Tutorial 2: GitHub & Copilot Hands-On 🚀

> **Personal Note to Mauricio**: Hey champion! 👋 Remember Tutorial 1? That was the foundation. Now we're building something HUGE: you'll learn version control (how professionals track code changes) and AI pair programming (having an AI write code WITH you!). At PE Club Brussels, when they see you using Git and Copilot, they'll know you're the real deal! - Dad ❤️

📊 Progress Tracker

Where you are in the course:

- ✅ Tutorial 1: VS Code Basics (Complete!)
- 🔥 **Tutorial 2: GitHub & AI Coding (You are here!)**
- ⬜ Tutorial 3: Data Analysis with Python
- ⬜ Tutorial 4: Building DCF Models
- ⬜ Tutorial 5: Power User Skills

Course completion: 20% → 40% 🎯

🎯 What You'll Learn (90 minutes)

By the end of this tutorial, you'll be able to:

✅ Version Control (Git & GitHub)

- [ ] Create and manage GitHub repositories
- [ ] Track changes to your code (like "Track Changes" in Word, but way better!)
- [ ] Save checkpoints (commits) - like save points in video games!
- [ ] Push code to cloud (backup + collaboration)
- [ ] Never lose work again!

✅ AI-Assisted Coding (GitHub Copilot)

- [ ] Set up your AI pair programmer
- [ ] Write code 10x faster with AI suggestions
- [ ] Build financial calculators with AI help
- [ ] Understand and modify AI-generated code
- [ ] Look like a coding wizard! 🧙

✅ Real-World Skills

- [ ] Professional Git workflow (how tech companies work)
- [ ] Collaborate like a developer
- [ ] Build financial tools with AI assistance

💡 **Why This Matters at PE Club Brussels**: Every serious tech-forward finance firm uses Git. When you say "I pushed my model to GitHub," you're speaking the language of modern finance. Copilot will make you code faster than analysts who've been coding for years. This is your competitive edge! 🚀

📝 Prerequisites

Before starting:

- ✅ Completed Tutorial 1 (VS Code Basics)
- ✅ VS Code installed and comfortable
- ✅ Python working
- ✅ Know basic keyboard shortcuts (`Ctrl+P`, `Ctrl+S`, `Ctrl+``)

New Requirements (we'll set up today!):

- ⬜ GitHub account (we'll create this!)
- ⬜ Git installed on Windows (we'll do this!)
- ⬜ GitHub Copilot activated (we'll set this up!)

⏱️ **Estimated Time**: 90 minutes (take breaks!) ☕ **Suggested**: Have coffee/water ready. This is a longer session!

Part 1: Setting Up GitHub - Your Code's Home in the Cloud (15 minutes)

> 💡 **What is GitHub?** Think of it like Google Drive for code. You save your code there, track all changes, and can access it from any computer. At PE Club, you'll share models with teammates through GitHub!

Step 1: Create Your GitHub Account

Follow these steps carefully:

1. **Go to**: https://github.com

2. **Click**: "Sign up" (top right corner)

3. **Enter**:

    - Email: Use your professional email (not temporary!)
    - Username: Choose wisely (potential employers see this!)
        - ✅ Good: `mauricio-gonzalez`, `mau-finance`, `mauricio-pe`
        - ❌ Bad: `coolkid123`, `partyboy`, `temp-account`
    - Password: Strong password (GitHub is professional portfolio!)

4. **Verify** your email (check inbox/spam)

5. **Choose FREE plan** (perfect for learning!)

🖼️ **[Screenshot: GitHub signup page with annotations]**

✅ **Success check**: Can you log into github.com and see your profile? If yes → Continue!

Step 2: GitHub Student Developer Pack (FREE Copilot!)

🎓 If you're a student or educator, this is AMAZING:

What you get FREE:

- ✅ GitHub Copilot (normally $10/month!)
- ✅ GitHub Pro features
- ✅ $100s worth of developer tools
- ✅ Cloud credits for AWS, Azure
- ✅ Access to premium courses

How to get it:

1. **Go to**: https://education.github.com/students

2. **Click**: "Sign up for Student Developer Pack"

3. **Prepare documents** (one of these):

    - Student ID card (photo)
    - School transcript (PDF)
    - Enrollment letter (official)
    - School email address (.edu address helps!)

4. **Upload proof** and wait for approval

    - Usually takes 1-3 days
    - Check email for approval notification

5. **Once approved**:

    - GitHub Copilot activates automatically!
    - Check: https://github.com/settings/copilot
    - You'll see "You have access to GitHub Copilot"

🚨 If you're NOT a student:

- Don't worry! GitHub Copilot has 30-day FREE trial
- After trial: $10/month (worth it if you code regularly!)
- At PE Club, your firm might pay for it!

🖼️ [Screenshot: GitHub Education benefits page]

Step 3: Install Git on Windows

What is Git?

- Git = Version control system (tracks changes locally on your PC)
- GitHub = Cloud service (stores your Git repositories online)
- **Analogy**: Git is like "Track Changes" in Word. GitHub is like saving that Word doc to OneDrive.

Installation steps:

1. **Download Git**:
   - Go to: https://git-scm.com/download/win
   - Click "Click here to download" (64-bit version)
   - Save file: `Git-2.xx.x-64-bit.exe`
2. **Run installer** (double-click downloaded file)
3. **IMPORTANT: Choose these options** (installer has MANY screens!):

   Screen: "Select Components"

   - ✅ Windows Explorer integration
   - ✅ Git Bash Here
   - ✅ Git GUI Here
   - ✅ Git LFS (Large File Support)
   - ✅ Associate .sh files to be run with Bash

   Screen: "Choosing the default editor"

   - ⚠ CHANGE from Vim to: **"Use Visual Studio Code as Git's default editor"**
   - (Scroll down in dropdown to find VS Code!)

   Screen: "Adjusting your PATH environment"

   - ✅ Select: "Git from the command line and also from 3rd-party software"

   Screen: "Choosing HTTPS transport backend"

   - ✅ Select: "Use the OpenSSL library"

   Screen: "Configuring the line ending conversions"

   - ✅ Select: "Checkout Windows-style, commit Unix-style line endings"

   Screen: "Configuring the terminal emulator"

   - ✅ Select: "Use MinTTY (the default terminal of MSYS2)"

   Screen: "Choose the default behavior of 'git pull'"

   - ✅ Select: "Default (fast-forward or merge)"

   Screen: "Choose a credential helper"

   - ✅ Select: "Git Credential Manager"

   Screen: "Configuring extra options"

   - ✅ Enable file system caching
   - ✅ Enable symbolic links
4. **Click "Install"** and wait (~1 minute)

5. **Finish** and close installer

🖼️ **[Screenshot: Key installer screens with correct selections highlighted]**

Step 4: Verify Git Installation

Open VS Code:

1. Press `Ctrl+`` (open terminal)
2. Type this command:

   ```
   git --version
   ```

3. **Expected output**:

   ```
   git version 2.43.0 (or similar)
   ```

✅ **If you see version number → Git installed successfully!**

❌ **If you see "git is not recognized":**

- Close VS Code completely
- Reopen VS Code
- Try again
- Still not working? See troubleshooting section below!

