

Teaching with AI

Kerry Back

1. Should we have courses on AI, and, if so, what should they cover?
2. To what extent and how should we incorporate AI into other courses?
3. How can we use AI to enhance learning?
4. How can we assess students?

- We should expect and encourage students to adopt AI as their go-to tool.
- Rather than opening a spreadsheet, they should open a chat (or maybe a chat in a spreadsheet).
- Every course should be about collaborating with AI to generate analysis, recommendations, and reports. But AI should not take center stage. It should just be the standard tool for analysis and reporting.
- We have to assess students on whether they can defend the work they submit – whether they can convince an audience that the reasoning and analysis is reliable. It is possible that AI could help us in the assessment process.

- At present, we need courses focusing on AI.
- Once AI has been integrated into the curriculum, we may not need stand-alone courses.
- I will describe the course I am teaching (preparing to teach) at Rice: mgmt675.kerryback.com. Much of it should soon be standard stuff that everyone does.
- I'll come back to enhancing learning and also to using AI to simplify and accelerate our research and course development.

Course Introduction

Derek Waldron, JP Morgan Chief Analytics Officer

What we're working towards is that every employee will have their own personalized AI assistant; every process is powered by AI agents, and every client experience has an AI concierge.

You'll still have people at the top who are managing and have relationships with clients, but many, many of the processes underneath are now being done by AI systems.

Workers would shift from being creators of reports or software updates, or 'makers' ... to 'checkers' or managers of AI agents doing that work.

The New Skills Landscape

Brookings Institute, 2025

As AI models begin to handle underwriting, compliance, and asset allocation, the traditional architecture of financial work is undergoing a fundamental shift.

As job descriptions evolve, so does the definition of financial talent. Excel is no longer a differentiator. Python is fast becoming the new Excel.

But technical skills alone will not cut it. The most in demand profiles today are those that speak both AI and finance.

Why AI is Useful for Finance

- AI can write code for financial analysis and building things (vibe coding)
- AI can generate Excel workbooks
- AI can draft reports (Word docs) and presentation decks (PowerPoint)
- AI can look up and answer questions about company practices, past company reports, communications, etc.
- AI can search the web, pull data from databases, or call APIs
- AI can analyze documents, contracts, or financial statements
- AI can automate repetitive tasks and workflows
- AI can communicate directly with clients (with strong controls)

Collaborating with AI

The Mindset Shift

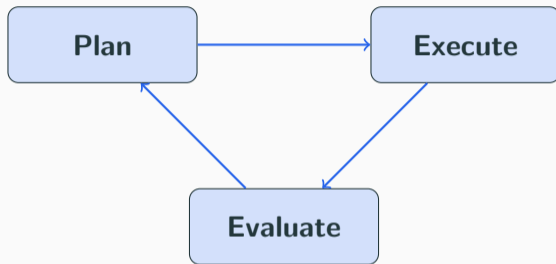
- AI is not a search engine—it's a **collaborator**
- Do not try to craft the “perfect prompt”
- Instead: have a **conversation**
- Think of AI as a capable colleague, not a vending machine
- The goal is *iterative refinement*, not one-shot perfection

Ask the AI What It Can Do

- AI systems know their own capabilities
- Ask: “What information do you need from me to do this?”
- Ask: “What are the different ways you could approach this?”
- Ask: “What should I consider before we start?”
- Let the AI **guide the conversation**—it often knows what questions to ask

“I want to build a portfolio optimizer. What do you need to know to help me?”

The Plan-Execute-Evaluate Cycle



- **Plan:** Define the task, gather requirements, outline approach
- **Execute:** Let AI do the work with your guidance
- **Evaluate:** Check results, identify gaps, refine
- Repeat until satisfied

Cross-Evaluation with Other AI Conversations

- Start a **new conversation** to evaluate the plan and output
- Even with the *same model*, a fresh context can catch errors
- Ask the new conversation to review, critique, or verify
- Different phrasing may reveal blind spots
- Consider using a *different model* for additional perspective

"I received this analysis from another session. Can you review it for errors or questionable assumptions?"

