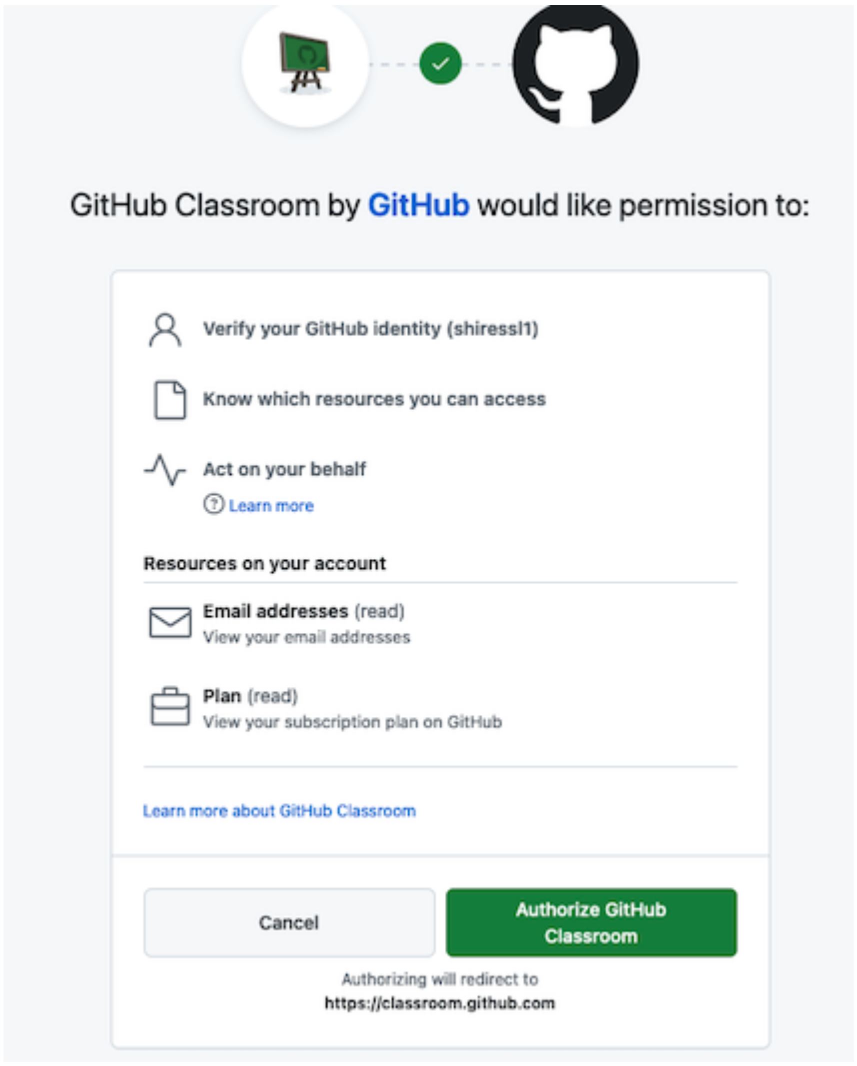
All coursework repositories for COMP0034 and COMP0035 are created using GitHub Classroom.

You click on a URL that creates a copy of a template repository with some starter files that you will need for the coursework.

Repositories that are created this way are stored in an organisation called **ucl-comp0035**, not in your personal GitHub account. They are created as **private** which in this case means *you, the course tutor and PGTAs* can see your code; but other students cannot.

Create the repository

1. Login to GitHub (https://github.com/login)
2. After you log in, click on this practice GitHub classroom assignment link: https://classroom.github.com/a/XDkrDasZ (https://classroom.github.com/a/XDkrDasZ)
3. The first time that you use GitHub classroom you need to:
    - authorise your GitHub account to use GitHub Classroom
    - join the organisation that owns the classroom, **ucl-comp0035**
    - If you see the following please authorise the classroom:


GitHub Classroom by GitHub would like permission to:

Verify your GitHub identity (shiressl1)

