DATASCI 350 - GitHub Tutorial

Danilo Freire

Contents

1 Introduction

This tutorial will guide students new to computer programming through the process of installing Visual Studio Code (VS Code) and connecting it with Anaconda. We will cover each step in detail to ensure a smooth setup process. If you have any questions or encounter issues during the installation, please feel free to ask for help from me (danilo.freire@emory.edu) or the teaching assistants.

2 Why VS Code?

VS Code is a popular, free code editor developed by Microsoft. According to the 2023 Stack Overflow Developer Survey, VS Code is the most preferred integrated development environment (IDE) among developers. Here are some reasons to choose VS Code for Python development:

- Extensive Features and Extensions: VS Code offers several built-in features such as syntax highlighting, IntelliSense, and debugging. It also features hundreds of extensions that can increase its functionality, such as GitHub Copilot.
- Cross-Platform Compatibility: VS Code is available for Windows, macOS, and Linux, so you can use it regardless of your operating system.
- Active Community and Support: The active community around VS Code means continuous updates and many community-driven extensions now and in the future.
- Integration with Git and Other Tools: VS Code seamlessly integrates with version control systems like Git (as Microsoft owns GitHub), making it easier to manage your code repositories and collaborate with others.

With that said, let us start the installation process.

3 Download and Install VS Code

- Please visit the official Visual Studio Code website: https://code.visualstudio.com/.

- Click on the “Download” button for your operating system (Windows, macOS, or Linux).

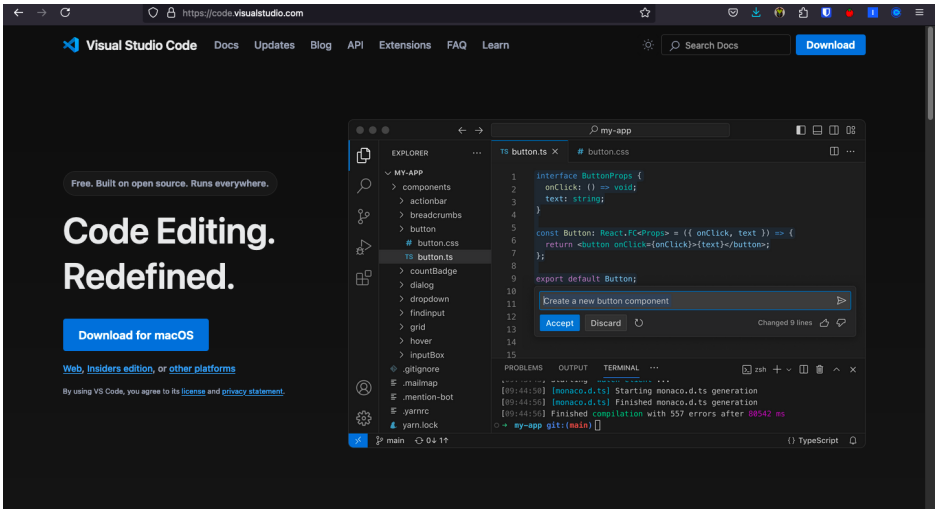

Figure 1: VS Code Website

3.1 For Windows Users

- Run the “VSCodeUserSetup-{version}.exe” file.

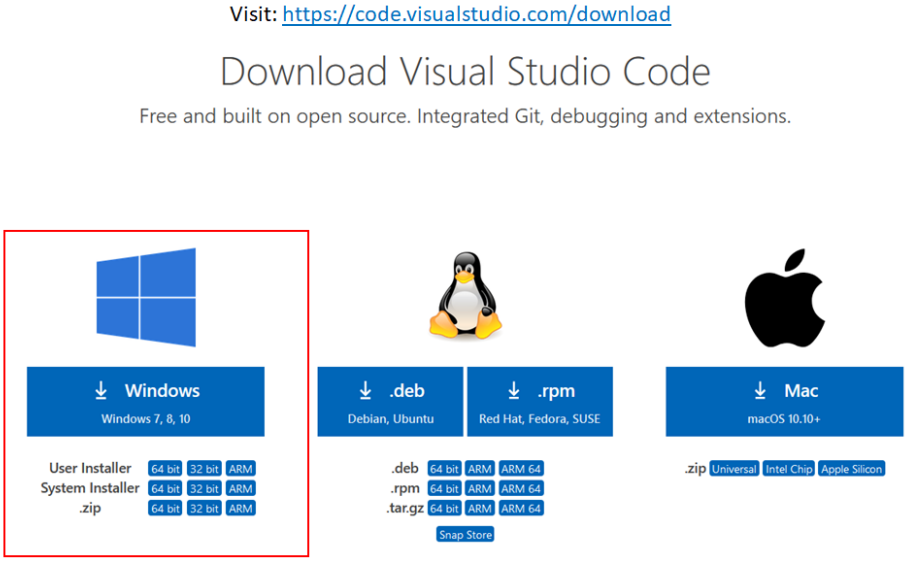

Figure 2: Windows Installation

- Accept the license agreement and click “Next”.

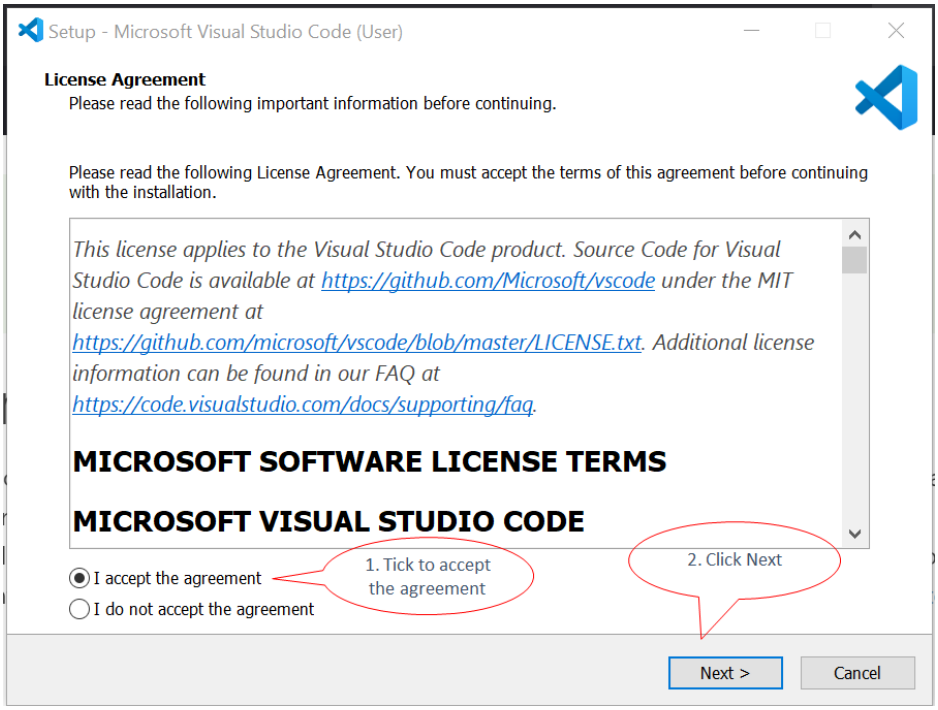

Figure 3: License Agreement

- Choose the installation location (default is recommended) and click “Next”.

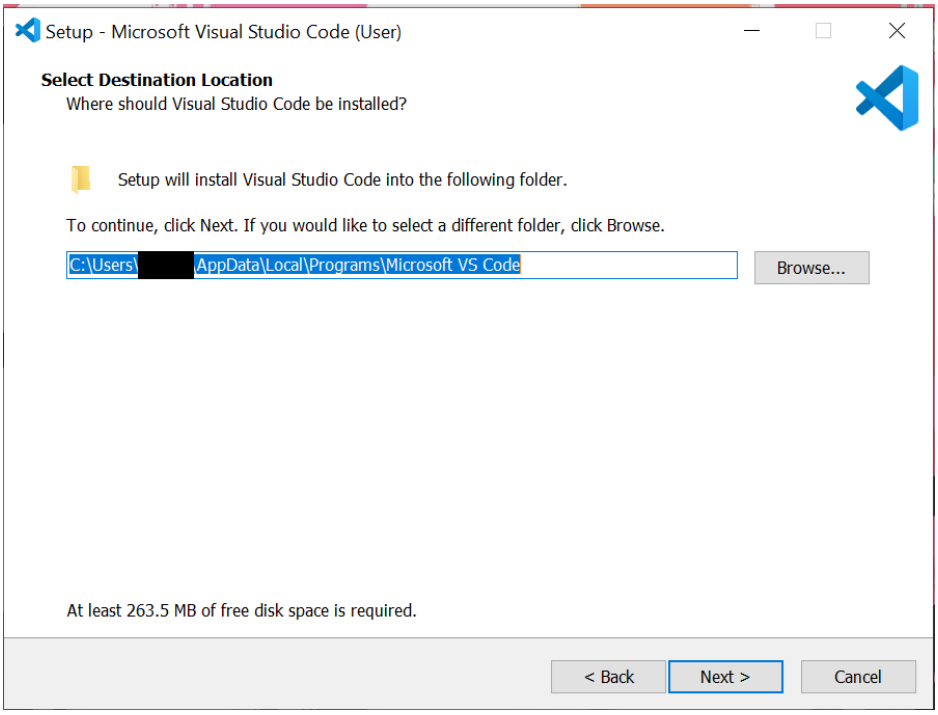

Figure 4: Installation Location

- Select additional tasks if desired (e.g., adding “Open with Code” action) and click “Next”.

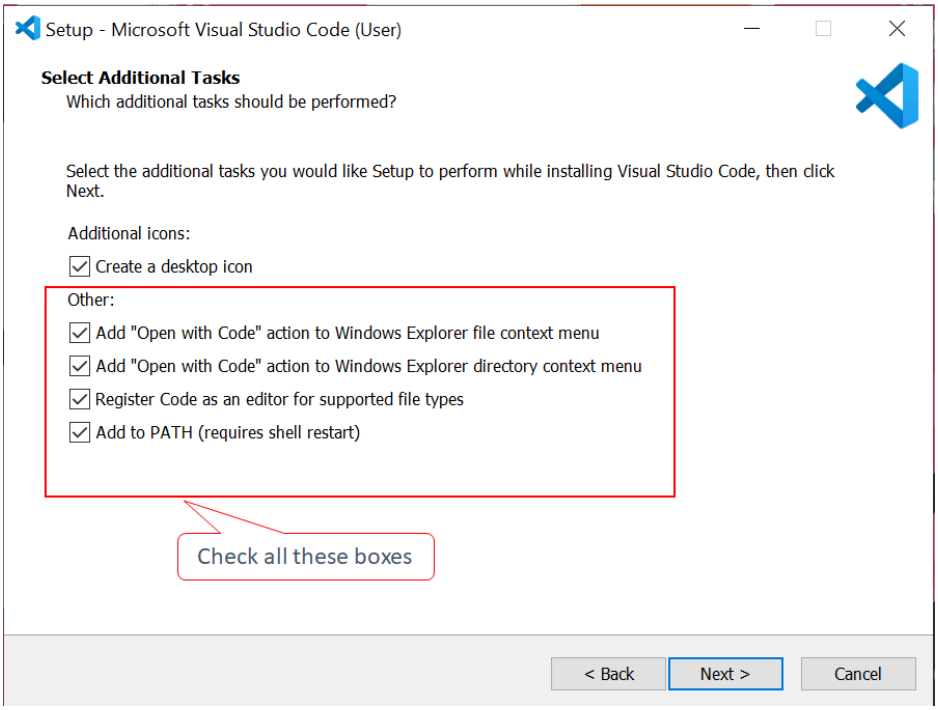

Figure 5: Additional Tasks

- Click “Install” to begin the installation process.

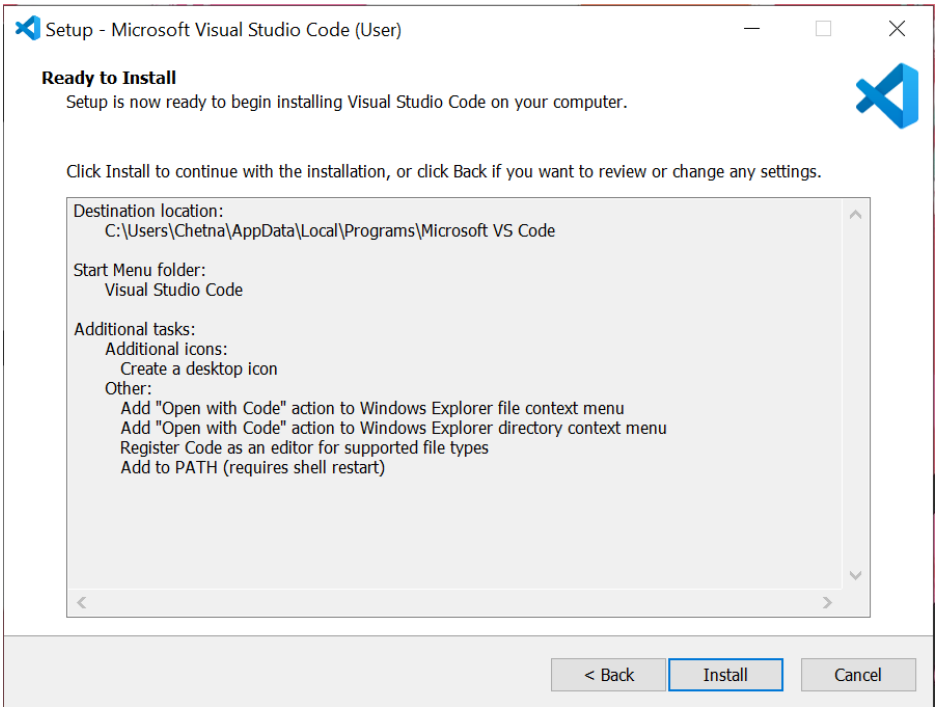

Figure 6: Installation Progress

3.2 For macOS Users

- Open the downloaded .zip file and drag the VS Code application to the Applications folder.