AI with Code Execution

Benefit of Bundling Code Execution with AI

- LLMs can write code, but writing is not the same as running
- Attaching a code execution tool enables:
 - Data analysis with real calculations
 - Visualizations and charts
 - File processing (Excel, CSV, PDF)
 - **Iterative debugging—run, fix, repeat**
- Transforms chatbots into computational tools

Fortune, Jan 29, 2026

Boris Cherny, head of Claude Code at Anthropic:

“For me personally, it has been 100% for two+ months now, I don’t even make small edits by hand.”

Across Anthropic, Cherny says *“pretty much 100%”* of code is AI-generated ... An Anthropic spokesperson said company-wide the figure is between 70% and 90%.

In the Cloud

- ChatGPT, Claude, Gemini
- Upload files, ask questions
- Server-side sandbox
- No internet access (LLM can reach internet, but Python cannot)

Locally

- Cursor, GitHub Copilot, Claude Code
- LLM is in the cloud but code runs on your machine
- Full environment access
- Internet, databases, APIs

Claude.ai and Desktop: Two Code Execution Features

Artifacts

- JavaScript in your browser
- Fast, lightweight
- Can save and publish artifacts

Python Execution

- Python/Bash on server
- Full linux environment
- **Can install packages (pip)**
- No internet access

"I'm uploading an Excel file with monthly returns for WMT, the excess market return (MKT-RF) and the risk-free rate (RF). Calculate excess returns for WMT and regress on MKT-RF to estimate the beta of WMT. Plot the data and the regression line."

Claude will write Python code, run it, show results, and let you download the chart.

Data for Exercise

- Interactive content in a panel next to the chat
- Codes in React/JavaScript—runs in your browser
- View, edit, and iterate in real-time
- **Save** artifacts to your account for later use
- **Publish** to create a public URL others can use (with charges to **their account, not yours.**)
- Create and check once. Use forever!

Claude Artifact Exercise

“Create an artifact that allows me to upload a csv file containing the return of a stock, the market excess return, and the risk-free rate over a given time period. Compute excess returns of the stock and regress them on the market excess return. Display the beta of the stock and a plot of the data with the regression line.”

Claude creates an interactive artifact with file upload, regression calculations, and a chart—all running in your browser.

Spreadsheets and AI

AI Can Create Spreadsheets with Python

- Python libraries like `openpyxl` create/modify Excel files
- AI writes Python code that:
 - Inserts data values into cells
 - Writes Excel formulas (e.g., `=SUM(A1:A10)`)
 - Applies formatting (fonts, colors, borders)
 - Creates charts and pivot tables
- Result: fully functional spreadsheet with **live formulas**

Two Ways AI Interacts with Spreadsheets

Inside Excel (Add-ins)

- Sidebar panel in Excel
- Sees your current workbook
- Modifies cells directly
- Context-aware suggestions
- Examples: Claude for Excel, Microsoft Copilot

Outside Excel (Python)

- Runs in terminal or IDE
- Creates/modifies .xlsx files
- You open result in Excel
- Full programming power
- Examples: Claude Code, ChatGPT

Claude for Excel Add-in

- Runs Python in a server-side sandbox
- Reads multi-tab workbooks, explains calculations
- Modifies cells while preserving formula dependencies
- **Does not require OneDrive**—works with local files
- Requires Claude Pro (\$20/month), Max, Team, or Enterprise plan

Claude Add-In for Excel

Exercise for Claude.ai or Claude Desktop

“Create an Excel workbook to illustrate two-stage DCF analysis.”

- Claude “skills” are instructions that Claude reads when needed
- The `xlsx` skill instructs Claude to:
 - Use Excel formulas instead of hardcoded values
 - Apply professional formatting (color coding, number formats)
 - Verify zero formula errors (`#REF!`, `#DIV/0!`, etc.)
 - Follow financial modeling standards
- Many available, easily customizable or created (just text files)
- [View the full `xlsx` skill documentation](#)

Creating Web Apps

Using Streamlit for Apps

Streamlit Overview

- Python library for web apps
- No HTML, CSS, or JavaScript needed
- Designed for data apps and dashboards
- Interactive widgets built-in
- Hot reload during development

Perfect For

- Data visualization dashboards
- Machine learning demos
- Financial analysis tools
- Interactive reports
- Chatbot interfaces

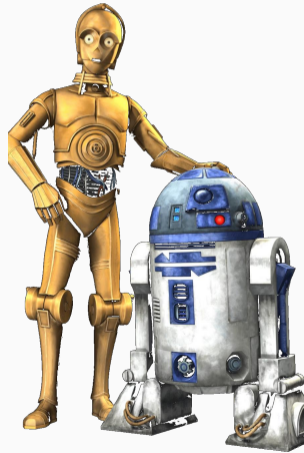
From Idea to Public App

The complete workflow to create and deploy a Streamlit app—all handled by Claude Code with simple natural language requests.

