Connecting Your Computer to AI

MGMT 675: Generative AI for Finance

Kerry Back

What is Claude Code?

Three Ways Claude Runs Code

Chat (Analysis)

- Sandboxed Python in the browser
- No local file access
- Must upload data manually
- State lost between chats
- Good for quick calculations

Cowork

- Runs in a remote VM
- Can install packages
- Files synced to/from VM
- VM is temporary
- Higher token cost

Code

- Runs on **your machine**
- Full local file access
- Persistent state
- Version control built in
- Most token-efficient

Code mode gives Claude direct access to your machine—no sandbox, no VM, no upload step.

Local Execution: Advantages

- **No sandboxing**—access databases, APIs, local files
- **Install anything**—pip install, npm, system packages
- **Large datasets**—no upload limits
- **Persistent state**—pick up where you left off
- **Version control**—code saved in your repo
- **Production path**—code is ready to deploy

Code Local

Step 5: Code (Local) — Efficient Agentic Work

Switch to the **Code** tab → select **Local** → choose a project folder.

How it differs from Cowork

- Runs code **directly on your machine** (no VM overhead)
- More token-efficient — fewer planning/orchestration tokens
- **Requires Python** (or R, Node, etc.) installed locally
- Coding-oriented interface, but works for data analysis

Example

"Read the file sp500_monthly.csv. Run a Fama–French regression for each of the 10 industry portfolios. Save the results to a table in results.xlsx and create a plot of the estimated betas with confidence intervals."

Code Local — Setup Requirements

One-time setup

1. Install Claude Desktop (already done from Step 3)
2. Install Python: `python.org/downloads`
3. Recommended: install key packages

```
pip install pandas numpy matplotlib statsmodels openpyxl
```

Or let Claude install them for you — it will run `pip install` as needed.

Token comparison

Surface	Relative token cost	Why
Chat / Artifacts	Low	No code execution
Code (Local)	Medium	Direct execution, lean context

Code Remote & GitHub

Step 6: Code (Remote) — Cloud Execution via GitHub

Switch to the **Code** tab → select **Remote** → choose a GitHub repository.

How it works

1. Your project lives in a GitHub repository
2. Claude clones it on Anthropic's cloud servers
3. Claude runs code, installs packages, pushes results back
4. You pull the results or view them in Claude Desktop

When to use this

- Python won't install on your machine (fallback)
- You want to work on a project you haven't cloned locally
- You want cloud compute for intensive tasks
- Collaborative work: Claude creates a pull request you can review

A Brief Detour: Why GitHub?

GitHub in 30 seconds

- Cloud storage for code and data files
- **Version control**: every change is tracked
- **Collaboration**: multiple people can work on the same project
- Free for students (GitHub Education)
- Industry standard in finance: quant teams, fintech, open-source models

Finance workflow

1. Push your data & scripts to GitHub
2. Tell Claude: *"run the event study in event_study.py and fix any errors"*
3. Claude clones, runs, fixes, pushes
4. You pull the clean results

Even if you never use Code Remote, learning GitHub is valuable. Every quantitative finance job expects it.

Comparison

Comparison: Which Tool When?

Task	Chat	Artifacts	Cowork	Code Local	Code Remote
Explain WACC formula	✓				
Draft an investment memo	✓	✓			
Interactive DCF calculator		✓			
Analyze CSV, create Excel			✓	✓	✓
Run Fama–French regressions			✓	✓	✓
Organize 50 PDF 10-Ks			✓		
Event study on CRSP data				✓	✓
Team project with version control					✓

Rule of thumb: Start with Chat. If you need a visual, use Artifacts. If you need file I/O and code execution, use Code Local. Save Cowork for complex multi-file tasks

Slash Commands

Built-in Slash Commands

Type / in Claude Code to see all available commands. These work in the Code tab, the CLI, and the VS Code extension.

Command	What It Does
`/help`	Show all available commands (built-in + custom)
`/clear`	Clear conversation history and start fresh
`/compact`	Compress the conversation to free up context window space
`/cost`	Show token usage for the current session
`/model`	Switch between models (Sonnet, Opus)
`/review`	Ask Claude to review your code for issues
`/init`	Create a `CLAUDE.md` file for your project
`/memory`	Edit your `CLAUDE.md` memory files
`/agents`	Create, browse, or run custom subagents
`/mcp`	Manage MCP server connections

Useful Slash Commands for Finance Work

Session Management

- `/clear` — start a new task without leftover context from the previous one
- `/compact` — keep working but free up space when context gets long
- `/cost` — check how much of your token budget you've used
- `/model sonnet` — switch to a faster, cheaper model for simple tasks

Project Setup

- `/init` — creates a `CLAUDE.md` file that gives Claude persistent context about your project (e.g., "this folder contains SEC filings")
- `/memory` — edit these instructions later
- `/review` — ask Claude to review code it wrote for errors before you rely on it

Tip: Use `/clear` between unrelated tasks and `/compact` within a long task. This keeps Claude focused and saves tokens.

