

Claude Code for Everyone

A practical online course on Claude Code: an AI agent on your own computer, in five live Zoom sessions, taught hands-on in the Claude app

Dates 12.11 • 15.11 • 22.11 • 29.11 • 6.12.2026	Time 17:30–20:00 • 5 × 2.5 hours	Format 12.5 hours, live on Zoom, recorded
Lecturer Shiri Zohar	Registration fee ₪100 (non-refundable)	Open to The entire Technion community

// Course Overview

This 12.5-hour course teaches participants how to use Claude Code, Anthropic's AI agent, as a general-purpose assistant for everyday knowledge work, personal projects, and small interactive tools. The center of gravity of the course is Claude Code inside the Claude app: the Code tab of the Claude Desktop app, a visual interface where you chat with the agent, approve its plans, and review every change it makes, with no terminal and no code editor required. Alongside it, participants use Cowork, the same agent for document-centric everyday work, and learn when each fits the task.

As a complement, the course also introduces Visual Studio Code (VS Code), the most widely used code editor in the world, with the Claude Code extension. Participants learn what it is, when it is worth opening, and how to use it to see and edit the files Claude works on, so they can continue in a professional environment if and when they need it.

No programming background is needed. Participants learn to direct an agent: describe an outcome, review the plan, and let Claude do the building, including small working web tools they keep and reuse. Every session produces something you can open in a browser, not just a document.

A central theme of the course is **context engineering**: giving an AI agent the right instructions, the right files, and the right structure so it produces the work you actually want. The course's second theme is **reuse**: turning one-off successes into skills, that is, packaged workflows you can run again, share, or import from a public marketplace.

Format. 5 live online sessions on Zoom × 2.5 hours (12.5 hours in total), held on 12.11, 15.11, 22.11, 29.11 and 6.12.2026. The course is delivered entirely via Zoom, not in person: participants join from their own computers, so everyone works hands-on in the Claude app on their own machine from the first minute. Each session follows a workshop structure: a presentation with a live screen-shared demonstration, followed by a focused hands-on segment, with a short break in the middle. All practice happens during the live sessions: there are no home tasks and no final project. Sessions 1 and 2 cover foundations (agents, the Claude app as a working environment, and context engineering); Session 3 covers reuse (skills and Cowork); Sessions 4 and 5 cover connection and orchestration (MCP and Connectors, sub-agents, and sharing your work). Installation is completed before the course begins, so Session 1 starts with everyone already installed and ready to work.

Syllabus. Pace, depth, and emphasis are adjusted to the group: what people find hard and what they're most interested in. Topics may be expanded, condensed, reordered, or swapped as the course progresses.

Source material. The course is based directly on the official Claude Code and Cowork documentation (<https://code.claude.com/docs> and <https://claude.com/docs/cowork>). Specific pages are referenced once per session to guide independent study.

// Lecturer

Shiri Zohar, AI Team Lead at monday.com. Shiri leads an AI team at monday.com and brings that hands-on, real-world experience into every session, with live screen-shared demonstrations throughout the course.

// Where We Work: The Claude App First, VS Code as a Complement

Most of the course takes place in the Claude app, because it is the most accessible way to work with Claude Code: everything is visual, nothing needs to be typed into a terminal, and the same app also hosts Cowork. In the Claude app, participants learn to:

- **Start and manage Claude Code sessions:** open a project folder, give the agent a task, and follow its progress step by step.
- **Review plans and changes:** approve a plan before anything happens, and read exactly what changed in each file before accepting it, the core habit of working safely with an agent.
- **Control permissions:** decide how much the agent may do on its own, from asking before every step to working freely inside an approved plan.
- **Use skills, connectors and Cowork:** all from the same app, with the same account and the same files.

VS Code is taught as a complement, not as the main environment:

- **What it is and when to use it:** a short, gentle introduction for non-programmers, and a clear rule for when it is worth opening.
- **A window into your files:** seeing the folder Claude works in, opening and editing the markdown files that configure it (CLAUDE.md, skills, sub-agents) by hand.
- **The Claude Code extension:** the same agent inside VS Code, for those who want to continue in a professional development environment after the course.