1. Create the Streamlit app
2. Initialize a Git repository
3. Create a GitHub repository
4. Commit and push the code
5. Deploy to Koyeb with auto-deploy
6. Get your public URL

Ask Claude to do each step—no commands to memorize

Chatbots and Agents



What is a Chatbot?

Simple Definition

A chatbot is a program that:

1. Accepts user input (prompt)
2. Adds system prompt and sends to an LLM via API
3. Displays the response
4. Captures prompts and responses in a loop and sends history to LLM each time

- ChatGPT, Claude, Gemini are chatbots
- The LLM is the “brain”
- The chatbot is the interface
- You can build your own!

OpenRouter: One API, Many Models

What is OpenRouter?

- Unified API for 100+ models
- OpenAI, Anthropic, Google, Meta, etc.
- Single API key for all models
- Some models are free!

`openrouter.ai`

Free Models on Hugging Face

- Hugging Face hosts open models
- OpenRouter provides free access
- Examples:
 - `mistralai/mistral-7b-instruct:free`
 - `meta-llama/llama-3-8b-instruct:free`
 - `google/gemma-7b-it:free`

Basic Structure

```
import requests

response = requests.post(
    "https://openrouter.ai/api/v1/chat/completions",
    headers={"Authorization": f"Bearer {API_KEY}"},
    json={
        "model": "mistralai/mistral-7b-instruct:free",
        "messages": [{"role": "user", "content": prompt}]
    }
)
answer = response.json()["choices"][0]["message"]["content"]
```

Messages Are a List of Dictionaries

```
messages = [  
    {"role": "user", "content": "My name is Alice"},  
    {"role": "assistant", "content": "Nice to meet you!"},  
    {"role": "user", "content": "What's my name?"}  
]
```

Roles

- user: Human messages
- assistant: LLM responses
- system: Instructions (special)

- Each message has role + content
- Order matters (chronological)
- Send full list each API call

The System Prompt

The **system prompt** is a special message that defines the chatbot's personality, knowledge, and behavior. It's the key to customization.

What It Does

- Sets the chatbot's persona
- Defines its expertise
- Specifies response style
- Adds domain knowledge
- Sets guardrails/rules

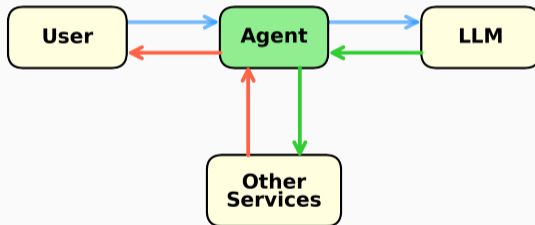
Placement

- First message in the list
- Role is system
- Sent with every API call
- User never sees it directly

Anthropic's Published System Prompts for Claude

What is an AI Agent?

- A chatbot equipped with tools to perform actions beyond conversation.
- Code execution (Python, etc.)
- File generation (Excel, Word, PDF)
- Database access
- Web search and API calls
- System commands (copy, move files, etc.)



What Are Tools?

Tools are functions the agent can call to interact with the outside world.

Example Tools

- Execute SQL queries
- Run Python code
- Read/write files
- Search the web
- Send emails
- Call APIs

How Tools Work

1. LLM decides to use a tool
2. Returns tool name + parameters
3. Agent executes the tool
4. Result sent back to LLM
5. LLM continues reasoning

How Coordination Works

The agent combines **LLM responses** with **pre-programmed logic** to decide the next action.

