HyperionDev

Installation, Sharing, and Collaboration

Additional Reading

Visit our website

Introduction

This guide will introduce you to the command line and help you install your coding environment. It also gives steps for what to do if the installation fails.

If you want to install extra third-party packages, you should know the best practices for installation, like using package managers and containerization. This guide contains information on both.

Finally, this guide contains information on using Google Colab and spaCy.

You may find it necessary to refer back to seconds of this document throughout your bootcamp.

Please note that the instructions in this guide may address different courses. Make sure to follow the steps that are specific to your course.

Table of contents

Screen sharing in Google Meet

Google Meet is the tool we use for video calls between students and mentors.

You will often need to share your screen with a mentor during a Google Meet call. For example, you might need help with installation if something goes wrong. You could also show your coding environment to a mentor. This way, you can get help debugging or running your code.

To share your screen in Meet on a Windows computer using the Chrome browser, follow these steps:

1. Join a Meet video meeting.
2. At the bottom, select "Present now ⬆."
3. Choose "Your entire screen," "A window," or "A tab."
    a. If you present a Chrome tab, it will share the tab's audio by default.
    b. To present a different tab, select the tab you want and click "Share this tab instead."
4. Select "Share."

⚠ Sharing your screen in macOS

When using a Mac, sharing your screen through Google Meet in Chrome is more complex. You will need to enable certain permissions before sharing. Follow **this guide** on how to screen share in macOS.

Set up your development environment

Now onto the exciting part! Once you have set up your development environment, you are ready to start programming.

You will use Visual Studio Code (**VS Code**) as your integrated development environment (**IDE**) to open all text files (**.txt**) and Python files (**.py** – used in data science and software engineering) or JavaScript files (**.js** – used if you are a web development student).

There are a range of other editors that you can use such as vi, emacs, Notepad++, and PyCharm, but we cannot guarantee that your peers will be familiar enough with them to assist you with them or that our mentors will be able to consistently review your work, therefore, we recommend you stick to VS Code.

Take note

Please follow the steps below carefully. These instructions are for Windows, macOS, and Linux versions that are still supported by their developers.

If your operating system is not supported because it is too old, or if you are using an unsupported operating system like ChromeOS, you will need to use **Replit** as your IDE, in which case, you can skip the following setup instructions.

If you are a data science student, you will also need to use **Google Colab** when you work with Jupyter notebooks later in your bootcamp.

if you encounter any errors with the instructions below, consider using **Replit** as an alternative.

Installing VS Code

1. Navigate to **Visual Studio Code.**

2. Download the version of VS Code that matches your operating system (OS). Alternatively, you can follow the instructions for your specific OS by navigating to one of the following links.

   a. **Running Visual Studio Code on Windows**

   b. **Running Visual Studio Code on macOS**

Installing Python

Unix-like operating systems such as macOS and Linux often come with a pre-installed version of Python. It is generally discouraged to use the distributions of Python that are shipped with macOS, as they may either be outdated or have customisations that might give you issues further down the line. Please follow the guidelines at **Get Started Tutorial for Python in Visual Studio Code** and note the following:

- If you are on **Windows** or **macOS**, ensure that you have installed the latest stable version of Python using as described in the tutorial.

- If you are on **Linux** and the prepackaged Python version is not the latest stable version, ensure that you get a package provider for your operating system that uses the latest stable version. Being behind by 1 or 2 minor versions is fine on operating systems such as Fedora. However, on operating systems such as Ubuntu, we strongly recommend that you use a Personal Package Archive (PPA), such as **"deadsnakes" team PPA**.

  NOTE: Please avoid the flatpak or snap versions as they give you troubleshooting problems. Only proceed with flatpaks if you are sure of how they work.

  Per the guidelines linked above, ensure that you install the latest stable version of Microsoft's Python extension available from **Python - Visual Studio Marketplace**, so that you get tooltips and other useful tooling that help you as you program.

If you run into any trouble, submit a query via your dashboard for assistance.

Installing Node

There are several ways to install **Node.js** depending on your operating system, but we recommend using **Node Version Manager** (NVM). NVM offers greater flexibility, allowing you to easily install and switch between different Node.js versions. Additionally, it provides a consistent installation process across different operating systems.

For Linux/WSL and MacOS users

Before we get started with installing NVM, please take note that for this example we are installing a specific version of NVM. However the current version might differ, to get updated installation instructions based on the current version NVM, please refer **here**. Otherwise please continue following along with the instructions below:

1. Open your terminal
2. Install NVM by running the following command:

```
curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
```

3. Restart the terminal, by closing and reopening the terminal. This will make the NVM command available within the terminal.
4. Verify that NVM has been installed by running the following command:

```
nvm --version
```

   ○ You should see a NVM version number displayed.
5. After verifying that NVM has been installed, we can continue on to install Node.js using NVM, this can be done by entering the following command within your terminal:

```
nvm install node
```

   ○ This will install the latest version of node, for more information, please refer to the documentation **here**.

For Windows users

To install NVM on Windows, we will be using **NVM for Windows** (NVM4W), a tool which has been specifically designed for managing Node.js versions on Windows. This allows you to enjoy the same benefits of NVM such as easily installing and switching between different Node.js versions on your Windows machine.