Step 5: Configure Git (Tell Git Who You Are)

Why? Every time you save code (commit), Git records WHO made the change. This is critical for teamwork!

Open VS Code terminal (`Ctrl+``) and run these commands:

```
# Set your name (will appear on all your commits)
git config --global user.name "Mauricio Gonzalez"

# Set your email (MUST match your GitHub email EXACTLY!)
git config --global user.email "mauricio@example.com"

# Set VS Code as Git's editor (makes Git use VS Code for messages)
git config --global core.editor "code --wait"

# Set default branch name to 'main' (modern standard)
git config --global init.defaultBranch main

# Make output colorful (easier to read!)
git config --global color.ui auto
```

⚠ **CRITICAL**: Replace `"Mauricio Gonzalez"` and `"mauricio@example.com"` with YOUR actual name and GitHub email!

Verify your configuration:

```
git config --list
```

Expected output (abbreviated):

```
user.name=Mauricio Gonzalez
user.email=mauricio@example.com
core.editor=code --wait
init.defaultbranch=main
color.ui=auto
...
```

✅ **Checkpoint**:

- [ ] Can you see `git version` when running `git --version`?
- [ ] Did `git config --list` show your name and email?
- [ ] Does your email match your GitHub account exactly?

If 3/3 checked → Git is ready! Let's create your first repository! 🎉

🚨 Troubleshooting: Git Installation Issues

Problem 1: "git is not recognized as a command"

Symptoms: After installing Git, terminal says `'git' is not recognized`

Solution 1 (Restart):

1. Close VS Code completely (File → Exit)
2. Reopen VS Code
3. Open terminal (`Ctrl+``)
4. Try `git --version` again

Solution 2 (Check PATH):

1. Windows key → Search "Environment Variables"
2. Click "Edit the system environment variables"
3. Click "Environment Variables" button
4. Under "System variables", find "Path"
5. Click "Edit"
6. Look for entry containing: `C:\Program Files\Git\cmd`
7. If missing:

    - Click "New"
    - Add: `C:\Program Files\Git\cmd`
    - Click OK on all windows
8. **Restart VS Code**

Solution 3 (Reinstall):

- Uninstall Git (Windows Settings → Apps → Git → Uninstall)
- Download again from git-scm.com
- Install with VS Code as editor option!

Problem 2: Wrong email configured

Symptoms: Configured email doesn't match GitHub

Fix:

```
# Change email to correct one
git config --global user.email "correct.email@example.com"

# Verify
git config --global user.email
```

Problem 3: Git Bash instead of PowerShell

Symptoms: Terminal opens Git Bash but you want PowerShell

Fix:

1. In VS Code, click terminal dropdown (says "bash" or "powershell")
2. Click "Select Default Profile"
3. Choose "PowerShell"
4. Click `+` to create new terminal with PowerShell

Part 2: Your First Repository - Save Your Code! (20 minutes)

> 💡 **What is a repository?** A folder tracked by Git. Every file change is recorded. Think of it like a project folder with superpowers - you can see all past versions, who changed what, and when!

Understanding Git Workflow (Before We Start)

The three states of files in Git:

```
┌─────────────┐      ┌─────────────┐      ┌─────────────┐
│  Working    │      │   Staging   │      │ Repository  │
│  Directory  │─────→│   Area      │─────→│ (Committed) │
│             │ add  │             │commit│             │
│ Your files  │         │ Staged files│         │ Saved forever│
└─────────────┘         └─────────────┘         └──────────────┘
```

Simple explanation:

1. **Working Directory**: Files you're editing right now
2. **Staging Area**: Files ready to be saved (like items in shopping cart)
3. **Repository**: Permanently saved snapshot (like completed purchase)

Commands you'll use:

- `git add` → Move files to staging area
- `git commit` → Save staging area to repository permanently
- `git push` → Upload repository to GitHub cloud

💡 **Analogy for PE work**:

- Working = Draft Excel model
- Staging = Final version ready to email
- Commit = Archived version with date stamp
- Push = Upload to team SharePoint

Step 1: Create Local Project Folder

Open VS Code terminal (`Ctrl+``):

```
# Navigate to your Documents folder
cd Documents

# Create project folder
mkdir my-first-finance-repo

# Enter the folder
cd my-first-finance-repo

# Verify you're in the right place
pwd
```

Expected output:

```
C:\Users\Mauricio\Documents\my-first-finance-repo
```

🎯 **Alternative (Visual Method)**:

1. Open File Explorer
2. Go to Documents

3. Right-click → New → Folder
4. Name it: `my-first-finance-repo`
5. In VS Code: File → Open Folder → Select `my-first-finance-repo`

Step 2: Initialize Git Repository

Make this folder a Git repository:

~~~~
```bash
# Initialize Git (creates hidden .git folder)
git init
~~~~

Expected output:

```
Initialized empty Git repository in C:/Users/Mauricio/Documents/my-first-
finance-repo/.git/
```

✅ **Success!** Your folder is now a Git repository!

🔍 **What just happened?**

- Git created a hidden `.git` folder
- This folder tracks ALL changes you make
- You can now save "checkpoints" (commits)

🎯 **Verify it worked:**

```
# Check Git status
git status
```

Expected output:

```
On branch main
No commits yet
nothing to commit (create/copy files and commit them)
```

Step 3: Create Your First Files

Let's create a simple financial calculator!

Create file 1: `README.md`

Press `Ctrl+N`, type this, save as `README.md`:

~~~
# My First Financial Calculator

A simple Python calculator for financial calculations.

## Features
- Present value calculation
- Future value calculation
- Compound annual growth rate (CAGR)

## How to Use
```bash
python calculator.py
```

## Author

Mauricio - Learning Git and Python!

