Software: VS Code

This guide provides instructions for setting up **Visual Studio Code (VS Code)** for CEE6501.

VS Code will be our primary workspace for:

- opening and running **Jupyter notebooks** (`.ipynb`)
- editing Python scripts and course files
- working with the course GitHub repository

Repo Link: https://github.com/Bruun-Automation-Research-Lab/CEE6501

The goal is to get you to a point where you can open the course repo, select the correct Python environment, and run notebooks smoothly.

Learning Objectives

By the end of this guide, you will:

- Install VS Code
- Install the key VS Code extensions for Python + Jupyter
- Select the **CEE6501** Conda environment as your interpreter/kernel
- Run Jupyter notebooks inside VS Code
- Run Jupyter notebooks outside of VS Code (JupyterLab and Colab)

🔧 Install VS Code

Windows

- Download VS Code: https://code.visualstudio.com
- Run the installer (`.exe`)
- Recommended options during install:
    - ✅ Add to PATH
    - ✅ Register Code as an editor for supported file types
    - ✅ Add “Open with Code” to Explorer context menu

---

macOS

- Download VS Code: https://code.visualstudio.com
- Open the `.dmg` and drag **Visual Studio Code** into **Applications**
- (Recommended) Enable the `code` command:
    1. Open VS Code
    2. Press `Cmd + Shift + P`
    3. Type: **Shell Command: Install 'code' command in PATH**

📂 Open the Course Repository

Open the Repo Folder in VS Code

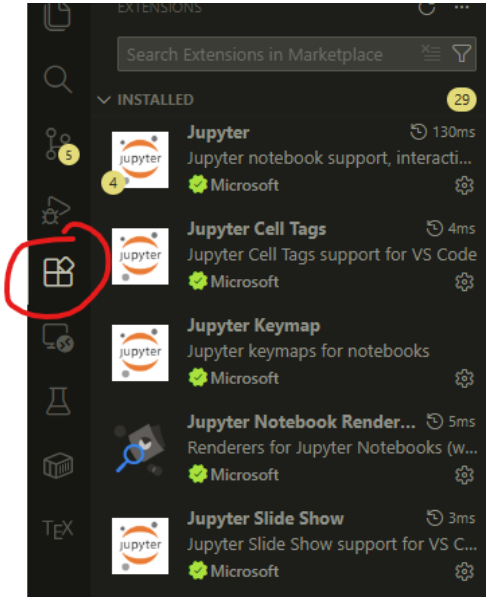
1. In VS Code, go to **File** → **Open Folder...**
2. Select the folder where you cloned the course repo:
    - Example: `.../Documents/GitHub/CEE6501`
3. You should now see the repository files in the Explorer sidebar.

```
Important: Always open the *repo root folder* (CEE6501), not an individual
notebook file.
```

Install Extensions

Extensions

Recommended extensions are provided in the `.vscode/extensions.json` file. Open the Extensions panel (left sidebar) and install all the recommended extensions:



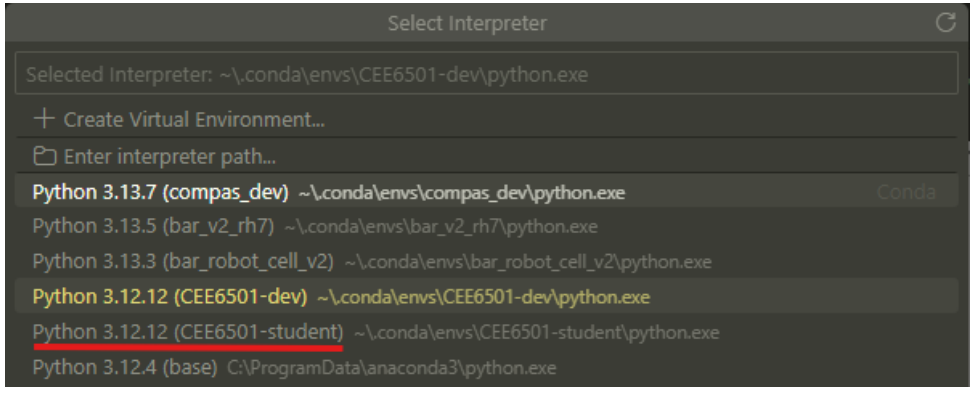
Note: After installing extensions, restart VS Code to make sure everything loads cleanly.

