Setup:

We install Cursor IDE and UV (python venv like tool). For the UV install we run in powershell:

powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"

Then, in the terminal opened in the Cursor terminal (like in VSCode) we run:

uv sync

It should build the env. You’ll see .venv directory in your project directory.

To run a python script just do: > uv run <script\_name>

Anthropic distinguishes 2 types:

1. Workflows – are systems where LLMs and tools are orchestrated through predefined code paths.
2. Agents – are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

They also identify 5 workflow design patterns:

1. Prompt chaining – Decompose into fixed sub-tasks. You can have an LLM making some task and its output can be given to a different LLM call (see slides for more details). (there is a blur between a workflow and agent)
2. Routing - Direct an input into a specialized sub-task ensuring separation of concerns.
3. Parallelization – Breaking down tasks and running multiple sub-tasks concurrently (not selecting as in routing). Doesn’t need to be different sub-tasks, just to split the main job.
4. Orchestration worker – not a code makes the above, but an LLM does it.
5. Evaluator-Optimizer – LLM is doing a job and another is used to evaluate and check the work of the previous LLM and it can accept or reject (and rerun the first). To build a higher accuracy, predictability and robustness. And have better guaranties.

Agents by contrast:

1. Open ended
2. Feedback loops
3. No fixed path

Applies actions on the env and get feedback until it completes its task.

Risks of agent frameworks

Unpredictable path

Unpredictable output

Unpredictable costs (how many API calls?)

Monitor – many ways how to see what is done under the hoods.

“Guardrails ensure your agents behave safely, consistently and within your intended boundaries”

Day 3

Transcript:  
So the first model needs no introduction really. It is of course, the model GPT 4o mini from open AI. It's for sure the most well known of the models out there.

And of course, there's also GPT 4o, the bigger cousin.

And then there are the reasoning models, which are models that have been trained to think through steps in an Agentic like way, in like a workflow of thinking through the different steps before they arrive at their conclusion.

Because it turns out that when you ask an LLM to think through its steps, you get much better outcomes.

So, we may take a look at some point at 1o and 3o-mini, but it's less essential for this course.

We're going to be sticking with GPT four mini now and OpenAI's great rival.

We're going to be using Gemini-2.0-flash.

There is also the Pro version of that too, but I think we'll stick with flash. And as of right now, flash is actually free at least as long as you use it within certain usage limits.

I don't know how long that will be the case for, but by all means, if you want to use a frontier model without paying for it, then Gemini might be your path.

Do look into that Deep Seek.it is the Chinese upstart startup that shocked us all by coming up with such a powerful model in the form of Deep Seek V3 and R1.

And it's important to understand that what made Deep Seek so sensational was not necessarily that their model was the strongest in the world, because it wasn't.

It was slightly behind the latest from open AI, but that they developed such powerful techniques to train to be that good, and it cost them a fraction of the spend that OpenAI had spent to train GPT 4.

Deep seek was able to achieve very similar performance, pretty much comparable at a fraction. I think it's like 30 times less spend. That was the true innovation. That's the remarkable thing about deep seek.

And also that they open sourced the model so that you can use it. But the main the major model has 671 billion parameters, which means it's far too big for anyone to run that on their computers.

But there are versions of it, small versions of it called the distilled versions, which are in fact themselves just smaller models. There are versions of Llama and Quen, two different models that have been finetuned on data generated by the big Deep Seek and those smaller distilled versions of Deep Seek for sure available free of charge.

Groq - We will also be using groq, and there are two groqs.

Confusingly, if you don't know this grok spelt with a k at the end is the name of the model that comes from the company formerly known as Twitter. We might use grok with a k at some point as well.

Not in today's lab, but grok with a Q is something different. Groq with a Q is a company that has come up with a really cheap, fast way to run inference runtime models like llama 3.3, which is the massive version of llama with 70 billion parameters. So, you can run llama 3.3 really fast, really low cost on Groq’s infrastructure and along with many other

open source models including Deep Seek variants.

So grok is great to use for that.

Olama - is itself a more of a platform. It's something that you can use to run something locally that provides endpoints locally that are consistent, very similar to the endpoints that OpenAI and other models here have so that you can make local calls to an API, which is in fact going to just run an open source model locally on your computer in high performance optimized C++ using a library called llama CPP.

Vellum leaderboard can compare the open/closed source models for more details.