



**TASK**

# **Additional Reading - Installation and Screen Sharing**

Visit our website

## INTRODUCTION

### Installation

This guide will help you locate and use an installation script to set up your coding environment, and also provides instructions for what to do in rare cases where the installation script fails. If you would like to install additional third-party packages, you should be aware of best practices relating to installation. The use of a package manager is highly recommended, and containerisation is important when different projects or applications require specific requirements.

### Screen Sharing

You will often find it necessary to share your screen with a mentor via a Google Meet call, for example, to get help with installation if something goes wrong, or to share your coding environment to show a mentor the code you have written and get help debugging or running your code. A guide to screen sharing in the Chrome browser (recommended) on Windows and macOS is provided at the end of this document.

## SETTING UP YOUR DEVELOPMENT ENVIRONMENT

To **set up your development environment**, please follow the instructions on GitHub, selecting the link that matches your bootcamp from the list below:

- [Data Science](#)
- [Software Engineering](#)
- [Web Development](#)

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 either Python files (.py - used in Data Science and Software Engineering) or JavaScript files (.js - used in Web Development).



### 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](#).

## SETTING UP YOUR DEVELOPMENT ENVIRONMENT IF THE SCRIPT FAILS

If the above script does not work for your particular operating system, please systematically and carefully follow the steps just below. Remember to read the following instructions in full before you implement them:

1. Visit <https://code.visualstudio.com/>
2. Download the version of VS Code that matches your operating system (OS). Alternatively, you can follow the instructions stated at the following links for the corresponding operating system families:
  - a. macOS: <https://code.visualstudio.com/docs/setup/mac>
  - b. Linux: <https://code.visualstudio.com/docs/setup/linux>
  - c. Windows: <https://code.visualstudio.com/docs/setup/windows>
3. 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:

<https://code.visualstudio.com/docs/python/python-tutorial>.

- a. Ensure that if you are on Windows or macOS, you have installed the latest stable version of Python using the prescribed means on the link above.
- b. Ensure that if you are on Linux and the prepackaged Python version is not the latest stable version, 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 PPA (Personal Package Archive) such as <https://launchpad.net/~deadsnakes/+archive/ubuntu/ppa>.
  - i. 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.

- c. For all operating systems, ensure that your environment paths are up-to-date with regards to your installation.
  - d. Per the guidelines linked above, ensure that you install the latest stable version of Microsoft's Python extension available from <https://marketplace.visualstudio.com/items?itemName=ms-python.python> so that you get tooltips and other useful tooling that help you as you program.
4. Use the compulsory task to ensure that your setup is working as expected. If unsure, please check with an academic staff member or a peer.
  5. Use <https://code.visualstudio.com/docs/languages/python> to learn how to use Visual Studio Code with Python. If you've never programmed before, we strongly recommend you watch the videos.
  6. 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 the academic staff members will be able to consistently review your work.
  7. If you're concerned about opt-out telemetry with Visual Studio Code, please turn it off by using the instructions from:

[https://code.visualstudio.com/docs/getstarted/telemetry#\\_disable-telemetry-reporting](https://code.visualstudio.com/docs/getstarted/telemetry#_disable-telemetry-reporting).

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

### Notes for 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.

### Notes for JavaScript Users

You can visit the [JavaScript 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.

## 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.

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

As mentioned, programming languages typically have language-specific package managers. Examples of some programming languages and commonly used package managers are summarised in the table below:

Language	Package Manager	Software Repository
Python	pip	<a href="#"><u>PyPi</u></a> (Python Package Index)
Python	conda	<a href="#"><u>Anaconda Packages</u></a>
Java	Maven	<a href="#"><u>Maven Central</u></a>
Java	Gradle	<a href="#"><u>Maven Central</u></a>
JavaScript	npm	<a href="#"><u>Node Package Manager</u></a>
Javascript	yarn	<a href="#"><u>Yarn Registry</u></a>

### Python

**pip** is the general-purpose package manager for Python. It is used to install, upgrade, and manage Python packages from the PyPI repository. On the other hand, **conda** is more specialised towards data science programming, providing easy installation of both Python packages and other data science tools. This makes it an ideal choice for data-related tasks, scientific computing, machine learning, and other data-intensive projects.