Know which resources you can access

Act on your behalf

Learn more

Resources on your account

Email addresses (read)
View your email addresses

Plan (read)
View your subscription plan on GitHub

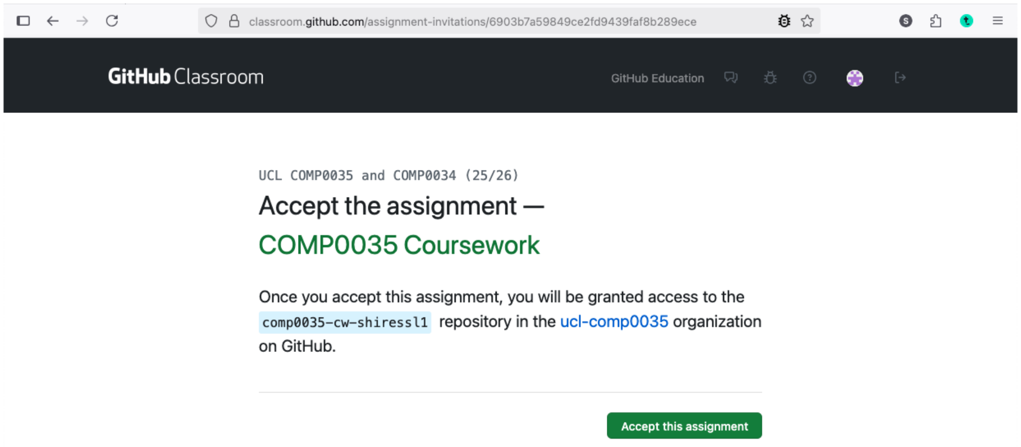
Learn more about GitHub Classroom

Authorizing will redirect to
https://classroom.github.com

0-1. Authorise GitHub

- If you see the following then click on the green button to join the organisation:

You should see something similar to the following:


classroom.github.com/assignment-invitations/6903b7a59849ce2fd9439faf8b289ece

UCL COMP0035 and COMP0034 (25/26)

Accept the assignment — COMP0035 Coursework

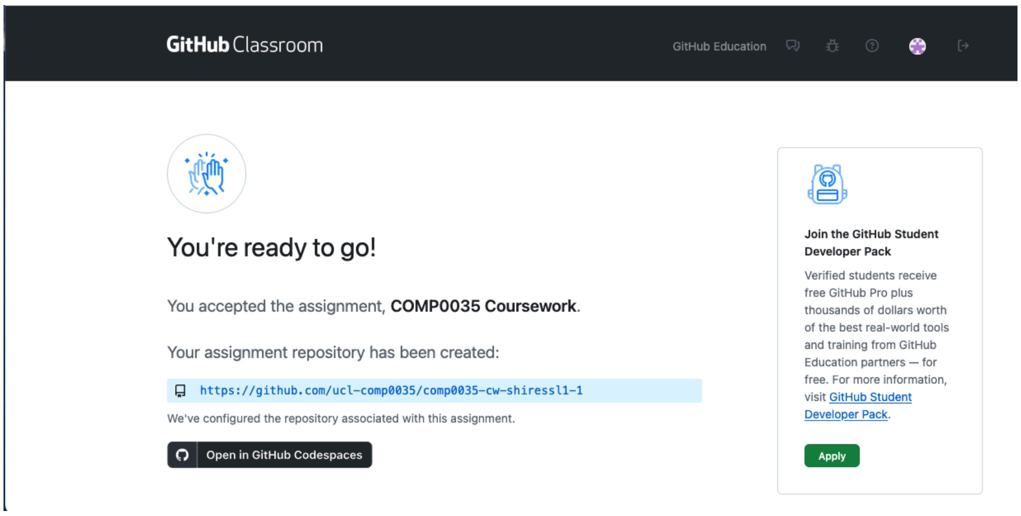
Once you accept this assignment, you will be granted access to the `comp0035-cw-shiressl1` repository in the ucl-comp0035 organization on GitHub.

0-1. Accept the assignment

4. Press ‘Accept this assignment’.

   It may take a minute for the repository to be created, and you may need to refresh the browser page to see when it has finished.

Once finished it will look something like this:



GitHub Classroom

You're ready to go!

You accepted the assignment, **COMP0035 Coursework**.

Your assignment repository has been created:

https://github.com/ucl-comp0035/comp0035-cw-shiressl1-1

We've configured the repository associated with this assignment.