Select the Course Python Environment

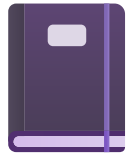
Select the Interpreter / Kernel

VS Code needs to know which Python environment to use.

1. Open the Command Palette:
    - **Windows:** `Ctrl + Shift + P`
    - **macOS:** `Cmd + Shift + P`
2. Run: **Python: Select Interpreter**
3. Choose the course environment: `CEE6501-student`

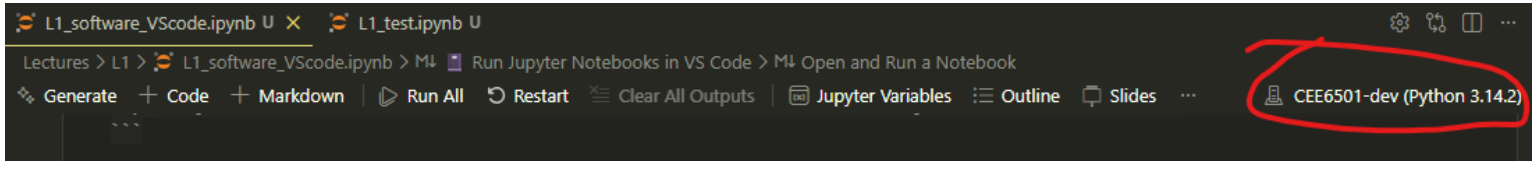


Note: If you don’t see the environment, make sure it was created successfully in the Conda tutorial.

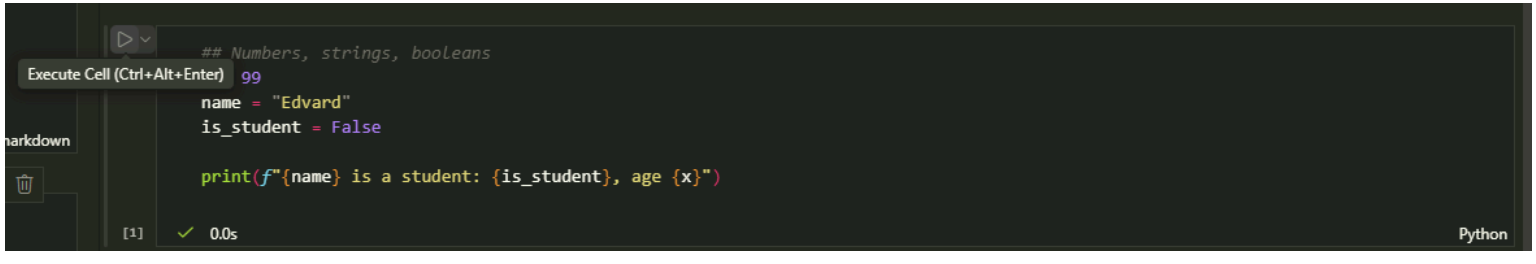
 Run Jupyter Notebooks in VS Code

Open and Run a Notebook

1. In the repo, open a notebook (example):
    - `Code/L1/L1_JupyterTest.ipynb`
2. At the top-right of the notebook, confirm the **kernel** matches your course environment



3. Run a cell using:
    - the ▶️ run button
    - or `Shift + Enter` when in the cell



Note: If the kernel prompts you to install something, you likely selected the wrong interpreter.

Run the rest of the cells in the notebook

In-class Practice

1. Write a function `cube(n)` that returns `n ** 3`.
2. call the function with a variable
3. print the results

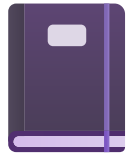
In [2]:
```python
def cube(n):
    return n ** 3

# Define a variable
value = 4

# Call the function and print the result
result = cube(value)
print(result)
```

```
64
```

Export From VS Code:

Click on the **"Export..."** button in the top-right to export as:

- PDF (requires TeX installed)
- HTML (easier, can then print to PDF)
- Python script (`.py`)



Run Jupyter Notebooks in JupyterLab

Open the Integrated Terminal in VS Code

In this section, you will launch **JupyterLab** using the **integrated terminal in VS Code**. The integrated terminal behaves exactly like a normal system terminal, but is embedded directly inside VS Code so you can work in one place.

