Financial Modelling: Prereading Materials - Intro, installation, and learning suggestions

Introduction to Financial Modelling

The purpose of this course is to make you better at financial modelling.

We accomplish this through:

- giving you skills in using Python as a financial modelling tool; and
- exploring some specific applications

There a certain areas of finance such as portfolio theory, yield curves, and option pricing that we deep-dive into. This will deepen your understanding of these areas. But the more important outcome is that you will develop the ability to harness a powerful toolset with broad application.

About the tools we are teaching you to use

If you have not done any programming before, you will face quite a steep learning curve in this subject. You will learn how to use the programming languate, Python. You will learn the basics of using GitHub - a development and collaboration tool. You will learn how to use VSCode - an integrated development environment that helps you be successful in using Python (and other tools and languages) in financial modelling.

Why Python?

You might be thinking, "Python is for software engineers. I know excel and that's all I will need to do well in Finance." This may be true just as there were many people in my generation who did well in finance without being any good at Excel. But for many roles, strong excel skills were a way to differentiate oneself in my day. Strong Python skills with all the capability that brings can be source of competitive advantage for you in banking, strategy, trading, investing, and other finance areas.

There are several ways in which Python is superior to Excel for finance applications:

1. Automation and Efficiency

Python allows for automation of repetitive tasks, which can save a significant amount of time compared to manual operations in Excel.

```
In [1]: # Example: Automating data fetching from a web API
        import requests

        response = requests.get('https://api.exchangerate-api.com/v4/latest/USD')
        data = response.json()
        print(data)
```

```
{'provider': 'https://www.exchangerate-api.com', 'WARNING_UPGRADE_TO_V6': 'http
s://www.exchangerate-api.com/docs/free', 'terms': 'https://www.exchangerate-api.c
om/terms', 'base': 'USD', 'date': '2026-01-10', 'time_last_updated': 1768003201,