Figure 7: Unzip file

Figure 8: Drag to Applications folder

3.3 For Linux Users

- Follow the distribution-specific instructions provided on the VS Code website.

3.4 Launch VS Code

- After installation, launch VS Code.
- You should see the welcome screen with options to start a new project or open existing files.

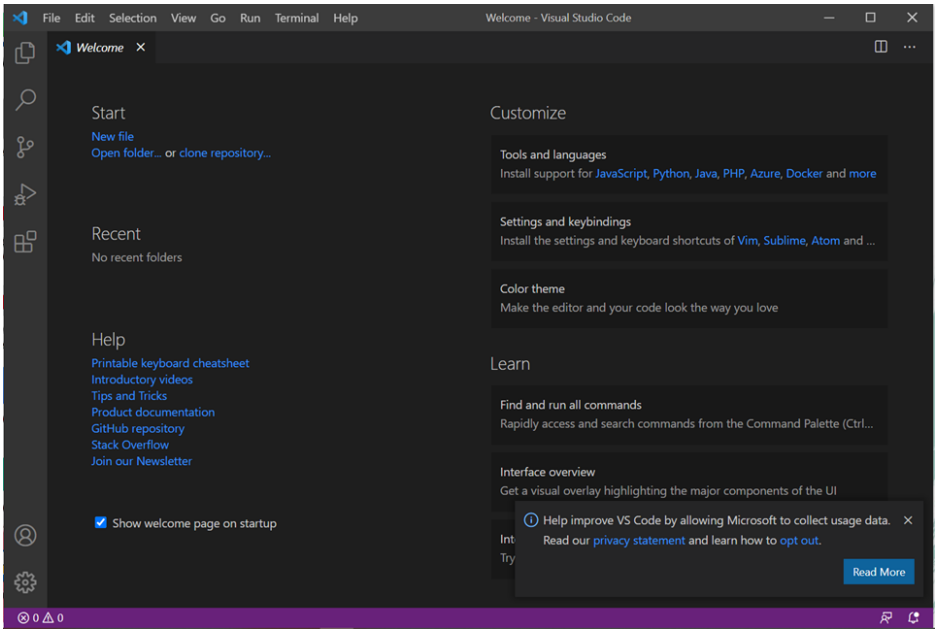

Figure 9: VS Code Welcome Screen

3.5 Getting Started

Microsoft has published a series of introductory videos to help you get started with VS Code. You can also read the official documentation for more detailed information.

4 Installing Anaconda

Anaconda is a distribution of Python that includes many popular packages for data science and scientific computing. It also provides a convenient way to manage different Python environments.

4.1 Download Anaconda

- Visit the Anaconda website: `https://www.anaconda.com/products/distribution`. You can skip the registration if you prefer.

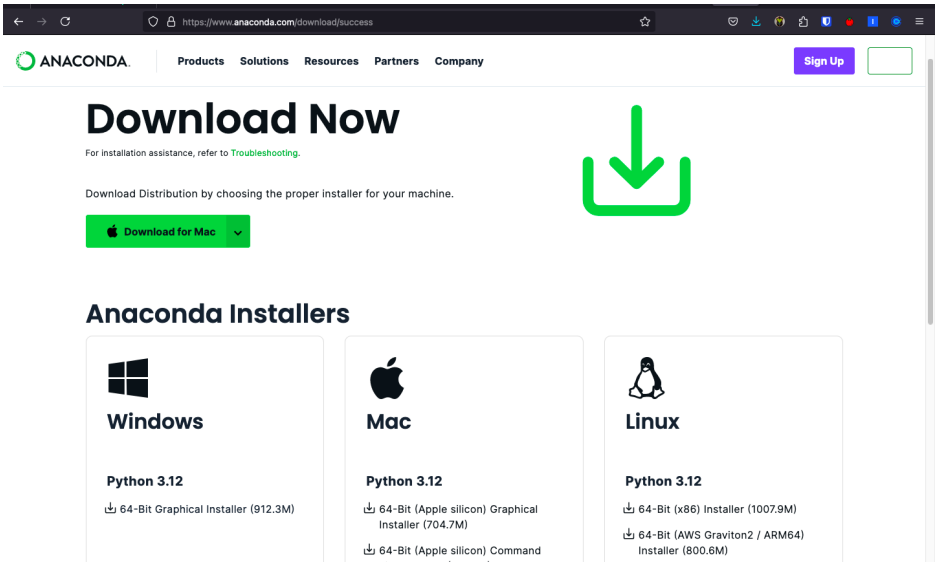

Figure 10: Anaconda Website

- Click on the “Download” button for your operating system.

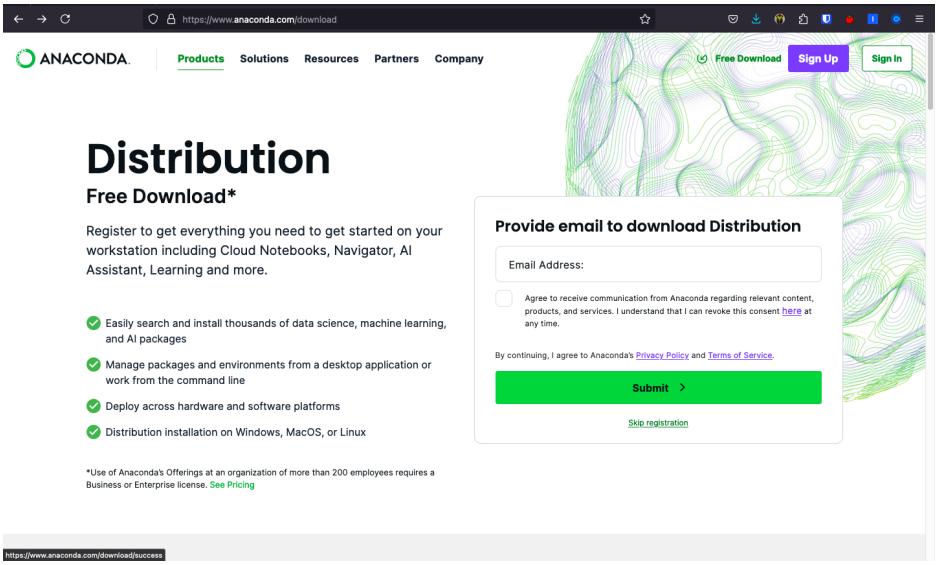

Figure 11: Download Anaconda

4.2 For Windows Users

- Choose “Just Me” for the installation type (recommended).

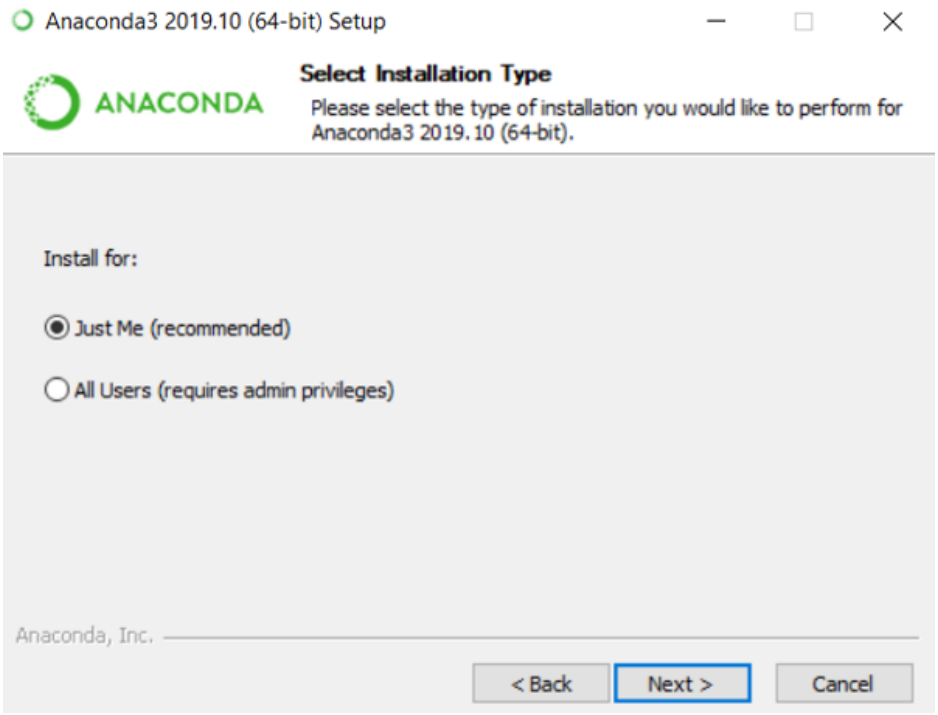

Figure 12: Anaconda Installation

- Select the installation location (default is fine).

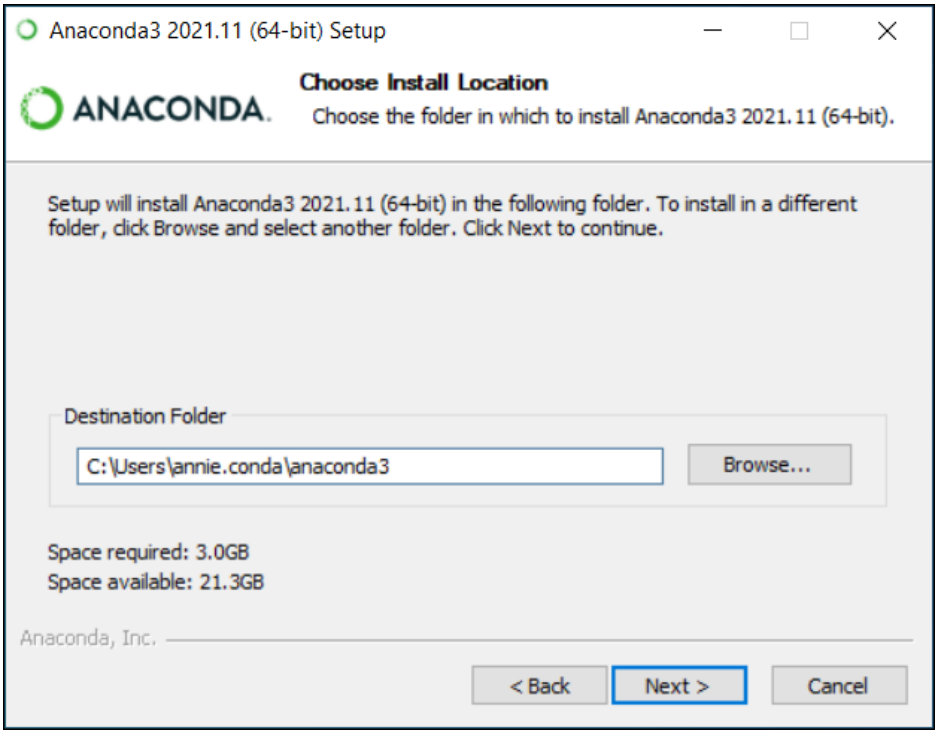

Figure 13: Anaconda Installation

- In the “Advanced Options” section, check “Add Anaconda to my PATH environment variable” and “Register Anaconda as my default Python”.

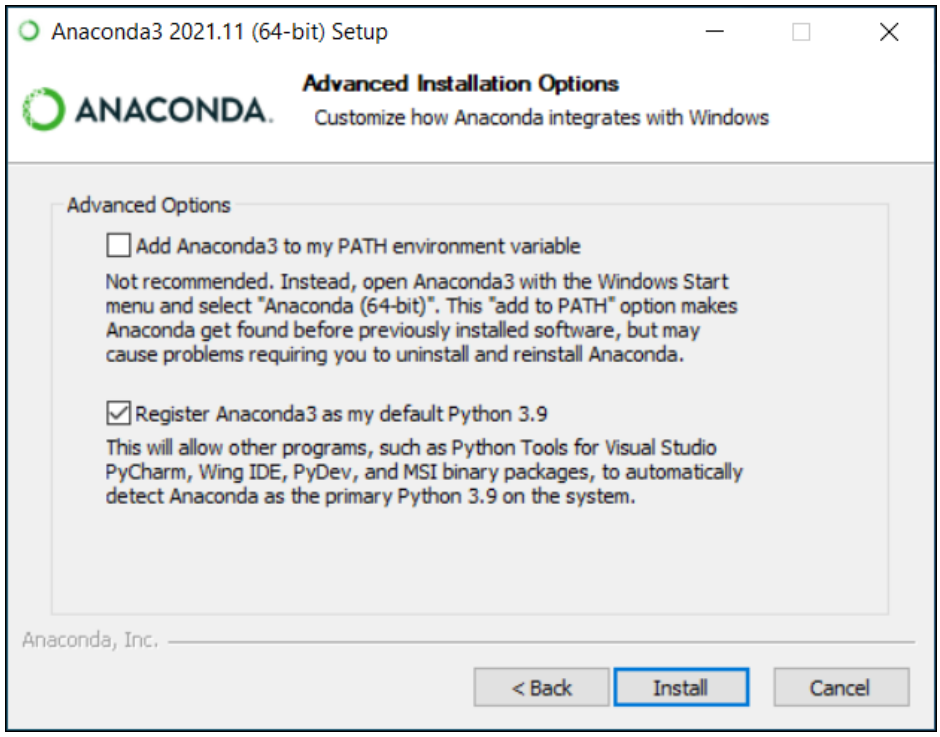

Figure 14: Advanced Options

4.3 For macOS Users

- Open the downloaded .pkg file and follow the installation instructions. You can install Anaconda for all users.