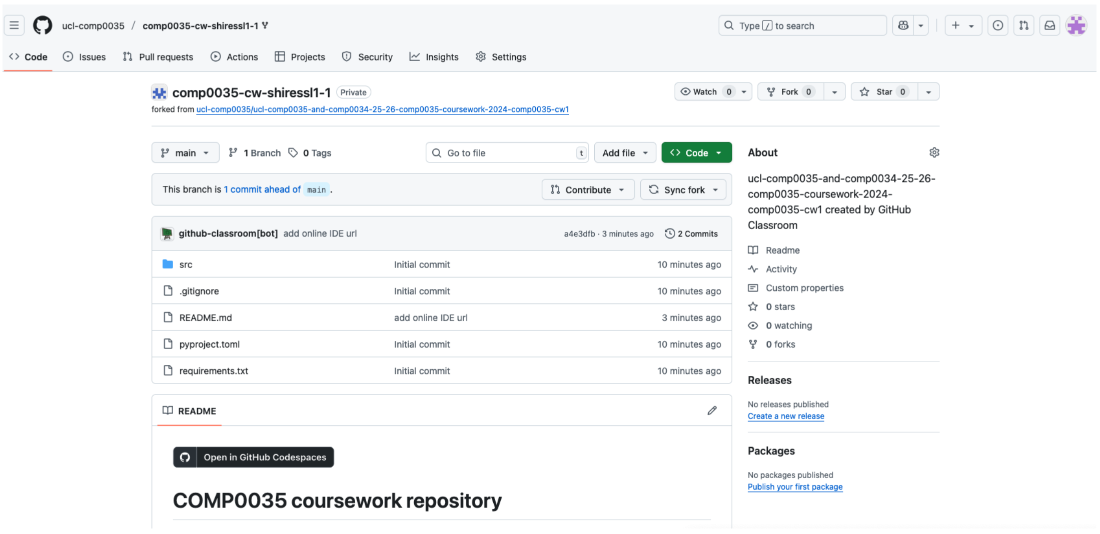
Join the GitHub Student Developer Pack

Verified students receive free GitHub Pro plus thousands of dollars worth of the best real-world tools and training from GitHub Education partners — for free. For more information, visit GitHub Student Developer Pack.

0-1. Assignment ready

If you click on the link it should go to the newly created repository.

Note: The ‘Open in Codespaces’ button allows you to work with the code in an online version of the VS Code IDE instead of on your computer.



comp0035-cw-shiressl1-1 Private

forked from ucl-comp0035/ucl-comp0035-and-comp0034-25-26-comp0035-coursework-2024-comp0035-cw1

This branch is 1 commit ahead of `main`.

github-classroom[bot] add online IDE url a4e3dfb · 3 minutes ago 2 Commits

src	Initial commit	10 minutes ago
.gitignore	Initial commit	10 minutes ago
README.md	add online IDE url	3 minutes ago
pyproject.toml	Initial commit	10 minutes ago
requirements.txt	Initial commit	10 minutes ago

About

ucl-comp0035-and-comp0034-25-26-comp0035-coursework-2024-comp0035-cw1 created by GitHub Classroom

Readme
Activity
Custom properties
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

COMP0035 coursework repository

0-1. Created repo in GitHub

Accessing the repository

This repository is in the ucl-comp0035 organisation and NOT in your personal GitHub account.

The URL to access it will be something like this but with your GitHub username at the end: `https://github.com/ucl-comp0035/comp0035-cw-yourgithubusername`

If you joined the ucl-comp0035 organisation then you can also find it through your GitHub page going to the top right menu and selecting ‘Organisations’.

Create the project in your IDE

Use the instructions in activities 3 and 4 to:

- Clone the repository from GitHub to your IDE
- Create and activate a venv

- Install the requirements and the code: `pip install -e .` (you don't need to do `pip install requirements.txt` as well as pyproject.toml has the dependencies listed)

Make a change and push that change to GitHub

Now that you have a local copy and a remote copy on GitHub, learn how to keep your files in synch.

Open the README.md file and add a line of text. You can write anything. Save the file.

Create a new python file in the coursework 1 folder, e.g. data_prep.py and write a 1 line comment in it.

You now need to:

- **add** or **stage** the new Python file so that git knows it should be tracked (your IDE may have prompted you to do this when you saved the new file)
- **commit** the changes and add a **commit message** to say what the change is e.g. "Practice commit"
- **push** or **sync** the changes with GitHub

The instructions for this differ by IDE:

- VS Code (https://code.visualstudio.com/docs/sourcecontrol/overview#_commit)
- PyCharm (https://www.jetbrains.com/help/pycharm/commit-and-push-changes.html#commit)

Make a change on GitHub and pull the change to your IDE

This step is optional.

You can edit code on GitHub directly. If you do this then you need to pull changes from GitHub to your local computer.

- Find the repository on GitHub.
- Click on the README.md file
- Use pencil icon to open in Edit mode.
- Add some text
- Select the 'commit changes' button
- Edit the commit message if you wish, then press 'commit'
- Now go to your IDE and find the option to 'pull', 'fetch' or 'synch' changes
  - VS Code (https://learn.microsoft.com/en-us/azure/developer/javascript/how-to/with-visual-studio-code/clone-github-repository?tabs=activity-bar)
  - PyCharm (https://www.jetbrains.com/help/pycharm/sync-with-a-remote-repository.html#fetch)

Conflicts

If you edit the same file on your local repo and the remote repo then when you try to pull/push/sync you will get a merge conflict. This means that git will need you to decide which version of the changes to keep.

Again, how you deal with this varies by IDE.

Typically, IDEs provide a view that shows both versions of the file and indicates the lines that are different and will have a mechanism that allows to select which version to keep.

- VS Code (https://code.visualstudio.com/docs/sourcecontrol/overview#_merge-conflicts)

- PyCharm (https://www.jetbrains.com/help/pycharm/resolve-conflicts.html)

Activity 6 (optional): Computer setup for COMP0035 and COMP0034

Skip to next (1-07-opt-integrate-IDE-GitHub.md) if you already completed this.