'rates': {'USD': 1, 'AED': 3.67, 'AFN': 66.09, 'ALL': 82.68, 'AMD': 381.34, 'AN
G': 1.79, 'AOA': 919.63, 'ARS': 1452.25, 'AUD': 1.5, 'AWG': 1.79, 'AZN': 1.7, 'BA
M': 1.68, 'BBD': 2, 'BDT': 122.18, 'BGN': 1.62, 'BHD': 0.376, 'BIF': 2967.93, 'BM
D': 1, 'BND': 1.29, 'BOB': 6.93, 'BRL': 5.38, 'BSD': 1, 'BTN': 90.2, 'BWP': 13.8,
'BYN': 2.95, 'BZD': 2, 'CAD': 1.39, 'CDF': 2243.85, 'CHF': 0.801, 'CLF': 0.0227,
'CLP': 896.92, 'CNH': 6.98, 'CNY': 6.99, 'COP': 3732.73, 'CRC': 497.18, 'CUP': 2
4, 'CVE': 94.75, 'CZK': 20.87, 'DJF': 177.72, 'DKK': 6.41, 'DOP': 63.46, 'DZD': 1
29.94, 'EGP': 47.28, 'ERN': 15, 'ETB': 154.51, 'EUR': 0.859, 'FJD': 2.28, 'FKP':
0.746, 'FOK': 6.41, 'GBP': 0.746, 'GEL': 2.7, 'GGP': 0.746, 'GHS': 10.73, 'GIP':
0.746, 'GMD': 73.88, 'GNF': 8748.31, 'GTQ': 7.67, 'GYD': 209.15, 'HKD': 7.79, 'HN
L': 26.37, 'HRK': 6.47, 'HTG': 130.95, 'HUF': 331.11, 'IDR': 16816.53, 'ILS': 3.1
6, 'IMP': 0.746, 'INR': 90.22, 'IQD': 1310.61, 'IRR': 42077.27, 'ISK': 126.57, 'J
EP': 0.746, 'JMD': 158.05, 'JOD': 0.709, 'JPY': 157.78, 'KES': 128.97, 'KGS': 87.
39, 'KHR': 4018.65, 'KID': 1.5, 'KMF': 422.73, 'KRW': 1458, 'KWD': 0.307, 'KYD':
0.833, 'KZT': 510.51, 'LAK': 21686.57, 'LBP': 89500, 'LKR': 309.6, 'LRD': 179.14,
'LSL': 16.5, 'LYD': 5.42, 'MAD': 9.23, 'MDL': 16.87, 'MGA': 4597.91, 'MKD': 52.6
7, 'MMK': 2103, 'MNT': 3559.6, 'MOP': 8.03, 'MRU': 39.97, 'MUR': 46.62, 'MVR': 1
5.44, 'MWK': 1745.37, 'MXN': 17.99, 'MYR': 4.07, 'MZN': 63.59, 'NAD': 16.5, 'NG
N': 1423.19, 'NIO': 36.8, 'NOK': 10.1, 'NPR': 144.33, 'NZD': 1.75, 'OMR': 0.384,
'PAB': 1, 'PEN': 3.36, 'PGK': 4.26, 'PHP': 59.29, 'PKR': 282.12, 'PLN': 3.62, 'PY
G': 6713.89, 'QAR': 3.64, 'RON': 4.37, 'RSD': 100.62, 'RUB': 79.45, 'RWF': 1459.7
7, 'SAR': 3.75, 'SBD': 8.02, 'SCR': 14.29, 'SDG': 510.54, 'SEK': 9.22, 'SGD': 1.2
9, 'SHP': 0.746, 'SLE': 23.84, 'SLL': 23836.61, 'SOS': 570.19, 'SRD': 38.18, 'SS
P': 4691.39, 'STN': 21.05, 'SYP': 114.02, 'SZL': 16.5, 'THB': 31.42, 'TJS': 9.23,
'TMT': 3.5, 'TND': 2.89, 'TOP': 2.39, 'TRY': 43.12, 'TTD': 6.76, 'TVD': 1.5, 'TW
D': 31.64, 'TZS': 2449.69, 'UAH': 43.06, 'UGX': 3575.08, 'UYU': 38.96, 'UZS': 121
24.02, 'VES': 330.38, 'VND': 26198.54, 'VUV': 120.26, 'WST': 2.74, 'XAF': 563.64,
'XCD': 2.7, 'XCG': 1.79, 'XDR': 0.731, 'XOF': 563.64, 'XPF': 102.54, 'YER': 238.
4, 'ZAR': 16.5, 'ZMW': 19.69, 'ZWG': 25.72, 'ZWL': 25.72}}
```

