

Contents

Tutorial 2: GitHub & Copilot Hands-On	1
<input type="checkbox"/> What You'll Learn (90 minutes)	1
Part 1: Setting Up GitHub (15 minutes)	1
Part 2: Your First Repository (20 minutes)	2
Author	3
Part 3: Setting Up GitHub Copilot (10 minutes)	5
Part 4: Your First Code with Copilot (20 minutes)	5
Part 5: Git Workflow Practice (15 minutes)	6
Part 6: Working with Branches (10 minutes)	7
Part 7: Real Project with Copilot (20 minutes)	8
Part 8: Copilot Best Practices (10 minutes)	9
Part 9: Copilot Keyboard Shortcuts	10
Part 10: Real-World Example (15 minutes)	11
<input type="checkbox"/> Skills Checklist	12
<input type="checkbox"/> Git Commands Cheat Sheet	12
<input type="checkbox"/> What's Next?	13
<input type="checkbox"/> Pro Tips	13
<input type="checkbox"/> Bonus: Copilot Prompts for Finance	14

Tutorial 2: GitHub & Copilot Hands-On

What You'll Learn (90 minutes)

By the end of this tutorial, you'll: - Create your first GitHub repository - Master the Git workflow (add, commit, push) - Set up GitHub Copilot - Use AI to write financial code - Build a real financial calculator with Copilot's help - Feel confident using AI-assisted coding

Prerequisites: - Completed Tutorial 1 (VS Code Basics) - VS Code installed - GitHub account created

Part 1: Setting Up GitHub (15 minutes)

Create GitHub Account

1. Go to <https://github.com>
2. Click Sign up
3. Use your professional email
4. Choose username (e.g., john-smith-finance or jsmith-analyst)
5. Verify email

GitHub Student Benefits (FREE Copilot!)

If you're a student or educator:

1. Go to: <https://education.github.com/students>
2. Click Sign up for Student Developer Pack
3. Upload proof:
 - Student ID
 - Transcript
 - Enrollment letter
4. Wait for approval (usually 1-3 days)
5. Get FREE GitHub Copilot! (normally \$10/month)

If not a student: - Start 30-day free Copilot trial - Subscribe for \$10/month if you continue

Install Git on Windows

1. Download: <https://git-scm.com/download/win>
2. Run installer
3. Important selections:
 - Use Visual Studio Code as Git's default editor
 - Git from the command line and also from 3rd-party software
 - Use bundled OpenSSH
 - Use the OpenSSL library
 - Checkout Windows-style, commit Unix-style line endings
 - Use MinTTY (terminal)
 - Default (fast-forward or merge)
 - Git Credential Manager
 - Enable file system caching
4. Verify installation (in VS Code terminal):

```
git --version
```

Should see: git version 2.x.x

Configure Git

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

```
# Set your name (appears in commits)
git config --global user.name "Your Full Name"

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

# Set VS Code as default editor
git config --global core.editor "code --wait"

# Make default branch 'main'
git config --global init.defaultBranch main

# Verify settings
git config --list
```

Part 2: Your First Repository (20 minutes)

Create Local Project

1. Create project folder:

```
cd Documents
mkdir my-first-finance-repo
cd my-first-finance-repo
```

2. Open in VS Code:

```
code .
```

3. Create initial files:

File 1: 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
```
python calculator.py

```

### Author

[Your Name]

```
File 2: `calculator.py`
```python
"""
Simple Financial Calculator
My first Git project!
"""

def present_value(future_value, rate, years):
    """Calculate present value"""
    return future_value / (1 + rate) ** years

def future_value(present_value, rate, years):
    """Calculate future value"""
    return present_value * (1 + rate) ** years

# Test
if __name__ == "__main__":
    print("Financial Calculator v1.0")
    pv = present_value(10000, 0.10, 5)
    print(f"Present Value: ${pv:.2f}")

```

Initialize Git Repository

In terminal:

```
# Initialize Git
git init
```

```
# Check status
git status
```

You'll see red files (unstaged)

The Git Workflow

Remember: ADD → COMMIT → PUSH

```
# 1. ADD (stage files)
git add .

