Module 1: Setting Up Your Financial Modeling Environment

Lesson 1: Getting Started with VS Code

What is Visual Studio Code?

Visual Studio Code (VS Code) is a powerful, free code editor that has become the tool of choice for modern financial analysts and quants. Unlike Excel, VS Code allows you to:

- Write reproducible, scalable financial models
- Track changes with version control (Git)
- Automate repetitive tasks
- Handle large datasets efficiently
- Create interactive visualizations
- Collaborate with teams seamlessly

Why VS Code for Financial Modeling?

Advantages over Excel:

1. **Scalability**: Handle millions of rows without crashing
2. **Automation**: Run models with a single click
3. **Version Control**: Track every change to your models
4. **Transparency**: Code is self-documenting and auditable
5. **Integration**: Connect to databases, APIs, and cloud services
6. **Speed**: Calculations run orders of magnitude faster

Why Python? (And What About Other Tools?)

This course focuses on **Python** because it's the most accessible and versatile language for financial modeling. Python strikes the perfect balance between ease of learning and professional capability - exactly what you need at PE Club.

That said, quants and financial professionals use various tools:

- **Python** ⭐ **(This Course)** - Best all-around choice for data analysis, modeling, and automation
- **R** - Statistical analysis and econometrics (popular in academia and quant research)
- **SQL** - Database queries and data extraction (essential for working with financial databases)
- **C++/C#** - High-frequency trading and performance-critical systems
- **MATLAB** - Quantitative research (still used in some hedge funds and academic settings)
- **Julia** - Emerging high-performance language for computational finance
- **Excel/VBA** - Still widely used in traditional finance (you already know this!)

Why we're starting with Python:

- Most in-demand skill for modern finance roles
- Gentlest learning curve for beginners

- 80% of what you need for PE analysis
- Largest community and best resources
- Perfect foundation for learning other languages later

Future courses could cover: SQL for financial databases, R for advanced statistics, or building high-performance trading systems. But Python first - it's the gateway to everything else.

Installation Guide

Step 1: Install VS Code

1. **Download VS Code**
    - Visit: https://code.visualstudio.com/
    - Download for Windows (User Installer, 64-bit)
    - Run the installer (.exe file)
    - **Important**: Check "Add to PATH" during installation
    - Check "Create a desktop icon" for easy access
2. **Launch VS Code**
    - Open from Desktop or Start Menu
    - Pin to Taskbar for easy access

Step 2: Install Python

1. **Check if Python is installed**
    - Open VS Code
    - Go to Terminal → New Terminal (or press Ctrl+`)
    - Type: `python --version`
    - If you see a version number (3.8+), you're good to go
2. **Install Python (if needed)**
    - Download from: https://www.python.org/downloads/
    - Choose Python 3.11 or later (64-bit recommended)
    - Run the installer (.exe)
    - **CRITICAL**: Check "Add Python to PATH" before installing
    - Also check "Install pip"
    - Click "Install Now"
3. **Verify installation**

    ```
    python --version
    pip --version
    ```

    Note: On Windows, use `python` and `pip` (not `python3` and `pip3`)

Step 3: Essential VS Code Extensions

Install these extensions for financial modeling:

1. **Python** (by Microsoft)
    - Click Extensions icon (left sidebar)
    - Search "Python"
    - Install the official Microsoft Python extension
    - Provides IntelliSense, debugging, and Jupyter support
2. **Jupyter** (by Microsoft)
    - Search "Jupyter"
    - Install for notebook support
    - Essential for interactive financial analysis
3. **Excel Viewer** (by GrapeCity)
    - View and edit Excel files in VS Code
    - Useful for comparing Excel models to Python models
4. **GitLens** (by GitKraken)
    - Advanced Git integration
    - Track changes to your models
5. **Pylance** (by Microsoft)
    - Advanced Python language support
    - Better autocomplete and type checking

To install extensions:

- Click the Extensions icon (Ctrl+Shift+X on Windows)
- Search for each extension
- Click "Install"

Step 4: Set Up Python Environment

What is a Python Virtual Environment? A virtual environment is like a separate workspace for your Python projects. It keeps your financial modeling libraries separate from other Python projects, preventing conflicts and making your work reproducible.