Before we get started with installing NVM4W, please note that we will be installing NVM4W using the **latest installer** (you can also refer **here** to read more about installing NVM4W):

1. To start with, download the "nvm-setup.exe" by clicking on the "nvm-setup.exe" file for the latest version of NVM4W listed in the **releases** page.
2. Proceed to install NVM4W by running the downloaded "nvm-setup.exe" installer. Follow the prompts that appear on your screen to complete the installation.

HyperionDev

3. After successfully installing NVM4W, open a new command line instance and verify that NVM4W has been successfully installed using the below command.

```
nvm --version
```

- You should see a NVM4W version number displayed.

4. Now that the installation of NVM4W has been verified we can proceed with using NVM4W to install Node.js:

```
nvm install node
```

- This will install the latest version of node, for more information, please refer to the documentation **here**.

5. After installing the latest version of Node.js with NVM, you need to set it as the active version. To do this, use the following command:

```
nvm use node
```

Verifying Node.js installation

Now that Node.js has been installed using NVM or NVM for Windows, you should verify that it has been successfully installed and activated. Before running the following commands, you may need to restart your command line interface by closing and reopening it.

To verify that a specific version of Node.js has been activated, use the following command:

```
node --version
```

Using Node Version Manager (NVM)

One of the benefits of using NVM to install Node.js is that it enables you to install and switch between different versions of Node.js. To switch to a specific version of Node.js, the following command can be used within the command line:

```
nvm use <version>
```

The `<version>` should be replaced with a specific version which you would like to use. However, you will need to first install the specified version of Node.js before attempting to switch Node.js versions. This can be done using the following steps:

1. Start by installing a specific version of Node.js.

    ```
    nvm install 20
    ```

2. Switch to the newly installed Node.js version.

    ```
    nvm use 20
    ```

3. Verify that the switched version of Node.js is in use.

    ```
    node --version
    ```

    -   The displayed version of node should now have been updated.

To check which versions of Node.js are installed and available for you to switch to, use the following command:

```
nvm ls
```

VS Code for different users

Python users

You can visit the **Python overview** page to learn how to use VS Code with Python. If you've never programmed before, we strongly recommend that you watch the **introductory videos**. If you have a particular problem, **Stack Overflow** has a number of posts about VS Code.

JavaScript users

You can visit the **JavaScript overview** page to learn how to use VS Code with JavaScript. If you've never programmed before, we strongly recommend that you watch the **introductory videos**. If you have a particular problem, **Stack Overflow** has a number of posts about VS Code.

 Take note

Opt-out telemetry in VS Code refers to the practice of automatically collecting usage data from users of the software, and sharing it with the VS Code developers to help them identify bugs, improve performance and make other changes that will benefit users.

If you have concerns about your privacy, data, and usage habits, please turn it off by following the **instructions provided on the VS Code webpage**.

# Using Live Share for collaborative coding

What is Live Share?

On our coding bootcamps, you are encouraged to use VS Code as your integrated development environment (IDE). Installing the VS Code Live Share extension is an extremely useful thing, as it enables you to work collaboratively on code with mentors, peers, or lecturers. It is quick and easy to install, and will accept either a GitHub account or a Microsoft account as login credentials. Once installed, the user can create a link to share their live coding instance with one or more other people, enabling them to both

work on the same VS Code instance, edit the same code, etc. On GitHub, students can open a VS Code environment from inside any repository by simply pressing the full-stop/period key. Then Live Share can be easily installed from the extensions option on the leftmost vertical menu.

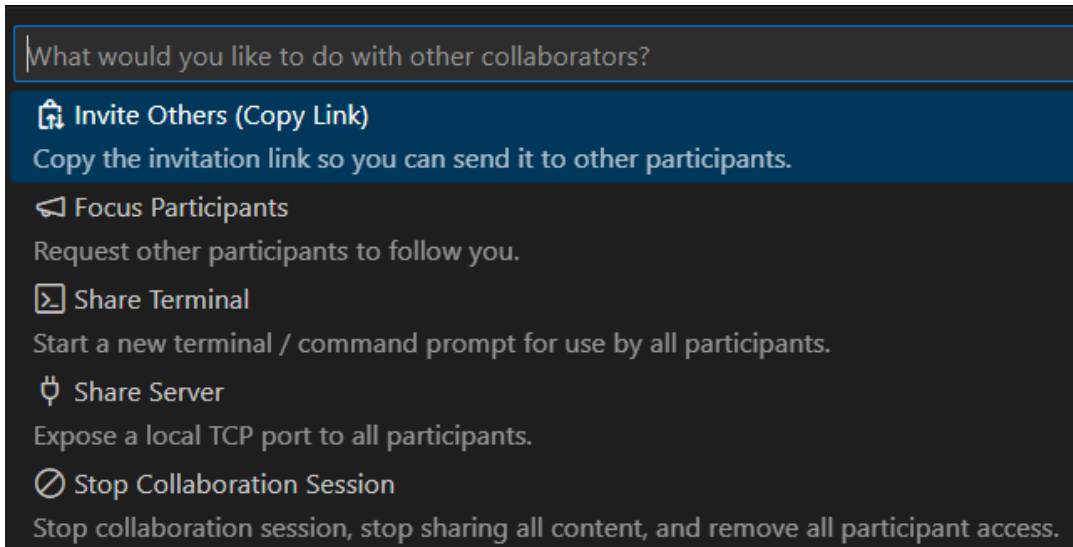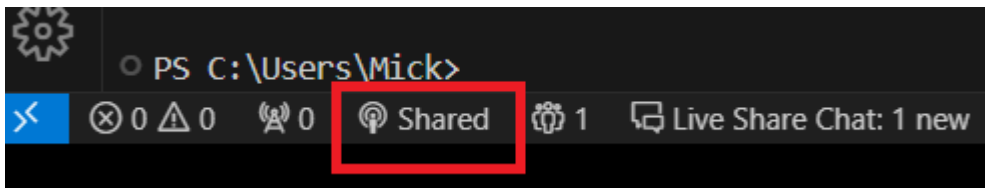## Installing and using Live Share on your device