## Created

November 2025 - PE Club Brussels

~~~

Create file 2: `calculator.py`

Press `Ctrl+N`, type this, save as `calculator.py`:

```python
"""
Simple Financial Calculator
My first Git project!
Created for PE Club Brussels
"""

def present_value(future_value, rate, years):
    """
    Calculate present value

    Args:
        future_value: Cash flow in the future
        rate: Discount rate (as decimal, e.g., 0.10 for 10%)
        years: Number of years

    Returns:
        Present value
    """
    return future_value / (1 + rate) ** years

def future_value(present_value, rate, years):
    """
    Calculate future value
```

```
    Args:
        present_value: Cash flow today
        rate: Growth rate (as decimal)
        years: Number of years

    Returns:
        Future value
    """
    return present_value * (1 + rate) ** years

# Test the functions
if __name__ == "__main__":
    print("=" * 40)
    print("   FINANCIAL CALCULATOR v1.0")
    print("=" * 40)

    # Example: What's $10,000 in 5 years worth today at 10% discount rate?
    fv = 10000
    rate = 0.10
    years = 5

    pv = present_value(fv, rate, years)
    print(f"\nPresent Value: ${pv:,.2f}")
    print(f"Future Value: ${fv:,.2f}")
    print(f"Discount Rate: {rate:.1%}")
    print(f"Years: {years}")
```

🎯 **Test it works:**

```
python calculator.py
```

Expected output:

```
========================================
   FINANCIAL CALCULATOR v1.0
========================================

Present Value: $6,209.21
Future Value: $10,000.00
Discount Rate: 10.0%
Years: 5
```

✅ **If you see output → Great! Let's save this to Git!**

Step 4: Your First Git Commit - Save Your Work!

Check what Git sees:

```
git status
```

Output:

```
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        calculator.py

nothing added to commit but untracked files present
```

📍 **What does this mean?**

- 🔴 **Red files** = Git sees them but isn't tracking them yet
- They're "untracked" - not saved in Git yet

Step 5: The Git Workflow - ADD → COMMIT → PUSH

Remember this magic formula:

```
📝 EDIT FILES → 🎯 ADD → 💾 COMMIT → ☁️ PUSH
```

Step 5a: ADD (Stage files for commit)

```
# Add all files to staging area
git add .

# Check status again
git status
```

Output now:

```
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   calculator.py
```

📍 **What changed?**

- ✅ **Green files** = Staged and ready to commit!
- Files are in "shopping cart", ready to "purchase" (commit)

Step 5b: COMMIT (Save snapshot permanently)

```
# Commit with descriptive message
git commit -m "Initial commit: Create financial calculator"
```

Output:

```
[main (root-commit) a1b2c3d] Initial commit: Create financial calculator
 2 files changed, 42 insertions(+)
 create mode 100644 README.md
 create mode 100644 calculator.py
```

🎉 **You just made your first commit!**

📍 **What happened?**

- Git took a "snapshot" of your files
- Saved with message "Initial commit: Create financial calculator"
- Created unique ID (a1b2c3d) for this checkpoint
- You can ALWAYS come back to this exact state!

Verify:

```
git status
```

Output:

```
On branch main
nothing to commit, working tree clean
```

✅ **"Clean working tree" = All changes saved! Perfect state!**

See your commit history:

```
git log
```

Output:

```
commit a1b2c3d4e5f6g7h8i9j0 (HEAD -> main)
Author: Mauricio Gonzalez <mauricio@example.com>
Date:   Sat Nov 30 14:30:00 2025 +0100

    Initial commit: Create financial calculator
```

💡 **Pro tip**: Write clear commit messages! Think "If I need to find this change in 6 months, what would I search for?"

Step 6: Create GitHub Repository (Put Your Code in the Cloud!)

Now let's upload this to GitHub!

Go to GitHub:

1. **Open browser** → https://github.com
2. **Click** + (top right corner) → "New repository"
3. **Fill in**:
   - Repository name: `my-first-finance-repo`
   - Description: `Learning Git with financial calculator - PE Club Brussels`
   - **Visibility**:
     - ✅ **Private** (for learning/practice)
     - Or **Public** (if you want to show off! 😎)
   - ⚠ **DON'T check** "Initialize with README" (we already have one!)
   - **DON'T** add .gitignore or license yet
4. **Click "Create repository"**

🖼️ **[Screenshot: GitHub new repository page with correct settings]**

✅ **Repository created!** GitHub now shows you setup instructions.

Step 7: Connect Local Repository to GitHub

GitHub shows you commands - let's use them!

In VS Code terminal:

```
# Add GitHub as "remote" (cloud destination)
git remote add origin https://github.com/YOUR-USERNAME/my-first-finance-
repo.git
```

```
# ⚠ IMPORTANT: Replace YOUR-USERNAME with your actual GitHub username!
```

Verify it worked:

```
git remote -v
```

Output:

```
origin  https://github.com/YOUR-USERNAME/my-first-finance-repo.git (fetch)
origin  https://github.com/YOUR-USERNAME/my-first-finance-repo.git (push)
```

📍 **What is "origin"?**

- "origin" = Nickname for your GitHub repository
- Like saving a bookmark with a short name
- You can have multiple remotes (origin, backup, etc.)

Step 8: PUSH to GitHub - Upload Your Code!

The moment of truth!

```
# Push your commits to GitHub
git push -u origin main
```

What happens:

1. Git asks for GitHub credentials (first time only)
2. Browser opens for authentication
3. You authorize VS Code
4. Git uploads your code!

Expected output:

```
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.2 KiB | 1.2 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/YOUR-USERNAME/my-first-finance-repo.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

🎉 **SUCCESS! Your code is now on GitHub!**

Visit your repository:

```
https://github.com/YOUR-USERNAME/my-first-finance-repo
```

You should see:

- ✅ Your README.md displayed
- ✅ calculator.py file
- ✅ Your commit message
- ✅ Green "Code" button

📸 **Take a screenshot and send to Dad! First GitHub repo! 🎉**

Step 9: Understanding What Just Happened

Your code exists in 3 places now:

```
┌─────────────────┐
│  Your Computer  │  ← Working files (edit here)
│  (Local Repo)   │
└─────────────────┘
        │ git push
        ↓
┌─────────────────┐
│     GitHub      │  ← Cloud backup (access anywhere)
│  (Remote Repo)  │
└─────────────────┘
```

The full workflow:

```
1. Edit files locally
2. git add .               (stage changes)
3. git commit -m "..."     (save checkpoint)
4. git push                (upload to GitHub)
```

✅ **Checkpoint Questions:**

- [ ] Can you create files and commit them?
- [ ] Do you understand add → commit → push workflow?
- [ ] Can you see your code on GitHub?
- [ ] Can you explain what "origin" means?