// Application Areas

The course is designed to be useful to anyone, in any field. Participants apply Claude Code to tasks drawn from their own lives and work, including:

- **Personal productivity:** organizing folders and photo libraries, taming a messy Downloads directory, planning trips and events, summarizing documents.
- **Small interactive tools:** single-file web apps built for you by the agent: flashcard decks, quizzes, trackers, dashboards, calculators for whatever you care about.
- **Writing and research:** drafting in your own voice, comparing documents, building digests from files, email, or the web.
- **Light automation at work or home:** recurring reports, briefings from connected sources like calendar and mail, reusable workflows packaged as skills.

// Learning Outcomes

By the end of the course, participants will be able to:

1. **Understand what an AI agent is:** the three building blocks of tools, memory, and tasks, and how Claude Code embodies them.
2. **Work confidently with Claude Code in the Claude app:** start sessions on a project folder, review plans and changes, and steer the agent to a finished result.
3. **Practice context engineering:** write effective prompts and author a CLAUDE.md file (the project instruction file Claude reads at the start of every session) that shapes Claude's behavior.
4. **Use permission modes deliberately:** from asking before every step to working freely inside an approved plan, with plan mode as the standing habit for anything consequential.
5. **Build and import skills:** create a reusable skill of their own, and install skills and plugins from public marketplaces.
6. **Use Claude Cowork:** run the same agent for document-centric work, and choose between Cowork and Claude Code for a given task.
7. **Connect Claude to external services:** via MCP (Model Context Protocol) servers and Connectors (Gmail, Google Drive, Calendar), and to the open web via search and fetch.
8. **Build and orchestrate sub-agents:** define specialized agents and coordinate two of them on one job.
9. **Know when and how to use VS Code:** open the same project in VS Code, read and edit its files, and use the Claude Code extension there when a professional environment is needed.
10. **Share a working setup on GitHub:** publish a project folder with a README so someone else can use it.

// Course Structure

Session	Topic	Core focus	Date
1	What Is an AI Agent? Claude Code in the Claude App and Your First App	Agents (tools, memory, tasks), the Claude app as a working environment, a first working web app, a first look at VS Code	12.11
2	Context Engineering: Permission Modes and CLAUDE.md	Prompting, permission modes, plan mode, authoring CLAUDE.md, memory layers	15.11
3	Skills and Cowork: Package, Import, Reuse	Skill anatomy, building skills, plugins and marketplaces, running skills in Cowork, choosing the surface	22.11
4	Connecting Claude to Your World: MCP and Connectors	MCP servers, Connectors (Gmail/Drive/Calendar), web search, a connected dashboard	29.11
5	Sub-Agents, Orchestration, VS Code and Sharing Your Work	Sub-agents, orchestrating a two-agent job, VS Code with the Claude Code extension, publishing to GitHub, what's next	6.12

Each session is 2.5 hours, for a total of 12.5 hours, all live on Zoom. Each session follows the same workshop rhythm: a presentation with a live demonstration, then hands-on practice, with a short break in the middle and a brief recap at the end.

// In-Class Practice

There are no home tasks and no final project. All practice happens live during the Zoom sessions, in each session's hands-on segment, with the lecturer available for questions and screen-share troubleshooting. Every session's labs apply that session's material to something from your own life: extending your own mini-app, taming one of your own folders, packaging your own workflow as a skill, building a mini-assistant from your own connected sources, and publishing your own two-agent workflow. By the end of Session 5 you leave with a small portfolio of working tools rather than one big deliverable.

// Prerequisites and Tools

Prerequisites

- A computer running macOS or Windows (10/11) with at least 8 GB RAM, a stable internet connection, and Zoom installed (a second screen is recommended, so you can follow the screen share while working in the Claude app).
- A Claude Pro account or higher (<https://claude.ai>). Claude Code is not available on the free plan: you need at least Pro.
- No prior programming or terminal experience required.