1. Navigate to the **Live Share page**, which includes installation instructions and a download link.

2. Download and install the extension as per the instructions. Once complete, you'll see Live Share appear in your status bar.

3. You can now begin collaborating with others immediately. Once an invited collaborator joins, you'll get a notification, something like the one below.
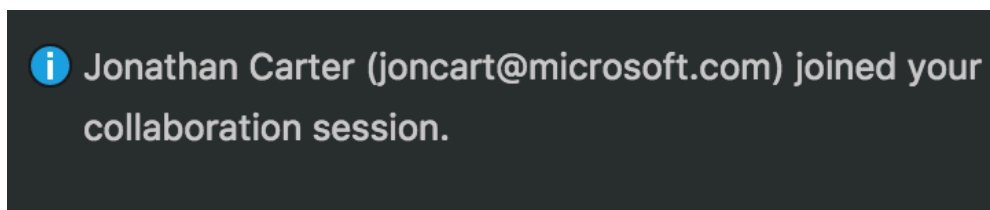

4. You will see the new collaborator's cursor appear in your currently opened file. All collaborators will start out 'following' you, which means that as you scroll or navigate between files, they will track those actions as well. This makes it easy to orient them with the issue/question/task you're about to start collaborating on.

5. Notice that a menu option for Live Share has appeared on the bottom left of your VS Code environment? You can chat with collaborators by selecting the Live Share Chat:
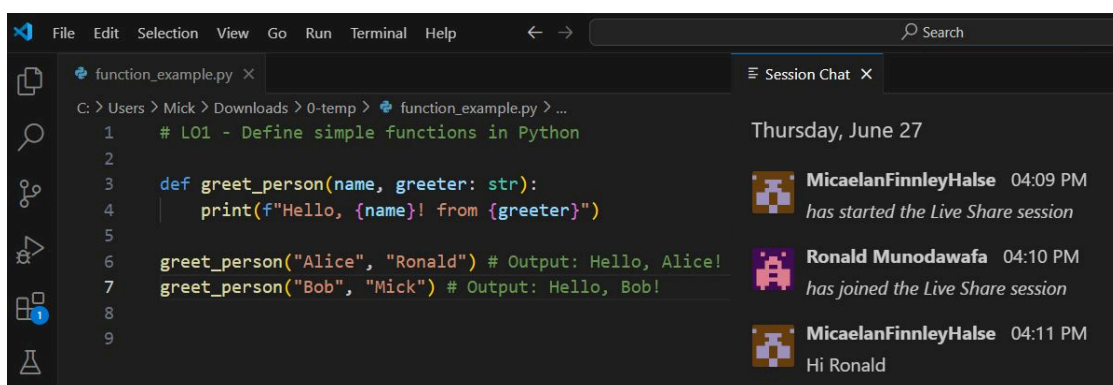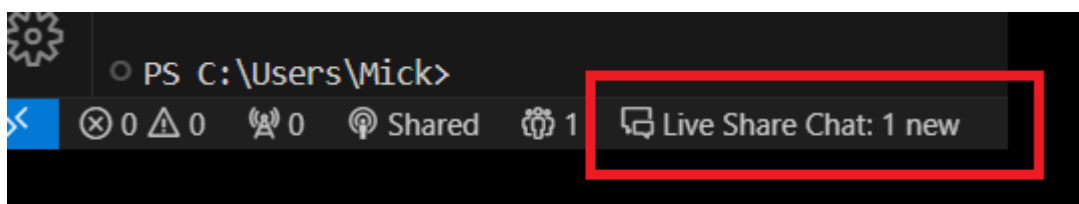
6. You can also access a menu of further collaboration options by clicking on the "Shared" option.