If 3/4 checked → You understand Git basics! Amazing progress! 🚀

🚨 Troubleshooting: Git & GitHub Issues

Problem 1: "git push" asks for username/password repeatedly

Cause: HTTPS authentication not cached

Solution 1 (Use Git Credential Manager):

```
git config --global credential.helper manager-core
```

Solution 2 (Use SSH instead of HTTPS):

1. Generate SSH key: https://docs.github.com/en/authentication/connecting-to-github-with-ssh
2. Change remote to SSH:

```
git remote set-url origin git@github.com:YOUR-USERNAME/my-first-finance-
repo.git
```

Problem 2: "Permission denied" when pushing

Cause: Not authenticated with GitHub

Fix:

1. `git push` → Opens browser
2. Sign in to GitHub
3. Authorize "Git Credential Manager"
4. Try `git push` again

Problem 3: "Updates were rejected"

Full error:

```
! [rejected]        main -> main (fetch first)
error: failed to push some refs
```

Cause: GitHub has commits you don't have locally

Fix:

```
# Download GitHub changes first
git pull origin main

# Then push yours
git push origin main
```

Problem 4: Can't find repository on GitHub

Check:

1. Correct URL? `https://github.com/YOUR-USERNAME/my-first-finance-repo`
2. Logged into correct account?
3. Repository is private? (need to be logged in to see it!)

Part 3: Setting Up GitHub Copilot - Your AI Coding Partner! (10 minutes)

> 💡 **What is Copilot?** Think of it as having a senior Python developer sitting next to you, suggesting code as you type. At PE Club, while others struggle with syntax, you'll have AI writing code FOR you! 🤖

Step 1: Install Copilot Extensions in VS Code

Open Extensions panel:

1. Press `Ctrl+Shift+X` (or click Extensions icon on left sidebar)
2. Search: `GitHub Copilot`
3. Find **"GitHub Copilot"** by GitHub (verified publisher ✓)
4. Click **"Install"**
5. Wait for installation (~30 seconds)

Also install Copilot Chat:

6. Search: `GitHub Copilot Chat`
7. Find **"GitHub Copilot Chat"** by GitHub
8. Click **"Install"**

🖼️ **[Screenshot: VS Code Extensions with Copilot installed]**

Step 2: Sign In to GitHub Copilot

After installation:

1. **Look for notification**: "Sign in to use GitHub Copilot"
2. **Click "Sign in"**
3. **Browser opens** → You may need to authorize
4. **Authorize "GitHub Copilot"** in browser
5. **Return to VS Code**

Alternative method (if notification doesn't appear):

1. Click Copilot icon in status bar (bottom right) - looks like "><"
2. Click "Sign in to GitHub"
3. Follow authentication flow

Step 3: Verify Copilot is Active

Check status bar (bottom right corner):

✅ **Good signs:**

- Copilot icon ("><") visible
- Icon is NOT red
- Hover over icon → "GitHub Copilot is active"

❌ **Problems:**

- Red X on icon = Not authenticated
- No icon = Extension not installed
- Gray icon = Inactive for this file type

Test Copilot is working:

1. Create new file: `test_copilot.py`
2. Type this comment:

   ```
   # Function to calculate compound interest
   ```

3. Press `Enter`
4. **Wait 1-2 seconds...**
5. **See gray text suggestion?** That's Copilot! ✨
6. Press `Tab` to accept suggestion

If you see gray suggestions → Copilot is working! 🎉

Step 4: Understanding Copilot Suggestions

Copilot shows suggestions in gray text:

```python
# You type this:
def calculate_npv

# Copilot suggests (in gray):
def calculate_npv(cash_flows, discount_rate):
    """Calculate Net Present Value"""
    npv = 0
    for i, cf in enumerate(cash_flows):
        npv += cf / (1 + discount_rate) ** (i + 1)
    return npv
```

How to use suggestions:

- `Tab` → Accept entire suggestion

- `Ctrl+→` → Accept next word only
- `Esc` → Reject suggestion
- `Alt+]` → Next suggestion (if multiple)
- `Alt+[` → Previous suggestion

🎯 **Try it:**

1. Type: `# Function to calculate IRR`
2. Wait for suggestion
3. Press `Alt+]` to cycle through different suggestions
4. Pick your favorite with `Tab`!

Step 5: Copilot Chat - Your AI Finance Tutor

Open Copilot Chat:

```
Ctrl+Shift+I    (or click chat icon in sidebar)
```

Try these questions:

1. **Ask**: `Explain how to calculate WACC`
   - Copilot explains in plain English!
2. **Ask**: `Write a function to calculate WACC with all parameters`
   - Copilot writes the code!
3. **Ask**: `What's the difference between NPV and IRR?`
   - Learn finance concepts!
4. **Ask**: `How do I read an Excel file in Python?`
   - Get instant code examples!

💡 **Pro tip**: Be specific! Instead of "help with DCF", ask "write a Python function to calculate DCF terminal value using perpetuity growth method"

🖼️ **[Screenshot: Copilot Chat answering a finance question]**

✅ Copilot Setup Checkpoint:

- [ ] Copilot icon visible in status bar?
- [ ] Can you see gray code suggestions when typing?
- [ ] Can you accept suggestions with Tab?
- [ ] Can you open Copilot Chat with Ctrl+Shift+I?
- [ ] Did Copilot answer a finance question?

If 4/5 checked → Copilot is ready! Time to code with AI! 🚀

🚨 Troubleshooting: Copilot Issues

Problem 1: "Copilot is not available"

Symptoms: Red X on Copilot icon

Solutions:

1. **Check subscription**:
   - Go to: https://github.com/settings/copilot
   - Verify you have active subscription or Student Pack
2. **Re-authenticate**:
   - Click Copilot icon → "Sign out"
   - Click again → "Sign in"
   - Complete authentication
3. **Restart VS Code**:
   - Close completely
   - Reopen
   - Check icon again

Problem 2: No suggestions appearing

Symptoms: Type code, but no gray suggestions

Causes & Fixes:

Cause 1: File type not supported

- ✅ Works: `.py`, `.js`, `.java`, `.cpp`, etc.
- ❌ Doesn't work: `.txt`, `.md` (limited support)
- **Fix**: Make sure file has `.py` extension!

Cause 2: Copilot disabled for this file

- Check status bar - might say "Copilot: Disabled"
- Click Copilot icon → "Enable Completions for [language]"

Cause 3: Internet connection

- Copilot needs internet!
- Check connection
- Try again

Cause 4: VS Code window not focused

- Click into VS Code window
- Try typing again

Problem 3: Suggestions are irrelevant/bad

This is normal! Copilot isn't perfect.

How to get better suggestions:

1. **Write better comments**:
    - ❌ `# calc npv`
    - ✅ `# Calculate Net Present Value using discount rate for array of cash flows`
2. **Give context**:

```
# Financial Calculator for PE Club Brussels
# Focus on LBO and DCF models

def calculate_npv(cash_flows, discount_rate):
    # Copilot now knows context!
```

3. **Use type hints**:

```
def calculate_npv(cash_flows: List[float], discount_rate: float) ->
float:
    # Copilot uses type info for better suggestions!
```

Problem 4: Copilot Chat not responding

Fix:

1. Check internet connection
2. Close and reopen chat panel
3. Restart VS Code
4. Check GitHub status: https://www.githubstatus.com

Part 4: Your First Code with Copilot - AI Magic! (20 minutes)

> 💡 **Goal**: Let AI write financial functions FOR you. You'll be amazed! This is how modern developers work - they guide AI, review code, and focus on logic instead of syntax.