- **Windows / macOS:** `View → Terminal`
- or use the shortcut: `Ctrl + ` ` (backtick)

A terminal panel will open at the bottom of the VS Code window. Anything you can do in a normal terminal can also be done here.

⚠️ Conda on Windows (Important)

On **Windows**, the default VS Code terminal uses `cmd.exe`, which does **not** recognize Conda commands unless it has been initialized.

To avoid this issue, the course repository includes a VS Code configuration (`.vscode/settings.json`) that defines a custom terminal profile which launches **Anaconda Prompt** inside VS Code.

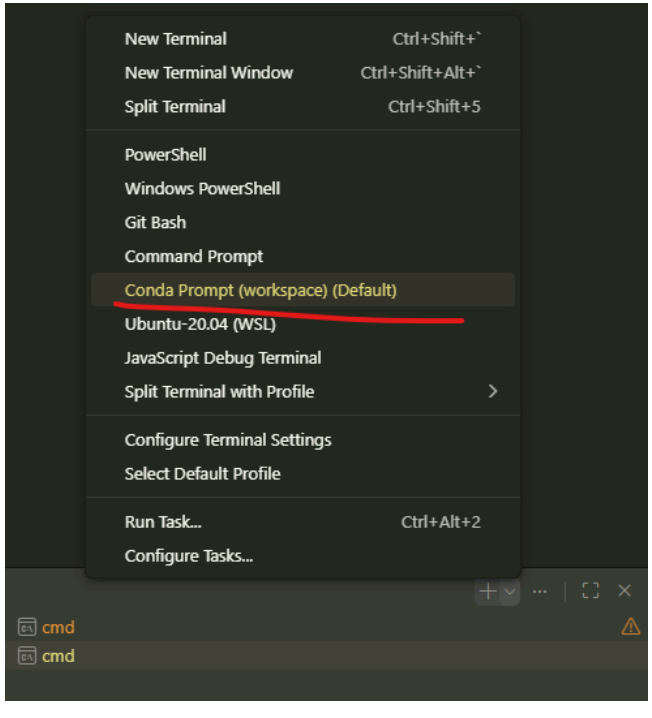
When this configuration is active:

- A terminal called **Conda Prompt (workspace)** will be available in VS Code
- This terminal automatically has access to Conda
- `conda activate CEE6501-student` will work as expected

What you should see

After opening the integrated terminal in VS Code, you should see **Conda Prompt (workspace)** as an available terminal profile.

If this terminal is available, you can safely run conda commands

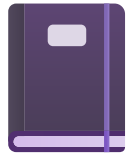


Launch JupyterLab

In the integrated terminal

```
jupyter lab
```

Opens the browser-based interface as in the example from the python install module

 Run Jupyter Notebooks in Google Colab

What is Google Colab?

Google Colab (Colaboratory) is a **cloud-based Jupyter notebook environment** that runs entirely in your web browser.

It allows you to run Python notebooks **without installing anything locally**.

- A hosted Jupyter notebook service provided by Google
- Runs on remote machines managed by Google
- Accessible from any device with a web browser
- **All you need is a Google account**

Colab is especially useful when:

- you cannot install software locally
- you are working on a different computer
- you want a quick, zero-setup way to run code
- share code quickly with collaborators

Why Use Colab?

Google Colab provides:

- A ready-to-use Python environment
- Built-in support for Jupyter notebooks
- Free access to computing resources (Can run code on GPUs and TPUs)
- Easy sharing and collaboration

For this course, Colab is a **convenient alternative** to running notebooks locally.

You will be submitting links to Colab notebooks for your coding assignments

Opening Course Notebooks in Colab

For **code-based lecture notebooks** and **coding assignments** links to Google Colab are provided directly in the course GitHub repository.

