COMPLETE CHEATSHEET

Claude Code for Beginners

The Complete Beginner's Cheatsheet

Everything you need to get started with AI-powered coding.
Install it, give it a memory, teach it your workflows,
and plug it into the tools you already use.

Difficulty: Beginner · Read Time: ~20 min · Tool: Claude Code · Cost: from $20/month

@learnwithseb
AI Growth Systems for Creators

What's in this cheatsheet

01 What is Claude Code?

02 How to install it — desktop app or VS Code

03 The CLAUDE.md file — your project's memory

04 Skills — just ask Claude to build one

05 Skills vs slash commands (read this one)

06 MCP servers and connectors

07 Essential tips and tricks

08 Common beginner mistakes

09 Plans and pricing

10 Quick reference

© 2026 Learnwithseb — Claude Code for Beginners

What is Claude Code?

Claude Code is an **AI-powered coding assistant** made by Anthropic. It lives in your code editor or your terminal, and it can read, write, and edit code for you. Think of it as a colleague sitting at your machine rather than a chatbot you paste snippets into.

What can Claude Code do?

- **Write code for you** — describe what you want in plain English
- **Fix bugs** — paste an error message and Claude finds and fixes the problem
- **Explain code** — highlight something you don't understand and ask about it
- **Refactor code** — clean up, reorganise, or improve what's already there
- **Run commands** — install packages, run tests, build, deploy
- **Read your files** — it sees your project and understands the full codebase
- **Create entire projects** — scaffold a whole app from scratch

Claude Code vs Claude.ai (web chat)

	CLAUDE CODE	CLAUDE.AI WEB CHAT
Where it lives	Your editor or terminal	A browser tab
Your files	Reads and edits them directly	Only what you paste in
Commands	Can build, test and deploy	Cannot touch your computer
Context	Understands the whole project	Only what you share
Best for	Building software, fixing bugs	Questions, writing, research

TIP

You don't need to be a programmer to use Claude Code. Plenty of non-technical people use it to build websites, automate tasks, and manage projects. If you can type a sentence, you can start.

Who needs Claude Code?

- **Developers** who want an AI pair-programmer to speed up their workflow
- **Non-technical founders** who want to build prototypes without hiring a developer
- **Content creators** who want to build tools, sites, or automate workflows
- **Students** learning to code who want an always-available tutor
- **Product managers** who want to prototype ideas and understand codebases

© 2026 Learnwithseb — Claude Code for Beginners

How to install Claude Code

Claude Code is one engine with several front doors. The two worth starting with are the **desktop app** and the **VS Code extension**. Both are set out in full below — pick the one that matches how you already work.

ROUTE	TERMINAL NEEDED	BEST FOR
Desktop app	No	Starting from scratch. Buttons, panes, visual diffs.
VS Code extension	No	You already work in VS Code all day.
Terminal (CLI)	Yes	Speed, once you're comfortable.

NOTHING IS WASTED IF YOU SWITCH

Desktop, VS Code, the terminal and claude.ai/code all run the same engine and share the same CLAUDE.md files, skills, MCP servers and settings. You can even run two at once on the same project.

What you need before you start

- ✓ A computer — Mac, Windows or Linux all work
- ✓ A Claude account — sign up at claude.ai
- ✓ A paid plan — Claude Code needs at least Pro ($20/month)
- ✓ On Windows: Git for Windows, from git-scm.com
- ✗ Any coding knowledge
- ✗ Any terminal experience

Option A — the desktop app

The friendliest way in. A proper application window with a sidebar, panes you can drag around, visual diffs and a built-in terminal. No editor required.

1. **Download it** from **claude.com/download**. macOS gets a universal build, Windows an x64 installer with a separate ARM64 option, Linux is in beta via apt.
2. **Run the installer**, then launch Claude from Applications on Mac or the Start menu on Windows. Sign in with your Anthropic account.
3. **Click the Code tab** at the top. The app has three: **Chat** for conversation, **Cowork** for background agent work, and **Code** for building software.

© 2026 Learnwithseb — Claude Code for Beginners

4. **Choose Local** as your environment, then click **Select folder** and pick the project you want to work in.
5. **Pick a model and a permission mode** from the dropdowns next to the send button. Start on **Manual** so Claude asks before every change.
6. **Type what you want** and press Enter. When Claude edits something, a **+12 -1** indicator appears — click it to open the diff and accept or reject each change.