2. Data Handling and Analysis

Python, with libraries like Pandas, provides powerful tools for data manipulation and analysis that are more flexible and efficient than Excel.

```
In [2]: # Example: Data manipulation with Pandas
        import pandas as pd

        # Creating a DataFrame
        data = {'Date': ['2021-01-01', '2021-01-02', '2021-01-03'],
                'Price': [100, 101, 102]}
        df = pd.DataFrame(data)
        print(df)

        # Calculating the percentage change
        df['Pct Change'] = df['Price'].pct_change()
        print(df)
```

```
         Date  Price
0  2021-01-01    100
1  2021-01-02    101
2  2021-01-03    102
         Date  Price  Pct Change
0  2021-01-01    100         NaN
1  2021-01-02    101    0.010000
2  2021-01-03    102    0.009901
```

3. Powerful Numerical Computation

NumPy is a Python library that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. This is more efficient than Excel for numerical computations.

In [3]:
```
# Example: Numerical computation with NumPy
import numpy as np

# Creating an array
prices = np.array([100, 101, 102, 103, 104])

# Calculating the log returns
log_returns = np.log(prices[1:] / prices[:-1])
print(log_returns)
```

```
[0.00995033 0.0098523  0.00975617 0.00966191]
```

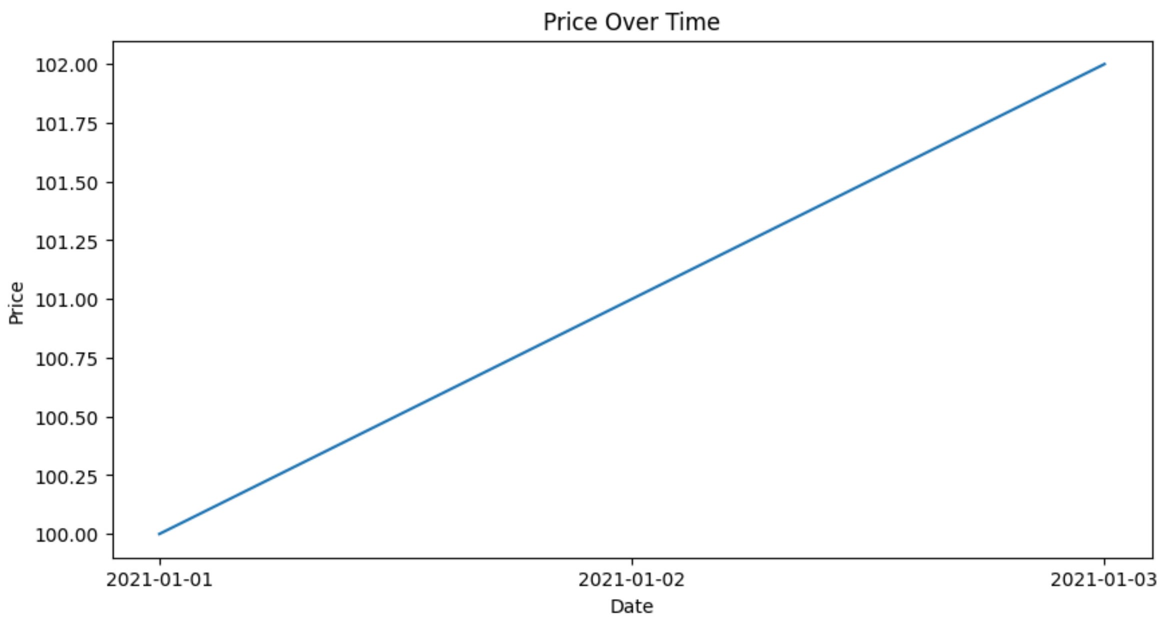
4. Powerful and Flexible Data Visualization

Python libraries, Matplotlib and Seaborn powerful in creating a wide range of static, animated, and interactive visualizations. They offer more customization and flexibility compared to Excel charts.

In [4]:
```
# Example: Data visualization with Matplotlib and Seaborn
import matplotlib.pyplot as plt
import seaborn as sns

# Sample data
data = {'Date': ['2021-01-01', '2021-01-02', '2021-01-03'],
        'Price': [100, 101, 102]}
df = pd.DataFrame(data)

# Line plot
plt.figure(figsize=(10, 5))
sns.lineplot(x='Date', y='Price', data=df)
plt.title('Price Over Time')
plt.show()
```



5. Scalability

Python can handle larger datasets more efficiently than Excel, which can become slow and unwieldy with large amounts of data.

In [5]:
```python
# Example: Handling large datasets with Pandas
import numpy as np

# Creating a large DataFrame
large_data = {'A': np.random.rand(1000000), 'B': np.random.rand(1000000)}
large_df = pd.DataFrame(large_data)
print(large_df.head())
```

```
          A         B
0  0.373371  0.846992
1  0.233230  0.136690
2  0.784277  0.919052
3  0.701315  0.988682
4  0.630909  0.319793
```

6. Advanced Analytics and Machine Learning

Python has extensive libraries for advanced analytics and machine learning, such as Scikit-Learn, TensorFlow, and PyTorch, which are not available in Excel.

In [6]:
```python
# Example: Simple linear regression with Scikit-Learn
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([1, 2, 1.3, 3.75, 2.25])

# Creating and training the model
model = LinearRegression()
model.fit(X, y)

# Making predictions
```

```
predictions = model.predict(X)
print(predictions)
```

```
[1.21  1.635 2.06  2.485 2.91 ]
```

7. Reproducibility and Version Control

Python scripts can be version-controlled using Git, ensuring reproducibility and collaboration, which is more challenging with Excel files.

8. User Interfaces

Python can create interactive user interfaces using libraries such as `Streamlit`, `Dash`, and `Tkinter`. These libraries allow you to build web applications and dashboards that make data analysis more accessible and interactive.

Why GitHub?