LLM Decides

- Which tool to call next
- What parameters to pass
- When the task is complete
- What to ask the user

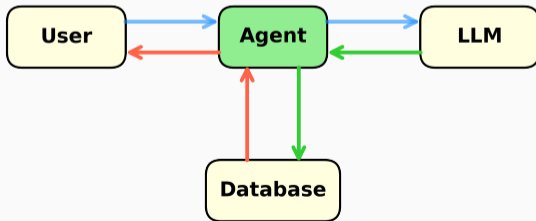
Code Decides

- If tool_call → execute tool
- If error → retry or report
- If question → prompt user
- If done → return response

The agent is part LLM intelligence, part traditional programming

- Prompt + System Prompt → LLM
- LLM → SQL code or a question for the user
- Agent → SQL code to database or → question to the user
- Eventually, SQL code → database
- Database → data or error message
- Error message → LLM
- Eventually data → user

Visit data-portal.rice-business.org
Get access token, Log in, Ask for data



Local Execution: Python, Claude Code, VS Code

- Python 3.12, VS Code, Git, Quarto, TinyTeX, Node.js, Claude Code
- VS Code extensions: Python, Jupyter, Claude Code, Quarto, LaTeX Workshop, GitHub Copilot
- Command line interfaces for GitHub, Koyeb

Class Software Installer

Opening a Folder in VS Code

- VS Code works with **folders**, not individual files
- File → Open Folder → select your project folder
- The folder appears in the Explorer sidebar (left panel)
- All files in the folder are accessible
- This is your workspace for a project

Tip: Create a dedicated folder for course work

Opening Claude Code

- **Spark icon:** Click the spark icon in the top-right corner of any open file
- **Status bar:** Click “Claude Code” in the bottom-right corner
- **Command Palette:** Ctrl+Shift+P → type “Claude Code”
- **Keyboard shortcut:** Cmd+Esc (Mac) / Ctrl+Esc (Windows)

What Claude Code Can Do

- Explain code and answer questions
- Write new code from descriptions
- Fix errors and debug problems
- Edit files (with your approval)
- **Run commands in the terminal**
- Create and modify Jupyter notebooks, Python scripts, LaTeX files, ...

Exercise: Estimating Betas

- In VS Code, open the folder containing betas.xlsx (or copy it into the folder currently open in VS Code)
- Ask Claude Code to compute WMT's excess returns and run a regression to estimate its beta.
- Ask for a Word doc containing a scatterplot of the data with the regression line and a discussion of why the beta is what it is.

Data for Exercise

Exercise: Aggregating Tables

- Download aggregation.zip into the folder open in VS Code.
- Extract all the workbooks in the zip file.
- Each workbook contains a table with similar data. Some tables are missing some columns and the names of some of the columns vary somewhat across the tables. Ask Claude Code to combine the tables into a single table, to include all columns, and to reconcile the varying names.

Data for Exercise

Exercise: Mean-Variance App

Build and deploy a Streamlit app for mean-variance portfolio optimization.

User Inputs

- Risk-free rate
- Number of risky assets
- Expected returns (means)
- Standard deviations
- Correlation matrix

App Outputs

- Tangency portfolio weights
- Downloadable image of:
 - Frontier of risky assets
 - Capital allocation line

AI for Class Preparation and Research

Some Claude Code prompts:

- Using this data ... compute these statistics ... and put them in a latex table like this ... and insert them in my paper in ...
- Using this data ... generate figures like this ... and insert them in ...
- I don't like the way I worded things in this paragraph ... Please clean it up.
- I want to add a slide to my lecture about ... It should say ... in bullet points
- Use the Google API to get an image that represents ... and insert it into ...
- Make this edit to ... and then render, commit, and push to update this website ...
- Build my latex file. Find and fix any errors.
- Using sympy, simplify this integral ...

Enhancing Learning: NotebookLM

- In addition to using AI for financial analysis, students should use AI to learn.
- Can ask them to chat with a chatbot about a subject in lieu of reading assignments.
- Can use retrieval-augmented generation (chatbot answers questions from specific sources)
- NotebookLM is an easy way to do this. Example: [chatbot about Pricing and Hedging Derivative Securities \(online textbook by Back, Liu, and Loewenstein\)](#)
- NotebookLM provides other tools, including AI-generated slide presentations. Example: [Presentation of Getting Paid to Hedge: Why Don't Investors Pay a Premium to Hedge Downturns? by Kapadia, Ostdiek, Weston, and Zekhnini, JFQA, 2019](#)