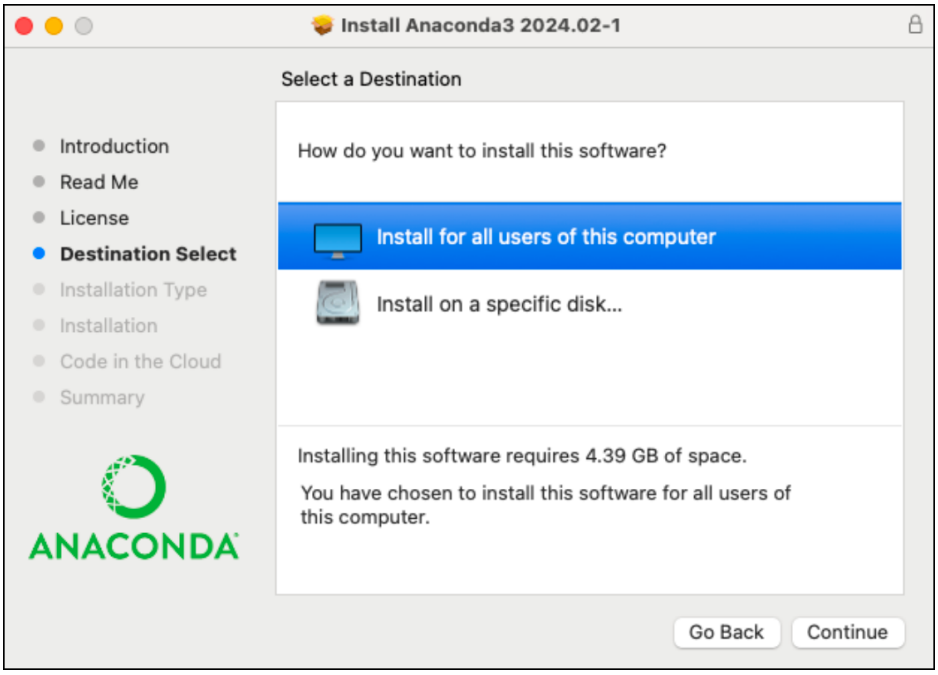

Figure 15: Anaconda Installation

- Follow the on-screen instructions, accepting the default options.

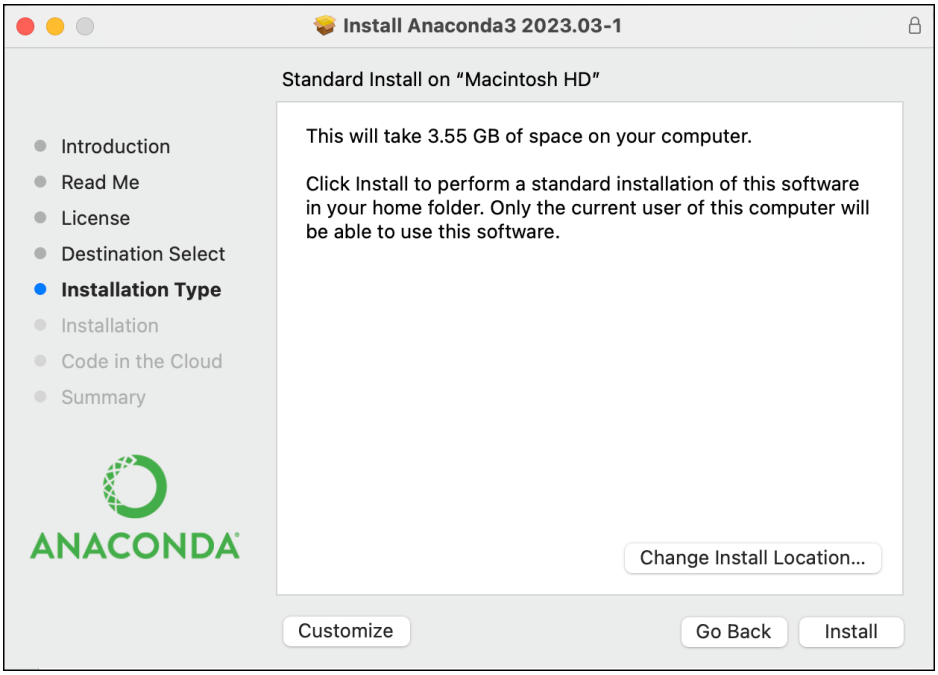

Figure 16: Anaconda Installation

4.4 For Linux Users

- Please follow the distribution-specific instructions provided on the Anaconda website.

4.5 Verify Anaconda Installation

- Open a new terminal or command prompt. On VS Code, you can do it by clicking on "Terminal" > "New Terminal" in the top menu.
- Type `conda --version` and press Enter. You should see the Conda version number.
- Type `python --version` and press Enter. You should see the Python version installed by Anaconda.

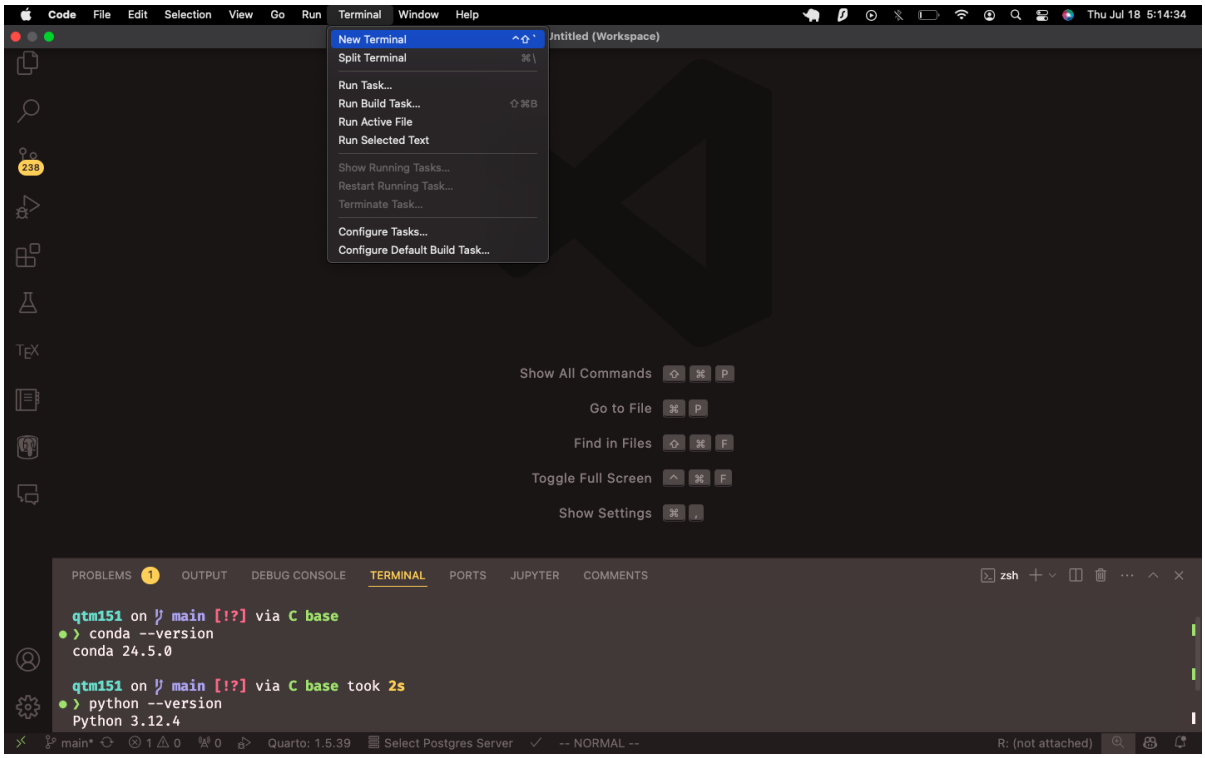

Figure 17: Verify Anaconda Installation

5 Connecting VS Code with Anaconda

Now that both VS Code and Anaconda are installed, we will connect them to use Anaconda's Python distribution within VS Code. If they are connected correctly, you should be able to see the Conda and Python versions again in the VS Code output window.

5.1 Install the Python and Jupyer Extensions in VS Code:

- Open VS Code and click on the Extensions view icon on the left sidebar (it looks like four squares). Search for "Python" in the search bar.

Figure 18: Python Extension in VS Code

- Find the official Python extension by Microsoft and click “Install”.
- Please also install the Jupyter extension for VS Code to work with Jupyter notebooks.

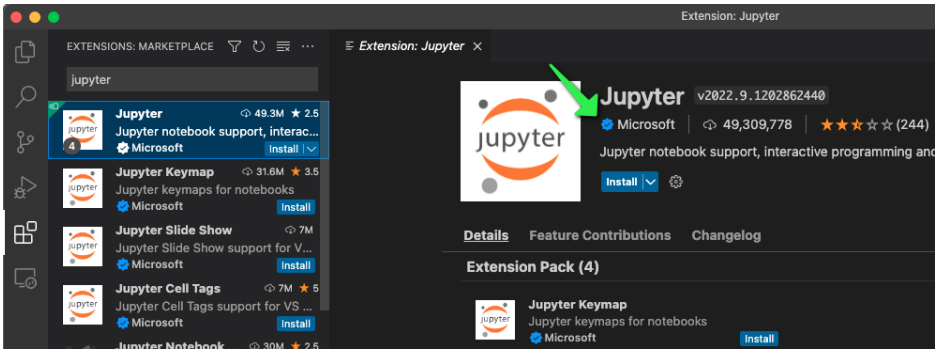

Figure 19: Jupyter Extension in VS Code

5.2 Select the Anaconda Python Interpreter:

- Create a Python file in VS Code. Click on the “New File” button in the top left corner and save it with a `.py` extension.

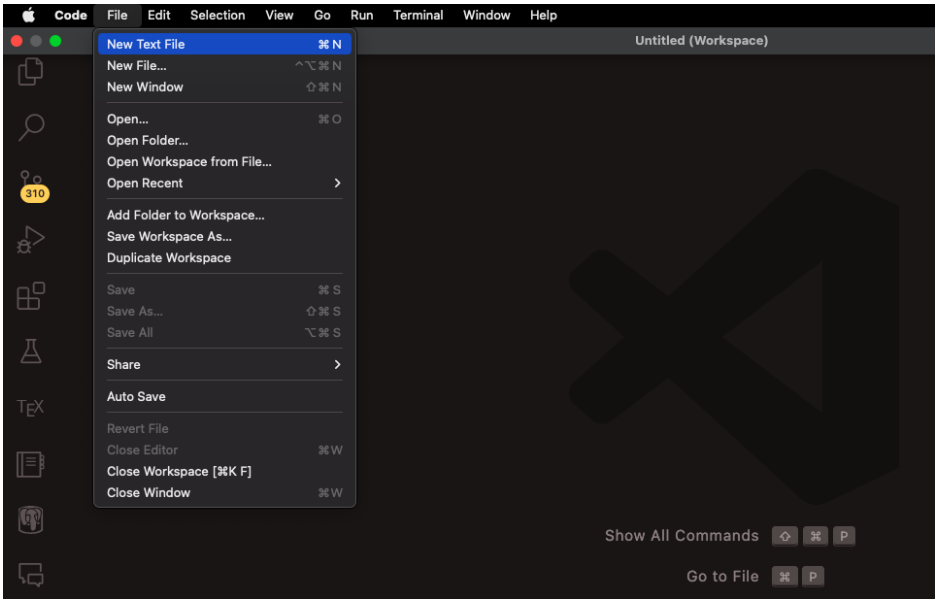

Figure 20: New Python File in VS Code

- Include the following code in the file:

```
import sys
print(sys.version)
print(sys.executable)
```

- Save the file again. Here I saved it as `testing-anaconda.py`.

Figure 21: Save Python File in VS Code

- Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (macOS) to open the Command Palette.

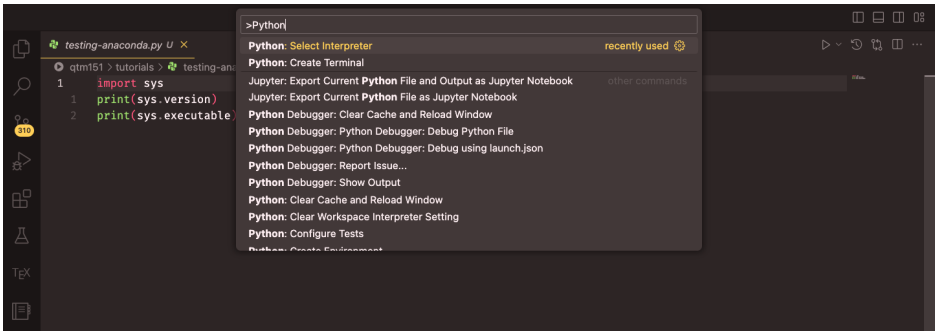

Figure 22: Command Palette in VS Code

- Type “Python: Select Interpreter” and select it from the list.

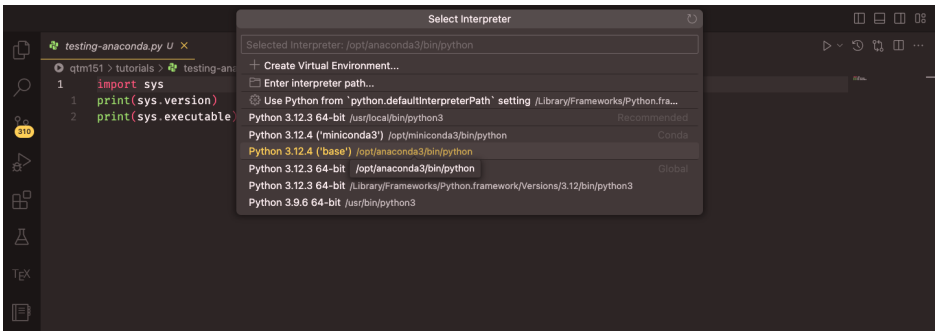

Figure 23: Select Python Interpreter

- Choose the Anaconda Python interpreter from the list. It should be labelled something like “Python 3.x.x (‘base’) Conda”.