Exercise 1: Copilot Completes Simple Functions

Create new file: `time_value_money.py`

Step 1: Type this comment and press `Enter`:

```
# Function to calculate net present value of an array of cash flows
```

Step 2: **Wait 1-2 seconds...**

Step 3: **See gray suggestion appear!**

Step 4: Press `Tab` to accept

You should get something like:

```python
def calculate_npv(cash_flows, discount_rate):
    """
    Calculate Net Present Value of cash flows

    Args:
        cash_flows: List of cash flows (negative for outflows, positive
for inflows)
        discount_rate: Discount rate as decimal (e.g., 0.10 for 10%)

    Returns:
        Net present value
    """
    npv = 0
    for i, cf in enumerate(cash_flows):
        npv += cf / (1 + discount_rate) ** i
    return npv
```

🤯 **COPILOT JUST WROTE AN ENTIRE FUNCTION!**

🎯 **Test it works:**

```python
# Add at bottom of file:
if __name__ == "__main__":
    # Example: Initial investment of -100K, then cash inflows
    cash_flows = [-100000, 30000, 35000, 40000, 45000]
    rate = 0.12

    npv = calculate_npv(cash_flows, rate)
    print(f"NPV at {rate:.0%} discount rate: ${npv:,.2f}")
```

Run it: `python time_value_money.py`

Expected output:

```
NPV at 12% discount rate: $8,417.91
```

```
```

✅ **It works! Copilot wrote production-ready code!** 🎉

Exercise 2: Let Copilot Write Multiple Functions

In same file, type each comment below (press Enter after each, Tab to accept):

```
# Function to calculate internal rate of return using Newton-Raphson
method
```

Wait... Tab to accept

```
# Function to calculate compound annual growth rate (CAGR)
```

Wait... Tab

```
# Function to calculate monthly mortgage payment
```

Wait... Tab

```
# Function to calculate bond price given yield to maturity
```

Wait... Tab

🎉 **Copilot wrote 4+ financial functions in minutes!**

What would take you hours, Copilot does in seconds!

Exercise 3: Using Copilot Chat for Complex Tasks

Open Copilot Chat: `Ctrl+Shift+I`

Ask Copilot:

```
Write a Python function to calculate WACC (Weighted Average Cost of
Capital).

Include parameters for:
- Market value of equity
- Market value of debt
- Cost of equity (using CAPM: risk-free rate, beta, market return)
- Cost of debt (interest rate)
- Tax rate

Include full docstring and type hints.
```

Copilot responds with complete code!

Copy the code into a new file `wacc_calculator.py`

🎯 **Try asking**:

- "Explain this WACC function line by line"
- "Add error handling for negative values"
- "Create a test case with example data"

Copilot does it all!

Exercise 4: Build Complete Calculator with Copilot

This is the BIG test!

Create new file: `investment_analyzer.py`

Type this detailed comment at the top:

```python
# Complete Investment Analysis Calculator for PE Club Brussels
#
# Required Features:
# 1. Time Value of Money - PV, FV, NPV calculations
# 2. Investment Returns - IRR, MOIC, CAGR calculations
# 3. Loan Calculations - Payment, amortization schedule
# 4. Bond Pricing - Price and YTM calculations
# 5. DCF Valuation - Terminal value, enterprise value
#
# Requirements:
# - Use type hints for all functions
# - Include comprehensive docstrings
# - Add input validation and error handling
# - Include example usage in main section
# - Make it user-friendly with formatted output

import numpy as np
from typing import List, Tuple, Optional
from datetime import datetime
```

Now press Enter and WAIT...

Copilot will suggest an ENTIRE program!

- Class structure

- All methods
- Documentation
- Error handling
- Test cases

Press Tab repeatedly to accept each section!

🤯 **IN 5 MINUTES, YOU HAVE A COMPLETE FINANCIAL TOOLKIT!**

This would take a junior developer DAYS to write from scratch!

Exercise 5: Review and Learn from Copilot Code

⚠ **IMPORTANT: Always understand the code!**

Pick one of Copilot's functions and:

1. **Read it line by line**
2. **Ask Copilot Chat**: `Explain this function: [paste function]`
3. **Add comments** explaining complex parts
4. **Test it** with different inputs
5. **Modify it** to fit your needs

Example - Understanding NPV function:

Ask Copilot Chat:

```
Explain this NPV function and tell me:
1. Why do we enumerate cash_flows?
2. What does ** i do?
3. What if discount_rate is negative?
4. How would I handle monthly cash flows?
```

Copilot teaches you! 🎓

✅ Copilot Coding Checkpoint:

- [ ] Can Copilot complete functions from comments?
- [ ] Can you accept/reject suggestions?
- [ ] Can Copilot Chat explain code?
- [ ] Can Copilot generate entire programs?
- [ ] Do you test and understand generated code?

If 4/5 checked → You're an AI-assisted developer! The future is here! 🚀

Make Changes and Commit

1. **Edit `calculator.py`** - Add a new function:

   ```
   # Let Copilot write this:
   # Function to calculate loan amortization schedule
   ```

2. **Check what changed:**

   ```
   git status
   ```

   Shows: `modified: calculator.py`

3. **See detailed changes:**

   ```
   git diff calculator.py
   ```

   Shows exactly what changed (green = added, red = removed)

4. **Stage and commit:**

   ```
   git add calculator.py
   git commit -m "Add loan amortization function"
   ```

5. **Push to GitHub:**

   ```
   git push
   ```

View History

```
# See all commits
git log

# See last 3 commits (prettier)
git log --oneline -3

# See what changed in last commit
git show
```

Undo Changes (Before Commit)

```
# Make a mistake in calculator.py
# Add this line: print("OOPS! Delete me!")
```

To undo:

```
# Discard changes to file
git checkout -- calculator.py
```

File reverts to last committed version!

Part 6: Working with Branches (10 minutes)

Why Use Branches?

