FRACTAL ACCELERATOR · NYC

The *Syllabus*

"An instrument whose music is ideas." A six-week course in making the computer truly yours.

fractalaccelerator.com/syllabus/content · Saturdays, 10am–5pm · Williamsburg, Brooklyn

Introduction

The inventors of computers had big dreams for their machines. One described the computer as "an instrument whose music is ideas." Computers could extend thought, creativity, and agency. They could amplify what makes you human.

That hasn't quite happened. Using a computer often feels tedious and frustrating instead of expressive and inspiring. You can't teach your computer how you think and work. Your ambition is limited by mass-produced software that wasn't designed for you. You have ideas that should exist but no way to build them. Your creativity and focus gets crowded out by clicking, copying, searching, and organizing.

That's starting to change. You can now sit down at your computer and begin to make it truly yours. You can craft your own tools, automate busy work, and create just by speaking aloud. You can teach your computer to adapt and grow with you. The computer is becoming an instrument of ideas, and you don't need to be a programmer to play it. In this course, we'll teach you how to play.

Class Details

Summary of requirements and sessions

DATES	The next cohort begins Saturday, September 12, 2026 and runs 6 weeks. Enrollment is limited to 25 participants.
TIME	Saturdays 10am-5pm: class 10am-1pm, followed by office hours and coworking 1-5pm. There are no weekday office hours. Outside class, plan for ~15 hours each week, applied to your real work.
ACCESS	Fractal Campus is available 24/7 during the program and for four months afterward.
LOCATION	Fractal Campus, a sunny 4,200 sq ft workspace (plus rooftop) in Williamsburg, Brooklyn.
PREREQUISITES	None. This course is built for people with domain expertise, not technical expertise. You do not need previous experience with programming.

What you will know how to do, and have done, by the end of the class

- You'll build a mental model of how modern AI actually works. You'll be able to explain how the "mind" of a language model works - how they think, remember, and forget. You'll understand how to help LLMs succeed, and what causes them to fail.
- You'll understand the architecture of AI agents - the loop of thinking, acting, and observing that makes them fundamentally different from chatbots like ChatGPT.
- You'll understand the history of computing and AI, and why it matters for where the technology is going.
- You'll be able to track new developments in AI, and understand how to separate hype from reality.
- You'll learn foundational software development skills and workflows. These will let you start building personal software, understand how it works under the hood, and get unstuck when things break.
- You'll begin to use the computer by talking with AI agents instead of clicking through menus. You'll learn how to connect AI agents to all of your apps, so agents can take action for you.
- You'll use your computer by speaking out loud. You'll experience the special kind of flow that comes from doing work just by using your voice.
- You'll learn how to communicate with AI effectively. You'll understand the art and science of "prompting" AI to get the outcomes you want. The way you talk to AI matters, and can make the difference between bad and good performance in your agents.

- You'll make your AI agent genuinely useful by teaching it who you are. Your work, your expertise, your preferences - stored in memory systems that persist across sessions and compound over time.
- You'll learn how to give agents new skills and capabilities through custom instructions and software, making them an expert in your work and the repetitive tasks you do everyday on the computer.
- You'll give your agent access to data that's currently locked inside the apps you use - email, banks, social media, work tools. Once it can reach your data, your agent can answer questions that span multiple sources, spot patterns you'd miss, automate work across apps, and watch for things that need your attention.
- You'll build and publish real websites and personal software - tools, dashboards, games, visualizations, automations, and workflows - starting from nothing but a plain-language description of what you want. You'll ship software that solves problems from your own life and keeps working long after the course ends.
- Your computer will keep working when you walk away from it. You'll set up agents that run in the background, handle routine tasks on their own, and stay reachable from your phone - so you can check in on what they've done from anywhere.
- You'll experience your computer differently, as an extension of your cognition, creativity, and will.

Subjects covered in class

The following is a general outline of subjects that we will cover in class. Additions and subtractions will be made according to student interest and competency.

01

Understanding AI

Tokens and context

- Tokenization - how text gets broken into pieces the model processes
- The context window - system prompts, conversation history, tool results
- Why the model "forgets" - no memory between sessions, finite context, compaction
- How long conversations degrade and when to start fresh
- Developing intuition for how context shapes what the model gets right and wrong

How AI is built

- Pre-training - learning patterns from internet-scale text
- Post-training - fine-tuning into a helpful assistant
- RLHF - how human feedback shapes model behavior
- Training data and data cutoffs

The model landscape: Claude, ChatGPT, Gemini, Grok

- Model sizes - small, fast, and cheap vs. large, slow, and powerful
- Claude's model family: Haiku, Sonnet, Opus
- Reasoning vs. non-reasoning models and when reasoning matters
- Form factors - web chat, desktop apps, terminal tools, mobile
- Survey of coding tools: Cursor, Copilot, Windsurf, Devin
- Pricing, metering, and how costs work

How an LLM drives your computer

- The agentic loop: prompt, tool call, tool response, next decision
- Tools - reading files, running commands, searching the web, calling APIs
- Tool descriptions - how the model knows what tools are available and when to use them
- Agents vs. chatbots - the difference between a response and a multi-step workflow
- Computer use - agents that see the screen and interact with GUIs

The spiky frontier

- U-shaped attention - why models pay less attention to the middle of long contexts
- Stuckness in patterns - when the model goes down a wrong path and stays there
- Hallucination and confabulation
- Prompt injection and security risks
- When to trust, when to verify, and when to intervene