5.3 Verify the Connection:

- Please click on the “Run” button at the top right corner of the editor to execute the code in your .py file. Select “Run Current File in Dedicated Terminal”

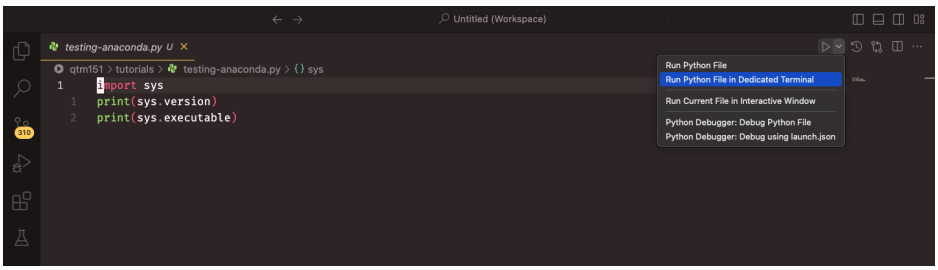

Figure 24: Run Python File in VS Code

- The output should show the Anaconda Python version and its location.

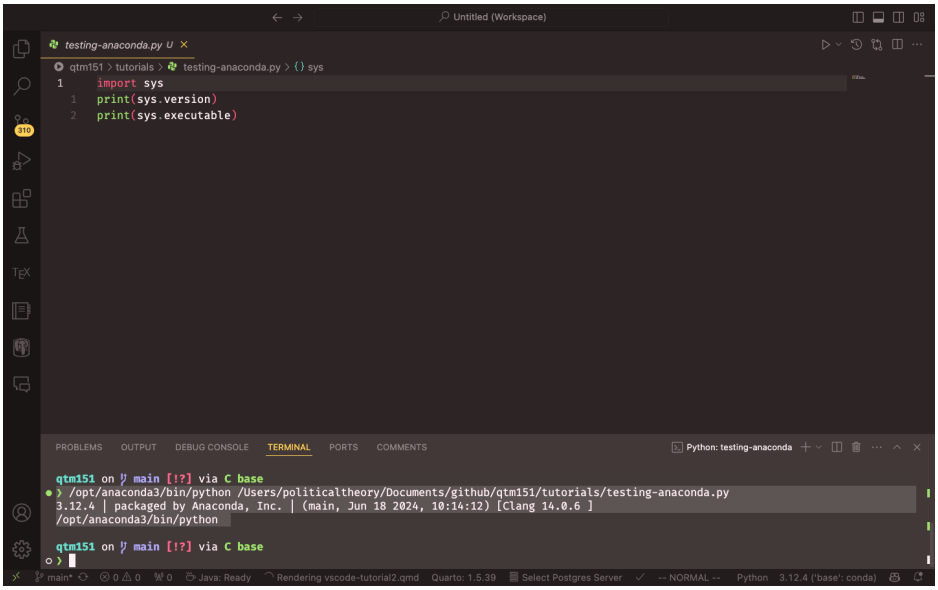

Figure 25: Python Output in VS Code

6 Optional: GitHub Copilot

GitHub Copilot is an AI-powered code completion tool that helps you write code faster and with fewer errors. It is available as an extension for VS Code. Detailed instructions are available on the 03-github-copilot-tutorial tutorial in the course repository, but I will provide a brief overview here. Please also refer to `https://code.visualstudio.com/docs/copilot/getting-started-chat` for further instructions.

To install GitHub Copilot, follow these steps:

- First, sign up for a free educational license at `https://github.com/education/students`. This will give you access to GitHub Copilot and other GitHub features.
- On the website, click on "Join GitHub Education" and follow the instructions to verify your student status. You will need to provide proof of enrollment, such as a school email address or a document from your institution. The educational license is valid for two years.
- Once you have verified your student status, you can install GitHub Copilot in VS Code. Go to the Extensions view in VS Code (the square icon on the left sidebar) and search for GitHub Copilot. Click on "Install". VS Code will install two extensions, GitHub Copilot and GitHub Copilot Chat.

- After installing the extensions, you will need to sign in to your GitHub account in VS Code. Click on the GitHub icon in the left sidebar and follow the prompts to sign in.
- Once you have signed in, you can start using GitHub Copilot. It will provide code suggestions as you type, based on the context of your code. You can accept the suggestions by pressing `Tab` or `Enter`. You can also use the Chat tab (the speech bubble icon) to ask questions in natural language, like you do with ChatGPT or other chatbots, and Copilot will provide code snippets in response.

GitHub Copilot can be very helpful for writing code and answering questions, but it also makes mistakes. It is important to review the code suggestions and ensure they are correct before using them in your projects.

7 Conclusion

You have now successfully installed Visual Studio Code and connected it with Anaconda. Remember to create and activate appropriate Conda environments for different projects to manage dependencies effectively.

For further information and advanced usage, refer to the following resources:

- VS Code Python Tutorial: `https://code.visualstudio.com/docs/python/python-tutorial`
- Anaconda Documentation: `https://docs.anaconda.com/`
- VS Code Documentation: `https://code.visualstudio.com/docs`

Happy coding!

8 Introduction

This tutorial will introduce you to Jupyter Notebook and Markdown. Jupyter Notebook is an interactive computing platform that allows users to create and share documents that contain live code, equations, visualisations, and narrative text. Markdown is a lightweight markup language that is used to format text, and can be used for everything - websites, documents, notes, books, presentations, email messages, and technical documentation. Even WhatsApp and Facebook Messenger use Markdown to format messages. So if you have already italicised a word or made a text bold on WhatsApp, you have used Markdown!

This tutorial is divided into two parts. The first part will introduce you to Jupyter Notebook and show you how to create a new notebook, run code cells, and format text cells using Markdown. The second part will provide a more in-depth look at Markdown and show you how to create headings, lists, links, images, and tables.

9 Introduction to Jupyter Notebook

9.1 What is Jupyter Notebook?

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. Jupyter Notebook supports over 40 programming languages, including Python, R, and Julia. Jupyter Notebook is widely used in data science, machine learning, scientific computing, and other fields.

Jupyter Notebooks provide an interactive development environment that allows you to write and execute code, see the results immediately, and create reproducible analyses. Jupyter Notebooks are organized into cells, which can contain code, text, equations, or visualisations. You can run individual cells or the entire notebook, and you can save your work as a notebook file (`.ipynb`) that can be shared with others.

9.2 Creating a New Notebook

First, please make sure you have Python, Jupyter Notebook, and VSCode installed on your computer. If you do not have these installed, please refer to the VSCode and Anaconda Tutorial for instructions on how to install them.

To create a new Jupyter Notebook in VSCode, please go to the tab "File" > "New File". You will see a prompt in the middle of the screen. Select the option "Jupyter Notebook".

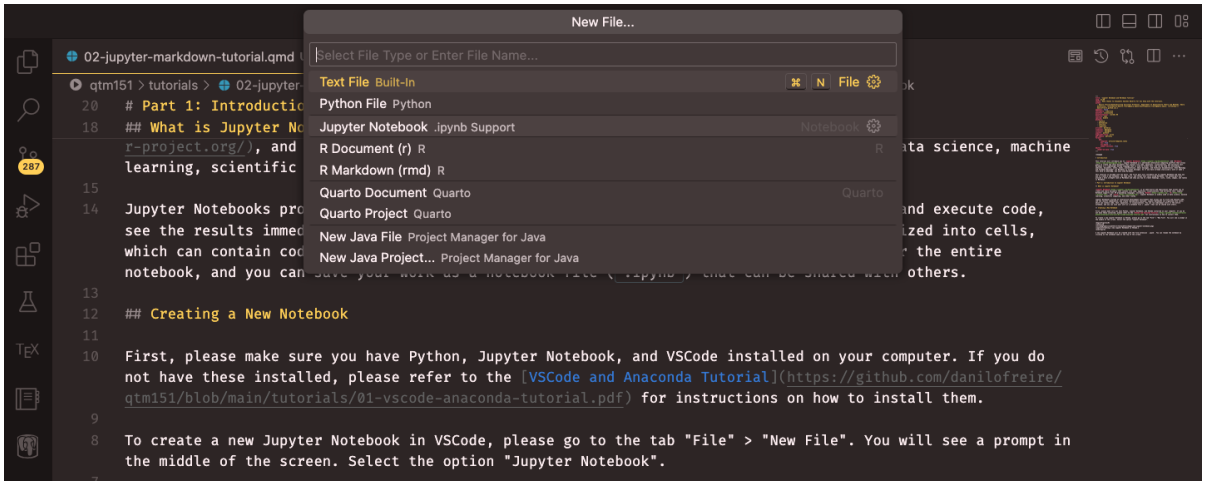

Figure 26: Creating a new Jupyter Notebook in VSCode.

A new Jupyter Notebook will be created with the file extension `.ipynb`. You can rename the notebook by clicking on the notebook name at the top of the screen. An empty notebook will look like this:

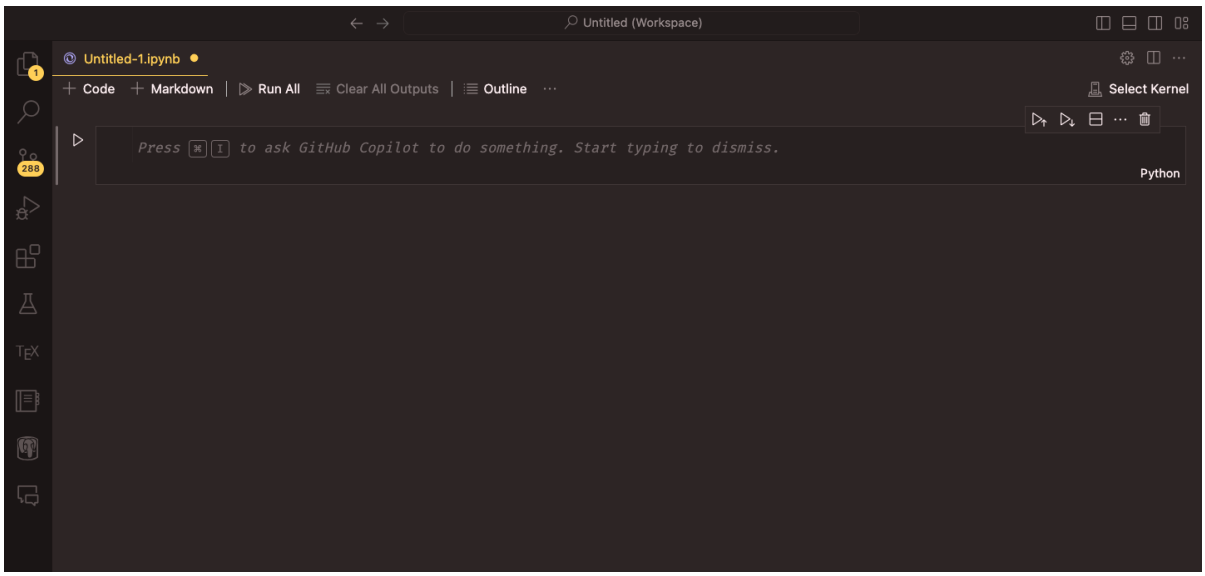

Figure 27: An empty Jupyter Notebook.

Please do not forget to select the Python interpreter that you want to use for the notebook. You can do this by clicking on the Python version at the top right corner of the screen. A prompt will appear, and you can select the Python interpreter that you want to use (in this case, Anaconda's "base").

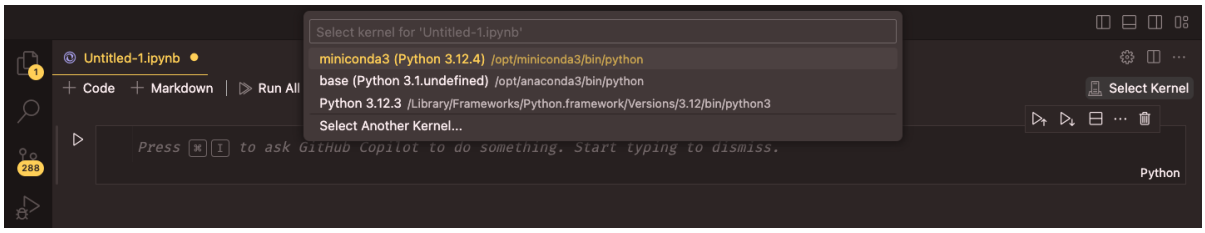

Figure 28: Selecting the Python interpreter for the notebook.

To create a chunk of Python code click on "+ Code." You will get an empty gray box which has Python on the lower-right corner. You can type Python code in this box and run it by clicking on the "Run" button on the left side of the box. You can also run the code by pressing "Shift + Enter" on your keyboard.

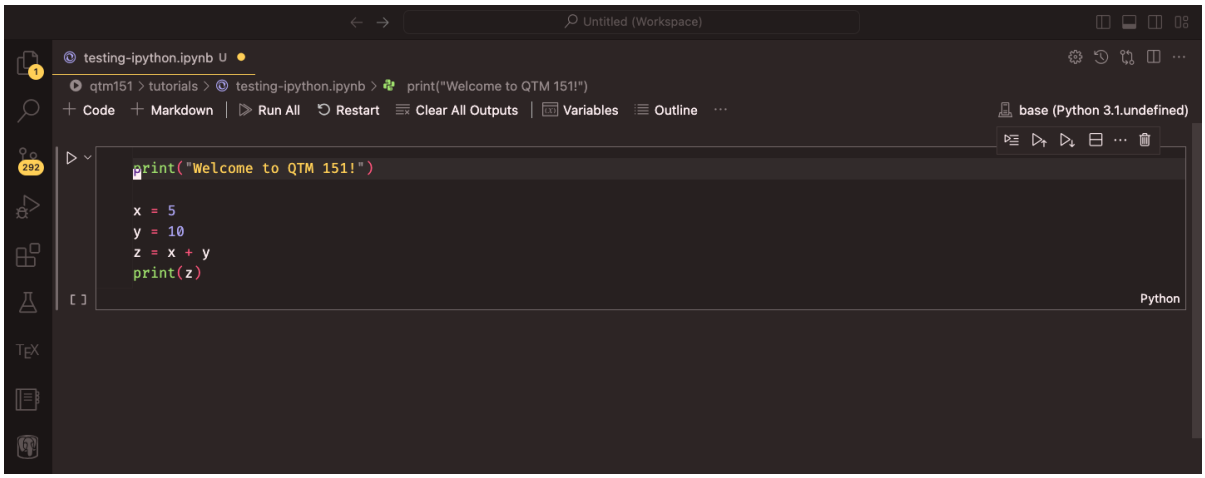

Figure 29: A code cell in a Jupyter Notebook.