- Experiment without breaking main code
- Work on features separately
- Safe to try new ideas

Create and Use Branch

```
# Create new branch
git branch add-dcf-calculator

# Switch to it
git checkout add-dcf-calculator

# Or do both at once
git checkout -b add-dcf-calculator
```

Work on Branch

Create new file: `dcf_calculator.py`

```
# Ask Copilot: "Create a complete DCF calculator class"
```

Commit on branch:

```
git add dcf_calculator.py
git commit -m "Add DCF calculator"
git push -u origin add-dcf-calculator
```

Merge Branch

```
# Switch back to main
git checkout main

# Merge feature branch
git merge add-dcf-calculator

# Push merged code
git push
```

Delete Branch (After Merging)

```
git branch -d add-dcf-calculator
```

Part 7: Real Project with Copilot (20 minutes)

Build: Complete Financial Analysis Tool

Create: `financial_toolkit.py`

Use Copilot to build this step-by-step:

Step 1: Type comment and let Copilot complete:

```python
# Complete Financial Analysis Toolkit
#
# Features:
# 1. Time Value of Money Calculations
# 2. Investment Returns (IRR, MOIC, CAGR)
# 3. Loan & Mortgage Calculations
# 4. Bond Pricing
# 5. Stock Valuation (DCF)
#
# Each function should have:
# - Type hints
# - Docstrings
# - Error handling
# - Example usage

import numpy as np
from typing import List, Tuple, Optional
```

Step 2: Let Copilot suggest the class structure:

```python
class FinancialToolkit:
    """Complete financial analysis toolkit"""

    # Copilot will suggest all methods!
```

Step 3: Add specific methods (Copilot completes each):

```python
    def calculate_pv(self, fv: float, rate: float, periods: int) -> float:
        """Calculate present value"""
        # Press Tab - Copilot writes it!

    def calculate_fv(self, pv: float, rate: float, periods: int) -> float:
        """Calculate future value"""
        # Press Tab

    def calculate_npv(self, cash_flows: List[float], discount_rate: float)
-> float:
        """Calculate NPV"""
        # Press Tab

    def calculate_irr(self, cash_flows: List[float]) -> float:
        """Calculate IRR"""
        # Press Tab
```

Step 4: Use Copilot Chat for complex parts:

Press `Ctrl+Shift+I` and ask:

```
"Add a method to calculate WACC with cost of equity using CAPM,
cost of debt, market values, and tax rate. Include full documentation."
```

Copy Copilot's response into your class!

Test Your Toolkit

```python
# Add at bottom of file:
if __name__ == "__main__":
    # Let Copilot write test code
    # Type comment: "Test all functions with example data"
```

Commit Your Work

```
git add financial_toolkit.py
git commit -m "Complete financial toolkit with Copilot assistance"
git push
```

Part 8: Copilot Best Practices (10 minutes)

1. Write Good Comments

❌ **Bad:**

```
# calculate wacc
```

✅ **Good:**

```python
# Calculate weighted average cost of capital (WACC)
# Formula: WACC = (E/V × Re) + (D/V × Rd × (1-Tc))
# where E = equity value, D = debt value, V = total value
# Re = cost of equity (CAPM), Rd = cost of debt, Tc = tax rate
def calculate_wacc(
```

Copilot generates better code with detailed comments!

2. Use Descriptive Names

```
# Copilot understands context from names
enterprise_value_to_ebitda_multiple = ...
debt_to_equity_ratio = ...
weighted_average_cost_of_capital = ...
```

3. Review Suggestions

Always:

- ✅ Read what Copilot suggests
- ✅ Understand the logic
- ✅ Test the code
- ✅ Verify financial formulas

Never:

- ❌ Blindly accept everything
- ❌ Skip testing
- ❌ Ignore errors

4. Use Copilot Chat for Learning

Great questions:

- "Explain the difference between NPV and IRR"
- "What's wrong with this DCF calculation?"
- "How do I handle circular references in financial models?"
- "Suggest improvements to this code"

5. Iterate with Copilot

```python
# First attempt
def calculate_return():
    # Basic version

# Ask Copilot Chat: "Add error handling to this function"
# Copilot suggests improvements

# Ask: "Add type hints and detailed docstring"
# Copilot enhances it further
```

Part 9: Copilot Keyboard Shortcuts

Essential Shortcuts

Shortcut	Action
`Tab`	Accept suggestion
`Esc`	Dismiss suggestion
`Alt+]`	Next suggestion
`Alt+[`	Previous suggestion
`Ctrl+Enter`	Show all suggestions panel
`Ctrl+Shift+I`	Open Copilot Chat
`Ctrl+I`	Inline Copilot chat

Practice: Cycling Through Suggestions

1. Type: `# Function to calculate mortgage payment`
2. Press `Enter`
3. Copilot suggests code
4. Press `Alt+]` to see next suggestion
5. Press `Alt+[` to go back
6. Press `Ctrl+Enter` to see all options
7. Choose the best one!

Part 10: Real-World Example (15 minutes)

Build: DCF Model with Copilot

Create: `simple_dcf.py`

Work with Copilot step-by-step:

```python
# DCF Valuation Model
# Calculate enterprise value using discounted cash flow method
#
# Inputs:
# - Projected free cash flows (5 years)
# - Discount rate (WACC)
# - Terminal growth rate
# - Current debt and cash
#
# Output:
# - Enterprise value
# - Equity value
# - Equity value per share

import pandas as pd
import numpy as np

class DCFModel:
    """Simple DCF valuation model"""

    def __init__(self, company_name: str):
        """Initialize DCF model"""
        self.company_name = company_name
        # Let Copilot suggest attributes

    def calculate_fcf(self, revenue, ebitda_margin, tax_rate,
                      capex_pct, nwc_pct, da_pct):
        """Calculate free cash flow from revenue"""
        # Let Copilot complete

    def calculate_terminal_value(self, final_fcf, wacc, growth_rate):
        """Calculate terminal value using perpetuity growth"""
        # Let Copilot complete

    def calculate_enterprise_value(self, fcf_list, wacc, terminal_value):
        """Calculate enterprise value"""
        # Let Copilot complete

    def calculate_equity_value(self, enterprise_value, debt, cash,
                               shares_outstanding):
        """Calculate equity value per share"""
        # Let Copilot complete

# Example usage - Let Copilot write this too!
```

```
if __name__ == "__main__":
    # Create model and run example valuation
```

Use Copilot Chat to enhance:

Ask in chat:

```
"Add a sensitivity analysis method that varies WACC and
terminal growth rate to create a valuation range"
```

Copilot will suggest the code!