WINDOWS USERS, READ THIS FIRST

Install **Git for Windows** from git-scm.com *before* you start a local session, then restart the app. Without it the Code tab won't run locally. Most Macs already have Git installed.

NO NODE.JS NEEDED

The desktop app ships with Claude Code inside it. You don't have to install Node.js or the CLI separately. If you later want **claude** in your terminal too, that's a separate install.

Option B — the VS Code extension

Best if VS Code is already where you live. Claude sits in a panel beside your files and automatically sees whatever you have selected. You'll need **VS Code 1.94 or later**.

1. **Open the Extensions panel.** Press **Ctrl+Shift+X** on Windows or Linux, **Cmd+Shift+X** on Mac.
2. **Search for "Claude Code"** and pick the one published by **Anthropic** — that's the official one. Click **Install**.
3. **Open the panel.** Look for the **Spark icon**: top-right in the editor toolbar when a file is open, or in the left Activity Bar, or click **Claude Code** in the status bar at the bottom-right.
4. **Sign in.** Click **Sign in** on the panel and finish in your browser. A **Learn Claude Code** checklist appears afterwards — it's worth ten minutes.
5. **Select some code and ask about it.** Claude sees your selection automatically. Press **Alt+K** (**Option+K** on Mac) to insert a precise reference like @app.js#12-18.
6. **Review the diff.** In Manual mode Claude shows a side-by-side comparison and waits. You can even edit its proposed code before accepting.

IF THE PANEL DOESN'T APPEAR

Restart VS Code, or run **Developer: Reload Window** from the Command Palette. The extension also installs in VS Code forks including Cursor. If your editor can't take it, install the CLI and run **claude** in the built-in terminal instead.

© 2026 Learnwithseb — Claude Code for Beginners

TRY IT — EITHER ROUTE

Once you're in, type a simple request like **"Create a hello world HTML page"**. Claude creates the file right in your project. Read the diff, accept it, and open the file to see the result. That's the whole loop in about a minute.

Key shortcuts to remember

ACTION	MAC	WINDOWS / LINUX
Cycle permission modes	Shift+Tab	Shift+Tab
Reference a file in a prompt	Type @filename	Type @filename
Open Extensions (VS Code)	Cmd+Shift+X	Ctrl+Shift+X
Jump editor ↔ Claude (VS Code)	Cmd+Esc	Ctrl+Esc
Reference selected lines (VS Code)	Option+K	Alt+K
New conversation (VS Code)	Cmd+N	Ctrl+N
Open the terminal pane (desktop)	Ctrl+`	Ctrl+`
Open the browser pane (desktop)	Cmd+Shift+B	Ctrl+Shift+B
Submit diff comments (desktop)	Cmd+Enter	Ctrl+Enter

The CLAUDE.md file

Your project's memory card

Every time you start a new conversation, Claude starts fresh with **no memory** of previous chats. The **CLAUDE.md** file solves this. It's a plain text file Claude reads **at the start of every session**, so it always knows your project's rules, preferences and context.

Think of it as a briefing document for a very capable new team member who has amnesia. You write it once, and Claude reads it every time.

Where does CLAUDE.md go?

LEVEL	LOCATION	WHAT IT'S FOR
Global	~/.claude/CLAUDE.md	Your personal preferences, across all projects
Project	your-project/CLAUDE.md	Team rules and project-specific context

© 2026 Learnwithseb — Claude Code for Beginners

LEVEL	LOCATION	WHAT IT'S FOR
Local	your-project/CLAUDE.local.md	Your own overrides, not shared with the team

TIP

If two levels conflict, the **more specific one wins**. Project overrides Global, and Local overrides Project.

What to put in it

Keep it short — under 200 lines is ideal. Here's a starter template:

```
# Project: My Awesome App

## What This Project Is
A task management web app built with React and Node.js.

## How to Run
- Install: npm install
- Start dev server: npm run dev
- Run tests: npm test

## Code Style Rules
- Use TypeScript for all new files
- Use 2-space indentation
- Always write tests for new features

## Important Notes
- Database is PostgreSQL, not MySQL
- We use pnpm, NOT npm
- Never commit .env files
```

Three ways to create one

1. **The easy way.** Type **/init**. Claude scans your project and generates a starter CLAUDE.md. Review it and delete anything obvious.
2. **The manual way.** Create a file called CLAUDE.md in your project root and add the sections above.
3. **The interview way.** Ask: *"Interview me to create a CLAUDE.md for this project."* Claude asks questions and builds the file from your answers.