Here you should enter the Python commands. For example, type the following lines of code in the code cell:

```
#| echo: true
#| eval: false
print("Welcome to QTM 151!")

x = 5
```

```
y = 10
z = x + y
print(z)
```

The results of the code will be displayed below the code cell. In this case, the output will be:

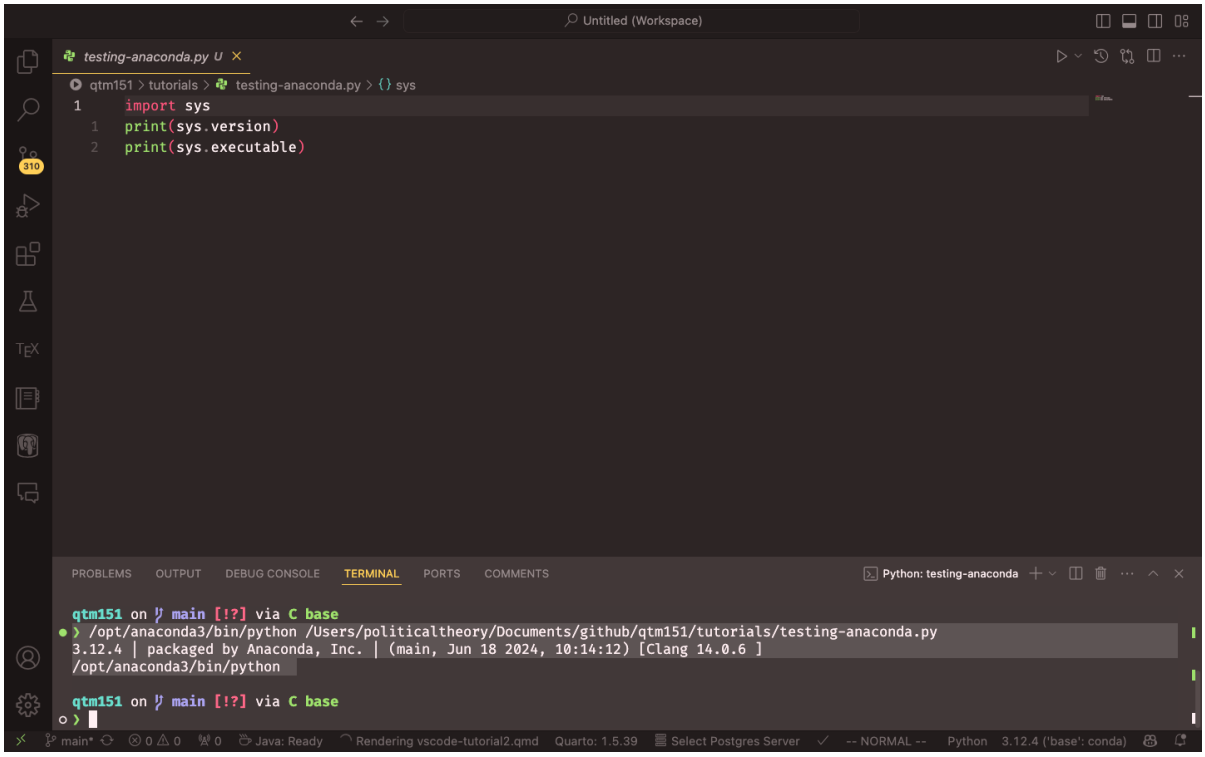

Figure 30: Jupyter Notebook Output

To create a text cell click on "+ Markdown." You will get an empty white box where you can type text. We will cover Markdown in more detail in the next section. But for now, you can type the following text in the Markdown cell:

```
# Welcome to QTM 151!

This is a Jupyter Notebook. You can write *text*, **equations**, and `code`
in [this notebook](https://github.com/danilofreire/qtm151/blob/main/tutorials/testing-ipython.ipynb).
```

The text will be displayed in the text cell like this:

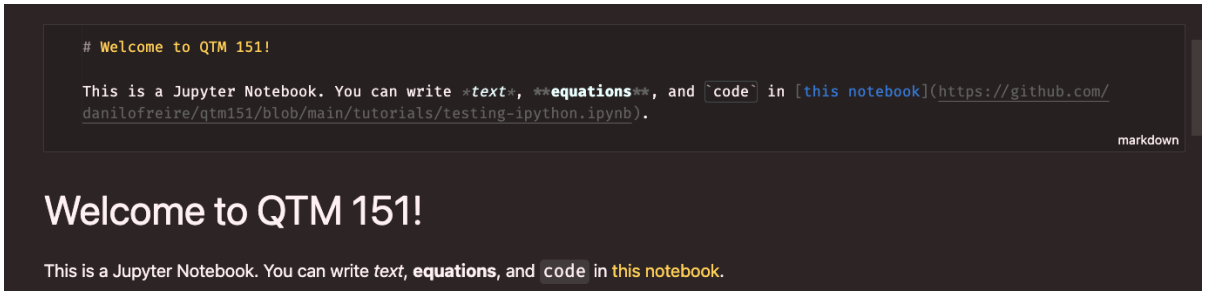

Figure 31: A text cell in a Jupyter Notebook.

You can run the text cell by clicking the "Run" button on the left side of the box or by pressing "Shift + Enter" on your keyboard. The first text block shows how the text looks before running the cell, and the second block shows how it looks after running the cell. Just double click on the space where the text is to edit the Markdown block. This is should open the gray box again.

10 Introduction to Markdown

10.1 Why Use Markdown?

Markdown is a great way to format text because it is simple, easy to learn, and widely supported. You can use Markdown to create headings, lists, tables, equations, and figures in your documents. Markdown is used in many different applications, including Jupyter Notebook, GitHub, and Slack. Learning Markdown will help you create well-formatted documents that are easy to read and share.

10.2 Headings

You can create headings using the # symbol. For example, `# Heading 1` creates a first-level heading, `## Heading 2` creates a second-level heading, and so on. You can create up to six levels of headings using the # symbol.

```
# Heading 1
## Heading 2
### Heading 3
```

10.3 Lists

To create an ordered list with nested unordered sub-items in Markdown, you can write the following code:

```
1. This is an ordered list.
2. This is the second item in the ordered list.
  - This is a sub-item in the unordered list.
    - This is a sub-sub-item in the unordered list.
```

1. This is an ordered list.
2. This is the second item in the ordered list.
    - This is a sub-item in the unordered list.
        - This is a sub-sub-item in the unordered list.

You can also create unordered lists:

```
- This is an unordered list.
- This is the second item in the unordered list.
  - This is a sub-item in the unordered list.
```

- This is an unordered list.
- This is the second item in the unordered list.
    - This is a sub-item in the unordered list.

10.4 Tables

You can create tables using the | symbol. For example:

```
Table: Your Caption

| A            | New              | Table          |
|:-------------|:----------------:|---------------:|
|left-aligned  |centre-aligned    |right-aligned   |
|*italics*     |~~strikethrough~~ |**boldface**    |
```

Table 1: Your Caption

A	New	Table
left-aligned	centre-aligned	right-aligned
italics	~~strikethrough~~	**boldface**

The : symbols in the second row of the table determine the alignment of the text in each column. You can use `left`, `center`, or `right` to align the text.

10.5 Creating Markdown Tables from Pandas DataFrames in Jupyter Notebooks

When working with pandas DataFrames, you can convert them into Markdown tables for better presentation. The method is also relatively simple. Here is how to do it:

10.5.1 Prerequisites

Ensure you have the following installed:

1. Jupyter Notebook
2. `pandas`
3. `tabulate` (for enhanced table formatting)

Jupyter Notebook and `pandas` are included in the Anaconda distribution. You can install `tabulate` using `conda install` in your terminal. If you are using VS Code, you can open a terminal by clicking on "Terminal" > "New Terminal" in the top menu. Then run the following command:

```
conda install tabulate
```

10.5.2 Basic Method: Using `pandas.DataFrame.to_markdown()`

`pandas` provides a built-in method `to_markdown()` for converting DataFrames to Markdown tables.

Step 1: Import pandas and create a DataFrame You can create a dataframe by passing a dictionary to the `pd.DataFrame()` constructor:

```
#| output: false
#| eval: false
# If necessary, install pandas and tabulate
# You should have pandas installed if you installed Anaconda,
# but if you do not, you can install both with the following terminal command:
# conda install pandas tabulate

# Import pandas
import pandas as pd
from tabulate import tabulate

# Create a sample DataFrame
data = {
    "Name": ["Alice", "Bob", "Charlie"],
    "Age": [25, 30, 35],
    "City": ["New York", "London", "Paris"]
}
df = pd.DataFrame(data)
```

Step 2: Convert DataFrame to Markdown Then you just need to call the `to_markdown()` method on the DataFrame:

```
#| output: false
#| eval: false

# Print the DataFrame as a Markdown table, excluding the index
markdown_table = df.to_markdown(index=False)
print(markdown_table)
```

This will output:

```
| Name    | Age  | City     |
|:--------|-----:|:---------|
| Alice   |   25 | New York |
| Bob     |   30 | London   |
| Charlie |   35 | Paris    |
```

Step 3: Display in Jupyter Notebook To display the Markdown table in a Jupyter Notebook cell, use the `display()` function from the `IPython.display` module:

```
#| output: false
#| eval: false

# You should also have the IPython package installed if you installed Anaconda
# If not, you can install it with
# conda install ipython

# Import the Markdown display class.
# This is what allows us to display Markdown in Jupyter Notebooks
from IPython.display import display, Markdown

# Display the Markdown table
display(Markdown(markdown_table))
```

This will render a nicely formatted table in your notebook:

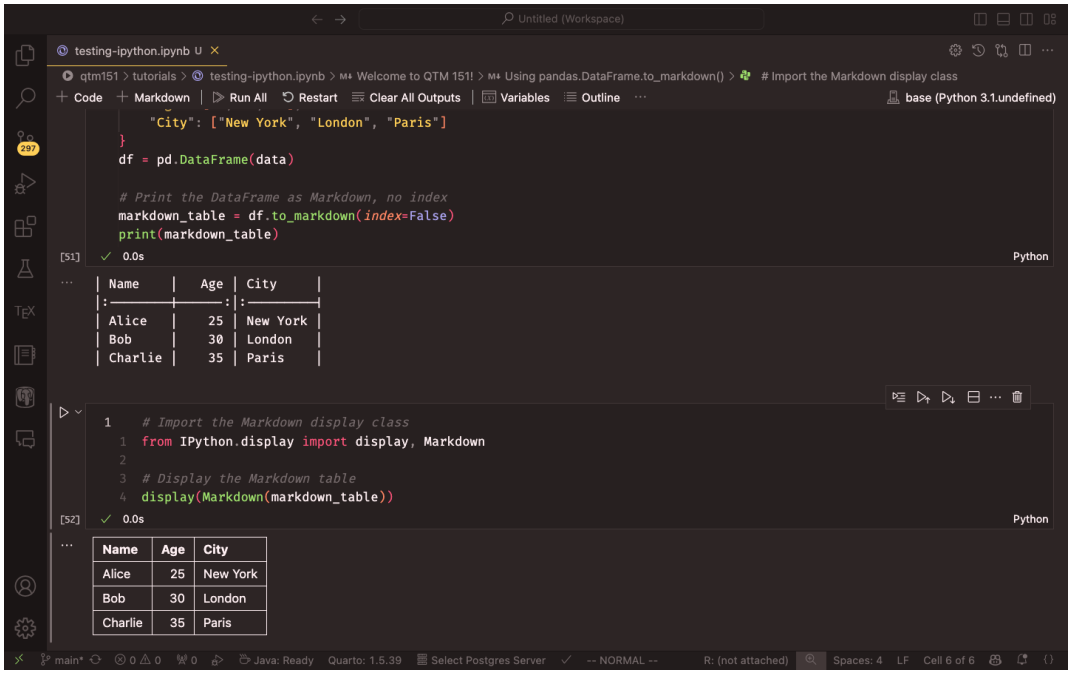

Figure 32: Markdown table in a Jupyter Notebook.

10.5.3 Advanced Formatting

You can customise the Markdown table using various parameters of `to_markdown()`:

```
#| output: false
#| eval: false
# Customising the Markdown table using the to_markdown() method
markdown_table = df.to_markdown(
    index=False,  # Don't include index
    tablefmt="pipe",  # Use pipe format
    floatfmt=".2f",  # Format floats to 2 decimal places
    headers=["Name", "Age (Years)", "City"],  # Custom headers
    colalign=("left", "center", "right")  # Align columns
)
display(Markdown(markdown_table))
```

10.5.4 Using tabulate for Enhanced Formatting

The `tabulate` library offers even more formatting options. While they are not necessary for basic tables, they can be useful for more complex tables. In our course, you will probably not need them, but I will show you how to use them for your reference. Please check their documentation for more information.

```
#| output: false
#| eval: false
from tabulate import tabulate

markdown_table = tabulate(df, headers='keys', tablefmt='pipe', showindex=False)
display(Markdown(markdown_table))
```

10.5.5 Best Practices

1. **Index**: Consider whether you need the index in your table. Often, it is cleaner to exclude it using `index=False`.
2. **Formatting**: Use `floatfmt` to control decimal places for numerical data.
3. **Headers**: Customize headers for clarity, especially if your DataFrame column names are not user-friendly.
4. **Table Format**: Experiment with different `tablefmt` options to find the most suitable for your needs.

5. **Large DataFrames**: For large DataFrames, consider displaying only a subset of rows or columns to maintain readability.