You need a python coding environment with the following.

Install/set-up in the order they are shown below.

Please do this before the first tutorial.

1. Install Python

Check if you already have this installed on your computer.

Open Terminal (Mac) or Powershell (Windows), and at the command prompt type and enter:

```
python3 --version
```

If Python 3 is installed you should see a version number returned e.g. Python 3.13.3

If you don't already have python 3.7 or later than please download and install the relevant version of python (https://www.python.org/downloads/) for your operating system.

2. Install Git

Git is the underlying source code control software that GitHub uses.

You need to have Git in order to work with code that is both on your computer and in GitHub.

Download the relevant version of Git (https://git-scm.com/downloads) for your operating system and install it on your computer.

3. Create a GitHub account

If you do not already have a GitHub account then please create a GitHub account (https://github.com/join) and optionally signup for a student developer pack (https://education.github.com/pack). Signing up for the student developer has benefits e.g. access to co-pilot.

Avoid usernames that would allow you to be easily identified if possible.

If you have an existing account you can use that for the course. To gain access to the student developer pack, you can also associate your UCL email with your account. The student developer pack which gives you access to additional resources, such as Copilot AI integration with VS Code.

Guidance for how to add a second email address (https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/adding-an-email-address-to-your-github-account#)

If you cannot use GitHub for a specific reason please contact the course tutor to try and find an alternative.

4. Install an interactive development environment (IDE) for Python

You need a coding environment to develop Python and in COMP0034 you will also need to edit SQL, HTML, CSS and possibly JavaScript.

The most powerful way to edit code is to use an integrated development environment (IDE). IDEs have features that go beyond simple code writing to make a developer's life easier. However, they can take some time to learn how to use, and if you are installing on your own computer they can require a significant amount of disk space.

If you already have a preferred IDE or editor that supports Python, Flask (python, Jinja2), SQL, HTML, CSS and JavaScript then feel free to continue using it.

If you do not have a suitable IDE then install one suitable for Python. Free options include:

- Visual Studio Code (https://code.visualstudio.com/download).
- PyCharm Professional (NOT the community version). You will need to apply for a free student license (https://www.jetbrains.com/community/education/#students) to be able to use the Professional version.
- Atom (https://atom.io)

The teaching materials are based on VS Code, though notes are usually included where there are differences for PyCharm. I use PyCharm Professional, most students use VS Code.

Jupyter is not recommended for these modules. While Jupyter is widely used in data science, and is useful for preparing and visualising datasets, it isn't as well suited to developing apps as an IDE. You are not permitted to submit .ipynb files for the coursework.

You may be able to use a basic text or code editor, however it is likely to be far more challenging to learn how to use Git, create and run unit tests, check code compliance, and other things that we will be looking at in this course.

Most students set up an environment on a computer to which they have regular access. If you are not able to install or configure software on a computer you have access to then you could try PythonAnywhere free beginner account (https://www.pythonanywhere.com/pricing/).

UCL computers are not ideal for this course as you are unlikely to be able to install the necessary packages.

Extensions to install for VS Code

For VS Code you may also need to install extensions:

- Python for VS Code (https://code.visualstudio.com/docs/languages/python)
- GitHub Co Pilot for VS Code (https://code.visualstudio.com/docs/languages/python#_enhance-completions-with-ai)
- SQLite support (https://marketplace.visualstudio.com/items?itemName=alexcvzz.vscode-sqlite) There are other options if you search for SQLite in the marketplace.
- flake8 (https://marketplace.visualstudio.com/items?itemName=ms-python.flake8). Optional. Linting support for flake8.
- Mermaid chart (https://marketplace.visualstudio.com/items?itemName=MermaidChart.vscode-mermaid-chart) Optional. Draw diagrams in markdown.
- gitignore (https://marketplace.visualstudio.com/items?itemName=codezombiech.gitignore) Optional. Create .gitignore files using templates.

Plugins to install for PyCharm Professional

For PyCharm Professional you may also need to install plugins:

- ignore for .gitignore files (https://plugins.jetbrains.com/plugin/7495--ignore)
- GitHub Co Pilot for PyCharm (https://plugins.jetbrains.com/plugin/17718-github-copilot)
- Mermaid (https://plugins.jetbrains.com/plugin/20146-mermaid) to view and edit ERD and UML diagrams in markdown.

SQLite database tools are included in the PyCharm Professional version.

Activity 7 (optional): Integrate your IDE with your GitHub account

Skip to next (1-08-opt-copilot-ide.md) if you have already completed this or do not want to integrate it.

You will be creating and editing files in your IDE and so you need to enable it to work with GitHub to manage your code.