Opening the VS Code Terminal:

1. **Open VS Code Terminal** using one of these methods:
    - **Menu:** Go to `Terminal` → `New Terminal` at the top menu bar
    - **Keyboard Shortcut:** Press `Ctrl+`` (that's Ctrl and the backtick key, usually above Tab)
    - **Command Palette:** Press `Ctrl+Shift+P`, type "Terminal", select "Terminal: Create New Terminal"

2. **You should see a terminal panel** appear at the bottom of VS Code
    - It looks like a command prompt window
    - You'll see a path (like `C:\Users\YourName\...`) followed by a cursor
3. **Choose Your Terminal Type:**
    - VS Code might ask which terminal to use: Choose **PowerShell** (recommended for Windows)
    - You can see which terminal you're using in the dropdown at the top-right of the terminal panel

Now, let's create your Python environment:

Step 4a: Create a Project Folder

In the terminal, type these commands **one at a time**, pressing Enter after each:

```
# Create a folder for your financial modeling projects
mkdir financial-modeling

# Navigate into that folder
cd financial-modeling
```

What just happened?

- `mkdir` created a new folder called "financial-modeling"
- `cd` changed your location to inside that folder
- You should see the path in your terminal change to include "financial-modeling"

Step 4b: Create the Virtual Environment

Now type this command (it will take 10-30 seconds to complete):

```
# Create a virtual environment called "venv"
python -m venv venv
```

What's happening?

- Python is creating a folder called `venv` with its own copy of Python
- You'll see some activity, then the command will finish
- No error messages = success! ✅

Step 4c: Activate the Virtual Environment

This is the most important step - you need to "turn on" your virtual environment:

```
# Activate the environment
.\venv\Scripts\Activate.ps1
```

How to know it worked:

- You should see `(venv)` appear at the beginning of your terminal line
- Example: `(venv) PS C:\Users\YourName\financial-modeling>`
- The `(venv)` means your virtual environment is active! 🎉

Troubleshooting: PowerShell Execution Policy Error

If you see an error like: `cannot be loaded because running scripts is disabled...`

Fix it with these steps:

1. **Open PowerShell as Administrator:**
   - Press Windows key
   - Type "PowerShell"
   - **Right-click** on "Windows PowerShell"
   - Select "Run as Administrator"
   - Click "Yes" when asked for permission
2. **In the Administrator PowerShell window, type:**

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

3. **Press Enter, then type `Y` and press Enter again to confirm**
4. **Close the Administrator PowerShell window**
5. **Go back to VS Code and try activating again:**

```
.\venv\Scripts\Activate.ps1
```

VS Code Terminal Tips:

- **Clear the terminal:** Type `cls` and press Enter (clears all text)
- **Close terminal:** Click the trash can icon in the terminal panel
- **New terminal:** Click the `+` icon in the terminal panel
- **Copy from terminal:** Select text, it copies automatically (or Ctrl+C)
- **Paste into terminal:** Right-click, or Ctrl+V
- **Split terminal:** Click the split icon to have multiple terminals side-by-side

Important: Always Activate Your Environment!

Every time you:

- Open VS Code
- Start a new terminal
- Want to work on your financial models

You must run: `.\venv\Scripts\Activate.ps1`

Look for `(venv)` at the start of your terminal line - that's your confirmation!

Step 5: Install Essential Python Libraries

What are Python libraries? Libraries are pre-written code packages that add powerful capabilities to Python. Instead of building financial models from scratch, you'll use professional-grade tools that quants worldwide rely on.

Before you start:

- ✅ Make sure your virtual environment is activated (you should see `(venv)` in your terminal)
- ✅ You should still be in the `financial-modeling` folder
- ❌ If you don't see `(venv)`, run: `.\venv\Scripts\Activate.ps1`

Now, let's install the libraries:

Type or copy-paste these commands **one at a time** into your terminal, pressing Enter after each:

1. Core Data Analysis (Most Important)

```
pip install numpy pandas
```

- **numpy**: Mathematical operations (think Excel formulas on steroids)
- **pandas**: Data tables and analysis (like Excel sheets, but better)
- This will take 30-60 seconds to install
- You'll see lots of text scrolling - this is normal!