7. Any number of collaborators can edit the code.
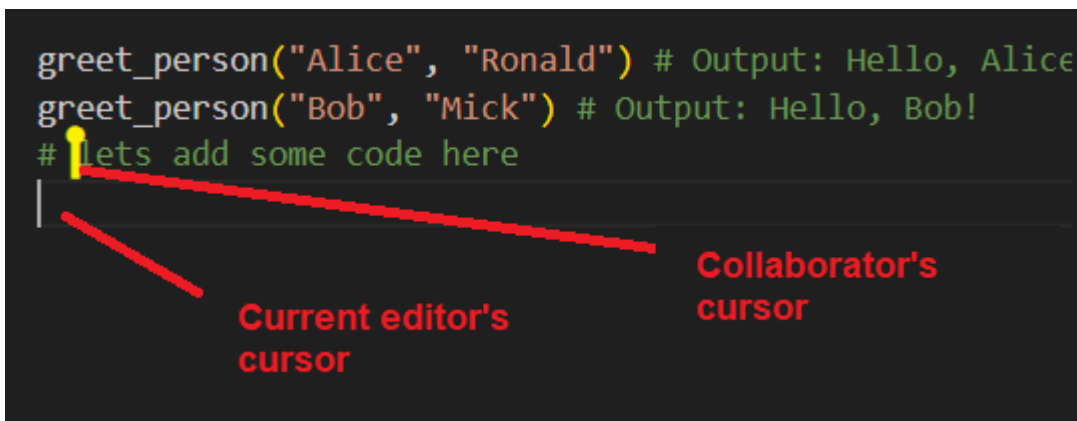
Installing and using Live Share on GitHub

1. Login to GitHub with your own credentials. If you still need to get a GitHub account, create one first.

2. Once logged in, navigate to **hyperiondev-com · GitHub**.

3. Enter a public repository, such as debugging practice, by clicking on the hyperlinked title (in light blue in the picture below).
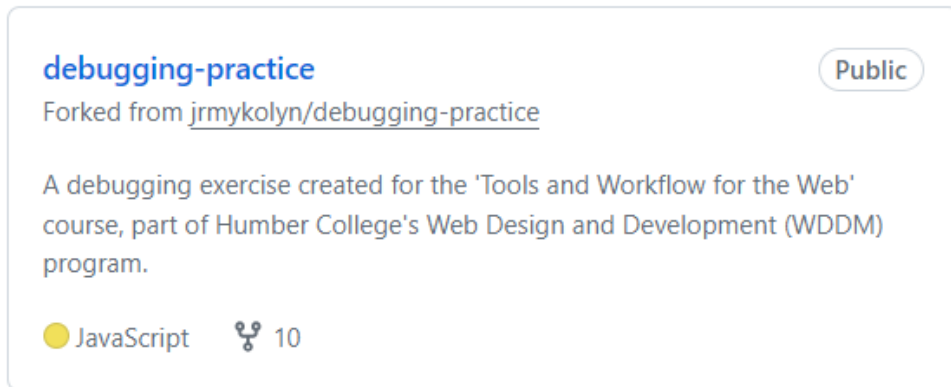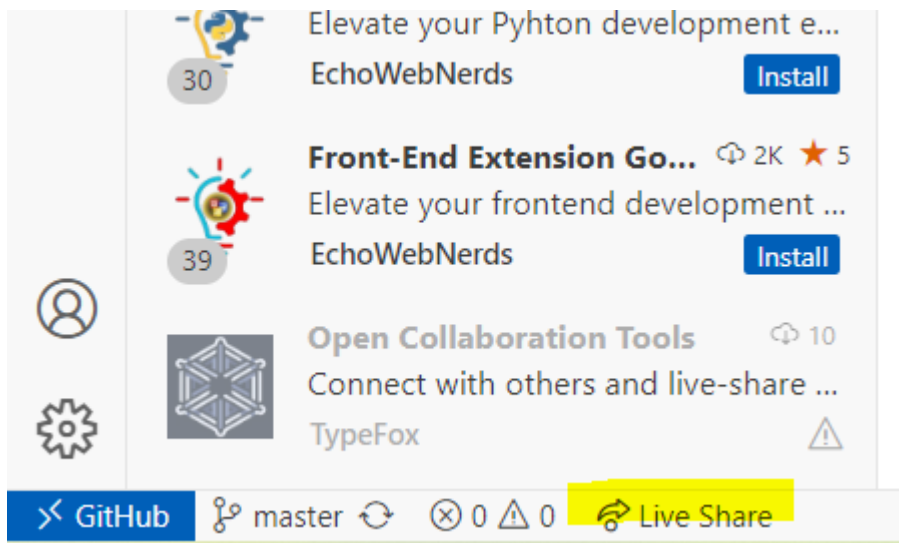