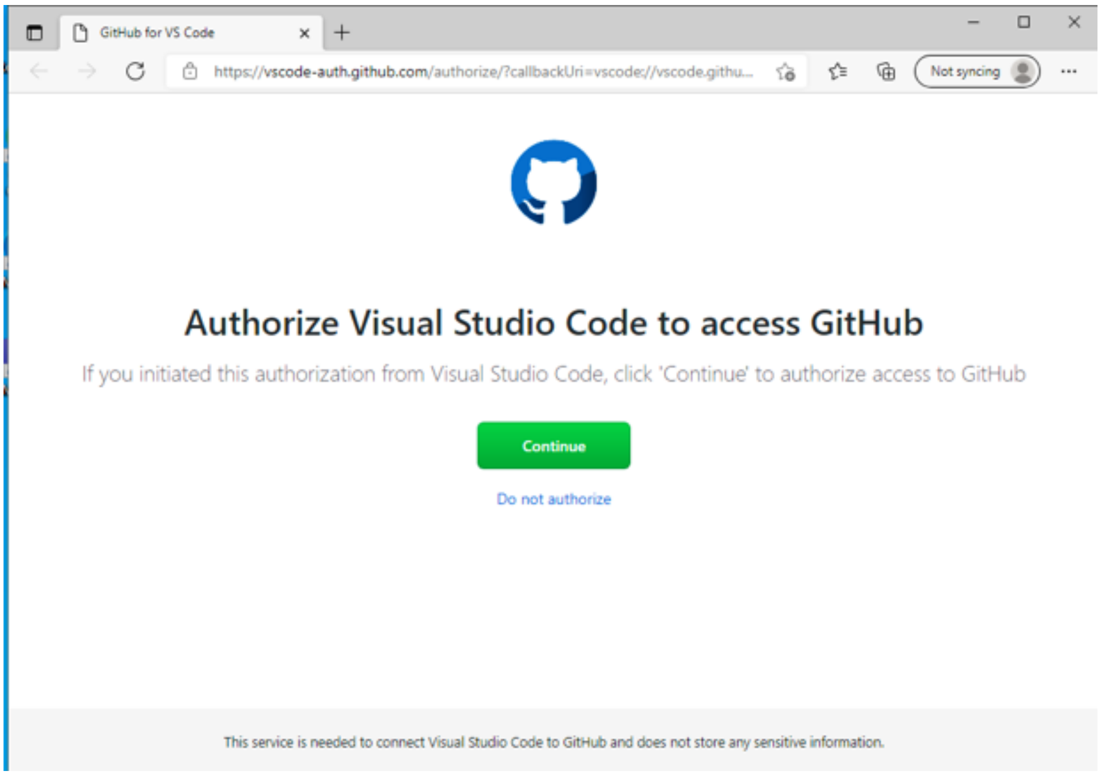
You will add authorisation in GitHub to allow your IDE to access your GitHub account using a GitHub personal access token (https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens) .

How you integrate your IDE with GitHub is specific to the particular IDE, so follow the relevant documentation to complete this:

- VS Code GitHub documentation (https://code.visualstudio.com/docs/editor/github)
- PyCharm Register an existing account with a token (https://www.jetbrains.com/help/pycharm/github.html#ef23dd64)

Notes on VS Code

If you are prompted to authorise your IDE to access GitHub then please accept e.g. the following shows the screens you may see using VS Code:


GitHub for VS Code
https://vscode-auth.github.com/authorize/?callbackUri=vscode://vscode.githu...
Not syncing
Continue
Do not authorize

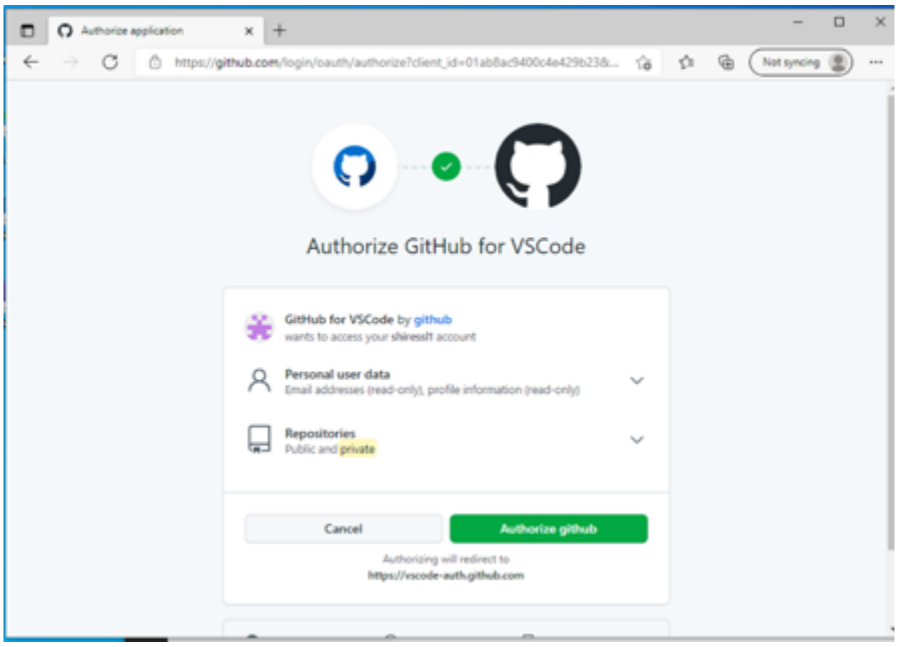
Authorize Visual Studio Code to access GitHub