Introduction

In today's digital age, proficiency in technology is crucial for success in any field, including finance. While GitHub is often associated with software development, it offers numerous benefits that can enhance the skill set of finance students. finma uses GitHub to store, manage and collaborate on our teaching content eventhough this is not software of even code. Similarly,research analysts can use GitHub to collaborate and manage their research production. More generally, this section outlines the reasons why finance students should consider learning GitHub, even if they are not planning on becoming professional software developers.

Version Control

GitHub provides a powerful version control system that allows you to track changes to your documents and collaborate with others seamlessly. This is particularly useful for:

- **Collaborative Projects**: Work on group assignments and ensure everyone is on the same page.
- **Research Papers**: Keep track of revisions and easily revert to previous versions if needed.

Collaboration

Collaboration is a cornerstone of GitHub's functionality, making it an ideal tool for finance students working on group projects or research. GitHub's collaboration features include:

- **Branching and Merging**: Work on different parts of a project simultaneously without interfering with each other's work. Branching allows you to create separate lines of development, and merging integrates changes back into the main project.

- **Pull Requests**: Propose changes to a project and discuss them with your team before integrating them. This ensures that all changes are reviewed and approved, maintaining the quality of the work.
- **Issues and Project Boards**: Track tasks, bugs, and enhancements using GitHub Issues. Organize and prioritize your work with project boards, making it easier to manage complex projects.
- **Content Reviews (including but not just code)**: Peer reviews are facilitated through pull requests, allowing team members to provide feedback and improve the quality of the work collaboratively.

Networking

GitHub is a platform that fosters networking and professional growth. By learning GitHub, you can:

- **Connect with Professionals**: Join a global community of professionals and gain exposure to industry best practices.
- **Showcase Your Work**: Create a portfolio of your projects and analyses, demonstrating your skills to potential employers.
- **Contribute to Open Source**: Participate in open-source projects, contributing to the community and gaining valuable experience.

Data Analysis and Automation

Finance increasingly relies on data analysis and automation. GitHub can help you:

- **Manage Code and Scripts**: Store and manage your data analysis scripts, ensuring they are organized and accessible.
- **Automate Tasks**: Use GitHub Actions to automate repetitive tasks, saving time and reducing errors.

Integration with VSCode

Visual Studio Code (VSCode) is a powerful, open-source code editor that integrates seamlessly with GitHub. The benefits of this integration include:

- **Ease of Use**: VSCode provides a user-friendly interface for managing your GitHub repositories, making it easier to commit changes, create branches, and manage pull requests.
- **Built-in Git Support**: VSCode has built-in Git support, allowing you to perform version control operations directly from the editor.
- **Extensions and Customization**: Enhance your workflow with a wide range of extensions available for VSCode, tailored to your specific needs.

GitHub Codespaces (Important and Recommended)

GitHub Codespaces is a cloud-based development environment that is fully integrated into the GitHub ecosystem. The benefits of using Codespaces include:

- **Instant Development Environments**: Spin up a development environment in seconds, pre-configured with all the tools you need.
- **Consistency**: Ensure that all team members are working in the same environment, reducing the "it works on my machine" problem.
- **Accessibility**: Access your development environment from anywhere, on any device, without the need for complex setup.
- Essentially VSCode online
- No need for local setup
- Cloud-based, can access from any device online.
- Automatically saves.
- Direct Co-pilot integration

GitHub Codespaces is available in the free tier of GitHub, but note you can join GitHub Education with your student credentials to get a pro-level of this feature and GitHub more broadly for free.

GitHub Copilot

GitHub Copilot is an AI-powered code completion tool that empowers users to do much more. This is particularly useful in extending the capability of less technical and more finance-oriented users. The benefits of GitHub Copilot include:

- **Code Suggestions**: Receive intelligent code suggestions as you type, helping you write code faster and with fewer errors.
- **Learning Aid**: Use Copilot as a learning tool to understand how to implement various coding tasks, even if you are not an expert.
- **Increased Productivity**: Focus on higher-level problem-solving while Copilot assists with the implementation details.

GitHub Copilot is a paid feature, but you can join GitHub Education with your student credentials to get this feature for free.