10.5.6 Example: Comprehensive Table Creation

Here is a more comprehensive example incorporating various best practices:

```
#| output: false
#| eval: false
# Import necessary packages
import pandas as pd
from IPython.display import display, Markdown
from tabulate import tabulate

# Create a sample DataFrame
data = {
    "Product": ["Laptop", "Smartphone", "Tablet"],
    "Price": [999.99, 599.50, 299.75],
    "Stock": [50, 100, 75],
    "Rating": [4.5, 4.8, 4.2]
}
df = pd.DataFrame(data)

# Create a formatted Markdown table
markdown_table = tabulate(
    df,
    headers=["Product Name", "Price ($)", "Stock Quantity", "Customer Rating"],
    tablefmt="pipe", # Format as Markdown table
    floatfmt=(".2f", ".2f", "d", ".1f"),
    showindex=False, # Don't show the index (row numbers)
    numalign="right", # Align numbers to the right
    stralign="left" # Align strings (text, first column) to the left
)

# Display the table in the notebook
display(Markdown("### Product Inventory Summary"))
display(Markdown(markdown_table))
```

This will produce a well-formatted table with custom headers, appropriate number formatting, and a title.

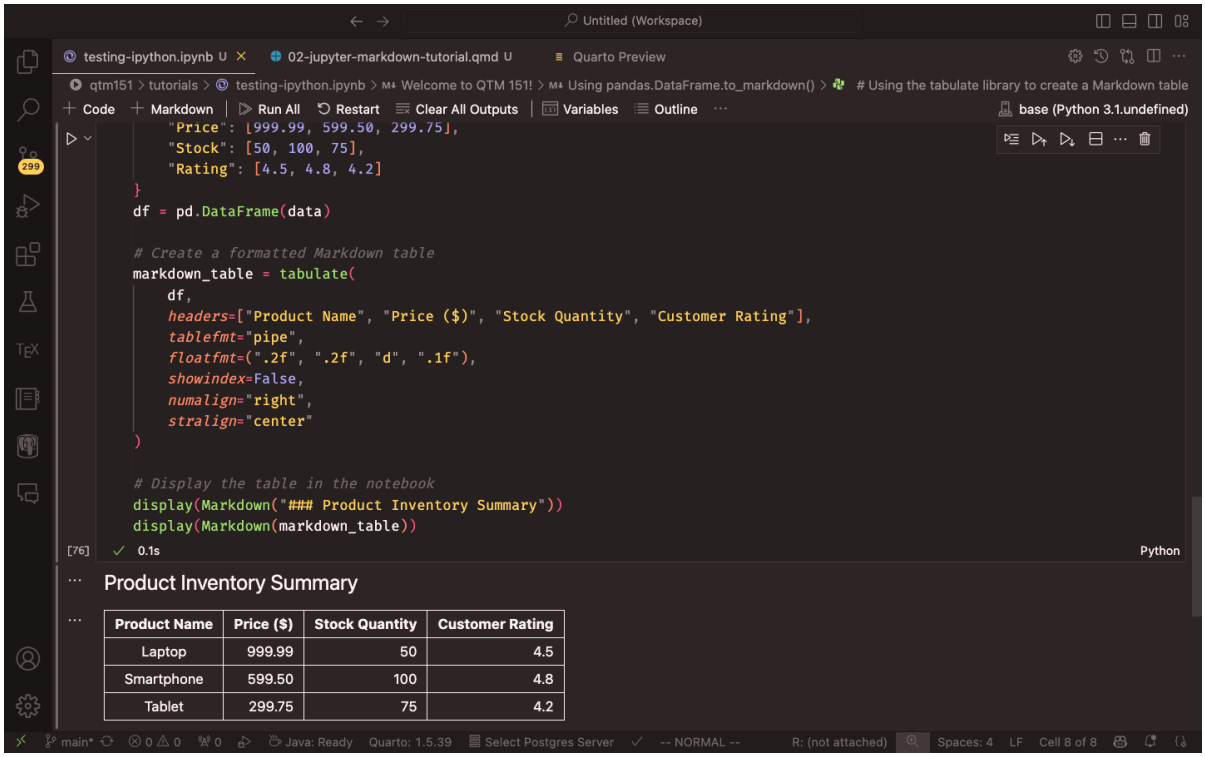

Figure 33: Comprehensive Markdown table in a Jupyter Notebook.

10.6 Equations

You can create equations using the $$ symbol. For example in Equation 1, we have the formula for the standard deviation of a population:

```
$$
\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}}
$$ {#eq-stddev}
```

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}} \tag{1}$$

You can also create equations inline by using the $ symbol. For example, `$\alpha = \beta + \gamma$` will render as $\alpha = \beta + \gamma$. To learn more about how to write equations in LaTeX using Markdown, you can refer to the Overleaf documentation.

10.7 Figures

You can include figures in your document using the `![Caption](path/to/image.png){#fig-label}` syntax. For example:

```
![This is a figure caption.](path/to/image.png){#fig-label}
```

This will include the image `path/to/image.png` in your document with the caption "This is a figure caption." You can refer to the figure using the label `fig-label`, but this is optional.

If you are creating plots on Jupyter Notebook, the graphs will appear after the code cell that generates them.

10.8 Citations

Although Markdown has excellent reference support with BibTeX files, there is no reliable way to include citations automatically in Jupyter Notebooks. The two packages I know that manage citations, cite2c and Jupyterlab Citation Manager, are not ideal. The first has not been maintained for years, while the second is not ready for most uses. Therefore, I suggest simply copying the citation from Google Scholar and pasting it into a Markdown cell titled "References" at the end of your document. The same approach applies for inline citations.

10.9 Footnotes

Jupyter and Markdown support inline footnotes[1]. To create a footnote, simply add a caret and brackets with a label inside, like this: `[^label]`. Then, you can define the footnote content anywhere in the document using the same label followed by a colon[2]. I usually include them at the end of a paragraph. Jupyter will automatically number and format your footnotes for you.

[1] This is an inline footnote.

[2] You can also include multiple paragraphs in a footnote by indenting the subsequent paragraphs.

11 Conclusion

I hope you find this tutorial helpful and that you can use Jupyter Notebook and Markdown to create beautiful documents. If you have any questions or feedback, please feel free to reach out to me. Happy coding! :)

12 Introduction

This tutorial will guide you through the process of creating a GitHub educational account, downloading and installing GitHub Desktop, and downloading the course materials from GitHub to your computer. The tutorial also covers the basics of Git and GitHub, such as creating a repository, committing changes, and pushing changes to GitHub. It will also show you how to use VS Code to edit and manage your code on GitHub. You can use any tool you prefer to version control your code.

13 What are Git and GitHub?

Git is a distributed version control system that allows you to track changes in your codebase, collaborate with others, and manage your projects. It was developed by Linus Torvalds (of Linux fame) in 2005 and has become the *de facto* standard for version control in the software development industry. While Git is a powerful tool, it can be a bit challenging to use, especially for beginners. That is where GitHub comes in.

GitHub is a web-based platform that provides hosting for Git repositories, allowing you to store different versions your code in the cloud and collaborate with others. GitHub also offers several useful features, such as issue tracking, project management, and continuous integration. It can even host your website for free. GitHub has become the go-to platform for open-source projects, and many companies use it to manage their codebase, such as Microsoft, Google, and Meta[3]. Think about GitHub as a social network for developers, where you can follow other developers, star their projects, and contribute to open-source projects.

We will use GitHub extensively in our course. So it is important that you become familiar with it. The good news is that GitHub is free for public and private repositories for educational purposes. Here are the steps to get started with GitHub:

1. **Create a GitHub account**. Go to `https://github.com/` and sign up (top right). You can use your Emory email address to get a free educational account (more on this later). Be careful selecting your username, this will be visible to the public.

[3] As a curiosity, the code for the Apollo 11 mission is also available on GitHub.

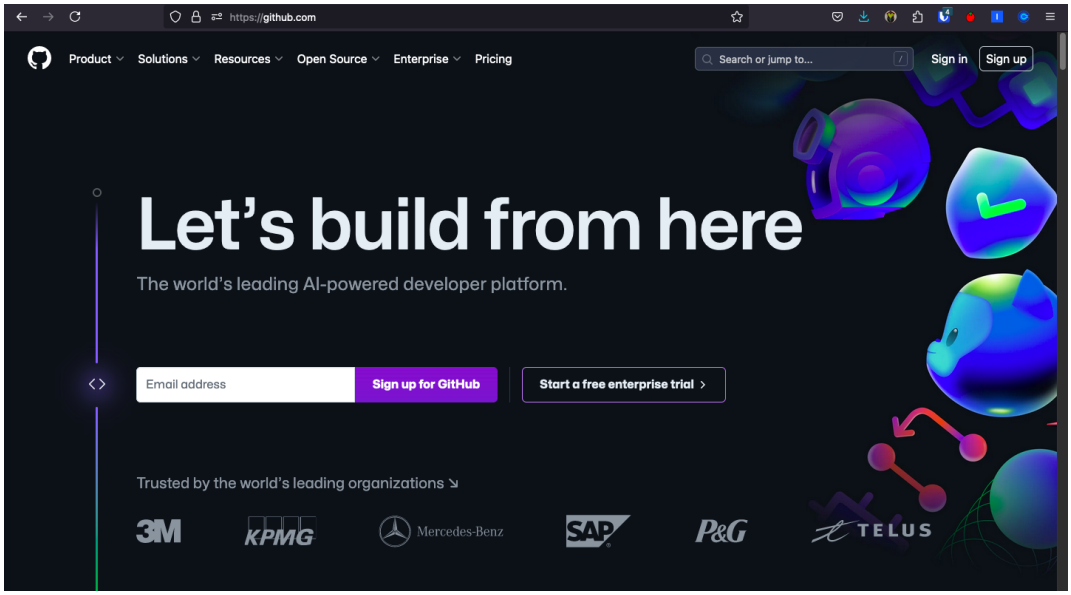

Figure 34: GitHub Signup

2. **Apply to GitHub Education**. Go to `https://github.com/education` and apply for the GitHub Student Developer Pack. This will give you access to several tools and services for free, such as a free domain name, cloud hosting, and more. You will need to provide proof of your student status, such as an ID card or transcript. Please click on "Join GitHub Education" and follow the instructions on the website.

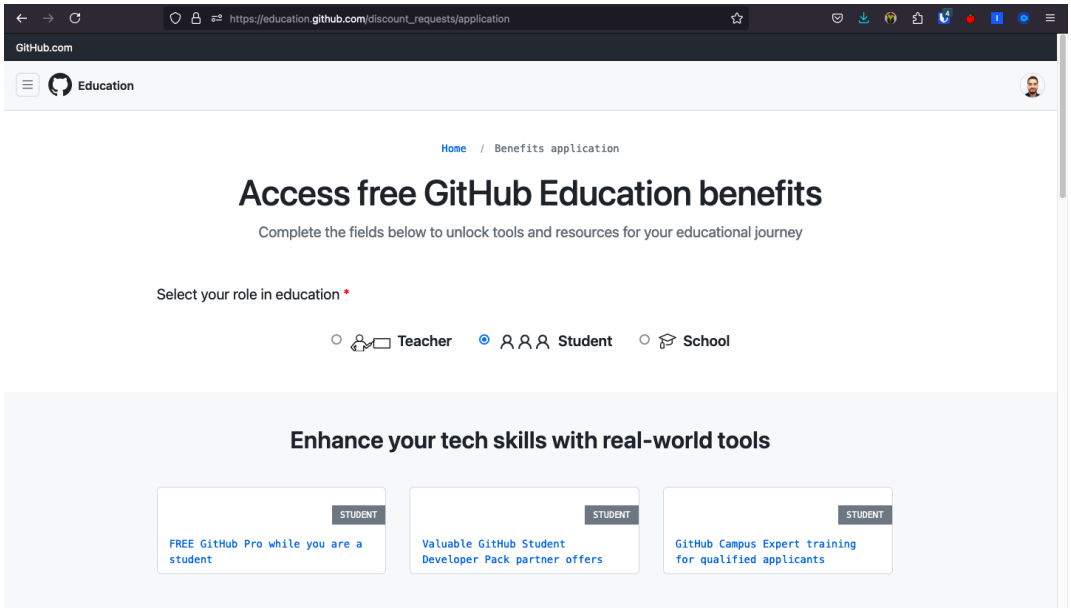

Figure 35: GitHub Education

3. **Download and install GitHub Desktop**. You can download GitHub Desktop from https://desktop.github.com/. GitHub Desktop is a graphical user interface (GUI) for Git that makes it easier to work with repositories. It is available for Windows and macOS. You can also use the command line if you prefer, but GitHub Desktop is more user-friendly and will be used in this tutorial.