© 2026 Learnwithseb — Claude Code for Beginners

BONUS: AUTO MEMORY

Claude Code also learns from your corrections automatically. Tell it *"use pnpm, not npm"* once and it saves that note for future sessions. Nothing to set up — check what it has saved with **/memory**.

Skills — your reusable workflows

A **skill** is a folder with a **SKILL.md** file inside it. You write instructions in plain markdown, and Claude adds them to its toolkit. No coding required.

Build one the moment you notice yourself pasting the same instructions twice, or when a section of your CLAUDE.md has grown from a fact into a procedure.

How Claude decides to use one



This is called **progressive disclosure**, and it's the whole trick. Only the short description sits in context up front. The full instructions load when something matches, which is why you can have a lot of skills without clogging the context window.

The easy way — ask Claude to build it

You almost never have to write a skill by hand. Describe the one you want in plain English and Claude creates the folder, writes the file, and tells you where it put it.

PROMPT — COPY & PASTE INTO CLAUDE CODE

Create a skill that summarises my uncommitted changes and flags anything risky.
I want to be able to run it by typing /summarize-changes.

That's the whole job. Read what it wrote, change any wording that doesn't sound like you, and it's live. No file formats to learn.

© 2026 Learnwithseb — Claude Code for Beginners

MAKE IT BETTER: THE SKILL-CREATOR PLUGIN

There's a purpose-built helper for this. Install it once with `/plugin install skill-creator@claude-plugins-official` and Claude will use it whenever you ask for a skill. It doesn't just draft one — it **tests and scores** it, checking whether the skill actually fires on the prompts it should and tightening the wording when it doesn't. If the install can't find the marketplace, run `/plugin marketplace add anthropics/claude-plugins-official` first.

ON A MAC? RECORD IT INSTEAD

In the desktop app go to **Customize** → **Skills** → **Add** → **Record your screen**, then do the task while Claude watches. It writes the skill from what you actually did. Pro, Max and Team plans, Mac only for now.

What Claude actually creates

Worth a look even if you never write one yourself. A folder, and one file inside it. The folder name becomes the command, so **summarize-changes** gives you **/summarize-changes**.

```
---
description: Summarises uncommitted changes and flags anything risky.
  Use when the user asks what changed, wants a commit message,
  or asks to review their diff.
---

## Current changes

!`git diff HEAD`

## Instructions

Summarise the changes above in two or three bullet points, then list
any risks you notice such as missing error handling, hardcoded values,
or tests that need updating. If the diff is empty, say so.
```

Put that folder in `~/.claude/skills/` and it works in every project. Put it in `.claude/skills/` inside a project and it stays with that project.

© 2026 Learnwithseb — Claude Code for Beginners

A NEAT TRICK IN THAT FILE

The !`git diff HEAD` line is **dynamic context injection**. Claude Code runs that command and pastes the real output in before Claude reads a word, so the answer is grounded in your actual work rather than a guess.

Frontmatter, in plain English

Every SKILL.md has two parts, and the jargon hides how simple it is.

The bit at the very top, fenced between two --- lines, is called the **frontmatter**. It's called that because it sits at the front. Think of it as **the label on a jar**. Everything below it is **the recipe inside**.

	THE LABEL (FRONTMATTER)	THE RECIPE (THE INSTRUCTIONS)
Where it is	Between the two --- lines	Everything below the second ---
What it looks like	One line per setting, name: value	Ordinary writing, plain markdown
When Claude reads it	Always, every session	Only once the skill is chosen
What it's for	Deciding which jar to open	What Claude actually follows
How long	A line or two	As long as it needs to be

That's the whole idea. Claude scans the labels on all your jars, opens the one that matches what you asked for, and follows the recipe inside. Nothing else gets opened, which is why a hundred skills cost you almost nothing.