Professional Development

Learning GitHub can enhance your professional profile by:

- **Showcasing Your Work**: Create a portfolio of your projects and analyses, demonstrating your skills to potential employers.
- **Continuous Learning**: Stay updated with the latest tools and technologies used in the finance industry.

Conclusion

Even if you are not planning to become a professional software developer, learning GitHub can provide you with valuable skills that are highly applicable in the finance industry. Embrace the opportunity to enhance your collaboration, data management, and professional development by integrating GitHub into your skill set. The integration with VSCode, the availability of Codespaces, and the power of GitHub Copilot further amplify the benefits, making GitHub an essential tool for finance students.

Why VSCode

Visual Studio Code (VSCode) is a powerful, open-source code editor developed by Microsoft. It has gained immense popularity among developers across various fields, including finance. This document outlines the reasons why VSCode is the best Integrated Development Environment (IDE) for financial modelling development.

User-Friendly Interface

- **Intuitive Design**: VSCode offers a clean and intuitive user interface that is easy to navigate, even for beginners. The layout is customizable, allowing you to arrange panels and tabs according to your workflow.
- **Integrated Terminal**: The built-in terminal allows you to run scripts and commands directly within the IDE, streamlining your development process.

Extensive Language Support

- **Multi-Language Support**: VSCode supports a wide range of programming languages, including Python, R, and SQL, which are commonly used in financial modelling. This makes it a versatile choice for finance professionals.
- **Syntax Highlighting and Autocompletion**: The editor provides syntax highlighting and autocompletion for various languages, enhancing code readability and reducing errors.

Powerful Extensions

- **Rich Extension Ecosystem**: VSCode has a vast library of extensions available through the Visual Studio Code Marketplace. These extensions can enhance your productivity and add new functionalities tailored to financial modelling.
  - **Python Extension**: Provides features like IntelliSense, linting, debugging, and Jupyter Notebook support.
  - **Excel Viewer**: Allows you to view and edit Excel files directly within VSCode.
  - **SQL Tools**: Integrates SQL databases, enabling you to run queries and manage databases seamlessly.

Integration with GitHub

- **Version Control**: VSCode integrates seamlessly with GitHub, allowing you to manage your repositories, commit changes, and handle pull requests directly from the IDE. This is crucial for collaborative financial modelling projects.

- **GitHub Codespaces**: With GitHub Codespaces, you can create cloud-based development environments that are pre-configured with all the tools you need. This ensures consistency and accessibility across different devices.

Debugging and Testing

- **Built-in Debugger**: VSCode comes with a powerful built-in debugger that supports multiple languages. You can set breakpoints, inspect variables, and step through your code to identify and fix issues efficiently.
- **Testing Frameworks**: The IDE supports various testing frameworks, allowing you to write and run tests to ensure the accuracy and reliability of your financial models.

Collaboration Features

- **Live Share**: The Live Share extension enables real-time collaboration with your team. You can share your workspace, edit code together, and debug issues collaboratively, making it easier to work on complex financial models.
- **Comments and Annotations**: You can add comments and annotations to your code, facilitating better communication and understanding among team members.

Customizability

- **Themes and Layouts**: VSCode offers a wide range of themes and layout options, allowing you to customize the appearance and functionality of the IDE to suit your preferences.
- **Keybindings**: You can configure custom keybindings to streamline your workflow and improve efficiency.

Integration with Data Science Tools

- **Jupyter Notebooks**: VSCode supports Jupyter Notebooks, enabling you to perform data analysis and visualization within the IDE. This is particularly useful for financial modelling and exploratory data analysis.
- **Data Visualization Libraries**: You can integrate popular data visualization libraries like Matplotlib, Seaborn, and Plotly to create interactive charts and graphs.

GitHub Copilot

- **AI-Powered Code Completion**: GitHub Copilot, an AI-powered code completion tool, is integrated into VSCode. It provides intelligent code suggestions, helping you write code faster and with fewer errors. This is especially beneficial for finance professionals who may not have extensive programming experience.

Conclusion

Visual Studio Code (VSCode) stands out as the best IDE for financial modelling development due to its user-friendly interface, extensive language support, powerful extensions, seamless integration with GitHub, robust debugging and testing capabilities, collaboration features, customizability, and integration with data science tools. The addition of GitHub Copilot further enhances its appeal by empowering users with AI-driven code suggestions. For finance professionals looking to develop accurate and efficient financial models, VSCode is the ideal choice.