Test and Commit

```
# Run it
python simple_dcf.py

# If works, commit
git add simple_dcf.py
git commit -m "Create DCF model with Copilot assistance"
git push
```

✅ Skills Checklist

After this tutorial, you can:

- [ ] Create GitHub account and authenticate
- [ ] Initialize Git repository
- [ ] Stage, commit, and push changes
- [ ] View commit history
- [ ] Create and merge branches
- [ ] Use GitHub Copilot for code generation
- [ ] Use Copilot Chat for explanations
- [ ] Review and verify AI suggestions
- [ ] Build financial calculators with AI help
- [ ] Maintain version control workflow

🎯 Git Commands Cheat Sheet

```
# Setup
git init                        # Initialize repository
git config --global user.name   # Set username
git config --global user.email  # Set email
```

```
# Daily Workflow
git status                      # Check status
git add .                       # Stage all changes
git add filename.py             # Stage specific file
git commit -m "message"         # Commit changes
git push                        # Upload to GitHub
git pull                        # Download from GitHub

# History
git log                         # View commits
git log --oneline              # Compact view
git diff                       # See changes
git show                       # Show last commit

# Branching
git branch                     # List branches
git branch name                # Create branch
git checkout name              # Switch branch
git checkout -b name           # Create and switch
git merge name                 # Merge branch
git branch -d name             # Delete branch

# Undo
git checkout -- file           # Discard changes
git reset HEAD file            # Unstage file
git reset --soft HEAD~1        # Undo last commit (keep changes)
git reset --hard HEAD~1        # Undo last commit (delete changes!)
```

🚀 What's Next?

🎉 **Mauricio, you just leveled up MASSIVELY!**

You now have:

- ✅ GitHub account and repository (your professional portfolio!)
- ✅ Git workflow mastery (add → commit → push like a pro!)
- ✅ GitHub Copilot AI assistant (coding superpower! 🤖)
- ✅ Financial calculators built with AI help
- ✅ Skills that 95% of finance professionals don't have!

At PE Club Brussels, you can now:

- Share models through GitHub (professional!)
- Track all changes to your work (never lose data!)
- Code 10x faster with Copilot (while others Google syntax!)
- Build financial tools in minutes (not days!)
- Look like the "technical analyst" on your team! 💼

📊 Skills Mastery Checklist

✅ GitHub & Git Skills:

- [ ] Create GitHub repositories
- [ ] Add, commit, and push changes
- [ ] Write clear commit messages
- [ ] Understand working directory vs staging vs repository
- [ ] Use branches for experiments
- [ ] Merge branches back to main
- [ ] View commit history with git log
- [ ] Undo changes when needed

✅ Copilot Skills:

- [ ] Accept/reject code suggestions
- [ ] Use Copilot Chat for learning
- [ ] Write good prompts/comments for better suggestions
- [ ] Review and understand AI-generated code
- [ ] Ask Copilot to explain complex concepts
- [ ] Generate complete functions from descriptions
- [ ] Test AI code before using it

If 12+ boxes checked → You're ready for serious development! 🚀

🎯 Practice Challenges (Do These This Week!)

Challenge 1: Build Your Finance Portfolio

Create repo: `finance-toolkit`

Include:

1. DCF calculator function
2. LBO returns calculator
3. Comparable companies analysis helper
4. Bond pricing tool

Use Copilot for all code!

Commit each function separately:

```
git add dcf_calculator.py
git commit -m "Add DCF terminal value calculator"
git push
```

Challenge 2: Contribute Daily

Goal: Make 1 commit every day for a week

Ideas:

- Day 1: Create project structure
- Day 2: Add one calculator function
- Day 3: Add tests
- Day 4: Improve documentation
- Day 5: Add error handling
- Day 6: Create examples
- Day 7: Write comprehensive README

Why?: GitHub shows your activity! Employers love to see consistent contributions! 🔥

Challenge 3: Learn from Copilot

Ask Copilot Chat these questions (learn finance + coding!):

1. "Explain the Black-Scholes formula and implement it in Python"
2. "What's the difference between NPV and IRR? Show code examples"
3. "How do I calculate Sharpe ratio for a portfolio?"
4. "Build a function to analyze profit margins from financial statements"
5. "Create a sensitivity table for DCF valuation"

Save each answer in a file, test the code, and commit!

💡 Git Command Reference Card (Print This!)

```
═══════════════════════════════════════════════════════════════
                    ESSENTIAL GIT COMMANDS
═══════════════════════════════════════════════════════════════

SETUP (One-time)
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
git config --global init.defaultBranch main

START PROJECT
git init                     # Initialize new repo
git clone <url>              # Copy existing repo

BASIC WORKFLOW
git status                   # Check what changed
git add .                    # Stage all changes
git add <file>               # Stage specific file
git commit -m "message"      # Save snapshot
git push                     # Upload to GitHub
git pull                     # Download from GitHub

VIEWING CHANGES
```

```
git log                        # See commit history
git log --oneline              # Compact history
git diff                       # See unstaged changes
git diff --staged              # See staged changes
git show                       # See last commit

BRANCHES
git branch                     # List branches
git branch <name>              # Create branch
git checkout <name>            # Switch branch
git checkout -b <name>         # Create + switch
git merge <name>               # Merge branch
git branch -d <name>           # Delete branch

UNDO CHANGES
git checkout -- <file>         # Discard file changes
git reset HEAD <file>          # Unstage file
git reset --soft HEAD~1        # Undo commit, keep changes
git reset --hard HEAD~1        # Undo commit, delete changes (⚠)

REMOTE REPOSITORIES
git remote -v                  # See remote URLs
git remote add origin <url>    # Add remote
git push -u origin main        # First push (sets upstream)
git push                       # Subsequent pushes

HELPFUL
git help <command>             # Get help
git config --list              # See all settings

═══════════════════════════════════════════════════════════════
💡 TIP: Commit early, commit often! Each commit is a save point
         you can return to. Think of it like autosave in games!
═══════════════════════════════════════════════════════════════
```

🎁 Copilot Best Practices for Finance

1. Write Descriptive Comments

❌ **Bad (vague)**:

```
# calculate wacc
```

✅ **Good (specific)**:

```
# Calculate weighted average cost of capital (WACC)
# Formula: WACC = (E/V × Re) + (D/V × Rd × (1-Tc))
# where E = market value of equity, D = market value of debt
# V = total firm value (E + D), Re = cost of equity (CAPM)
# Rd = cost of debt, Tc = corporate tax rate
```

Result: Copilot generates complete, well-documented function!

---

2. Use Type Hints

❌ **Without type hints**:

```python
def calculate_irr(cash_flows, guess):
    # Copilot less certain about types
```

✅ **With type hints**:

```python
def calculate_irr(cash_flows: List[float], guess: float = 0.1) -> float:
    # Copilot knows exactly what you need!
```

---

3. Provide Context in File Header

Start each file with context:

```python
"""
DCF Valuation Model for Technology Companies
Created for PE Club Brussels - Private Equity Analysis

This module implements a discounted cash flow valuation model
specifically designed for high-growth technology companies.

Author: Mauricio Gonzalez
Date: November 2025
"""

# Now Copilot knows to generate PE-focused code!
```

---

4. Break Down Complex Tasks

Instead of:

```python
# Build complete LBO model
```

Do this:

```
# Step 1: Create sources and uses table
# Step 2: Build debt schedule with term loan and revolver
# Step 3: Calculate cash flow waterfall
# Step 4: Compute IRR and MOIC returns
```

Copilot handles each step better!