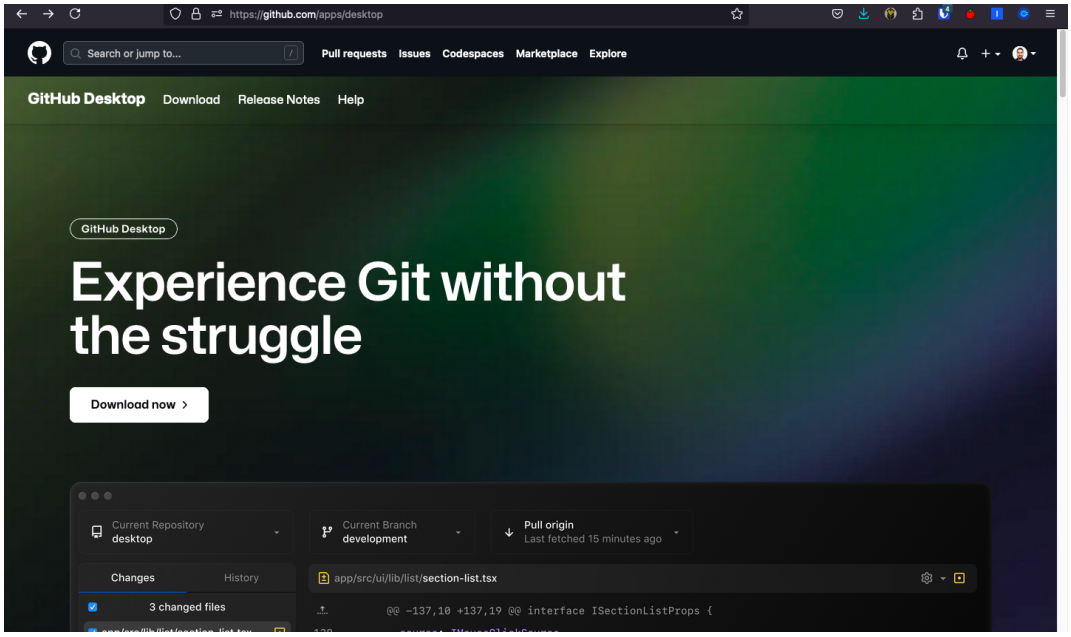

Figure 36: GitHub Desktop

After you install GitHub Desktop, you will need to sign in with your GitHub account. This will allow you to work with your repositories. Then you should see the following screen:

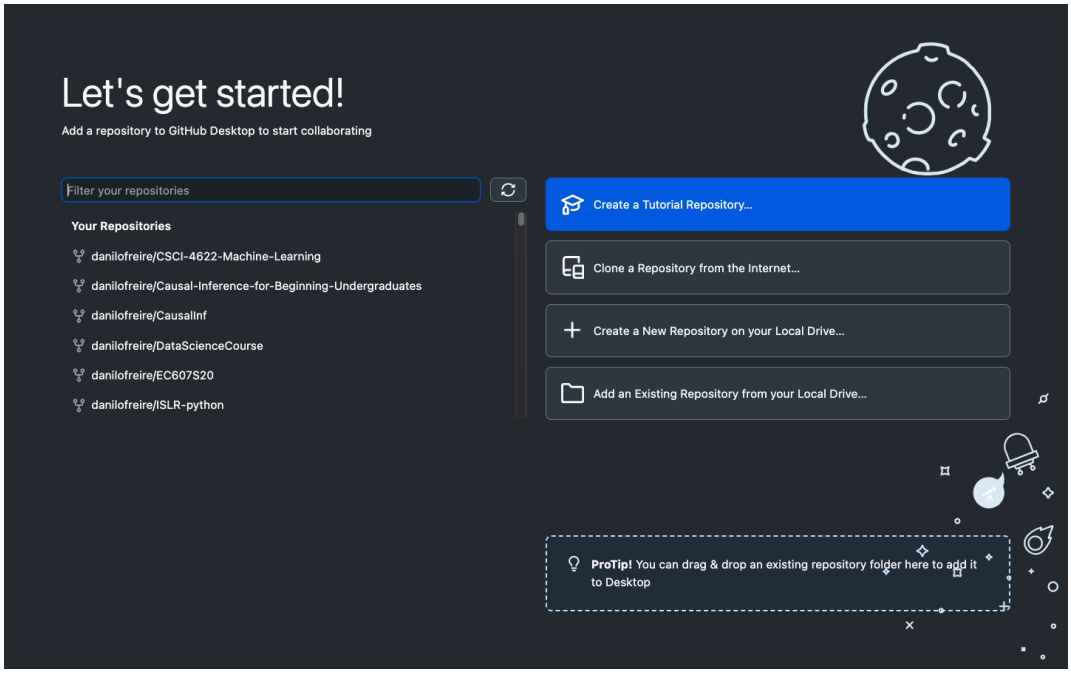

Figure 37: GitHub Desktop Sign In

Now you are ready to start using Git and GitHub. Let us move on to the next step.

14 Downloading the Course Materials

The course materials are available on GitHub at `https://github.com/danilofreire/qtm350`. You should "fork" the repository to your account. This will create a copy of the repository in your account that you can modify and push changes to. Please click on the top-right corner of the repository page and select "Fork".

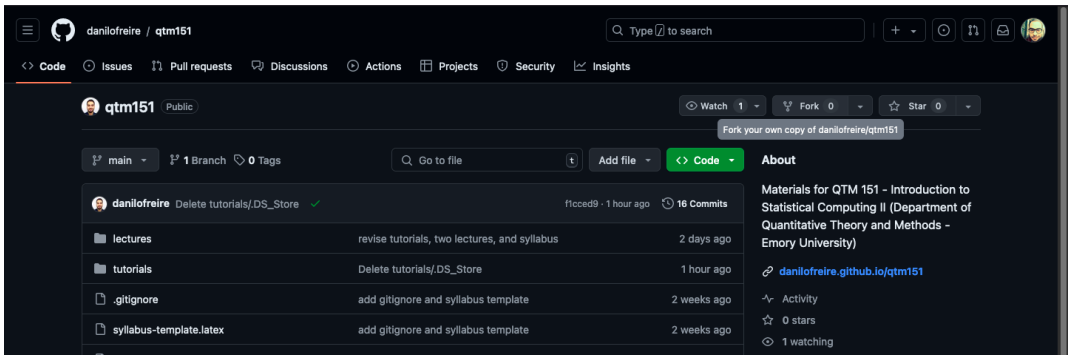

Figure 38: Fork the Repository

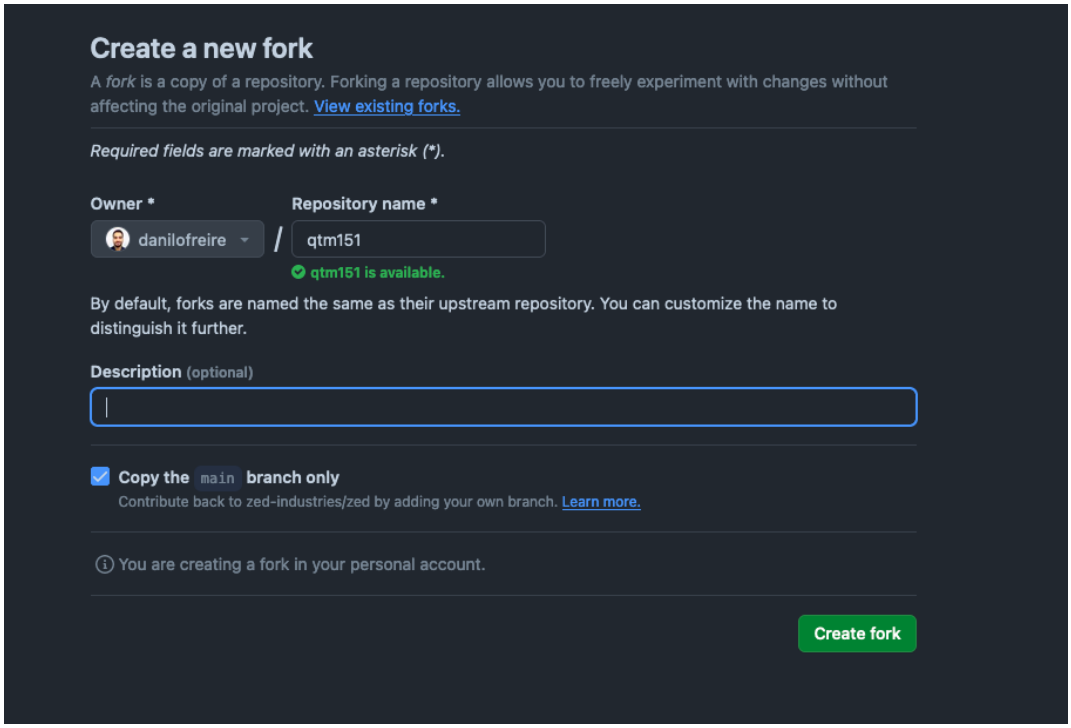

Figure 39: Create a Copy of the Repository

You will be asked to make your own copy of the lecture notes. You can add any name you like. I recommend you use the name same name I used for the repository (`qtm350`). This will make it easier to follow the instructions in the tutorials. Click on "Create fork".

You will only do this once. After you fork the repository, you will be able to pull changes from the original repository and push changes to your forked repository. This will allow you to keep your copy of the course materials up to date with the latest changes.

Now you need to clone your repository to your computer. This will create a local copy of the repository on your computer that you can work with. To do this, click on the green "Code" button and copy the URL of your forked repository. Again, please remember that you should clone *your forked repository*, not the original repository. The name in the URL should be your username.

Please open GitHub Desktop and click on the "Clone a Repository from the Internet" button. This will open a dialog where you can paste the URL of your repository and select a location on your computer to save the repository. Then click on "Clone".

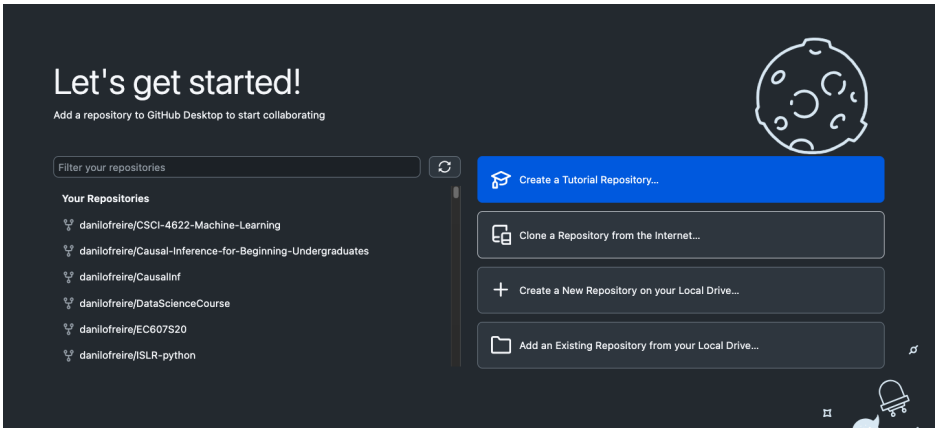

Figure 40: Click on "Clone a Repository from the Internet".

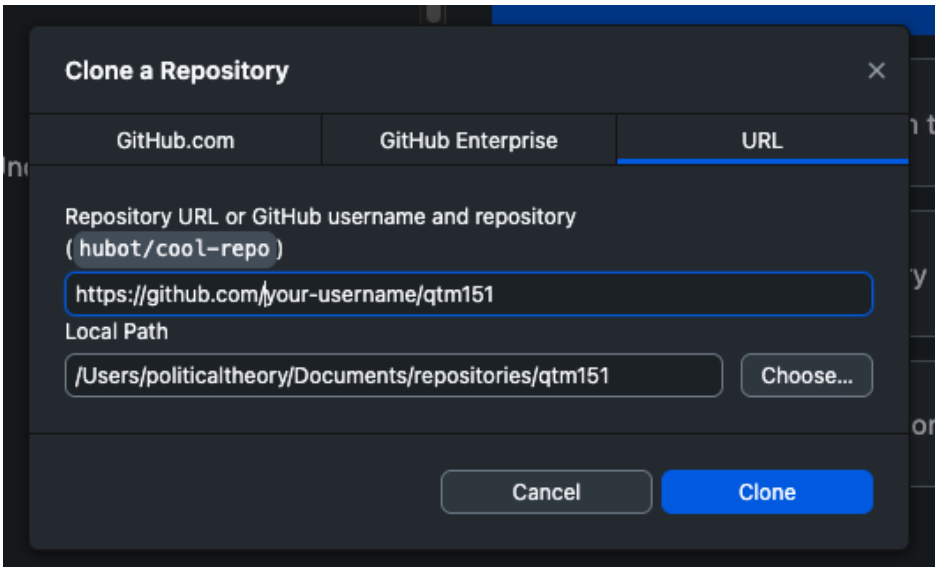

Figure 41: Click on URL and Paste the Repository URL. Then click on "Clone"'.

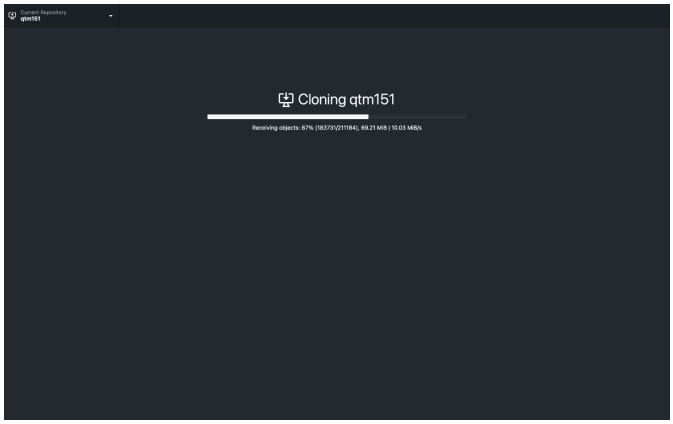

Figure 42: Cloning

Then select “To contribute to the parent project” and click on “Continue”. This option allows you to keep updating your lecture notes from the instructor’s repository. The other option creates a local copy that has no links to the original repository. If you accidentally choose this option, you will not be able to see any changes. To reverse: (a) remove the repository on your computer (by right-clicking on the repository on Github dekstop), (b) manually delete the directory, (c) go back to Step 2.