Installation and Setup

Setting up a GitHub account

This course requires you to have and use a GitHub account to complete projects. In addition we recommend that you use Codespaces within your GitHub account to do you work. GitHub will also be where you access the code base that will form a starting point for your projects. It will also be the platform on which you formally collaborate with your team members in completing your assignment together.

If you don't already have a github account, you can sign up for a free account.

Creating your first GitHub repository and launching a Codespaces for that repository

GitHub Codespaces provides a cloud-based development environment that comes pre-configured with VS Code. This eliminates the need for local installation and ensures consistent development environments across different machines.

Step 1: Create a New Repository

1. Navigate to GitHub and sign in to your account
2. Click the **"+"** icon in the top-right corner and select **"New repository"**
3. Fill in the repository details:
   - **Repository name**: Choose a meaningful name (e.g., `finma-FM-working`)
   - **Description**: Optional, but helpful (e.g., "General repo for testing python and jupyter concepts")
   - **Visibility**: Choose **Public** or **Private** based on your preference
   - **Initialize repository**: Check the box for **"Add a README file"** (recommended)
   - Optionally add a `.gitignore` template (select **Python**)
   - Optionally choose a license (e.g., MIT License)
4. Click **"Create repository"**

Step 2: Launch GitHub Codespaces

1. From your newly created repository page, click the green **"Code"** button
2. Select the **"Codespaces"** tab
3. Click **"Create codespace on main"** (or the name of your default branch)

4. Wait for the environment to build (this may take 1-2 minutes on first launch)
5. Once loaded, you'll have a fully functional VS Code environment in your browser

Step 3: Understanding Your Codespaces Environment

Your Codespaces environment includes:

- **Explorer**: File browser on the left sidebar
- **Editor**: Central area for writing code
- **Terminal**: Access via the bottom panel or `Terminal > New Terminal` menu

Important Notes

- **Free tier**: GitHub provides free Codespaces usage hours for personal accounts (check current limits on GitHub)
- **Automatic save**: Codespaces automatically saves your work
- **Stop when done**: Remember to stop your Codespace when not in use to conserve free hours (it will auto-stop after 30 minutes of inactivity)
- **Access anywhere**: You can access your Codespace from any device with a web browser

Installing Python and Jupyter on your Codespaces virtual machine

Once your Codespace is running, you need to install the Python and Jupyter extensions to enable full Python development and notebook capabilities.

Installing the Python Extension

1. Click the **Extensions** icon in the left sidebar (or press `Ctrl+Shift+X` / `Cmd+Shift+X`)
2. In the search bar, type **"Python"**
3. Look for the extension by **Microsoft** (it should be the first result with a blue checkmark)
4. Click **"Install"**
5. Wait for the installation to complete (typically 10-30 seconds)

Installing the Jupyter Extension

1. In the Extensions panel, search for **"Jupyter"**
2. Select the **Jupyter** extension by **Microsoft**
3. Click **"Install"**
4. This will also install related extensions like "Jupyter Keymap" and "Jupyter Notebook Renderers"

Verifying the Installation

After installing both extensions:

1. Create a new file with a `.py` extension or open an existing Python file
2. You should see Python version information in the bottom status bar
3. Create or open a `.ipynb` notebook file - you should be able to run cells interactively

Installing Python Itself

If Python is not already installed on your Codespace:

1. Open the terminal (`Terminal > New Terminal`)
2. Run the following commands:
   ```
   sudo apt update
   sudo apt install python3 python3-pip -y
   ```
3. Verify installation:
   ```
   python3 --version
   pip --version
   ```

Installing Python Packages

To install some of the python the packages you will need for your work:

Using the terminal:

```
pip install pandas numpy matplotlib requests yfinance scipystats
seaborn
```

Note that you can also install these packages one-by-one. For example:

```
pip install yfinance
```

It is a very easy process, as you can see, to install both extensions (such as Python and Jupyter) and python packages such as numpy. So don't feel you need to have everything you can think of installed up front. You can simply install as you need things.