Tools used

- **Claude Desktop app (Code tab):** Claude Code in a visual interface, the primary working environment throughout the course.

- **Claude Desktop app (Cowork):** the same agent for document-centric everyday work, used from Session 3 onward.
- **Visual Studio Code with the Claude Code extension:** introduced as a complement, for viewing and editing project files and for continuing in a professional environment.

All software is installed before Session 1; a short setup guide is provided to participants ahead of the course.

Session 1.

12.11

What Is an AI Agent? Claude Code in the Claude App and Your First App

Learning objectives

- Understand the three pillars of an AI agent: tools, memory, and tasks.
- Find your way around Claude Code in the Claude app: sessions, the project folder, plans, and the change review.
- Understand that the selected folder is Claude's entire world (folder scoping as a safety feature).
- Direct Claude to build a complete, working single-file web app.
- Get a first look at VS Code and understand its role as a complement in this course.

Presentation & live build

1. **What "AI" means today:** large language models, what they're good at, what trips them up: hallucinations, the context window, and the fact that a model on its own can't do anything until you give it tools.
2. **From model to agent:** the three pillars. Tools: actions the AI can take (read a file, run a command, search the web). Memory: what it retains across turns and sessions. Tasks: how it breaks a goal into steps. A chat answers; an agent acts.
3. **The Claude product family:** claude.ai in the browser, the Claude Desktop app and its modes (Chat, Cowork, Code), and Claude Code in VS Code. Why this course works mainly in the Claude app: everything is visual and accessible, with no terminal required.
4. **A tour of Claude Code in the Claude app:** starting a session on a folder, following the agent as it works, approving steps, and reviewing every change it made.
5. **"Your folder is Claude's world":** Claude only sees files inside the folder you selected. Choosing the folder is the first act of context engineering.
6. **Live build: a flashcards app.** From an empty folder and one well-formed prompt, Claude builds a single-file HTML flashcards app: ten flip cards, animations, no installation, opens in any browser. The point: a non-programmer directed an agent to working software in minutes.
7. **A first look at VS Code:** what a code editor is, opening the same folder in VS Code to see the files Claude created, and markdown in five minutes: every Claude customization in this course (instructions, skills, sub-agents) is a plain markdown file you can read and edit.

Hands-on

- **Lab 1.1:** Open the Claude app, start a Claude Code session on your course folder, verify you're signed in, send a hello.
- **Lab 1.2:** Build your own single-file web tool (a flashcard deck, a trivia game, a quiz, or a study timer) on any topic you care about. Prompt, review, open it in your browser.
- **Lab 1.3:** Iterate: ask Claude to change one thing (colors, card count, difficulty) and review exactly what changed before accepting it.
- **Lab 1.4:** Open the same folder in VS Code and find the file Claude built. Notice that it is the same project, seen from a different window.

Session 2.

15.11

Context Engineering: Permission Modes and CLAUDE.md

Learning objectives

- Write effective prompts: be specific, give examples, name the desired output.
- Understand the permission modes and pick the right one for a task; adopt plan mode as the standing habit.
- Author a CLAUDE.md file and understand the memory layers (project, user, auto).
- Watch instructions in a file visibly change what the agent builds.

Presentation & live build

1. **What a harness is:** the model itself just produces text; Claude Code wraps it with tools, permission rules, and the loop that keeps it going. Everything in this course (modes, CLAUDE.md, skills, MCP) is part of that wrapper.
2. **Context engineering, the basics:** three levers: the prompt, the files in scope, and the persistent instructions. The same task with a vague prompt versus a specific one, live.
3. **Permission modes:** the dial that controls how much Claude may do without asking: asking before editing files or running anything; plan mode (researches and writes a plan but changes nothing until you approve, the recommended starting habit); accepting edits automatically once a plan is approved; and more autonomous modes, explained with their safety trade-offs. Switching modes in the Claude app.
4. **The plan, approve, execute loop:** the safe rhythm for anything consequential: plan mode to survey and propose, approve, execute, then review the changes in the Claude app.
5. **CLAUDE.md:** the most important file in your project: a markdown file Claude reads at the start of every session in that folder. Anatomy of a good one: who you are, what the folder is for, how to talk to you, always/never rules, file conventions, common tasks. The single biggest mistake: vague rules.
6. **Memory layers:** project memory (./CLAUDE.md), user memory (applies everywhere), and Claude's auto memory. Where a given instruction belongs, and how to split a CLAUDE.md that grows too long.
7. **Live build: instructions that visibly steer.** Write a CLAUDE.md with concrete rules (a color palette, a tone, a single-file HTML rule), then ask Claude to build an interactive tool and watch the file's rules show up in the result. Change a rule, rebuild, see the difference.