If you initiated this authorization from Visual Studio Code, click 'Continue' to authorize access to GitHub

This service is needed to connect Visual Studio Code to GitHub and does not store any sensitive information.

0-1. GitHub request to authorise VS Code

If you are not already logged in to GitHub you will be prompted to log in and then Authorise GitHub:


Authorize application

https://github.com/login/oauth/authorize?client_id=01ab8ac9400c4e429b23&...

Authorize GitHub for VSCode

GitHub for VSCode by github
wants to access your shiresslt account

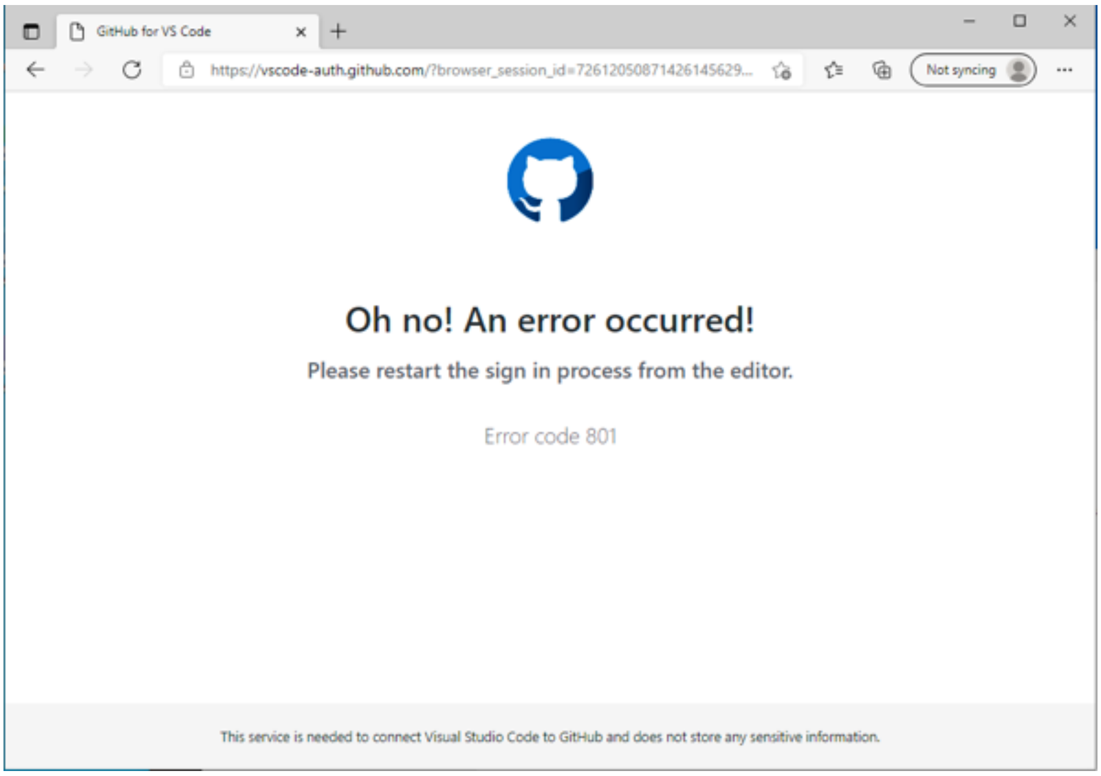
Personal user data
Email addresses (read-only), profile information (read-only)

Repositories
Public and private

Authorizing will redirect to
https://vscode-auth.github.com

0-1. GitHub request to authorise VS Code

I experienced the following error message:


GitHub for VS Code

https://vscode-auth.github.com/?browser_session_id=7261205087142614529...

Oh no! An error occurred!