Optional: Local machine installation of Python and VSCode

Python is the language. VSCode is the Integrated Development Environment (IDE)

Windows

Start by downloading and installing Python from the Python website.

https://www.python.org/downloads/

1. 'Download the latest version for Windows'
2. Run the executable download file.
3. Make sure to select Add python.exe to PATH option upon opening the installer
4. Click Install Now

Next we want to get Visual Studio Code (VSCode). This is a commonly used source code editor in industry and the one we will be using in this course.

https://go.microsoft.com/fwlink/?LinkID=534107

1. Download and run the installer with default settings
2. After downloading, run VSCode.

Mac

Macs come with Python pre-installed, however it is likely an old version. To check the version, enter the following in your terminal.

`$ python --version`

Check the message. If the version is 2.xxx or below 3.10, you should install a new version using the below steps.

https://www.python.org/downloads/

1. Download the latest version for Mac. This will be the MacOS 64 Bit universal installer.
2. Run the downloaded file.
3. Follow all prompts - it is recommended to keep the default install location.

To install VSCode, please follow the below link.

https://code.visualstudio.com/?wt.mc_id=vscom_downloads#alt-downloads

1. Download the appropriate installer for your Mac (Intel or ARM). If you aren't sure, choose the universal installer.
2. Extract the downloaded ZIP folder and run the installer.
3. Follow any prompts and open VSCode.

Installing pip

`pip` is the package installer for Python, allowing you to install and manage additional libraries and dependencies that are not included in the standard Python library.

It simplifies the process of downloading and installing packages from the Python Package Index (PyPI) and other repositories. With `pip`, you can easily add functionality to your Python projects by installing packages with a single command, such as `pip install package-name`. It also supports managing package versions, enabling you to specify and install specific versions of packages to ensure compatibility and stability in your projects. Additionally, `pip` can generate and use requirements files, which list all the dependencies for a project, making it easier to share and replicate environments across different systems.

We need to ensure we have `pip` installed. The process you will follow depends on if you utilise Windows or Mac.

Windows

Installed by default.

Mac

Macs now come with Python pre-installed, however this version may be outdated. Earlier, you would've installed the latest version of Python, however your Pip is likely outdated. To see if you have Pip installed, run the following in your Mac terminal:

```
pip --version
```

If this gives you a version number and directory, you have Pip installed. To ensure it's up to date, run the following in your terminal:

```
python3 -m pip install --upgrade pip
```

If you did not have Pip installed (no version could be found), follow these steps:

Open the Mac terminal and enter:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
```

Once the above has run, enter:

```
python3 get-pip.py
```

Installing Libraries

To install a library in Python, you can either utilise the terminal directly, or access the terminal through a Jupyter notebook using `!`. Installing through a Jupyter notebook is as follows: `!python -m pip install package`

To install from the terminal, you would enter: `$ python -m pip install package`

We have listed the libraries we will be using in this course below. Please run the following cell to install them on to your machine.