4. Press the full-stop/period key on your keyboard to launch VS Code in-browser from GitHub. Select the "Extensions" options from the leftmost vertical menu bar.
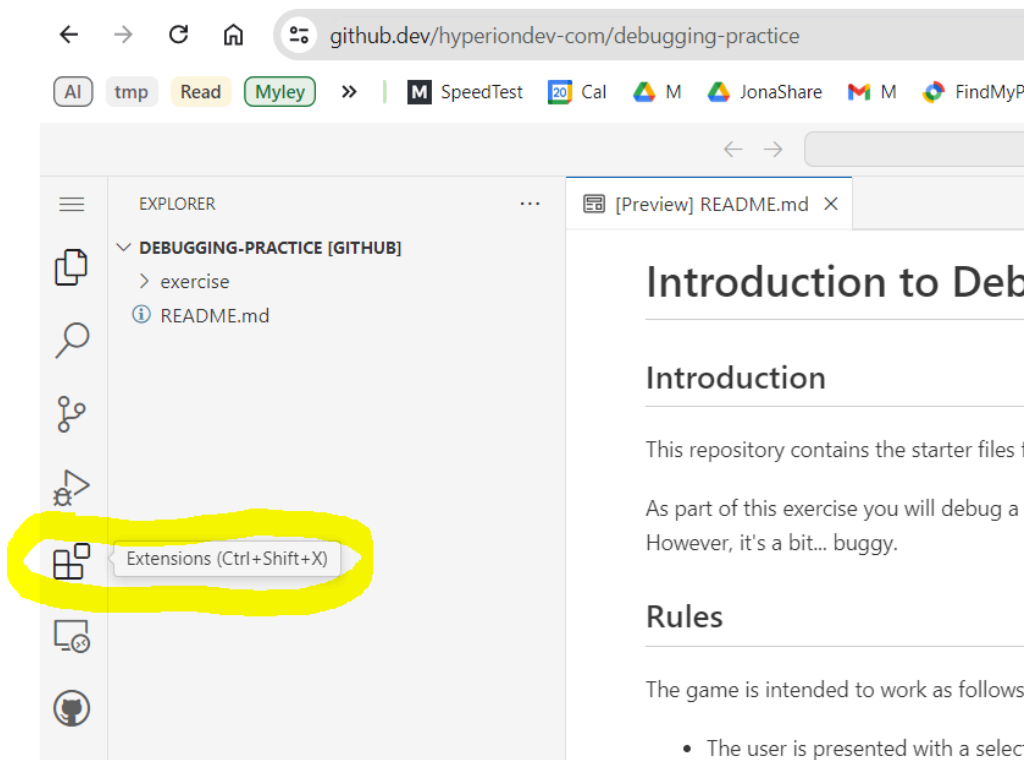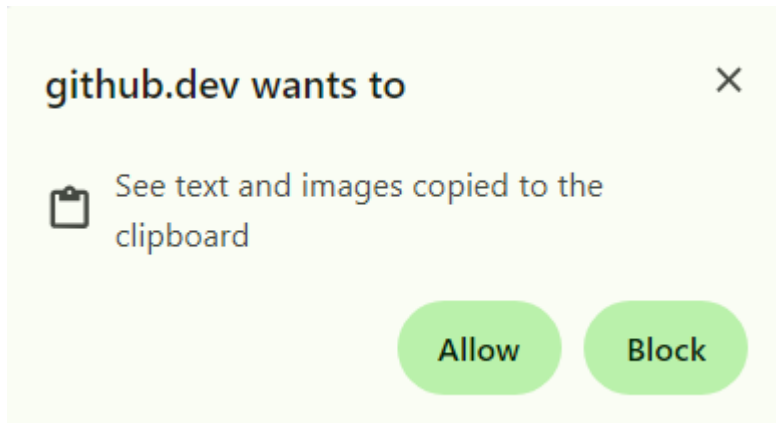
5. Under "Extensions", search for Live Share, and when it pops up, click on the "Install" option.
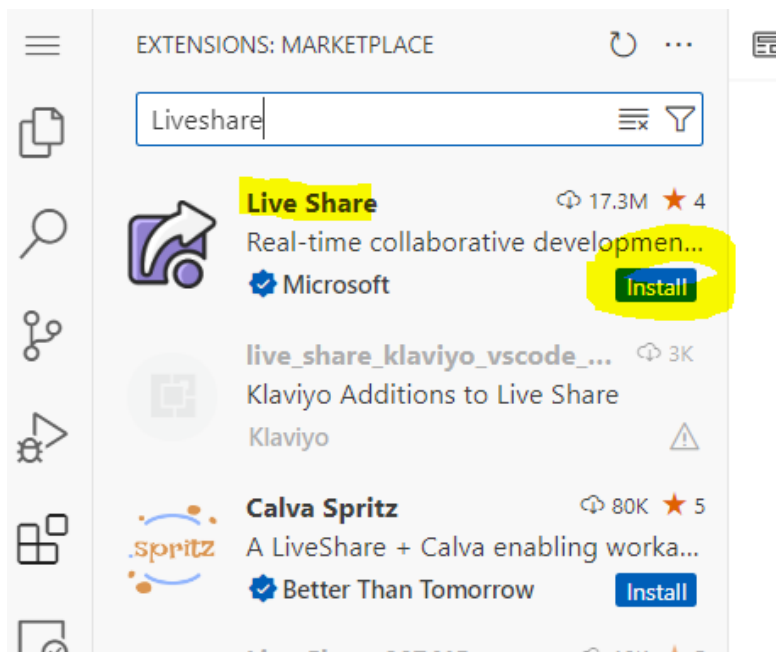
6. You will notice that Live Share becomes a menu option at the very bottom of the left-hand menu panel.