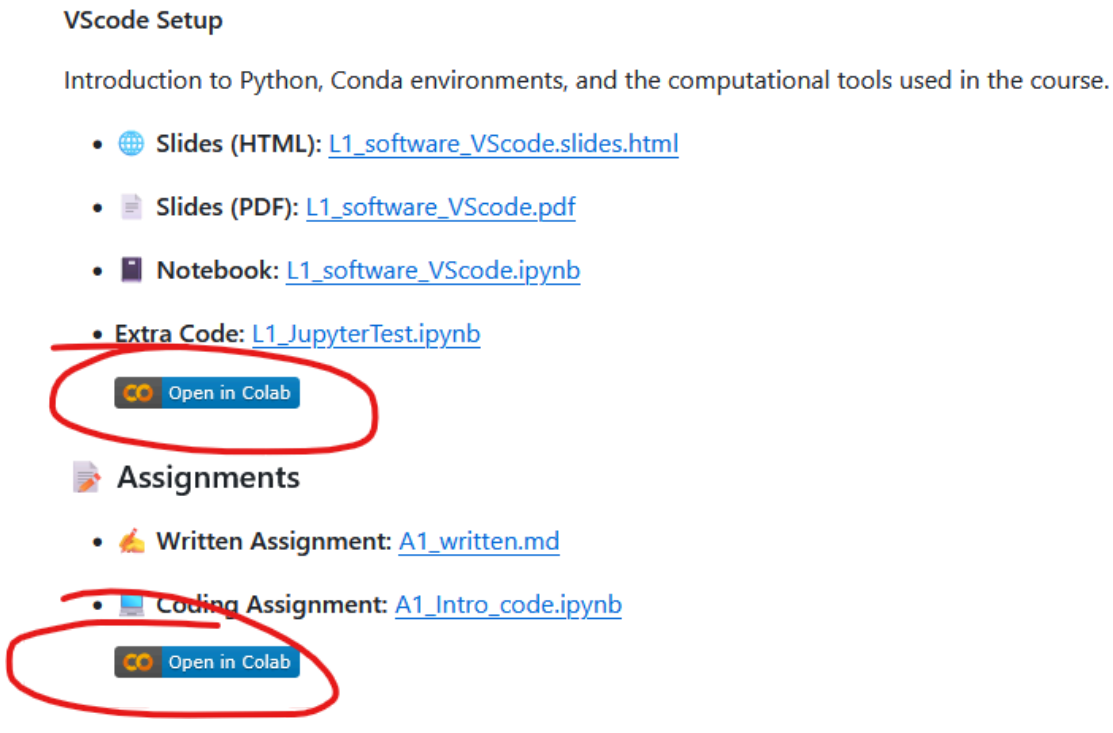
These links allow you to open a notebook in Colab with a single click.

You do **not** need to download the notebook to start working on it.

```
Note: Make sure to save a local copy of the notebook to avoid it being
overwritten.
```

Example: Colab Link in the Repository

Clicking the **Open in Colab** badge launches the notebook directly in Google Colab.



⚠️ Important: Package Environments Do Not Transfer

Google Colab runs notebooks in its **own Python environment**, which is **completely separate from Conda**. At the time of this course, Colab uses **Python 3.12.12**, which is why the course Conda environments are aligned to this version—even though newer Python releases may exist.

Because Colab does not have access to your local Conda environment:

- **Package versions do not automatically carry over**
- Packages may differ unless they are **explicitly installed inside the notebook**

Initializing the Colab Environment

To ensure consistent behavior across all students, code notebooks include a **Colab-specific setup cell** at the top that installs the required package versions when running in Google Colab.

- Make sure to run this cell when you **first open the notebook in Colab**
- You only need to run it **once per Colab runtime session**
- If you restart the runtime, you must run the setup cell again

When working in Colab, you should always run the notebook **from top to bottom** to ensure that all imports, variables, and package versions are initialized correctly.