```
In [ ]: !python -m pip install numpy
        !python -m pip install pandas
        !python -m pip install matplotlib
        !python -m pip install yfinance
        !python -m pip install scipystats
        !python -m pip install seaborn
        !python -m pip install requests
```

Setting up and using Jupyter notebooks in Python

Typically, Python files are stored in a `file.py` format. This is great for writing full applications, but not so great for learning Python where you will be working with small snippets of code.

In this course, we will be using Jupyter Notebooks, which are saved as `file.ipynb`.

Jupyter notebooks are structured with 'cells' that can be run one-by-one:

1. **Coding cells**: where the user can type and execute Python code.
2. **Markdown cells**: where the user can type text in markdown format.

To allow VSCode to understand how to read `.ipynb` files, we want to install the Jupyter Notebook extension. The easiest way to do this is to get VSCode to try and open an `.ipynb` file, which will automatically prompt the user to install the Jupyter Notebook library.

1. Make sure the Explorer menu is open in VSCode (either press CTRL/Command + Shift + E, or select the icon with the two pages at the top left of the screen)

2. Set your workspace folder by selecting 'Open Folder' and navigating to an empty folder on your hard drive.

3. Rick click the explorer pane on the left and select 'New File'

4. Name the file `notebook.ipynb`

5. Select `notebook.ipynb` and VSCode should prompt you to install the Jupyter Notebook extension. Proceed with installation.

6. These pre-reading materials are also available in `.ipynb` format. Now that we have Jupyter Notebooks working, we recommend following the remainder of the pre-reading materials in the `.ipynb` format. This will let you run the code cells yourself and install any necessary libraries. To do so, copy the pre-reading.ipynb into the current working directory folder and it will appear in the explorer pane. Click to open.

Navigating a Jupyter Notebook

Jupyter Notebooks are split into Code and Markdown Cells.

Creating Cells

To create a cell click on the `+ Code` or `+ Markdown` button at the top of the Window

Editing Cells

To edit a **code** cell, simply click into it as if its a text box and start typing away.

To edit a **markdown** cell, click on the text and press `Enter`

Executing Cells

To run a **code** cell, either:

- Press the Execute Cell (Triangle shaped) button to the left of the code cell; or
- Press `CTRL or Command + Enter`

Once we're finished editing a **markdown** cell, either:

- Press the tick button on the top right of the cell; or

- Press `CTRL or Command + Enter`

Resources to learn Python, VSCode, Jupyter Notebooks, and GitHub

We have created a separate Introduction to Python Jupyter notebook for you which covers many of the key Python knowledge areas that you will need for this course. And a lot of your learning will be "on the job" - extending the basics when you do the assignment. That said, we recommend a more comprensive approach to learning Python as well as VSCode, Jupyter notebooks and GitHub in order to develop a higher level of proficiency and do well.

There are excellent resources all over the web for learning the tools. Here are some suggestions that cover Python, Jupyter notebooks, and GitHub that have mostly been crafted for an audience that uses VSCode:

Visual Studio Code for Education

- Introduction to Python course.
  - This course is good if you are new to programming. It offers a comprehensive introduction to Python, featuring structured modules in a ready-to-code browser-based development environment.

Official VSCode Documentation:

- Python in Visual Studio Code.
  - This guide covers installation, configuration, and basic usage of Python in VSCode.
- Working with Jupyter Notebooks in Visual Studio Code.
  - This guide covers installation, configuration, and basic usage of Jupyter Notebooks in VSCode.
- Using Git in VS Code
  - This guide covers the basics of using Git and GitHub within VSCode, including cloning repositories, committing changes, and managing branches.

Microsoft Learn:

- Python in Visual Studio Code.
  - A comprehensive learning path that includes tutorials and hands-on exercises.
- Use Jupyter Notebooks in Visual Studio Code.
  - A comprehensive learning path that includes tutorials and hands-on exercises.
- Introduction to GitHub in Visual Studio Code
  - A comprehensive learning path that includes tutorials and hands-on exercises for using GitHub with VSCode.

Real Python:

- Python Development in Visual Studio Code (Setup Guide).

    - An in-depth guide on setting up and using VSCode for Python development.
- Jupyter Notebooks in VS Code
    - An in-depth guide on setting up and using Jupyter Notebooks in VSCode.

YouTube Channels:

- **Tech With Tim**: VSCode for Python Programming
- **Tech With Tim**: Jupyter Notebooks in VS Code
- **Microsoft Visual Studio Code**: Getting Started with Jupyter Notebooks in VS Code
- **Microsoft Visual Studio Code**: Getting Started with Git in VS Code
- **Traversy Media**: Git & GitHub Crash Course For Beginners
    - while not specific to VSCode, it provides a solid foundation for understanding Git and GitHub.

VSCode Python Extension:

- Visual Studio Code Marketplace - Python Extension. The extension page includes documentation, tutorials, and links to additional resources.

VSCode GitHub Extension:

- Visual Studio Code Marketplace - GitHub Pull Requests and Issues
    - The extension page includes documentation, tutorials, and links to additional resources for managing GitHub pull requests and issues directly within VSCode.

GitHub Learning Lab:

- Introduction to GitHub
- An interactive course that teaches you the basics of GitHub, which you can apply within VSCode.

GitHub Docs:

- GitHub Docs - GitHub and Visual Studio Code
    - Official GitHub documentation on using GitHub with VSCode, including setup and common workflows.