2. Financial Data

```
pip install yfinance pandas-datareader
```

- **yfinance**: Download stock prices, financial statements, and market data
- **pandas-datareader**: Access economic data from various sources

3. Visualization

```
pip install matplotlib seaborn plotly
```

- **matplotlib**: Create charts and graphs
- **seaborn**: Beautiful statistical visualizations
- **plotly**: Interactive charts for presentations

4. Excel Integration

```
pip install openpyxl xlrd
```

- **openpyxl**: Read and write modern Excel files (.xlsx)
- **xlrd**: Read older Excel files (.xls)
- Perfect for importing existing Excel models!

5. Jupyter Notebooks

```
pip install jupyter ipykernel
```

- **jupyter**: Interactive coding environment (like coding + Word document combined)
- **ipykernel**: Makes Jupyter work with your virtual environment

6. Advanced Analysis (Optional but Recommended)

```
pip install scipy scikit-learn
```

- **scipy**: Scientific computing and optimization
- **scikit-learn**: Machine learning for financial predictions

What to expect:

- Each installation shows downloading progress
- You'll see "Successfully installed..." when done
- Total time: 3-5 minutes for all libraries
- If you see warnings (yellow text) - that's usually fine
- If you see errors (red text) - double-check your virtual environment is active

Save Your Setup (Important!)

Create a "shopping list" of all installed libraries:

```
pip freeze > requirements.txt
```

What this does:

- Creates a file called `requirements.txt` with all your installed libraries
- Allows you (or Mauricio) to recreate this exact setup on any computer
- To reinstall everything later: `pip install -r requirements.txt`

Verify Everything Installed:

```
pip list
```

You should see a list of all installed packages. Look for:

- numpy
- pandas
- yfinance
- matplotlib
- jupyter

If you see these, you're ready to go! 🎉

VS Code Configuration for Finance

Why configure VS Code? These settings optimize VS Code for financial modeling - auto-saving your work, formatting code professionally, and setting up Python correctly.

Method 1: Quick Settings (Recommended for Beginners)

Step-by-step:

1. **Open Settings**
   - Click the gear icon ⚙️ in the bottom-left corner of VS Code
   - Select "Settings"
   - OR press `Ctrl+,` (Ctrl and comma)
2. **Configure these key settings:**

   Search for: "Auto Save"
   - Find "Files: Auto Save"
   - Change from "off" to "afterDelay"
   - ✅ Your work now saves automatically every few seconds!

   Search for: "Format On Save"
   - Find "Editor: Format On Save"
   - Check the box ✅
   - Your code will auto-format to look professional

   Search for: "Python Default Interpreter"
   - Find "Python › Default Interpreter Path"
   - Enter: `.\venv\Scripts\python.exe`
   - This tells VS Code to use your virtual environment

   Search for: "Terminal Profile Windows"
   - Find "Terminal › Integrated › Default Profile: Windows"
   - Select "PowerShell" from dropdown
   - Ensures consistent terminal experience

Method 2: Advanced Settings (Optional)

For users comfortable with JSON:

1. Press `Ctrl+Shift+P`
2. Type "Preferences: Open User Settings (JSON)"
3. Add these configurations:

```
{
    "python.defaultInterpreterPath": ".\\venv\\Scripts\\python.exe",
    "editor.formatOnSave": true,
    "python.formatting.provider": "black",
    "editor.rulers": [88],
    "files.autoSave": "afterDelay",
    "editor.minimap.enabled": true,
    "workbench.colorTheme": "Default Dark Modern",
    "terminal.integrated.defaultProfile.windows": "PowerShell"
}
```

Setting Up Jupyter Notebooks

What is a Jupyter Notebook? Think of it as a living document where you can mix:

- Python code that runs
- Results and charts
- Notes and explanations (like a financial model memo)

Perfect for building and presenting financial models!

Create Your First Notebook:

1. **Create a new file:**
    - Press `Ctrl+N` (new file)
    - Press `Ctrl+S` (save)
    - Name it: `test.ipynb` (the `.ipynb` extension is critical!)
    - Choose to save it in your `financial-modeling` folder
2. **Select Python Kernel:**
    - VS Code will show a popup: "Select a kernel"
    - Click "Python Environments..."
    - Choose the one that shows `venv` or `('venv': venv)`
    - If you don't see this popup, click "Select Kernel" in the top-right corner
3. **What you'll see:**
    - An empty "cell" (looks like a text box)
    - A "+ Code" button above it
    - A "+ Markdown" button (for adding notes)

4. **Test Your Setup:**

   In the first cell, type this code:

```
import numpy as np
import pandas as pd

print("Financial Modeling Environment Ready!")
print(f"NumPy version: {np.__version__}")
print(f"Pandas version: {pd.__version__}")
```

5. **Run the code:**

   - Click the ▶ play button to the left of the cell
   - OR press `Shift+Enter`
   - You should see output appear below the cell!