7. Allow the extension the permissions it requests.

8. You'll see the following message pop up on the bottom right-hand side of your screen. Your sharelink has been automatically copied to your clipboard and you can paste it to anyone you want to allow collaborative access to your code.
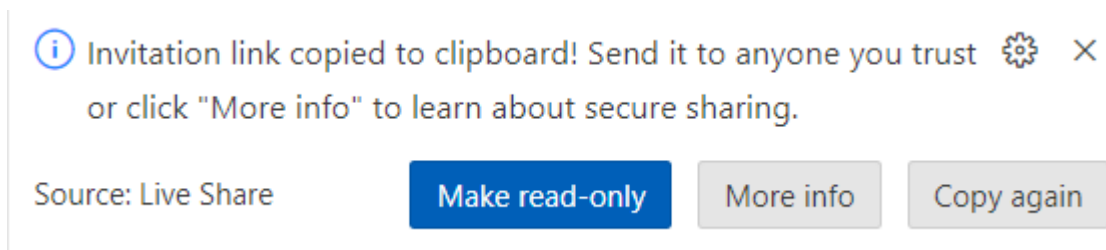
9. The recipient will see the below message, giving them the option to access your coding environment from their browser (useful from, for example, a tablet), or to open their own installation of VS Code on their main device such as a laptop.
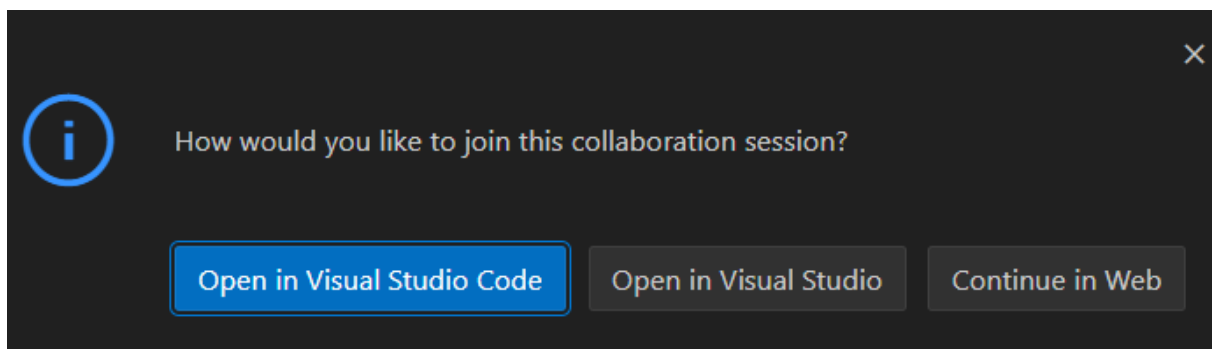
10. They can choose to sign in or continue as an anonymous user.
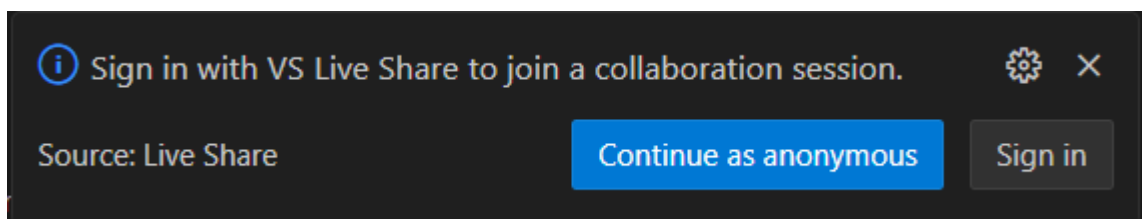
11. You may need to approve them joining if they select anonymous; it is generally a better idea for them to join as a signed-in GitHub user.

12. As the sharer, your view will now be something like this – the highlighted line is a text that has just been added:
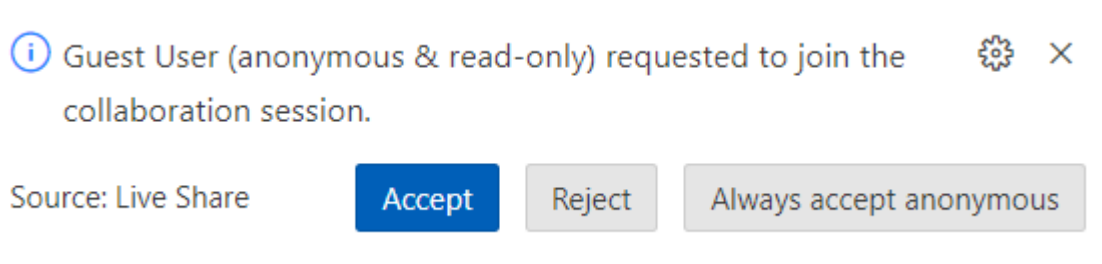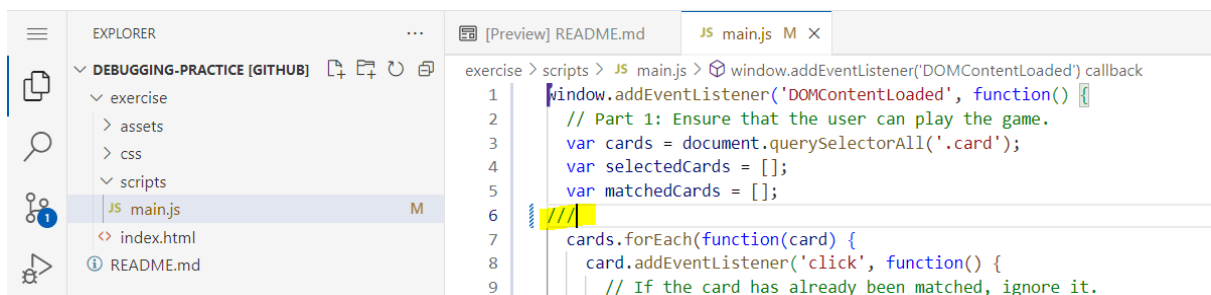