THE LINE THAT MATTERS MOST

You only ever need **one** label line: **description**. It's how Claude decides whether to reach for your skill, so write it using the words you'd actually say. *"Use when the user asks what changed"* beats *"summarises diffs"*. If a skill never fires, this is almost always why — and if you asked Claude to build it, it writes this line for you anyway.

© 2026 Learnwithseb — Claude Code for Beginners

What goes in the folder

```
my-skill/
■■■ SKILL.md        # required. the overview and the instructions
■■■ reference.md    # detailed docs, loaded only when needed
■■■ examples.md     # worked examples, same deal
■■■ scripts/
    ■■■ helper.py   # executed, never loaded into context
```

Point at the extra files from inside SKILL.md so Claude knows what each one holds. That's how a skill can carry a hundred pages of reference material and still cost almost nothing until the day you need it.

The label settings worth knowing

FIELD	WHAT IT DOES
description	What the skill does and when to use it. The one field you should always write.
when_to_use	Extra trigger phrases and example requests, appended to the description.
disable-model-invocation	Set true and only you can run it. Use for anything with side effects.
user-invocable	Set false and only Claude can run it. Use for background knowledge.
allowed-tools	Tools Claude may use without asking, just for that turn.
argument-hint	What to show in autocomplete, like [issue-number].

All optional. Only **description** really matters, and Claude writes that one for you.

Passing arguments

Anything you type after the skill name lands in **$ARGUMENTS**. A fix-issue skill containing *"Fix GitHub issue $ARGUMENTS following our coding standards"* turns **/fix-issue 123** into a real instruction.

Where skills live

SCOPE	PATH	WHO GETS IT
Personal	~/.claude/skills/<name>/SKILL.md	Every project on your machine
Project	.claude/skills/<name>/SKILL.md	This repo, and anyone you share it with
Plugin	<plugin>/skills/<name>/SKILL.md	Wherever that plugin is switched on

© 2026 Learnwithseb — Claude Code for Beginners

When two skills share a name, personal beats project.

KEEP IT SHORT

Once a skill loads, its body stays in context for the rest of the session, so every line is a recurring cost. Keep SKILL.md under 500 lines and push the detail into supporting files.

SKILLS AREN'T JUST FOR THE TERMINAL

In the desktop app and on claude.ai, open **Customize** → **Skills** to browse and switch on ready-made ones: Anthropic's own, partner skills from Notion, Figma and Atlassian, and anything your organisation has published. Skills follow the **Agent Skills** open standard at agentskills.io, so one you write is portable to other AI tools.

Skills vs slash commands

This is the bit that confuses everyone, and most guides still get it wrong. **They are not two competing systems.** Custom commands have been merged into skills. A file at .claude/commands/deploy.md and a skill at .claude/skills/deploy/SKILL.md both give you **/deploy**, and both behave the same way.

A slash command is simply **how you invoke** a skill. The / is the doorway, not the thing behind it.

	COMMAND FILE	SKILL FOLDER
Gives you /deploy	Yes	Yes
Can hold extra files	No, one file only	Yes, a whole folder
Claude can load it unprompted	No	Yes
Control over who invokes it	Limited	Full frontmatter
Still supported	Yes, nothing breaks	Recommended for new ones

The real distinction: who starts it

FRONTMATTER	YOU CAN INVOKE	CLAUDE CAN INVOKE	USE IT FOR
(default)	Yes	Yes	Most skills
disable-model-invocation: true	Yes	No	Side effects: /deploy, /commit, /send-slack-message

© 2026 Learnwithseb — Claude Code for Beginners

FRONTMATTER	YOU CAN INVOKE	CLAUDE CAN INVOKE	USE IT FOR
user-invocable: false	No	Yes	Background knowledge: conventions, legacy system context

RULE OF THUMB

If running it by accident would be bad, lock it to you with **disable-model-invocation: true**. You don't want Claude deciding to deploy because your code looks ready.

Built-in commands you get for free

COMMAND	WHAT IT DOES
/init	Scans your project and creates a starter CLAUDE.md
/clear	Clears the conversation history (saves tokens)
/compact	Summarises the conversation to free up context space
/cost	Shows how many tokens and how much money you've used
/model	Switch between Claude models
/memory	View and manage Claude's auto-saved memory notes
/mcp	Check and authenticate your connected MCP servers
/help	Shows all available commands

MCP servers and connectors

Skills teach Claude **how** to do something. MCP gives Claude **access** to something. The **Model Context Protocol** is an open standard for connecting AI tools to external services — think of it as a universal plug.