02

Your computing environment

The terminal

- Core navigation: pwd, ls, cd, absolute vs. relative paths, the home directory
- File manipulation: touch, mkdir, cp, mv, rm, cat, less
- Pipes and redirection (|, >, >>)
- The file system - files, directories, paths, extensions, permissions
- Commands, arguments, flags, and environment variables
- Tab completion and other shortcuts

Dev environment setup

- Installing developer tools and package managers
- Runtimes and languages
- GitHub account and authentication

Git

- The core workflow: add, commit, push, pull, status, log, diff
- Branching and merging
- Public vs. private repositories
- GitHub - repositories, profiles, collaboration

Claude Code

- Installation
- What Claude Code adds on top of the base model: permissions, MCP, slash commands, multi-file editing
- Keyboard shortcuts, slash commands, sessions, key repeat speed
- The permission model and auto mode
- cmux - running multiple agents in parallel
- Claude Code on your phone

Voice interaction

- Dictation and voice-to-text tools
- Voice as an input method for the terminal and for agents
- When voice is faster and when typing is better
- Unlocking flow with voice-driven computing

03

Working with your agent

Prompting

- Declarative vs. imperative prompting
- Moving up the ladder of abstraction - delegating details to the model
- Separating your working session from your learning session
- Using Claude to explain its own commands and code
- /learning mode
- Verification habits - spot-checking what the model states most confidently
- The research-plan-implement workflow
- Sub-agents for context isolation

Debugging and recovery

- Feeding errors to Claude
- Recognizing failure modes: incoherence, ignoring corrections, looping
- Context clearing and fresh sessions
- The "hand off" - summarizing progress for a new session

Teaching your agent about you

- CLAUDE.md - how Claude should behave: personality, rules, preferences
- MEMORIES.md - what Claude should know: facts about you, your work, your domain
- Notes, todo lists, and reference material as files
- The introduction exercise - writing CLAUDE.md like you're onboarding a new colleague

Skill files and recipes

- Writing a skill file - trigger, instructions, expected output
- Downloading and sharing skills (skills.sh)
- Examples: meeting notes processor, email drafter, weekly review
- Skills vs. scripts - natural language recipes vs. deterministic programs

MCP and external tool connections

- Model Context Protocol (MCP)
- Setting up MCP servers
- Calendar, email, messaging, office tools, design tools, knowledge management, file storage
- MCP vs. skills - access to services vs. instructions for tasks

04

Building personal software

Context engineering

- Structuring what goes into the context window
- Keeping sessions focused and short
- Frequent intentional compaction
- Using skills and scripts for context injection
- CLAUDE.md structure and organization

The compound engineering technique

- The loop: plan, work, review, compound
- The 50/50 rule - half building, half improving systems
- Encoding taste and judgment into CLAUDE.md, skill files, and scripts
- The three review questions: hardest decision, rejected alternatives, lowest confidence
- The adoption stages: chat-based → agentic with review → plan-first
- What compounds well vs. what doesn't

Building and publishing websites

- The request-response model: clients, servers, URLs, browsers
- Hosting, domains, and deployment
- Recommended languages and tools
- From local files to a live site on the internet

Building personal tools and software

- One-off tools, personal software, data visualizations, dashboards, presentations, games
- The practice loop: need, describe, iterate, deploy, use, improve
- Scoping down - building the smallest version first

Automating workflows

- When to script: things you've done manually three or more times
- Natural language recipes (skills) vs. deterministic programs (scripts)
- The Unix philosophy - composability, clear inputs and outputs, text interfaces
- Seeing like an LLM - designing tools for agent use

Leveraging existing software

- Open source software and package managers
- Finding and evaluating existing tools: stars, recent activity, documentation, community
- Off-the-shelf tools that compound - libraries, frameworks, plugins

APIs and data ingestion

- APIs and calling APIs from scripts
- Types of integrations: bots, webhooks, notifications
- Exporting data from apps and services you use
- Web scraping
- Passive and continuous data capture

Data storage and retrieval

- Flat files: markdown, JSON, CSV
- Local databases with sqlite
- Cleaning and normalizing data from different sources
- Search and retrieval tools
- Backups and sync

05

Your always-on assistant

Scheduled tasks and local automation

- Cron jobs and task schedulers
- Automated data pipelines: sync, indexing, ongoing capture
- Teaching your agent to scan for changes and flag what matters
- Defensive automation: logging, dry runs, alerts

Cloud deployment and remote computing

- VPS and home server setup
- Secure networking between your devices
- Syncing your environment: CLAUDE.md, skill files, scripts, MCP configurations
- Remote access from your phone and messaging apps
- Running multiple agents in parallel in the cloud

Patterns for autonomous computing

- The event queue - how your computer knows when something needs attention
- The heartbeat - intelligent checks vs. fixed scheduled actions
- The materialize pattern - converting data between formats automatically
- The verify pattern - deduplication, staleness checks, consistency validation
- Safety: logging, human-in-the-loop checkpoints, rollback

06

Advanced topics

Local models and local inference

- Running LLMs on your own hardware
- When local is useful: privacy, offline, cost
- When cloud models are still better
- Fine-tuning and the broader model ecosystem

Custom agent harnesses

- The minimal harness: an API call, a tool definition, a loop, a system prompt
- Purpose-built agents: bots, email processors, voice assistants
- Extensibility: skills, prompt templates, custom commands
- When to use a general-purpose agent vs. a custom harness

Keep building

This syllabus is a guide to the program's structure, outcomes, and curriculum.

READ MORE AT FRACTALACCELERATOR.COM/SYLLABUS/CONTENT