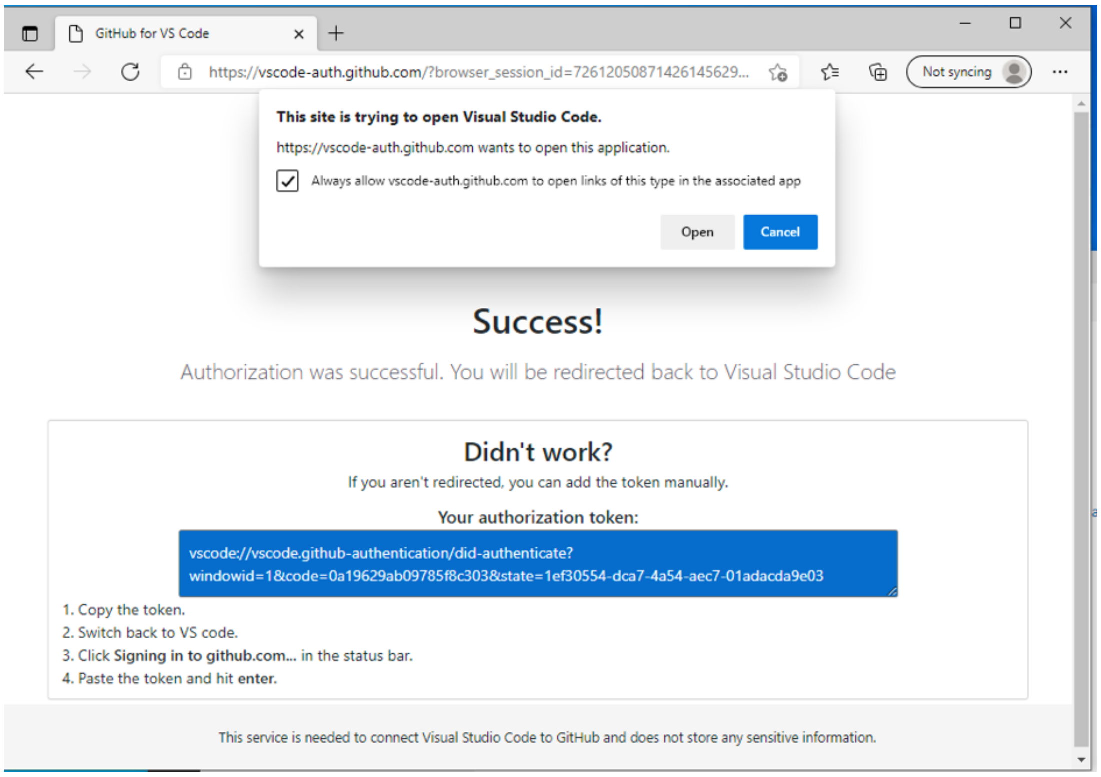
Please restart the sign in process from the editor.

Error code 801

This service is needed to connect Visual Studio Code to GitHub and does not store any sensitive information.

0-1. GitHub error

On repeating the steps I then accepted the following and was successful:


GitHub for VS Code
https://vscode-auth.github.com/?browser_session_id=72612050871426145629...
Not syncing
Open
Cancel

This site is trying to open Visual Studio Code.

https://vscode-auth.github.com wants to open this application.

☑ Always allow vscode-auth.github.com to open links of this type in the associated app

Success!

Authorization was successful. You will be redirected back to Visual Studio Code

Didn't work?

If you aren't redirected, you can add the token manually.

Your authorization token:

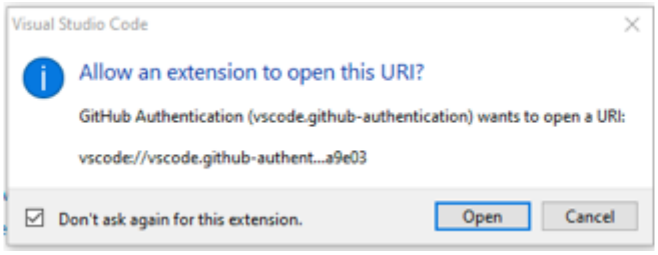
vscode://vscode.github-authentication/did-authenticate?windowid=1&code=0a19629ab09785f8c303&state=1ef30554-dca7-4a54-aec7-01adacda9e03

1. Copy the token.
2. Switch back to VS code.
3. Click **Signing in to github.com...** in the status bar.
4. Paste the token and hit **enter**.

This service is needed to connect Visual Studio Code to GitHub and does not store any sensitive information.

0-1. GitHub authorise success with token

Select open if you get this:



Allow an extension to open this URI?

GitHub Authentication (vscode.github-authentication) wants to open a URI:

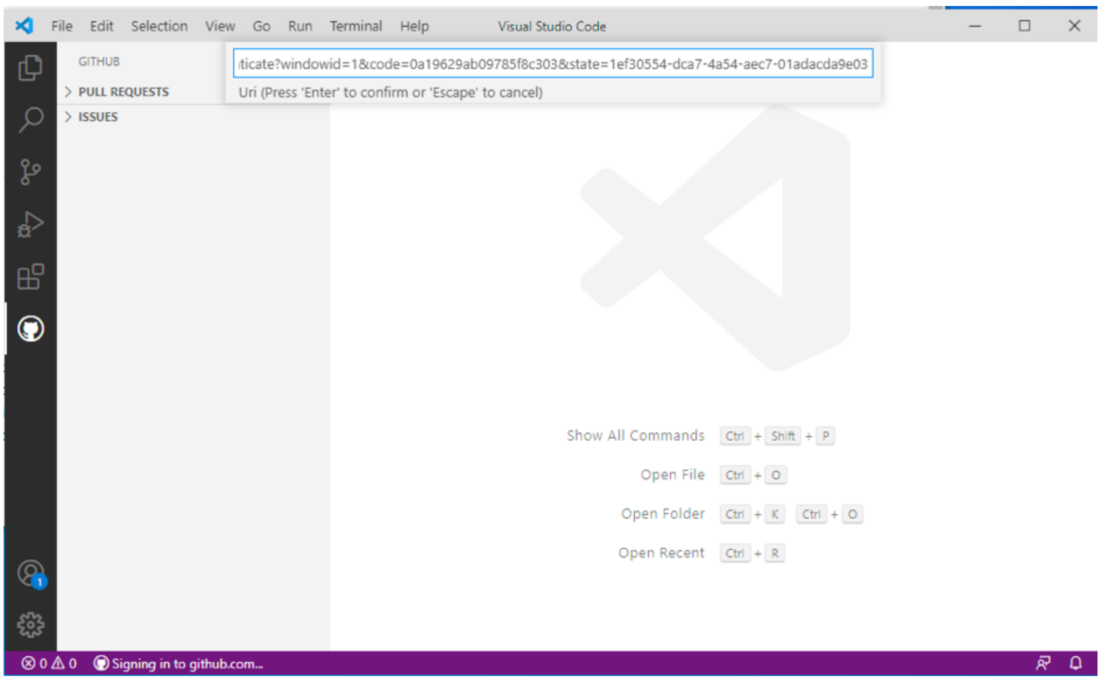
vscode://vscode.github-authent...a9e03

☑ Don't ask again for this extension.

0-1. VSCode allow

The process automatically generated the token which you need to copy before closing this screen. Follow the 4 steps show in the image above:

1. Copy the token
2. Switch back to VS Code
3. Click `Signining in to github.com...` in the status bar
4. Paste the token and hit enter


File Edit Selection View Go Run Terminal Help
Visual Studio Code
GITHUB
PULL REQUESTS
ISSUES
Show All Commands Ctrl + Shift + P
Open File Ctrl + O
Open Folder Ctrl + K Ctrl + O
Open Recent Ctrl + R

ticate?windowid=1&code=0a19629ab09785f8c303&state=1ef30554-dca7-4a54-aec7-01adacda9e03

Uri (Press 'Enter' to confirm or 'Escape' to cancel)

0 0 Signing in to github.com...

0-1. VSCode signing in to github.com status bar message

VS Code

Consider adding the GitHub respositories (https://marketplace.visualstudio.com/items?itemName=github.remotehub) extension to VS Code.

The VS Code documentation tells you to install the GitHub Pull Requests and Issues extension. This is optional. Pull requests are a mechanism when working with others to tell them about changes you pushed to a branch so others can review your changes before the branch is merged to the main branch. Working with branches is not covered in the course.

Activity 8 (optional): Using Copilot Pro in VS Code or PyCharm

Skip if you already completed this, or you do not wish to use and integrate Copilot.

To qualify for free Copilot Pro you need to be a verified student in GitHub Education (https://docs.github.com/en/copilot/how-tos/manage-your-account/get-free-access-to-copilot-pro).

To activate:

1. Verify GitHub Education status (student or teacher).
2. While signed in to GitHub, go to Copilot features (https://github.com/settings/copilot/features?) and enable Copilot Pro on the account.
3. Install the Copilot extension in your editor (VS Code recommended) and sign in when prompted.
4. Turn on inline suggestions and open Copilot Chat. Try a first prompt you can grade: "Write a failing unit test for function X and explain the assertion."

Use the GitHub 10-minute beginner video (https://www.youtube.com/watch?v=n0NlxUyA7FI&list=PL0lo9MOBetEFcp4SCWinBdpml9B2U25-f&index=8) to get up and running in VS Code.

Watch the full How to use GitHub Copilot (beginner's guide) (https://mailing.github.com/ls/click?upn=u001.bO3G-2FaeMIVe5lSdMnGODo3-2FQ6ALJoZReZPpMLK6cEoV4-2BxSVc46D7Oa-2BPtwZVTFS7k8XEIjVYNy
i_IsYeSBG-2BZ-2BTZ5It1-2BmoDsiwlumO5avdKROrFh8rWFtNzEx7-2B9y0qlyx3YZn-2BFBzGdolIkP-2Fq7i8tNoQGW9
for more. In particular understand how to write effective prompts for code assistance (https://youtu.be/LAF-lACf2QY?si=9HLcbIfv1dKNpvDl).