6. **Success looks like:**

```
Financial Modeling Environment Ready!
NumPy version: 1.26.x
Pandas version: 2.1.x
```

Jupyter Tips:

- Each cell runs independently - like Excel cells, but for code
- Add new cells with the `+ Code` button
- Add notes/explanations with `+ Markdown`
- Cells remember what you ran previously (useful for building models step-by-step)

Your First Financial Calculation in VS Code

Let's build something real! You'll create a Python script that calculates present value - a core finance concept. This replaces what you'd do in Excel with NPV formulas.

Step 1: Create the File

1. **Make sure you're in the right folder:**

   - In VS Code, click "File" → "Open Folder"
   - Navigate to and open your `financial-modeling` folder
   - You should see `venv` folder in the sidebar

2. **Create a subfolder for Module 1:**

   - Right-click in the Explorer panel (left sidebar)
   - Select "New Folder"
   - Name it: `Module_01_Setup`

3. **Create your Python file:**
    - Right-click on `Module_01_Setup` folder
    - Select "New File"
    - Name it: `test_environment.py`
    - The file will open in the editor

Step 2: Write Your First Financial Model

Copy this code into `test_environment.py`:

```
"""
Test your financial modeling environment
This calculates the present value of future cash flows
"""

import numpy as np
import pandas as pd
from datetime import datetime

# Simple DCF calculation
def present_value(cash_flow, discount_rate, year):
    """Calculate present value of a future cash flow"""
    return cash_flow / (1 + discount_rate) ** year

# Example: Value a series of cash flows
cash_flows = [100, 110, 121, 133, 146]  # Growing at 10%
discount_rate = 0.12
years = range(1, 6)

pv_cash_flows = [present_value(cf, discount_rate, year)
                 for cf, year in zip(cash_flows, years)]

print("Cash Flow Valuation Example")
print("-" * 40)
print(f"Discount Rate: {discount_rate:.1%}")
print(f"\nYear | Cash Flow | Present Value")
print("-" * 40)

for year, cf, pv in zip(years, cash_flows, pv_cash_flows):
    print(f"{year:4} | ${cf:8.2f} | ${pv:13.2f}")

total_pv = sum(pv_cash_flows)
print("-" * 40)
print(f"Total Present Value: ${total_pv:,.2f}")
```

Step 3: Run Your Code

Method 1: Play Button (Easiest)

- Look for the ▶ play button in the top-right corner of VS Code

- Click it
- The terminal will open and show your results

Method 2: Right-Click Menu

- Right-click anywhere in your code
- Select "Run Python File in Terminal"

Method 3: Keyboard Shortcut

- Press `Ctrl+Alt+N` (if you have Code Runner extension)

What You Should See:

```
Cash Flow Valuation Example
----------------------------------------
Discount Rate: 12.0%

Year | Cash Flow | Present Value
----------------------------------------
   1 |  $100.00 |       $89.29
   2 |  $110.00 |       $87.70
   3 |  $121.00 |       $86.15
   4 |  $133.00 |       $84.63
   5 |  $146.00 |       $82.87
----------------------------------------
Total Present Value: $430.64
```

What just happened?

- You imported professional financial libraries (numpy, pandas)
- Created a function to calculate present value (like a custom Excel formula)
- Ran the calculation on 5 years of cash flows
- Got formatted output that would take multiple Excel formulas

Congratulations! You just replaced an Excel model with Python code! 🎉

Workspace Organization

Create this folder structure for your course:

```
financial-modeling/
│
├── venv/                          # Virtual environment
├── Module_01_Setup/               # This module
├── Module_02_Python_Fundamentals/ # Python basics
├── Module_03_Data_Analysis/       # Data handling
├── Module_04_DCF_Modeling/        # DCF models
├── Module_05_LBO_Modeling/        # LBO models
├── Module_06_MA_Analysis/         # M&A analysis
```

```
├── Module_07_PE_Modeling/          # PE models
├── Module_08_Advanced_Topics/      # Advanced techniques
├── Module_09_Projects/             # Real-world projects
│
├── data/                           # Sample datasets
├── templates/                      # Model templates
├── outputs/                        # Generated reports
│
├── requirements.txt                # Python dependencies
└── README.md                       # Course overview
```