---

5. Ask Copilot to Explain Finance Concepts

Use Copilot Chat as your finance tutor:

In Copilot Chat:

```
Explain Enterprise Value vs Equity Value with examples.
Then write Python functions to calculate each.
```

Copilot teaches + codes! 🎓

---

6. Request Error Handling

Add to your comments:

```
# Calculate NPV with error handling for:
# - Empty cash flow list
# - Negative discount rate
# - Non-numeric inputs
# Raise appropriate exceptions with clear messages
```

Copilot adds robust error checking!

---

7. Ask for Tests

After Copilot writes a function:

```
# Generate pytest test cases for calculate_wacc function
# Include edge cases: zero debt, zero equity, negative tax rate
```

Copilot writes your tests too!

🚨 Common Mistakes to Avoid

Mistake 1: Blindly Trusting Copilot

❌ **Don't**:

- Accept all suggestions without reading
- Use code you don't understand
- Skip testing AI-generated functions

✅ **Do**:

- Read every line Copilot suggests
- Test with multiple inputs
- Ask Copilot to explain confusing parts
- Modify code to fit your exact needs

Remember: Copilot is a tool, not a replacement for thinking!

Mistake 2: Committing Sensitive Data

❌ **NEVER commit**:

- Passwords or API keys
- Private financial data
- Client information
- Personal credentials

✅ **Use `.gitignore`**:

Create file: `.gitignore`

```
# Sensitive files
config/secrets.json
*.env
private_data/
api_keys.txt

# Python
__pycache__/
*.pyc
.pytest_cache/

# VS Code
.vscode/settings.json
```

Add API keys to environment variables, not code!

Mistake 3: Poor Commit Messages

❌ **Bad commits**:

```
git commit -m "fix"
git commit -m "update"
git commit -m "changes"
git commit -m "asdf"
```

✅ **Good commits**:

```
git commit -m "Fix NPV calculation for negative initial investment"
git commit -m "Add WACC calculator with tax shield adjustment"
git commit -m "Refactor DCF model to handle multiple growth periods"
git commit -m "Update documentation with LBO return examples"
```

Rule: Commit message should complete this sentence: "If applied, this commit will ___________"

Mistake 4: Not Committing Often Enough

❌ **Don't**:

- Work for hours without committing
- Commit everything at end of day
- Combine unrelated changes in one commit

✅ **Do**:

- Commit after each feature
- Commit after each bug fix
- Make small, focused commits
- Aim for 5-10 commits per work session

Why?: Easier to find bugs, undo changes, understand history!

📚 Additional Resources

Git & GitHub Learning

Official:

- GitHub Skills: https://skills.github.com
- Git Documentation: https://git-scm.com/doc
- GitHub Docs: https://docs.github.com

Interactive:

- Learn Git Branching: https://learngitbranching.js.org
- Git Cheat Sheet: https://education.github.com/git-cheat-sheet-education.pdf

Video:

- Search YouTube: "Git and GitHub for Beginners"
- GitHub's official channel has great tutorials

GitHub Copilot Resources

Official:

- Copilot Documentation: https://docs.github.com/en/copilot
- Copilot Best Practices: https://docs.github.com/en/copilot/getting-started-with-github-copilot

Tips:

- Copilot works best with common patterns
- Finance-specific prompts might need more detail
- Review and test all generated financial formulas!

Finance + Python Learning

Combine both skills:

- Ask Copilot to teach you: "Explain [finance concept] and show Python implementation"
- Build mini-projects for each concept you learn
- Commit each project to GitHub portfolio

Example progression:

1. Basic calculators (PV, FV, NPV)
2. Bond and stock valuation
3. Portfolio analysis
4. DCF models
5. LBO models
6. M&A analysis tools

🎓 Next Steps - Continue Your Learning Journey

Immediate Next Steps (This Week):

1. ✅ **Create your GitHub portfolio repository**
   - Name it: `pe-finance-toolkit` or `financial-modeling-tools`
   - Make it public (show employers!)
   - Add professional README
2. ✅ **Complete Practice Challenges above**
   - Build 3-4 financial calculators
   - Commit each one separately
   - Use Copilot for all coding

3. ✅ **Practice Git workflow daily**
    - Make changes
    - Commit with good messages
    - Push to GitHub
    - Build that green contribution graph! 🟩

Continue to Tutorial 3:

→ `Tutorials/Tutorial_03_VS_Code_Data_Analysis.md`

What you'll learn:

- Reading financial data from Excel and CSV
- Using Pandas for data analysis (game changer!)
- Cleaning and preprocessing data
- Financial statement analysis automation
- Creating visualizations with charts

Why this matters: At PE Club, you'll analyze TONS of data. Tutorial 3 teaches you to do it programmatically instead of manually!

💬 Final Thoughts from Dad

Mauricio, you just learned skills that will set you apart at PE Club Brussels!

What you accomplished today:

- ✅ Professional version control (Git/GitHub)
- ✅ AI-assisted coding (Copilot)
- ✅ Built real financial tools
- ✅ Started your coding portfolio

Most finance people NEVER learn these skills. You're 25 and already ahead of 90% of analysts!

The combination of finance knowledge + technical skills is RARE and VALUABLE. Keep building on this foundation!

Remember:

- Commit daily (even small changes)
- Use Copilot to learn, not just to copy
- Build tools that solve real PE problems
- Share your GitHub profile proudly!

I'm so proud of you! 🎉 ❤️

Now go build something amazing! 🚀

Estimated completion time: 90 minutes Difficulty: Beginner-Intermediate ★★☆☆☆ Next: Data Analysis with Python ★★★☆☆ Course Progress: 40% Complete 📊

Made with ❤️ by Dad for Mauricio's success at PE Club Brussels

---

📋 Quick Reference: Complete Git Workflow

For every new feature/change:

```
# 1. Check current status
git status

# 2. Make your changes in VS Code
# (edit files, create new files, etc.)

# 3. See what changed
git status                # Overview
git diff                  # Detailed changes

# 4. Stage changes
git add .                 # All files
# or
git add specific_file.py  # Specific file

# 5. Commit changes
git commit -m "Clear description of what you did"

# 6. Push to GitHub
git push

# 7. Verify on GitHub
# Visit your repo URL and see your changes!
```

Repeat this cycle for every logical unit of work!

🎯 **You're now ready to code with AI and collaborate like a pro! See you in Tutorial 3!** 🚀