00: Orientation

What this course is, and how to use it

Why this matters

January 2025	Late 2026
AI as fancy autocomplete	Engineers direct several coding agents at once
A chat window you paste code into	Agents read the repo, run commands, test, and open PRs

Anthropic reports Claude wrote **more than 80%** of its merged code by May 2026.

Who this is for

- Students whose courses ignore or ban the tools, then jobs expect fluency
- Engineers who tried a tool in 2024 and found it wrong half the time
- Leads who want their teams to adopt these tools well

It runs from "what is a language model?" to parallel agents that are reviewed, secure, and affordable.

Part 1

What we mean by "vibe coding"

Where the phrase came from

> "a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists"

Andrej Karpathy, February 2, 2025

Describe it → accept the output → paste errors back → never read the diff

The disciplined version got its own names

When	Simon Willison's position
March 2025	If you review, test, and can explain it, it isn't vibe coding
October 2025	Proposes "vibe engineering" for the disciplined version
Early 2026	Reports that "agentic engineering" is winning as the name

So why keep the phrase?

People search for "vibe coding course," not "agentic engineering course."

We use the popular phrase and mean the disciplined version.

A Brief Introduction to Not Writing Code means you type much less code, so the engineering around it matters more.

What "vibe coding" means in this course

> **Bryan's take:** When I say vibe coding, I mean AI-accelerated coding, not really vibe coding. The AI does most of the typing. A human still decides what to build, reads what matters, makes sure it's tested, and owns the result. That's the only version I'd teach.

The spectrum

Mode	Who writes	Who reads	Good for
Hand-coding	You	You	Learning, tricky core logic
Autocomplete	You, with suggestions	You	Boilerplate, familiar code
Chat-assisted	Model writes, you paste	You	Explanations, one-off functions
Supervised agent	Agent edits the repo	You, every diff	Most professional work today
Parallel supervised agents	Several agents	You, plus agent cross-review	Experienced users, good tests
Background / cloud agents	Agent, unattended	You review the PR	Well-specified, well-tested tasks
Pure vibe coding	Model	Nobody	Prototypes, throwaways
Fully autonomous pipeline	Agents	Other agents	Contested

Why the human stays in charge

Reason	Why it holds
Accountability doesn't transfer	A model can't be fired, sued, or paged at 3 a.m.
The productivity evidence is mixed	Careful studies found experienced developers slower, while feeling faster
Learning requires effort	Letting AI do the work can reduce what you learn

Use AI in a way where you still get better, and where someone can answer for the result.

Part 2

How fast this moves, and what lasts

Models ship monthly

In 2026 alone (as of November 2026):

Lab	What shipped
Anthropic	Eleven generally available models across four tiers
OpenAI	GPT-5.3 through GPT-6.1
Google	Cancelled one flagship, launched another, announced a third

Google's third flagship is open only to vetted cyber defenders so far.

Everything else churns too

What	How fast
Task length	METR's "time horizon" doubling every few months
Vocabulary	Prompt → context → harness → loop → graph engineering, in ~18 months
Tools	Roo Code shut down (May 2026)
	Cursor acquired by SpaceX (August 2026)
	Gemini CLI moved to enterprise-only (June 2026)

Learn ideas that survive the next release.

Six ideas that last (1–3)

1. A language model predicts the next token. It has no built-in notion of truth.
2. The model only knows what's in its context. You control that.
3. Verification is your job. You can delegate the checking, not the responsibility.

Six ideas that last (4–6)

4. The harness matters as much as the model. Agent = model + harness.
5. Anything the model reads can give it orders. Prompt injection isn't solved.
6. Cost scales with context, not with the size of your question.

A two-minute history

Year	Milestone
2017	Transformer paper, "Attention Is All You Need"
2020	GPT-3 shows few-shot learning at scale
2021	GitHub Copilot brings autocomplete to IDEs
2022	ChatGPT launches (November 30)
2023	GPT-4, Claude 2, open-weight Llama models
2024	Claude 3.5 Sonnet, Cursor takes off, o1 reasoning model, MCP announced
2025	"Vibe coding" coined, Claude Code and Codex CLI, DeepSeek-R1, AGENTS.md, Agent Skills
2026	Parallel agents mainstream, Opus 4.6 to 5.5, GPT-5.3 to 6.1, Gemini 4, Kimi K3

How the modules fit together

Modules	About
01–05	Models: what they are, who makes them, how to judge them
06–10	Working with them: tools, context, workflows, multiple agents, review
11–15	Doing it responsibly: security, cost, ecosystem, keeping up, careers

Core path: 01 → 06 → 07 → 08 → 09 → 10

Part 3

How to use this course

Pick a path

You are	Start with	Time
A student, or new to AI tools	Student path	A semester, or ~15 hours self-paced
A working engineer	Engineer path	~6–8 hours, plus a two-week practice plan
A lead, manager, or AI champion	Leads path	~3 hours, plus the workplace playbook

Every module has three levels: Foundations, In practice, Going deeper.

Ground rules we assume throughout

1. You own what you ship. "The AI wrote it" is never an excuse.
2. Read every diff you merge, or have a check you trust read it.
3. No secrets or private data in tools not approved for that data.
4. Disclose AI use where your course, employer, or project asks.
5. A human approves anything irreversible.
6. When a tool surprises you, write it down.

What you need to follow along

- A computer with a terminal, git, and a code editor
- One frontier coding agent; a $20/month plan covers every lab
- Optionally, a second agent from a different lab, for cross-review
- No GPU. Nothing requires downloading model weights.

Reading this material with an AI

Please do, with two cautions.

- Many 2026 facts here postdate the model's training cutoff. Trust the sources.
- Ask it to quiz you instead of summarizing.

Try it

On your own (30 minutes)

- Ask an AI to explain a program you wrote, improve it, and test it
- Predict what the test will do before running it
- Note one thing it got right, and one it got wrong or glossed over

In a group (45 minutes)

- Each person finishes the sentence "Using AI to code is like…"
- Sort the analogies by which of the six ideas they capture or miss

Discussion

1. Karpathy's vibe coding meant *not* reading the code. When is that the right choice?
2. Which durable idea is most likely to be wrong in two years? Why?
3. If AI writes 80% of a team's code, what should the humans do?
4. What would convince you that a fully autonomous coding pipeline is trustworthy?

Read more

- Module: 00 Orientation
- Andrej Karpathy, "vibe coding" post (February 2, 2025)
- Simon Willison, "Not all AI-assisted programming is vibe coding" (March 19, 2025)
- Simon Willison, "Vibe engineering" (October 7, 2025; updated February 2026)