AI with Code Execution

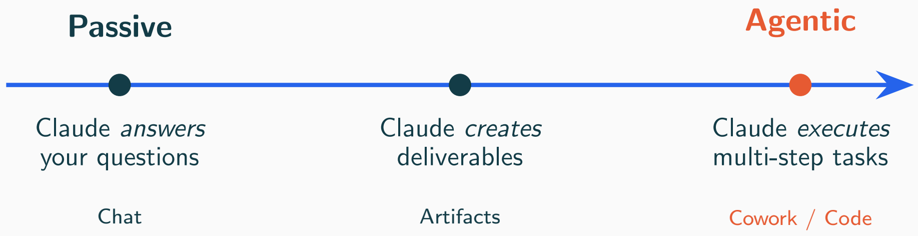
MGMT 675: Generative AI for Finance

Kerry Back

Roadmap

1. Claude.ai — Web Chat	*Chatbot + code execution*
2. Claude.ai — Artifacts	*Interactive charts and apps*
3. Claude Desktop — Chat	*Same, but installed on your machine*
4. Claude Desktop — Cowork	*Claude acts on your files autonomously*
5. Claude Desktop — Code	*Claude runs code on your machine*
6. Claude Code in VS Code	*AI coding inside an IDE*

The Big Idea



Passive
Agentic
Claude *answers* your questions
Chat
Claude *creates* deliverables
Artifacts
Claude *executes* multi-step tasks
Cowork / Code

Each step gives Claude more **capability** and more **access** to your work.

Claude.ai — Web Chat

Step 1: Claude.ai — Web Chat

Go to `claude.ai` in any browser. No installation required. Icon at top left opens sidebar.

What it can do

It's a chatbot with a code execution tool. It can answer questions, explain concepts, brainstorm, draft text, summarize documents you paste in. It can also generate charts, Excel files, Word docs, and PowerPoint decks. Code runs in a **sandbox**—can `pip install` packages but has **no internet access**.

Example

"Create an Excel file to illustrate two-stage DCF analysis."

Artifacts

Step 2: Artifacts — Interactive Charts and Apps

- Still in `claude.ai`, Claude can produce **interactive charts and apps** that appear in a side panel.
- Choose Artifacts → Create an Artifact → apps or websites.
- Artifacts aren't throwaway — they persist and can be shared.
- Click **Publish** to get a shareable link — anyone can view and interact with it, no Claude account needed

Artifact Examples

Example 1

"Here is an Excel file [upload this file to `claude.ai`] containing monthly returns for WMT and SPY and the one-month T-bill yield. Create an interactive scatter plot of AAPL excess returns vs. market excess returns, show the regression line, and display alpha and beta. Include the month in the hover data."

Example 2

"Create an app that lets a user input a stock's current price, the strike price, the risk-free rate, the volatility, and the time to expiration in years. Compute the Black-Scholes call and put option prices. Create an interactive plot showing how both option prices change as the stock price varies from 50% to 150% of the strike price."

Publish

Claude Desktop — Chat

Step 3: Claude Desktop — Chat

Download the Claude app from `claude.com/download`.

What changes from web?

- Same chat + artifact capabilities
- Installed application
- Foundation for Cowork and Code tabs (next steps)

Setup

Install the app → sign in with your Claude account → use the **Chat** tab. That's it. Same experience as the web, but now you have access to Cowork and Code, which we will investigate later.

Code Execution Capabilities

Code Execution: Sandbox vs. Local

All chatbot code execution runs in a **sandbox**—an isolated container with restricted network access.

Mode	`pip install`	**Internet**	**External APIs**
ChatGPT (Data Analysis)	**Yes***	**No**	**No**
Claude Chat (web)	**Yes**	**No**	**No**
Claude Cowork (VM)	**Yes**	**No**	**No**
Claude Code (local)	**Yes**	**Yes**	**Yes**

*ChatGPT added `pip install` in January 2026; previously limited to pre-installed packages.

Key insight: Only **Claude Code** (running on your machine) can call external APIs—Yahoo Finance, Alpha Vantage, Financial Modeling Prep, databases, etc.

Example: Live Stock Data with `yfinance`

With Claude Code running locally, you can fetch **live financial data**:

Prompt to Claude Code

"Get the last year of daily prices for AAPL and MSFT using yfinance. Plot the cumulative returns."

Python code Claude writes and executes:

```
import yfinance as yf
import matplotlib.pyplot as plt

data = yf.download(["AAPL", "MSFT"], period="1y")
rets = data["Close"].pct_change().cumsum()
rets.plot(title="Cumulative Returns (1 Year)")
plt.savefig("returns.png")
```

`yfinance` is free and requires no API key. This **fails in Chat, Cowork, and ChatGPT**—the sandbox blocks the network call.