## 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.

## JavaScript

**npm** is the de facto package manager for JavaScript and Node.js projects, providing a vast array of packages, libraries, and tools for frontend and backend 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.

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 vs 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.

## 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.

**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.

### 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. Additionally, Anaconda allows you to create specialised conda **virtual environments**, which is particularly useful for Data Science projects, to package and manage your projects efficiently.

### 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.

## THE COMMAND LINE

To install the packages you will need to become familiar with the command line.

The command line is a means of interacting with a computer program where the user issues commands to the program in the form of successive lines of text. With the command line, you can quickly issue instructions to your computer, getting it to do precisely what you want it to do. The command line is rarely used by most end users since the advent of the Graphical User Interface (a more visual way of interacting with a computer using items such as windows, icons, menus, etc.).

However, you will find it helpful to use the command line when installing third-party packages and other actions such as version control with Git.



### Take note:

You will need to ensure that you have **administrator access** on Windows or **superuser access** on macOS, Fedora or Ubuntu to install packages.

## Finding The Command Line Shell



On Windows, you can easily access the command line interface by clicking the Start menu and typing '**powershell**' (for Windows PowerShell) or '**cmd**' (for Windows Command Prompt) in the search box. Alternatively, you can find the command line interface under 'Programs' and open it by clicking the application.



With macOS, you can access the command line interface by opening the Applications folder, navigating to Utilities and then launching Terminal. Alternatively, you can search for '**terminal**' to find the application to launch.

## Common Commands

The following table provides a selection of commonly used PowerShell and Unix commands to help get you started with the command line.

For a comprehensive list of commands, visit [Powershell commands](#), [Command Prompt commands](#), and [Unix commands](#) to explore more command line operations.

Description	Windows cmd	Windows Powershell (alias)	Mac OS/Unix
Displays the current working directory	chdir	pwd	pwd
Changes the directory	cd	cd	cd
Move up one level in the directory	cd..	cd..	cd..
Displays a list of a directories	dir	dir	ls



files and subfolders			
Print contents of a text file	<code>type</code>	<code>type</code>	<code>cat</code>
Create a new directory	<code>mkdir</code>	<code>mkdir</code>	<code>mkdir</code>
Remove files and directories	<code>del / rmdir</code>	<code>del / rmdir</code>	<code>rm</code>
Move or rename files and directories	<code>move / ren / rd</code>	<code>move / ren</code>	<code>mv</code>
Copy files and directories	<code>copy</code>	<code>copy</code>	<code>cp</code>
Clear the screen	<code>cls</code>	<code>cls</code>	<code>clear</code>
Quit the terminal	<code>exit</code>	<code>exit</code>	<code>q</code>



## Code Hack

You can get detailed information, including instructions, examples and usage guidelines, on the various commands available in both Windows Powershell and Unix-based systems.

For **Windows Powershell**, open the terminal and run **Get-Help**. To get help on a specific command, run **Get-Help <cmd-name>**.

For **Windows Command Prompt**, open the terminal and run **help**. To get help on a specific command, run **help <cmd-name>**.

For **Unix-based systems**, open the terminal and run **man** to access the manual pages. To get help on a specific command, run **man <cmd-name>**. Additionally, you can also run the **whatIs <cmd-name>** to get a brief description of the specified command.


## SCREEN SHARING IN GOOGLE MEET

Google Meet is the standard tool we use to host video-call meetings between students and mentors. Note that the steps needed to share your screen in Meet differ between Windows and macOS.

### Windows

It's relatively simple to share your screen in Meet in the Chrome browser on a Windows machine - just follow these steps:

1. Join a Meet video meeting.

2. At the bottom, click Present now .
3. Select **Your entire screen, A window**, or **A tab**.
  - a. If you present a Chrome tab, it shares that tab's audio by default.
  - b. To present a different tab, select the tab you want to present and click **Share this tab instead**.
  - c. If you present a Slides presentation through a tab, you can [\*\*control it in Meet\*\*](#).
4. Click Share.

## macOS

To share your screen in Meet in Chrome is more complex when using a Mac. Here is a [\*\*guide to sharing your screen in a Meet call using Chrome on macOS\*\*](#).