Connect a server the moment you catch yourself copying data into chat from another tool. Once it's connected, Claude reads and acts on that system directly instead of working from what you paste.

© 2026 Learnwithseb — Claude Code for Beginners

What you can do with it

- **Work from issue trackers** — "Add the feature described in JIRA ENG-4521 and open a PR"
- **Query your database** — "Find the ten most recent signups in our PostgreSQL database"
- **Read monitoring data** — "Check Sentry for errors on the checkout page this week"
- **Pull in designs** — "Update the email template to match the new Figma design"
- **Automate across tools** — "Draft Gmail invites to these ten users"

Connectors and MCP servers are the same plumbing

	CONNECTORS	MCP SERVERS
Where you set them up	Claude desktop app and claude.ai	Claude Code, from the terminal
How you add one	Browse the directory, click Connect	claude mcp add
Authentication	Sign in through your browser	OAuth via /mcp
Underneath	MCP	MCP

A connector is a reviewed, one-click MCP server. Anything in the Anthropic Directory at claude.ai/directory can also be added to Claude Code by hand.

In the desktop app or on claude.ai

1. Open the sidebar and choose **Customize**.
2. Click **+** next to **Connectors**, then **Browse connectors**.
3. Find the service you want and click **Connect**.
4. Sign in when your browser opens. Claude now has that tool available.

In Claude Code

Most modern services expose an HTTP endpoint. That's the one to reach for:

```
# Add a remote HTTP server (the usual case)
claude mcp add --transport http notion https://mcp.notion.com/mcp

# Add a local server that runs on your machine
claude mcp add --env AIRTABLE_API_KEY=YOUR_KEY --transport stdio \
  airtable -- npx -y airtable-mcp-server

# See what you have, and whether it connected
claude mcp list

# Remove one
claude mcp remove notion
```

© 2026 Learnwithseb — Claude Code for Beginners

THE DOUBLE DASH MATTERS

For local (stdio) servers, the **--** separates Claude's own options from the command that runs the server. Everything after it is passed through untouched. Leave it out and Claude Code tries to parse the server's flags as its own.

Where a server is saved

SCOPE	LOADS IN	SHARED WITH TEAM	STORED IN
Local (default)	Current project only	No	~/.claude.json
Project	Current project only	Yes, via version control	.mcp.json in the repo
User	All your projects	No	~/.claude.json

Add `--scope project` to share a server with your team, or `--scope user` to have it in every project you open. Project-scoped servers ask for your approval the first time, since they arrive with someone else's config.

Signing in

Many hosted servers need authentication. Add the server, then type **/mcp** inside Claude Code and follow the browser prompts. Tokens are stored securely and refreshed automatically. Use **/mcp** again any time to check status, re-authenticate, or clear access.

ONLY CONNECT SERVERS YOU TRUST

An MCP server can read data and take actions on your behalf. Servers that fetch external content can expose you to **prompt injection**, where instructions hidden in a web page or ticket try to get Claude to do something you didn't ask for. Stick to the reviewed Anthropic Directory where you can, and treat an unknown third-party server the same way you'd treat an unknown browser extension.

WORTH KNOWING

Plugins are the third piece of the puzzle: a package that can bundle skills, connectors, agents and MCP servers together. Browse all three — skills, connectors and plugins — in one place under **Customize**, or at claude.ai/directory.

© 2026 Learnwithseb — Claude Code for Beginners

Essential tips and tricks

How to talk to Claude Code

- **Be specific.** "Add a login page with email and password fields" beats "make a login".
- **Reference files with @.** Type @filename.js to point Claude at a specific file.
- **Highlight code first.** Select code in your editor, then ask about it — Claude sees your selection.
- **Break big tasks into steps.** "Create the project structure", then "add the homepage", not "build me an app".
- **Review before accepting.** Claude shows you a diff of every change. Read it.

Context management

Claude has a **context window** — a limit on how much it can hold at once. Every message you send stays in it. Fill it up and answers get slower and vaguer.

1. **Use /clear often.** Start fresh when you switch tasks.
2. **Use /compact when sessions run long.** It summarises so you keep working without losing the thread.
3. **Keep CLAUDE.md concise.** Every line is paid for on every message.
4. **One task per conversation.** Don't jump between unrelated jobs in the same chat.