Example: Financial Modeling Prep API

APIs requiring **authentication** also work in Claude Code:

Prompt to Claude Code

"Use Financial Modeling Prep to get Apple's income statement. My API key is in the environment variable FMP_API_KEY."

Python code Claude writes and executes:

```
import os, requests, pandas as pd

key = os.environ["FMP_API_KEY"]
url = "https://financialmodelingprep.com/api/v3"
url += f"/income-statement/AAPL?apikey={key}"
resp = requests.get(url)
df = pd.DataFrame(resp.json())
print(df[["date", "revenue", "netIncome"]])
```

Free tier: 250 requests/day at `financialmodelingprep.com`.

Chatbots, System Prompts, and Skills

What is a Chatbot?

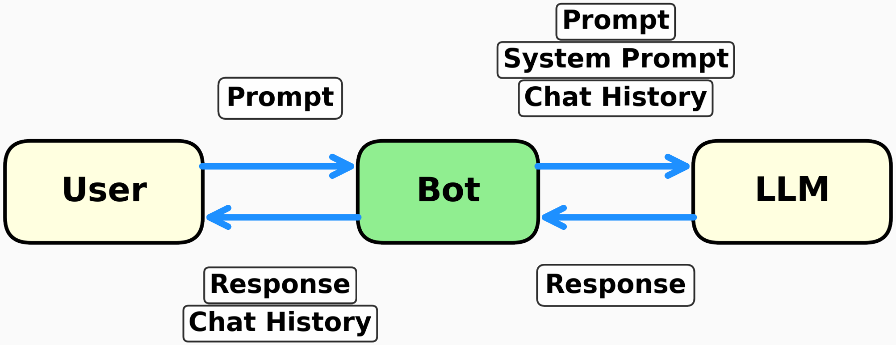
Simple Definition

A chatbot is a program that:

1. Accepts user input (prompt)
2. Adds system prompt and sends to an LLM (large language model)
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

Chatbot Architecture



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

How It Works

- Sent to LLM with every prompt
- User never sees it directly

Anthropic's Published System Prompts for Claude

Claude Skills

- Claude ‘skills’ are text files (instructions) that the chatbot sends to the LLM when needed
- An overview of available skills is part of the system prompt
- The overview tells LLM: ‘ask for this skill in these circumstances . . . ’
- In those circumstances, LLM sends response to chatbot: ‘send skill’
- Chatbot sends system prompt + prompt + chat history + skill
- LLM sends response
- This pattern is universal: Custom GPTs, Copilot instructions, `.cursorrules`, Gems
- A skill is just a system prompt extension—extra text the model sees

Claude Excel Skill

- The `xlsx` skill instructs Claude to:
  - Use Excel formulas instead of hardcoded values
  - Apply professional formatting (color coding, number formats)
  - Verify there are no formula errors (#REF!, #DIV/0!, etc.)
  - Follow financial modeling standards

View the full xlsx skill documentation

Effective Prompting

Effective Prompting = Collaboration with AI

- 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

Do’s and Dont’s for Effective Prompting

Don’t Do This

- Try to anticipate every edge case
- Give up when the first response is wrong
- Treat AI as a one-shot tool

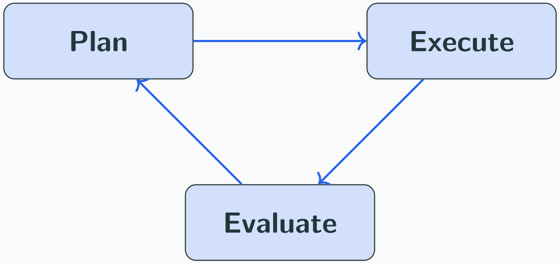
Do This Instead

- Start with a rough request
- Refine based on the response
- Ask follow-up questions
- Iterate until you get what you need

Ask the AI What It Can Do

- 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?”

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?"

Claude Pricing Options

Subscription Plans

- **Free:** Limited usage
- **Pro:** \$20/month — 5× free usage, Claude Code, Cowork, memory
- **Max:** \$100/month (5× Pro) or \$200/month (20× Pro)

API (Pay Per Token)

- **Sonnet 4.5:** \$3 input / \$15 output per million tokens
- **Haiku 4.5:** \$1 input / \$5 output per MTok
- **Opus 4.6:** \$5 input / \$25 output per MTok
- Does not include Code, Cowork, or Excel add-in (subscription only)

Example: What Does a Typical Query Cost via API?

A 1-page prompt ≈ 500 tokens. A 2-page response ≈ 1,000 tokens.
Using Sonnet 4.5: input $= 500 \times \$3/1\text{M} = \$0.0015$, output $= 1{,}000 \times \$15/1\text{M} = \$0.015$.