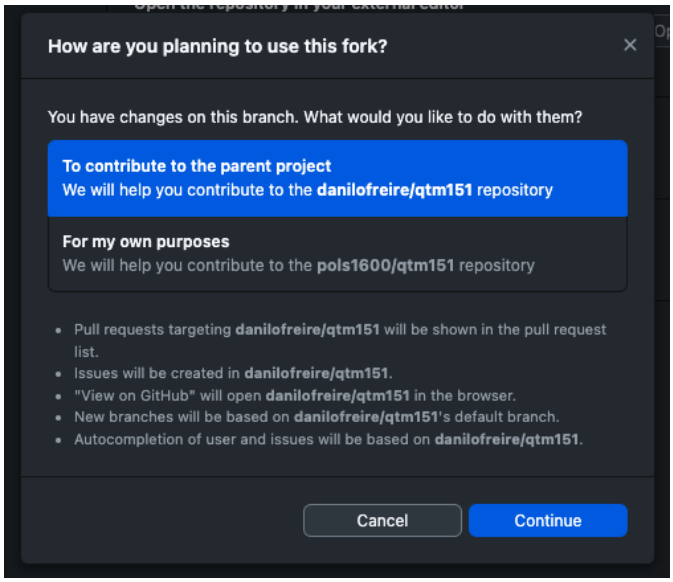

Figure 43: Cloning

And you are done! You now have a local copy of the course materials on your computer that you can work with. You can open the repository in VS Code by clicking on “Repository” > “Open in Visual Studio Code”. This will open the repository in VS Code, where you can edit and manage your code.

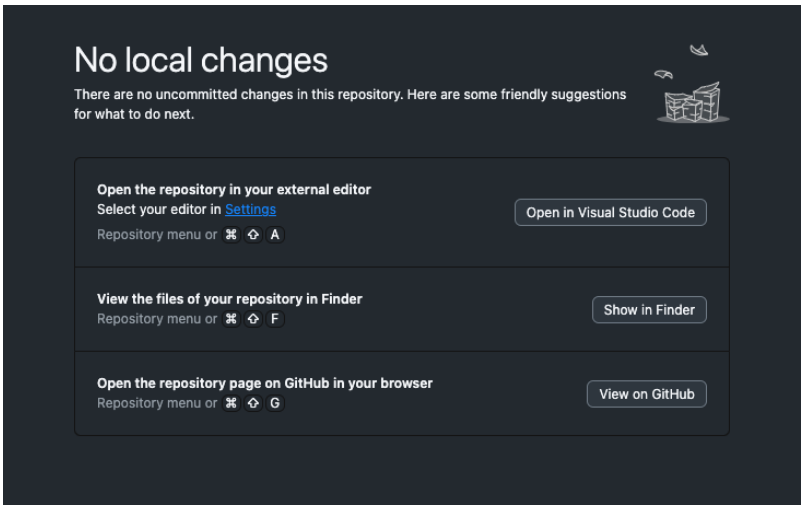

Figure 44: Open in Visual Studio Code

15 How to Add Changes to Your Repository

Now that you have the course materials on your computer, let us go over some of the essential Git and GitHub commands that you will need to use.

1. *A programmer makes some changes to their files*: In the photo you can see that I have made many changes to a few files. GitHub Desktop will show you the changes you made to your files. You can see the changes in the "Changes" tab.

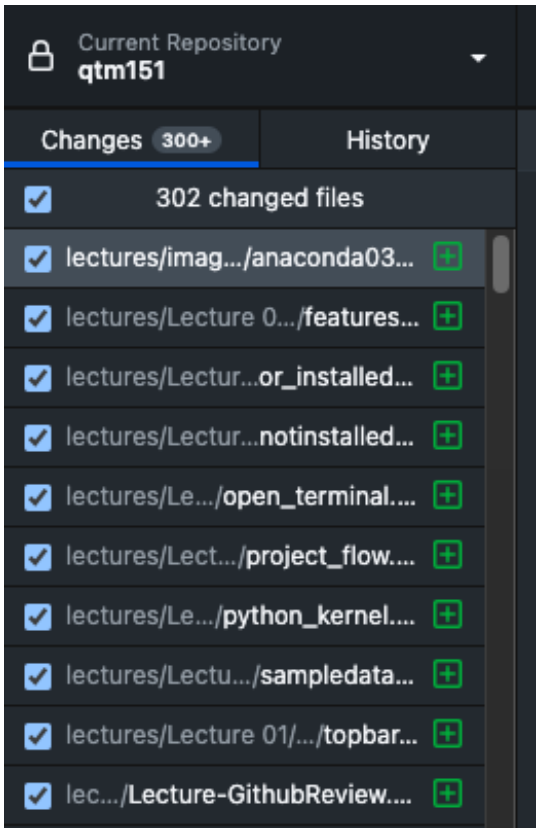

Figure 45: Changes

You can also see the history of your changes in the "History" tab. This will show you all the commits you have made to your repository. The green lines indicate the changes you made, and the red lines indicate the changes you removed.

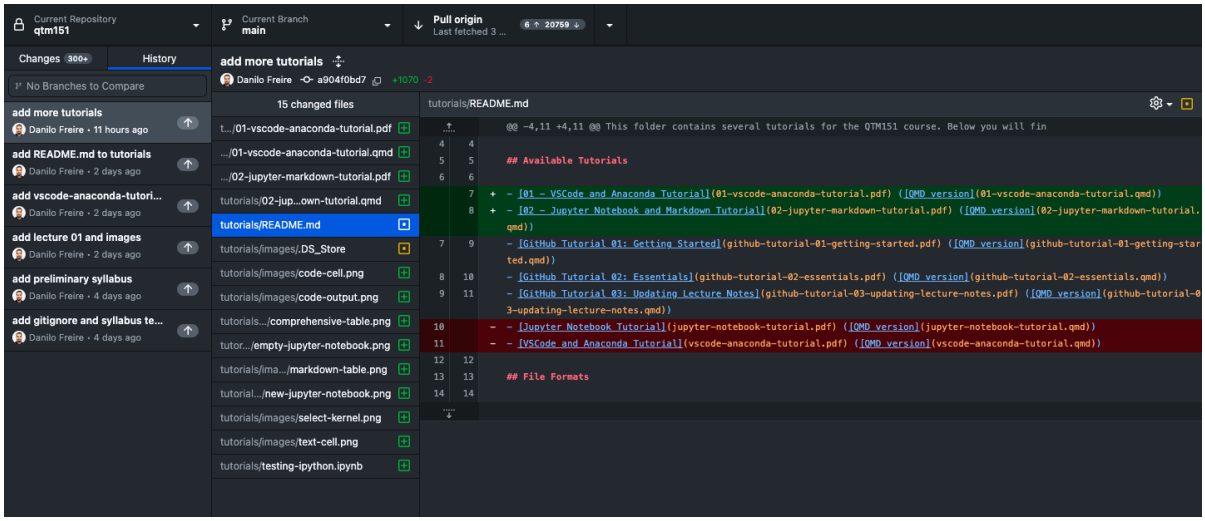

Figure 46: History

2. *The programmer commits the changes*: After making changes to your files, you need to commit them to your local repository. This creates a snapshot of your changes that you can revert to later if needed. To commit changes, open GitHub Desktop and you will see the changes you made. Write a summary of the changes in the "Summary" box and click on "Commit to main"[4]. It is always a good idea to write a descriptive summary of your changes so that you can easily track them later.

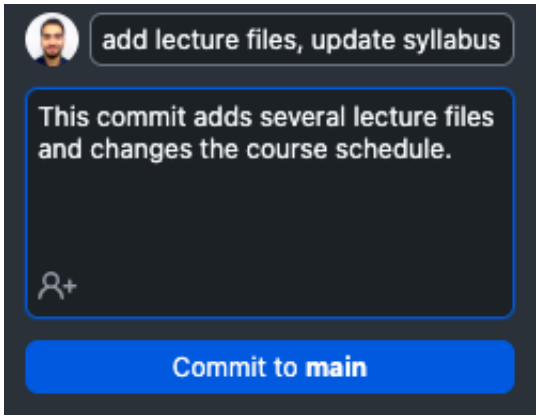

Figure 47: Commit Changes

As you can see, I have committed the changes I made to the files. You can see the commit message in the "History" tab. This will allow you to track the changes you made to your files.

[4]The default branch in GitHub is called `main`. This is the branch where you will make most of your changes. You can create other branches if you want to work on different features or bug fixes.

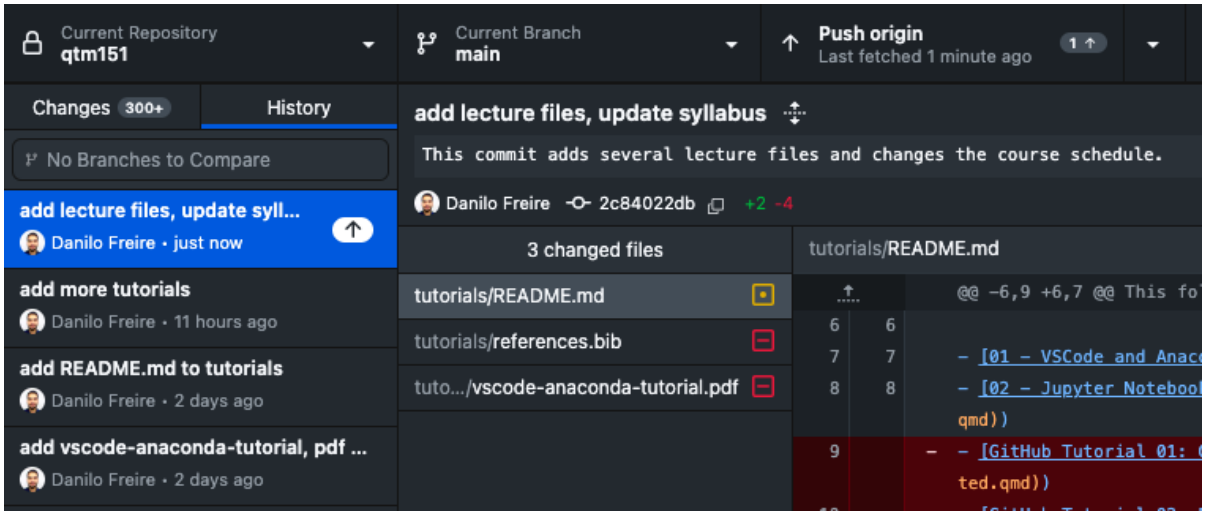

Figure 48: Commit Message

3. *The programmer pushes the changes to GitHub*: After committing your changes to your local repository, you need to push them to GitHub. This will update the remote repository with the changes you made. To push changes, click on "Repository" > "Push" in GitHub Desktop. This will push your changes to GitHub.

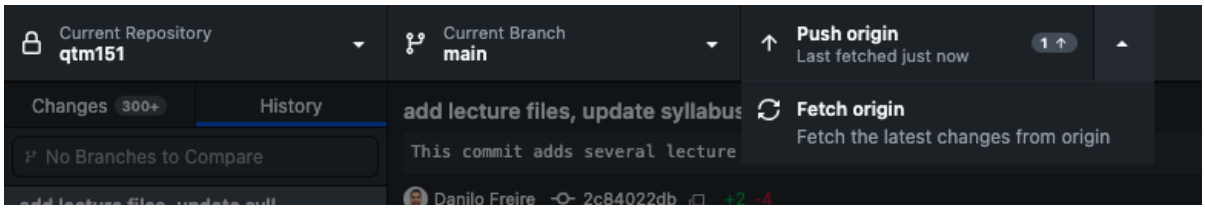

Figure 49: Push or Fetch Changes

As you can also see, there are two options: "Push" and "Fetch". "Push" will push your changes to GitHub, while "Fetch" will fetch ("download") changes from GitHub. You can use "Fetch" to update your local repository with the changes from GitHub. We will cover this in more detail in the next section.

16 Updating the Course Materials

The course materials will be updated regularly with new content, bug fixes, and improvements. To keep your copy of the course materials up to date, you need to pull changes from the original repository.

However, before proceeding, make sure that you have read the previous section and that you have committed all outstanding changes.

1. In the top panel go to the "Branch" tab. Click on the option "update from upstream/main".
    - "upstream" refers to the original repository (the instructor's).
    - "main" is usually the name given to the primary version of your repository.
    - *Note*: You can create different branches to work on different features or bug fixes. This allows you to work on different parts of your codebase without affecting the main branch. You can then merge the changes from the different branches into the main branch when you are ready.

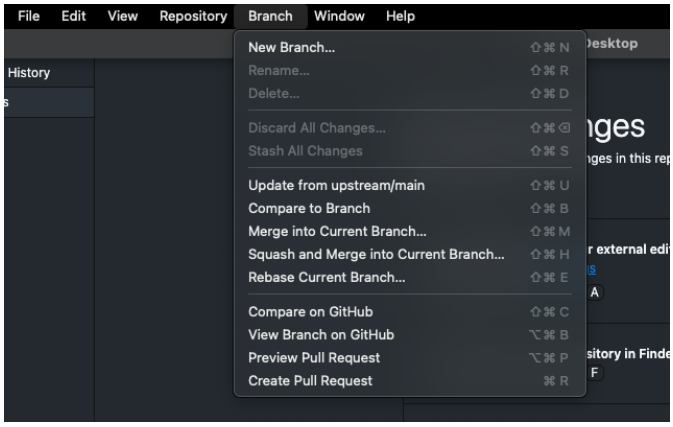

Figure 50: Update from Upstream

Some practical advice:

- Try to do the updating process *at the start of every class*, to make sure you have the latest version.
- Store any modifications to the files *with a separate file name.* This avoids your files *being overwritten by the instructor's version* when you do this process
- Privacy: Right now, your forked repository is publicly visible on your account. If you want to change this, you can make your repository private. How to do this:
    - Go to your repository on GitHub
    - Click on the "Settings" tab

- Scroll down to the “Danger Zone” section
- Click on “Make private”
- Confirm the change

And that is it! You now know how to use Git and GitHub to manage your code and collaborate with others. If you have any questions or need help, feel free to ask. I am here to help you. Good luck with your coding!

17 Recommended Readings and Resources

- GitHub Guides: A collection of guides to help you get started with Git and GitHub.
- GitHub Skills: A very useful website to help you improve your Git and GitHub skills.
- GitHub Desktop Documentation: The official documentation for GitHub Desktop.
- Pro Git: A free book that covers everything you need to know about Git.
- How to Use Git & GitHub Desktop Tutorial for Beginners - YouTube: A video tutorial that covers the basics of GitHub Desktop. There are many other tutorials on YouTube that can help you get started, so feel free to explore!