13. The person you've shared with will see something like the picture below. In this picture, the original sharer (you, when you are following these instructions) is "MickFinnHalse" and their cursor can be seen after the text that has just been added.
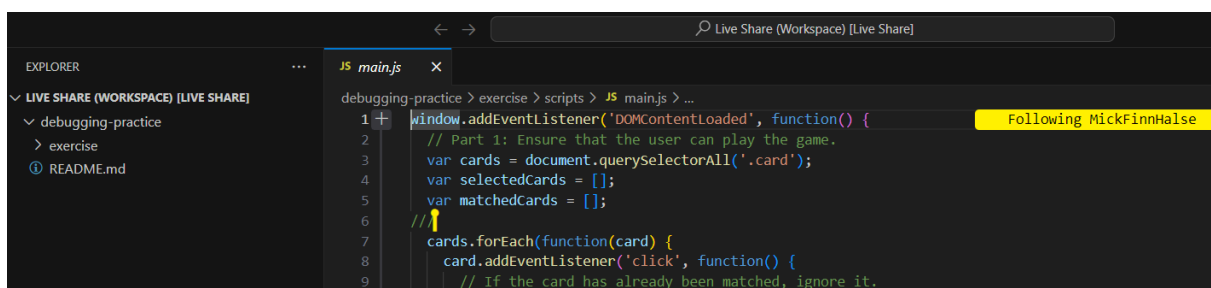
14. The person you've shared with can also annotate the code. Here, they have added a comment saying "Hello".
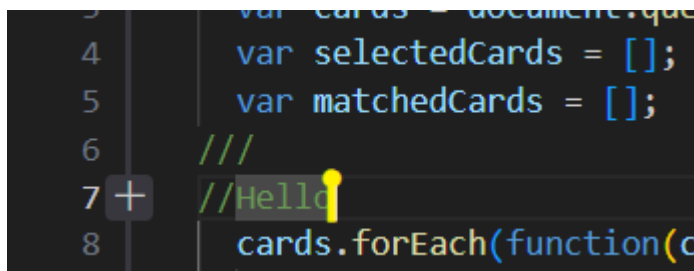
15. This will appear on your GitHub VS Code instance like this:

Installing Express

Previously, you will have installed Node.js and learned about NPM (the package manager for JavaScript). You will now install Express using NPM.

NPM is an extremely important tool for working with Node applications. It is used to fetch any packages (JavaScript libraries) that an application needs for development, testing, and/or production. It can also be used to run tests and Node modules used in the development process. Express is another package you need to install using NPM:

1. Open your terminal or command prompt.

2. Create a directory using the `mkdir` command and then navigate into it using the `cd` command.
   ```
   mkdir myapp
   cd myapp
   ```

3. Use the `npm init` command to create a package.json file for your application. We manage dependencies using a plain-text definition file named **package.json**. All the dependencies for a specific JavaScript package are listed in this file. The package.json file should contain everything NPM needs to fetch and run your application. The command `npm init` will prompt you to enter several details, such as the package name, version, description, entry point, test command, etc. Please note: `npm init` creates the initial `package.json` file. Please do that now:
   ```
   npm init
   ```

4. To display the package.json file, enter `cat package.json` or `type package.json` (depending on the operating system you are using) into your terminal. You can also view the package.json file in Visual Studio Code.

5. We will now install the Express library in the 'myapp' directory that you created, and save it in the dependencies list of the package.json file. Do this by entering the following command into the terminal or command prompt:
    `npm install express`

6. Enter `cat package.json` or `type package.json` into your terminal again. The dependencies section of your package.json will now appear in the **package.json** file and will contain Express.

```
cat package.json
{
  "name": "myapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.15.4"
  }
}
```

Package Managers

Package managers provide a means to install packages, organise them in your file system, and manage dependencies between packages. A dependency is when a package depends on another particular version of another package to function. If you have two different packages that rely on different versions of another package, this creates problems if you don't have a package manager. It can also be very time-consuming to find and install all the dependencies for a package. Package managers also check for known vulnerabilities that may pose a security risk. You can use a package manager to share packages you have created with others.

There are two types of package managers: those for **operating systems** and those for **programming languages**. Package managers are linked to software repositories, where the packages or software are stored.

Operating system package managers

Some common operating system package managers include:

- **Chocolatey** for Windows (**Chocolatey repository**)
- **Homebrew** for macOS (**Homebrew repository**)
- Linux – each distribution has its own package manager:
    - **DNF** for Fedora
    - **APT** for Ubuntu

Programming language package managers

As mentioned, programming languages typically have language-specific package managers.