# Check status again
git status

Now files are green (staged)

# 2. COMMIT (save snapshot)
git commit -m "Initial commit: Create financial calculator"

# Check status
git status
```

Should see: “nothing to commit, working tree clean”

Create GitHub Repository

Method 1: GitHub Website

1. Go to <https://github.com>
2. Click + (top right) → New repository
3. Name: my-first-finance-repo
4. Description: “Learning Git with financial calculator”
5. Keep it Private (for learning)
6. DON’T check “Initialize with README” (we already have one)
7. Click Create repository

Method 2: GitHub CLI (Advanced)

```
# Install GitHub CLI first
winget install --id GitHub.cli

# Authenticate
gh auth login

# Create repo
gh repo create my-first-finance-repo --private --source=. --remote=origin
```

Connect Local to GitHub

GitHub shows you commands. Copy and run:

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

# Verify
git remote -v

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

Your code is now on GitHub!

Visit: <https://github.com/YOUR-USERNAME/my-first-finance-repo>

Part 3: Setting Up GitHub Copilot (10 minutes)

Install Copilot Extensions

1. In VS Code, press **Ctrl+Shift+X** (Extensions)
2. Search: GitHub Copilot
3. Install GitHub Copilot (by GitHub)
4. Install GitHub Copilot Chat (by GitHub)

Sign In

1. Click “Sign in to use GitHub Copilot” notification
2. Opens browser → Authorize
3. Return to VS Code
4. Should see Copilot icon in status bar (bottom right)

Verify Copilot is Active

Look at status bar (bottom right): - Copilot icon visible - No red X on icon

If not working: - Click Copilot icon → “Check Status” - Verify GitHub subscription is active

Part 4: Your First Code with Copilot (20 minutes)

Exercise 1: Copilot Completes Your Code

Create new file: `time_value_money.py`

Step 1: Type this comment and press Enter:

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

Watch: Copilot suggests code in gray!

Accept: Press Tab

You should get something like:

```
def calculate_npv(cash_flows, discount_rate):  
    """Calculate net present value"""\n    npv = 0\n    for i, cf in enumerate(cash_flows):\n        npv += cf / (1 + discount_rate) ** (i + 1)\n    return npv
```

Exercise 2: Let Copilot Write Functions

Type each comment, press Enter, and press Tab to accept:

```
# Function to calculate internal rate of return
```

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

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

```
# Function to calculate bond price
```

Copilot writes each function for you!

Exercise 3: Copilot Chat

1. Press Ctrl+Shift+I (Open Copilot Chat)
2. Ask: "Explain how to calculate WACC"
3. Read Copilot's explanation
4. Ask: "Write a function to calculate WACC with all parameters"
5. Copilot generates the code!
6. Copy and paste into your file

Exercise 4: Build Complete Calculator

Create new file: `investment_analyzer.py`

Type this comment at the top:

```
# Complete investment analysis calculator with:  
# - Future value calculation  
# - Present value calculation  
# - NPV calculation  
# - IRR calculation (using scipy)  
# - CAGR calculation  
# - Annuity calculations  
# Include a main menu for user interaction
```

Press Enter and wait...

Copilot will suggest an ENTIRE program! Press Tab to accept.

Mind blown? That's the power of AI coding!

Part 5: Git Workflow Practice (15 minutes)

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:

```
# 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:

```
class FinancialToolkit:  
    """Complete financial analysis toolkit"""\n  
    # Copilot will suggest all methods!
```

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

```

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

```

# 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:

```

# 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

```
# 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:

```
# 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?

You now have: - □ GitHub repository - □ Git workflow mastery - □ Copilot generating code - □ Financial calculators built!

Next steps:

1. □ Practice: Commit every time you complete a feature
 2. □ Use Copilot for all new code
 3. □ Continue to: Tutorials/03_VS_Code_Data_Analysis.md
 4. □ Build your portfolio on GitHub
-

□ Pro Tips

Commit Often

- Commit after each feature
- Write clear messages
- Think: “If I need to undo, where should I go back to?”

Use Branches

- main = working code only
- feature-xyz = experiments
- Merge when ready

Trust but Verify Copilot

- Copilot is smart but not perfect
- Always test financial calculations
- Understand what code does
- Learn from suggestions

Build Portfolio

- Keep best projects public
 - Write good README files
 - Show on LinkedIn
 - Employers LOVE GitHub profiles!
-

Bonus: Copilot Prompts for Finance

Great Prompts to Try

```
# "Create a function to calculate Black-Scholes option pricing"  
  
# "Build a bond pricing calculator with yield to maturity"  
  
# "Write a function to calculate Sharpe ratio from returns"  
  
# "Create a portfolio optimization function using modern portfolio theory"  
  
# "Build a credit risk scoring model"  
  
# "Write a function to parse and analyze financial statements from Excel"
```

Ask in Copilot Chat: - “Explain the dividend discount model” - “How do I calculate beta for a stock?” - “What’s the difference between enterprise and equity value?” - “Help me build a LBO model in Python”

Congratulations! You’re now a Git + Copilot power user!

Next: Tutorials/03_VS_Code_Data_Analysis.md - Analyze real financial data

Estimated completion time: 90 minutes Difficulty: Beginner-Intermediate Next: Data Analysis with VS Code