Keyboard Shortcuts (Windows)

Master these VS Code shortcuts:

Shortcut	Action
Ctrl+P	Quick file open
Ctrl+Shift+P	Command palette
Ctrl+B	Toggle sidebar
Ctrl+J	Toggle terminal
Ctrl+/	Comment/uncomment
Alt+↑/↓	Move line up/down
Shift+Alt+↑/↓	Copy line up/down
Ctrl+D	Select next occurrence
Ctrl+F	Find
Ctrl+Shift+F	Find in files
F5	Run debugger
Ctrl+`	Toggle terminal

Git Setup (Version Control)

What is Git? Git is like "Track Changes" in Word, but for code. It saves every version of your financial models, lets you experiment without fear, and enables collaboration at PE Club.

Why use Git for finance?

- 📊 See exactly what changed in your model (and when)
- ⬅️ Undo mistakes - go back to any previous version
- 🤝 Collaborate with team members on the same model
- 📝 Audit trail - know who changed what (important for compliance)

Setting Up Git (Optional but Highly Recommended)

Step 1: Check if Git is Installed

In your terminal, type:

```
git --version
```

- If you see a version number: ✅ Git is installed, proceed to Step 2
- If you see an error: Install Git from https://git-scm.com/download/win

Step 2: Configure Git (First Time Only)

Tell Git who you are:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Replace with your actual name and email.

Step 3: Initialize Git in Your Project

Make sure you're in your `financial-modeling` folder, then:

```
# Initialize Git tracking
git init
```

You'll see: `Initialized empty Git repository...` ✅

Step 4: Tell Git What NOT to Track

Create a `.gitignore` file to exclude unnecessary files:

```
# Create .gitignore file (PowerShell)
@"
venv/
__pycache__/
*.pyc
.DS_Store
.ipynb_checkpoints/
outputs/
"@ | Out-File -FilePath .gitignore -Encoding utf8
```

What this ignores:

- `venv/` - Virtual environment (can be recreated)
- `__pycache__/` - Python temporary files

- `*.pyc` - Compiled Python files
- `.ipynb_checkpoints/` - Jupyter auto-saves
- `outputs/` - Generated reports (track code, not outputs)

Step 5: Make Your First Commit (Save Point)

```
# Stage all files for commit
git add .

# Create your first save point
git commit -m "Initial setup: Financial modeling environment"
```

What just happened:

- `git add .` - Staged all your files (like selecting files to save)
- `git commit -m "..."` - Created a save point with a description
- Your entire project is now versioned! 🎉

Git Basics You'll Use Daily:

```
# See what changed
git status

# Save changes
git add .
git commit -m "Built DCF model for Company X"

# View history
git log --oneline

# Undo changes to a file (before commit)
git checkout -- filename.py
```

Don't worry if this seems complex! You'll learn Git gradually. For now, just remember:

- `git add .` then `git commit -m "description"` to save your progress

Troubleshooting Common Issues

Problem 1: "Python not found" or "python is not recognized"

Symptoms: When you type `python --version`, you get an error

Solutions:

1. Python might not be in your PATH:
    - Reinstall Python from python.org
    - ⚠ CRITICAL: Check "Add Python to PATH" during installation

- Restart your computer after installation

2. Try the `py` command instead:

```
py --version
```

If this works, use `py` instead of `python` in all commands

3. Close and reopen VS Code after installing Python
4. Open a **new** terminal window (click the + icon in terminal panel)

Problem 2: Virtual Environment Won't Activate

Symptoms: When you run activation command, `(venv)` doesn't appear

Solutions:

1. **Make sure you're in the right folder:**

```
# Check current location
pwd

# You should see a path ending in "financial-modeling"
# If not, navigate to it:
cd path\to\financial-modeling
```

2. **Use the correct activation command for your terminal:**

PowerShell (Recommended):

```
.\venv\Scripts\Activate.ps1
```

Command Prompt:

```
venv\Scripts\activate
```

3. **PowerShell execution policy error:**

If you see: `cannot be loaded because running scripts is disabled...`

Fix:

- Open PowerShell as Administrator (Right-click → Run as Administrator)

- Run: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`
- Type `Y` and press Enter
- Close Administrator PowerShell
- Try activating again in VS Code

4. **Virtual environment doesn't exist:**

```
# Recreate it:
python -m venv venv
```

Problem 3: Extensions Not Working

Symptoms: Python extension installed but IntelliSense not working, or Jupyter not opening

Solutions:

1. **Reload VS Code:**
   - Press `Ctrl+Shift+P`
   - Type "Developer: Reload Window"
   - Press Enter
2. **Check extension is enabled:**
   - Click Extensions icon (Ctrl+Shift+X)
   - Search for "Python"
   - Make sure it says "Disable" (meaning it's currently enabled)
   - If it says "Enable", click to enable it
3. **Update extensions:**
   - In Extensions panel, click the "..." menu at the top
   - Select "Check for Extension Updates"
4. **Completely restart VS Code:**
   - File → Exit (or Alt+F4)
   - Reopen VS Code
5. **Select the correct Python interpreter:**
   - Press `Ctrl+Shift+P`
   - Type "Python: Select Interpreter"
   - Choose the one showing `('venv': venv)`

Problem 4: Import Errors ("No module named...")

Symptoms: When running code, you see `ModuleNotFoundError: No module named 'pandas'`

Solutions:

1. **Check virtual environment is activated:**
   - Look for `(venv)` at the start of your terminal line
   - If missing, run: `.\venv\Scripts\Activate.ps1`
2. **Reinstall the missing package:**

```
pip install pandas
```

3. **Reinstall all packages:**

```
pip install -r requirements.txt
```

4. **Check which Python is being used:**

```
# In terminal with (venv) active:
python -c "import sys; print(sys.executable)"

# Should show path with 'venv' in it
# If not, your virtual environment isn't properly activated
```

5. **VS Code using wrong Python:**
   - Click on Python version in bottom-left status bar
   - Select your `venv` Python interpreter

Problem 5: Jupyter Notebook Kernel Won't Start

Symptoms: "Failed to start kernel" or "Kernel is dead"

Solutions:

1. **Install ipykernel in your virtual environment:**

```
pip install ipykernel
```

2. **Select the correct kernel:**
   - In notebook, click "Select Kernel" (top-right)
   - Choose Python Environments → Select your venv
3. **Restart VS Code and try again**

Problem 6: Git Commands Not Working

Symptoms: `git: command not found` or `git is not recognized`

Solutions:

1. **Install Git:**
   - Download from: https://git-scm.com/download/win
   - Install with default settings
   - Restart VS Code
2. **Check Git is installed:**

```
git --version
```

Problem 7: Permission Denied Errors

Symptoms: "Permission denied" or "Access is denied"

Solutions:

1. **Close any programs using the files** (especially Excel, other code editors)
2. **Run VS Code as Administrator** (right-click → Run as Administrator)
3. **Check folder isn't read-only:**
   - Right-click folder in File Explorer
   - Properties → Uncheck "Read-only"
   - Apply to all subfolders

Still Stuck?

Best debugging approach:

1. Copy the exact error message
2. Google: `"your error message" vscode python`
3. Check Stack Overflow results (usually has solutions)
4. Or ask GitHub Copilot Chat: "I'm getting this error: [paste error]"

Remember: Everyone encounters these issues when starting. You're learning a professional toolset - it's worth the initial setup time!

Next Steps

You're now ready to start building financial models in Python!

Practice Exercise:

1. Create a new Jupyter notebook: `my_first_model.ipynb`
2. Import numpy and pandas
3. Create a simple time value of money calculator
4. Calculate the future value of $10,000 invested for 5 years at 8% annual return

Continue to: `Module_02_Python_Fundamentals/01_Python_Basics.md`

Additional Resources

- VS Code Documentation: https://code.visualstudio.com/docs
- Python for Finance: https://www.python.org/about/gettingstarted/
- Pandas Documentation: https://pandas.pydata.org/docs/
- Financial Modeling Best Practices: See `resources/` folder

Checkpoint: By completing this lesson, you should have:

- ✅ VS Code installed and configured
- ✅ Python environment set up
- ✅ Essential extensions installed
- ✅ Successfully run your first financial calculation
- ✅ Understood the advantages of VS Code over Excel

Estimated Time: 45-60 minutes