The package manager you use depends on your preferences and the project's specific needs. Some key considerations to keep in mind are:

- **Project requirements**: ensure that the package manager you choose supports the necessary libraries and dependencies.
- **Community support**: look for an active and supportive community to provide resources and troubleshooting assistance.
- **Performance and reliability**: assess performance factors like speed and consistency versus comprehensive package registry management – especially for large, complex projects.
- **Compatibility**: consider compatibility to ensure that it integrates seamlessly with your tools and workflows.
- **Security**: research the security measures and best practices followed by the package manager to safeguard your project from potential vulnerabilities.
- **Package versions**: check whether the package manager is frequently updated to ensure your project benefits from bug fixes and new features.
- **Documentation**: always review the documentation and ease of use for a smooth development process.

Examples of some programming languages and commonly used package managers are summarised in the table below:

Language	Package manager	Software repository
Python	`pip`	**PyPI** (Python Package Index)
Python	`conda`	**Anaconda Packages**
JavaScript	`npm`	**Node Package Manager**
JavaScript	`yarn`	**Yarn Registry**
Java	`Maven`	**Maven Central**
Java	`Gradle`	**Maven Central**

Python

`pip` is the general-purpose package manager for Python. It is used to install, upgrade, and manage Python packages from the PyPI repository.

JavaScript

`npm` is the de facto package manager for JavaScript and Node.js projects, providing a vast array of packages, libraries, and tools for front-end and back-end development. However, for projects that emphasise faster and more reliable package installation, `yarn` is a good alternative. Built on top of `npm`, `yarn` is known for its enhanced performance, parallel installations, and caching capabilities.

Java

maven is the primary package manager for Java projects. Among its vast collection of libraries and plugins, it excels in managing project dependencies, build automation, and project documentation. For projects that prioritise a build tool with flexibility and performance, `Gradle` is a good alternative. It offers powerful build customisation options and enjoys widespread usage in app development.

Containerisation

As you start working on software applications and larger projects that need to be deployed, you will also need to consider how to bundle your code with all the third-party packages it relies on.

Python

For Python projects, the venv module is a well-known containerisation method. The **venv module** allows you to create isolated virtual environments for Python projects, each with its own Python interpreter and package dependencies.

JavaScript

The primary containerisation method for JavaScript applications is the Node.js npm module, which seamlessly integrates with Docker containers. Additionally, the nvm **Node Version Manager** is another popular way to install, upgrade, and manage different Node.js and npm releases on Unix, macOS, and Windows WSL platforms.

Docker is a widely adopted and versatile method of containerisation that supports multiple programming languages, making it a popular choice among developers across various technology stacks. It enables you to create lightweight and portable containers that encapsulate your application, simplifying the deployment and management process across different environments.

Google Colab

Google Colab is a cloud-based platform that allows you to write and execute Python code in a collaborative environment. It provides access to high-performance computing resources, including GPUs, and is ideal for data science projects, including those involving spaCy for natural language processing (NLP).

Accessing Google Colab

Open your web browser and navigate to **Google Colab**. Sign in to your Google account. If you don't have one, you can create a new account for free.

Creating a new notebook

Once you're signed in, click on the "New notebook" button in the "Open notebook" pop-up window to create a new Python notebook. Alternatively, navigate to "File" and click on "New notebook".

You'll be presented with a new notebook interface, where you can write and execute Python code.

Installing spaCy

To install spaCy in your Google Colab notebook, simply add an exclamation mark to the command you would usually use in your terminal as shown below:

```
!pip install spacy
```

Similarly, if you need to download spaCy models, also add an exclamation mark. For example, you can install the `en_core_web_sm` language model using the following command:

```
!python -m spacy download en_core_web_sm
```

Working with datasets

You can upload datasets directly to your Google Colab environment using the following code:

```
from google.colab import files
uploaded = files.upload()
```

This will allow you to select and upload dataset files from your local machine. Explore other methods to load and save data from external sources using the **colab documentation**.

Note: The `google.colab` module is specific to Google Colab notebooks and won't work when the notebook is run locally. To ensure that the reviewer is aware that the code should be reviewed on Google Colab, please include the following README file within your submission.

Include a separate file named "README" in your submission, where you mention the requirement to use Google Colab prominently. You can use any text editor to create your README text file. Your README file could include the following text:

```
**Note to Reviewer:**

This code is designed to be run on Google Colab. Please review the code and
execute it on Google Colab for accurate assessment.
```

Once the dataset is uploaded, you can read it into your notebook using pandas or another appropriate library:

```
import pandas as pd
df = pd.read_csv('your_dataset.csv')
```

Using spaCy

To use spaCy to perform various NLP tasks on your dataset, import spaCy and load the desired model as follows:

```
import spacy
nlp = spacy.load('en_core_web_sm')
```

Now, you can apply spaCy's functionalities to your dataset, such as tokenisation, named entity recognition, and sentiment analysis.

Saving your work

Google Colab automatically saves your notebook as you work on it. However, you can also manually save your work by clicking on "File" > "Save" or by using the keyboard shortcut Ctrl+S.

Sharing your Notebook

Download your Google Colab notebook as an `.ipynb` file to upload to your task folder for review. You can also share your notebook with others by clicking on "Share" in the top-right corner of the interface. You can share it with specific people or make it accessible to anyone with the link.