Permission modes

Press **Shift+Tab** to cycle how much Claude does without asking. Start on Manual and move along as your trust grows.

MODE	WHAT HAPPENS
Manual	Asks before every file edit and command. Slowest, safest, best for week one.
Accept edits	Applies file edits automatically, still asks before running commands.
Plan	Explores and proposes an approach without touching your code.
Auto	Runs everything, with background safety checks reviewing the risky bits.

© 2026 Learnwithseb — Claude Code for Beginners

Power user features (when you're ready)

- **Plan mode** — have Claude map out an approach before it writes anything
- **Parallel sessions** — several conversations at once, each isolated from the others
- **Sub-agents** — Claude spawns helpers for large jobs, like reviewing while it writes
- **Scheduled tasks** — a daily code review or a weekly dependency audit, on a timer
- **Plugins** — bundle your skills, agents and MCP servers and share them as one package

Common beginner mistakes

WHAT TRIPS PEOPLE UP	WHAT TO DO INSTEAD
✗ Writing a huge CLAUDE.md file	✓ Keep it under 200 lines. Less is more.
✗ Never using /clear	✓ Clear between tasks to keep context fresh
✗ Accepting changes without reading	✓ Always read the diff before accepting
✗ Giving vague instructions	✓ Be specific: what, where, and how
✗ Doing everything in one prompt	✓ Break complex jobs into smaller steps
✗ Ignoring error messages	✓ Paste them in — it's excellent at debugging
✗ A skill description like "helps with code"	✓ Write the words you'd actually say out loud
✗ Connecting any MCP server you find	✓ Stick to the reviewed directory unless you know the source

Plans and pricing

Claude Code needs a paid subscription. The short version:

PLAN	PRICE	CLAUDE CODE?	BEST FOR
Free	**$0/month**	No	Trying Claude chat only
Pro	**$20/month**	Yes	Learning, small projects
Max 5x	**from $100/month**	Yes (more usage)	Regular professional use
Max 20x	**from $200/month**	Yes (heavy usage)	Full-time daily building

© 2026 Learnwithseb — Claude Code for Beginners

RECOMMENDATION

Start with **Pro at $20/month** to learn. Upgrade to Max only when you're hitting usage limits regularly. Check what you're using any time with **/cost**. Prices change — confirm at claude.com/pricing before you commit.

Quick reference

What a set-up project looks like

```
your-project/
■■■ CLAUDE.md           # project memory, shared with the team
■■■ CLAUDE.local.md     # your own overrides, gitignored
■■■ .claudeignore       # files Claude should never read
■■■ .mcp.json           # MCP servers shared with the team
■■■ .claude/
    ■■■ skills/         # your custom skills
    ■   ■■■ review/
    ■   ■   ■■■ SKILL.md    # becomes /review
    ■   ■■■ deploy/
    ■       ■■■ SKILL.md    # becomes /deploy
    ■■■ commands/       # older single-file commands, still work
        ■■■ notes.md    # becomes /notes
```

None of this is required on day one. Add each piece when you need it.

Essential links

RESOURCE	WHERE
Official docs	code.claude.com/docs
Download the desktop app	claude.com/download
VS Code extension	Search "Claude Code" in Extensions
Skills, connectors and plugins	claude.ai/directory
Sign up / account	claude.ai
Pricing	claude.com/pricing
Help and support	support.claude.com

© 2026 Learnwithseb — Claude Code for Beginners

Your first five minutes

1. **Install it** — desktop app or VS Code extension, both in section 02.
2. **Open a project folder** — any folder will do, or create a new one.
3. **Type /init** — let Claude write a CLAUDE.md for your project.
4. **Ask for something small** — try "create a simple webpage with a welcome message".
5. **Read the diff and accept** — then open the file and see the result.

REMEMBER

The best way to learn Claude Code is to **use** it. Start with small tasks, review every change, and take on bigger jobs as your trust grows. The gap between people who get value from this and people who don't is almost entirely reps.

BuildWithSeb — Claude Code for Beginners
Built by **@learnwithseb** · Start small. Read every diff. Build the habit.
© 2026 Learnwithseb. All rights reserved.