Hands-on

- **Lab 2.1:** Specificity test: the same request vague, then specific. Compare outputs.
- **Lab 2.2:** Run one small task fully in plan mode, approve, execute, and review the result in the Claude app.
- **Lab 2.3:** Write a CLAUDE.md for your course folder (ask Claude to draft it, or use the template). Start a new session and verify Claude follows it. Optionally, open and edit the file in VS Code.
- **Lab 2.4:** Tame a folder. Pick a real, messy folder you brought. In plan mode: survey it, approve a reorganization, execute it, and write a CLAUDE.md so future sessions know the rules.

Session 3.

22.11

Skills and Cowork: Package, Import, Reuse

Learning objectives

- Understand what a skill is and when to reach for one.
- Build a custom skill; understand where skills live (personal, project, plugin).
- Install skills and plugins from a marketplace; adopt the borrow, try, improve, share pattern.
- Use Cowork, run a skill there, and choose between Cowork and Claude Code for a task.

Presentation & live build

1. **What a skill is:** a packaged, reusable workflow: a folder with a SKILL.md file (a short header with a name and description, followed by instructions), plus optional helper files and scripts. Claude loads it automatically when the task matches the description. Rule of thumb: solved something twice? Package it as a skill.
2. **Skill anatomy, built live:** turn Session 1's HTML-tool workflow into a skill (name, description, steps, design rules). Run it with one short prompt and watch it reproduce the whole workflow.
3. **Where skills live:** personal (follows you everywhere), project (travels with the folder), and plugin (installed as a package).
4. **Building skills with a skill:** install the official skill-creator plugin and let it interview you into a well-formed skill.
5. **Plugins and marketplaces:** a plugin bundles skills into a shareable package; a marketplace is a catalog you add and install from. Tour of the official directory, including the document skills that produce real PowerPoint, Word, Excel, and PDF files.
6. **Cowork: the same agent for everyday work.** Cowork is the Claude Desktop app's agentic workspace: the same architecture as Claude Code, focused on documents and everyday tasks. It works on local files and produces professional outputs. Your skills work in Cowork too, the same SKILL.md you built in Claude Code. When to pick which: Cowork for document-centric, everyday tasks; Claude Code when you want to build and steer every file change.

Hands-on

- **Lab 3.1:** Package one thing you already do into a skill (with Claude or with skill-creator). Test it with a one-line prompt in a fresh session.
- **Lab 3.2:** Add the official marketplace, install one plugin, and use it (for example the pptx document skill to produce a five-slide deck from notes).
- **Lab 3.3:** Open Cowork, point it at the same folder, and run your skill there. Note what's different and what's identical.

Session 4.

29.11

Connecting Claude to Your World: MCP and Connectors

Learning objectives

- Understand what MCP (Model Context Protocol) is and what problem it solves.
- Connect an MCP server to Claude Code; check and use it.
- Connect everyday services (Gmail, Google Drive, Calendar) via Connectors, in Claude Code and in Cowork.
- Combine a skill with a connected source to produce a working dashboard.