📘 Lecture 1: Python + Jupyter Test

This in-class hands-on excercise introduces variables, data types, control flow, and functions.

```python
# --- Google Colab environment setup ---
# This cell runs only when the notebook is opened in Google Colab.
# Local Conda users will NOT be affected.

# Note: When running this notebook on Google Colab, required packages are installed automatically to ensure consistent versions across all students.

import sys
import subprocess

if "google.colab" in sys.modules:
    print("Running in Google Colab")
    print("Python version:", sys.version)
    print("Installing required packages...")

    # Single source of truth (kept verbatim)
    requirements = [
        "numpy==2.4.0",
        "scipy==1.16.3",
        "matplotlib==3.10.8",
        "pandas==2.3.3",
    ]

    # Remove duplicates while preserving order
    pip_requirements = list(dict.fromkeys(requirements))

    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "-q", *pip_requirements]
    )

else:
    print("Running outside Google Colab")
    print("Python version:", sys.version)

# --- Version check (always runs) ---
import numpy
import scipy
import matplotlib
import pandas
```

Local vs. Colab Workflows

- **VS Code + Conda**
  - Full control over environments
  - Best for long-term projects and assignments
- **Google Colab**
  - No local setup required
  - Best for quick experimentation and access from anywhere
  - Use to submit homework

Both workflows are supported in this course.

📘 Summary

You now have three ways to work with course notebooks and to write code:

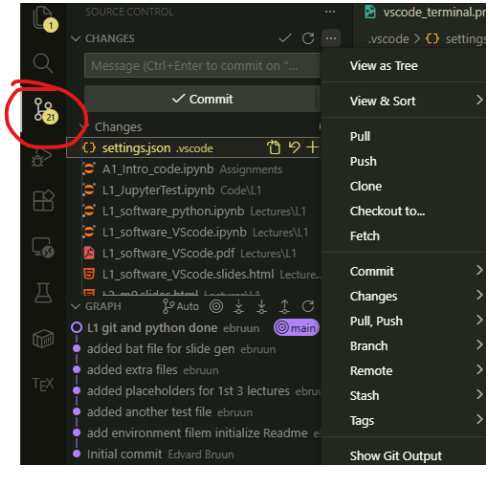
1. Locally in VS Code using Conda
2. In JupyterLab launched from your local environment
3. In Google Colab using the provided links

Basic Git in VS Code (Optional)

Viewing Changes

VS Code can show Git changes directly:

- Open the **Source Control** panel (branch icon)
- You can fetch and pull updates
- You can view modified files and diffs



Note: For this course, you will not push changes to the main repo. Treat the course repo as read-only and keep your own work in a separate folder.

⚠️ Important: Pulling Updates Will Overwrite Local Changes

When you **pull updates** from the course repository, Git will replace files in your local copy with the latest versions from the instructor.

This means:

- Any **local edits** you make to files inside the course repo (e.g. notebooks, scripts, notes) **may be lost**
- This repository is used to **distribute course materials only**
- You should treat it as **read-only**

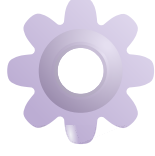
How to keep your own work safe

To avoid losing your work, use one of the following approaches:

- **Copy notebooks** you want to edit into a separate folder outside the course repo
- **Create your own repository** (recommended) and commit your work there
- Keep personal notes, experiments, and assignments **outside** the shared course repository

---

```
Note: Always assume that pulling updates will reset files in the course
repo. If you want to keep changes, save them somewhere else first.
```



VS Code Tasks (Optional)

Running Common Commands with One Shortcut

VS Code Tasks let you run repeatable commands without retyping them.

Examples (instructor workflows):

- Convert the currently-open notebook to Reveal.js slides
- Export the slides HTML to PDF

To run a task:

1. Open the Command Palette (`Ctrl/Cmd + Shift + P`)
2. Run: **Tasks: Run Task**
3. Select the desired task listed in `.vscode\tasks.json`

```
Note: Tasks are optional for students, but they are extremely useful for
keeping workflows consistent.
```

Troubleshooting

Common Issues

- **I can’t find the Conda environment in VS Code**
    - Restart VS Code
    - Re-run **Python: Select Interpreter**
    - Confirm the environment exists using `conda env list`
- **Notebook runs, but imports fail**
    - You likely selected the wrong kernel
    - Switch the kernel to `CEE6501-student`
- **VS Code asks to install Jupyter**
    - You are not in the course environment
    - Select the interpreter again and reopen the notebook

📘 What's Next

Now that VS Code is set up and connected to your course Python environment, you're ready to pull updates from GitHub and follow along with the lecture notebooks during and after class, as well as work on assignments.

From this point forward, VS Code will be your primary workspace for running notebooks, editing code, and organizing your course work throughout the semester.