Subagents

What Are Subagents?

A **subagent** is a specialized AI assistant that handles a specific type of task. Each subagent runs in its own context window with a custom system prompt and specific tools.

Built-in Subagents

- **Explore** — fast, read-only agent for searching files and understanding code
- **Plan** — researches your project before proposing an approach
- **General-purpose** — handles complex, multi-step tasks

Claude uses these automatically when appropriate.

Why Subagents?

- **Focus**: each agent has its own context window, so it doesn't get distracted
- **Expertise**: custom system prompt teaches it domain knowledge
- **Parallelism**: multiple agents can work on different subtasks at once
- **Safety**: you can restrict which tools each agent can use

The `/agents` Command

Type `/agents` to create, browse, or run custom subagents. Claude walks you through the setup interactively.

1. Type `/agents` and select **Create new agent**
2. Choose scope: **Project** (this folder only) or **User** (all projects)
3. Select **Generate with Claude** and describe what the agent should do
4. Claude generates the system prompt and configuration
5. The agent is saved as a markdown file:
   - Project: `.claude/agents/my-agent.md`
   - User: `~/.claude/agents/my-agent.md`
6. Invoke it anytime with `/agents` → select the agent

Example: Financial Data Analyst Agent

.claude/agents/analyst.md

```
---
name: financial-analyst
description: "Analyzes financial
  data files and produces
  reports with charts"
tools: Read, Write, Bash, Glob
model: sonnet
---

You are a financial data analyst.

# Workflow
1. Read the data file(s)
2. Clean and validate the data
3. Compute requested metrics
4. Create charts with matplotlib
5. Save results to Excel

# Standards
- Always annualize returns
- Use log returns for regressions
- Label all chart axes
```

How to Use It

- Type /agents → select `financial-analyst`
- Or Claude invokes it automatically when you ask for data analysis
- The agent follows your standards every time (annualized returns, log returns, labeled axes)

Other Finance Agent Ideas

- **SEC filing reader**: extracts tables and key metrics from 10-K/10-Q PDFs
- **Portfolio optimizer**: runs mean-variance optimization, outputs weights

When to Use Subagents

Use a Subagent When

- You repeat the same type of task often (e.g., "analyze this CSV the same way every week")
- The task has specific standards Claude should always follow
- You want to delegate a subtask while Claude works on something else
- The task is complex enough to benefit from a dedicated context window

Just Use Claude Directly When

- It's a one-off task with no recurring standards
- The task is simple (a quick question, a small edit)
- You want to iterate interactively on the approach
- You haven't done the task enough to know what standards to enforce

Start by using Claude directly. When you notice you're repeating the same instructions, that's the signal to create a subagent.

Managing Usage

Managing Your Token Budget

All Claude products share the **same usage pool**. Usage resets every 5 hours.

Strategies for the Pro plan ($20/month)

1. **Use Chat for questions** — "Explain Jensen's alpha" costs very few tokens
2. **Use Artifacts for quick visuals** — paste data, get a chart
3. **Use Code Local for analysis assignments** — lean and efficient
4. **Reserve Cowork for heavy-lift tasks** — multi-file, complex output
5. **Clear context between tasks** — type `/clear` in Code sessions
6. **Bundle related work** — don't start a new session for every subtask

If you hit your limit: Wait for the 5-hour reset, or fall back to Chat + Artifacts (much lower token cost) to keep working.

Getting Started

Getting Started — Your Setup Checklist

1. **Create a Claude account** at `claude.ai`
2. **Subscribe to Pro** ($20/month) via the account settings
3. **Download Claude Desktop** from `claude.com/download`
   - Windows (x64) or Mac
4. **Install Python** from `python.org/downloads`
   - Check "Add Python to PATH" during installation
5. **Try each mode:**
   - Chat tab: ask a finance question
   - Chat tab: paste data and request an artifact
   - Code tab → Local: point at a folder with a CSV, ask for analysis
6. **(Optional)** Create a GitHub account at `github.com`
   - Apply for GitHub Education (free Pro features for students)

Exercises

Exercise: Loan Portfolio Analysis

- Download loans.zip into your project folder.
- Extract the files. You will have a loan tape (CSV), a collateral appraisal report (PDF), and a policy exceptions memo (docx).
- In Claude Desktop, go to Code → Local and select your project folder.
- Ask Claude to load the loan tape, compute the weighted average interest rate and LTV ratio, and flag any loans that exceed the policy limits described in the memo.
- Ask it to cross-reference the collateral values from the appraisal report and produce a summary of loans where the appraised value has declined.

Download Data for Exercise