Presentation & live build

1. **What MCP is, simply:** a standard "plug" that lets Claude talk to external apps and services; the same plug works across Claude products. Host, client, and server; local versus remote servers.
2. **A caution before connecting:** anything a server feeds Claude becomes context. Treat instructions arriving from outside (web pages, documents, emails) with the same skepticism you'd give a stranger's note.
3. **Connecting a server:** adding an MCP server, checking its status, and authenticating, from the Claude app. Live demo with a documentation server: ask a question Claude could not have answered without it.
4. **Connectors:** the consumer-friendly face of MCP: Gmail, Google Drive, Calendar and more, connected through your Claude account with a consent flow, and available in Cowork as well as Claude Code. Live: pull the coming week from Calendar.
5. **The open web:** built-in web search and fetch, and when the web is the right connector.
6. **Live build: a connected dashboard.** A skill plus a connected source, chained: fetch this week's calendar (or a Drive folder, or a set of web pages), distill it, and hand the result to a report skill that renders a one-page HTML dashboard. One step's output feeds the next.

Hands-on

- **Lab 4.1:** Add the documentation MCP server, verify it is connected, and ask it something real.
- **Lab 4.2:** Connect one Connector of your choice (Drive, Gmail, or Calendar). Run one practical query, for example find a recent document and summarize it, or list this week's commitments.
- **Lab 4.3:** Mini-assistant. Chain your CLAUDE.md, one of your skills from Session 3, and the connected source into a workflow that produces an HTML dashboard or digest you'd genuinely reuse.

Sub-Agents, Orchestration, VS Code and Sharing Your Work

Learning objectives

- Understand sub-agents: why one agent isn't always enough, and what a sub-agent is (its own context, its own constrained tools).
- Build a sub-agent and understand that its description steers automatic delegation.
- Orchestrate two sub-agents on one job.
- Use VS Code with the Claude Code extension as a complement, and know when it is worth moving there.
- Publish a project folder to GitHub with a README, with Claude doing the technical steps.
- Know the current wider landscape: Claude Code on the web and mobile, scheduled tasks, Claude in Chrome, and checkpoints.

Presentation & live build

1. **From one agent to a team:** the limits of a single context window; why splitting a job into specialized roles keeps each role sharp. A skill is instructions the main agent follows; a sub-agent is a separate worker with its own context and its own (possibly restricted) tools.
2. **Building a sub-agent:** asking Claude to create one for you, and what is inside: a markdown file with a name, description, allowed tools, and instructions. The description is the steering wheel for automatic delegation; you can also mention an agent explicitly to force it.
3. **Orchestrating two sub-agents, live:** a read-only researcher (may read files and search, may not write) and a builder (turns the researcher's findings into a product). One prompt to the main agent; it delegates in sequence; out comes a reviewed, working HTML artifact.
4. **VS Code as a complement:** opening the course project in VS Code, reading and editing the sub-agent and skill files by hand, and a short demo of the Claude Code extension: the same agent, inside a professional editor. When to stay in the Claude app and when VS Code is worth it.
5. **Sharing your work: GitHub in plain language:** what GitHub is (a public shelf for project folders), repository and README defined, and publishing without ceremony: Claude runs the technical steps, you supply the intent and see every step. One caution: never publish secrets (keys, passwords, personal data), and how to check before you push.

Hands-on

- **Lab 5.1:** Build two sub-agents: a read-only specialist (summarizer, fact-checker, or data-scanner) and a builder. Test each alone.
- **Lab 5.2:** Orchestrate: one prompt that makes the main agent run the specialist first and hand its output to the builder, ending in an HTML artifact.
- **Lab 5.3:** Open the project in VS Code, find your sub-agent files, and make one small edit by hand. Run the agent again in the Claude app and see the effect.
- **Lab 5.4:** Publish your course folder to GitHub with a README a stranger could follow. Share the link in the Zoom chat and try a classmate's.

// What's next

A short close on where this is going, every item official and current: Claude Code on the web and mobile (start agents from a browser or phone); scheduled tasks that run agents in the cloud or on your machine; Claude in Chrome (the agent driving your browser); and checkpoints with rewind, the built-in undo that makes experimentation safe. The full arc in one line: prompt, context, skills, connections, agents, shared.



t:pro • Professional Training Program of **t:hub**, Technion Hub for Innovation & Entrepreneurship

t-hub.technion.ac.il