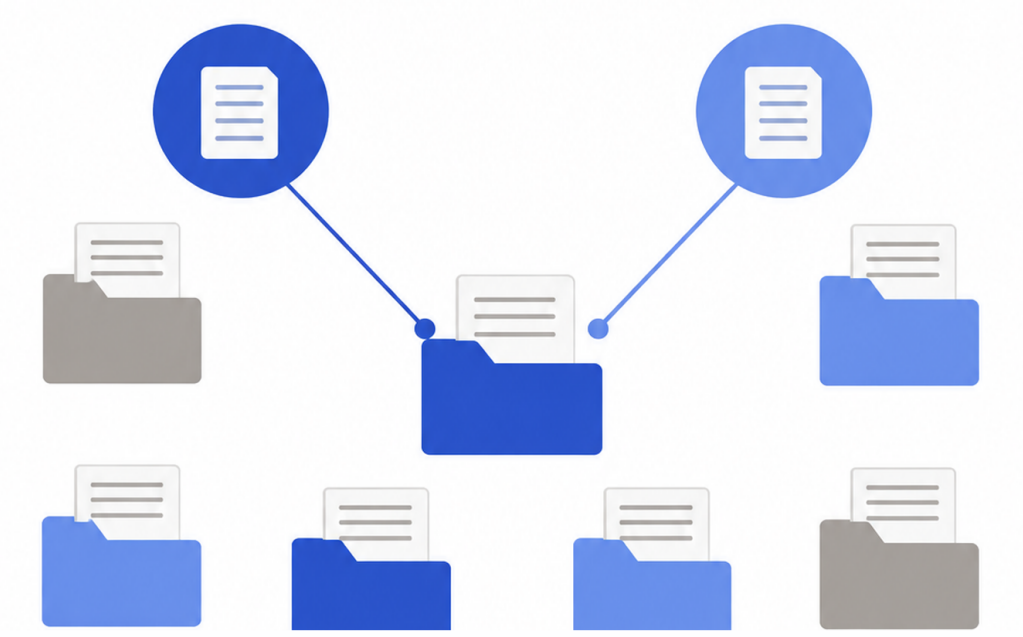
CLAUDE CODE × CODEX

The Claude Code & Codex Skills Guidebook

53 agent skills tested in both tools



Jay Jung Lim

BuildnWrite

The Claude Code & Codex Skills Guidebook

53 agent skills tested in both tools

Jay Jung Lim

About this book 5
About this edition 5
About the author 6
Copyright 6
00-1 How to read this book 7
Where to start by reader type 7
Reference versions 8
Notation 8
What this book does not cover 9
Update commitment 9
What is included and excluded? 9
Which Part should you read first? 9
How to read the ratings 10
Frequently asked questions 10
Sources 10
Recap and the next chapter 11
00-0-1 53 recommended Claude skills and Codex skills 12
53 skills in this book 12
00-2 What is an agent, and where do skills fit? 15
How is an agent different from a chatbot? 15
Six things that accompany an agent 16
When do they step in, and who calls them? 17
How is a skill different from a subagent? 18
Questions to help you choose 19
Where do they go in each product? 19
What went wrong? 19
Recap and the next chapter 20
Frequently asked questions 20
Sources 20
02-02 ★★☆ humanizer: Editing out machine-like prose 21
At a glance 21
Who it fits 21
Install 22
First run 22
What makes it useful 23
What's included 23
Watch out for 23
Frequently asked questions 23
Sources 24
05-01 ★★★ How to use superpowers: Development methodology skill pack 25
At a glance 25
Who it fits 25
Install 26
First run 26
What makes it useful 27
What's included 27

Watch out for....27
Frequently asked questions....27
Sources....28
End of the free sample....29

About this book

Agent skills are work manuals that `Claude Code` and `Codex` open when a task calls for them. Thousands of them sit on GitHub, but most come with a terse README and install steps written for developers. This book picks 53 skills worth your time and tests each one in both tools.

Every skill chapter answers the same questions in the same order:

- **Who it fits.** A table of situations where the skill is a good fit, an okay fit, or a poor fit, with a reason for each.
- **Install.** Commands for `Claude Code` and for `Codex`.
- **First run.** What to type and what the skill does step by step.
- **Watch out for.** Limits, risks, and what the skill will not do.
- **Sources.** Repository, license, and the files the chapter quotes.

The foundation chapters explain what agents, skills, subagents, MCP, and hooks are, how the two tools read the same skill folder, and how to share one skill folder between them. If you have never installed either tool, start with "Getting started with Claude Code and Codex."

About this edition

This is the English edition of a guidebook first published in Korean in September 2026. The Korean edition is free to read on Wikidocs, a Korean site for online technical books. AI curator @lucas_flatwhite, who has more than 15,000 followers on X, recommended it without being asked, and the post passed 14,000 views and 290 bookmarks (x.com/lucas_flatwhite/status/2101481809922207913). The official Wikidocs account on X shared it next. In September 2026 the book reached #3 on the Wikidocs weekly ranking, with more than 12,000 views that week.

Three chapters that only make sense for Korean readers were left out of this edition. Both tools change quickly, so the book is revised as skills and tools change. Buyers receive new editions free for at least 12 months after purchase.

Screenshots of open-source repository READMEs are included as snapshots to help you recognise each project. They belong to their authors and are used under each repository's MIT or Apache-2.0 license, named in the chapter's Sources. People's avatars and other companies' logos in them are blurred.

Want to hear when new chapters and editions come out? Get the free 10-minute checklist for sharing one SKILL.md across both tools, and you will be on the update list: jaylim5.gumroad.com/l/skill-md-checklist

Reference versions: `Claude Code` v2.1.273 and `Codex` v0.153.4. Reference environment: macOS with a terminal. On Windows, use the same commands in WSL2.

About the author

Jay Jung Lim is the founder of BuildnWrite, a Seoul company that coaches teams to build their own AI agents. Jay teaches this work for a living and writes about it.

- Corporate AI training and agent coaching since 2023, for more than 20 organizations
- Coauthor of “n8n Does It All” (Hanbit Media, 2026), a Korean book on workflow automation
- Instructor of an 18-part Claude video course on Hanbit+, Hanbit Media’s course platform (hanbit.co.kr/product/C2240643131)
- Co-instructor on two free Inflearn courses with more than 1,300 learners combined, as of October 2026 (n8n course, reading challenge)
- 6,275 recorded Claude sessions and 1,048 Codex sessions while writing and building with agents
- Uses 70 skills in daily work, several of which appear in this book as author’s skills

LinkedIn: linkedin.com/in/jayjunglim

Company: buildnwrite.com/en

Copyright

© 2026 BuildnWrite. All rights reserved.

Skill files quoted in this book follow the license of each repository. Product names belong to their owners. This book is not affiliated with Anthropic or OpenAI.

00-1 How to read this book

This chapter explains the book's reading order, reference versions, notation, and scope. For each skill, this book first asks which situations it fits and which it does not.

This book was compiled by Jay Jung Lim (BuildnWrite), an AI agent coach.

- Corporate AI training and agent coaching (since 2023)
- Coauthor of "n8n Does It All" (Hanbit Media, 2026)
- Time using Claude: 14 months (since August 2025)
- Claude session records: 6,275 (since March 20, 2026)

Public skills on GitHub are usually written in English for developers. I selected skills that have attracted attention and explained them in plain language so people outside software, too, can choose and install skills that fit their work.

Where to start by reader type

To choose a skill for your work first, see 00-0 Skill combination roadmap by role. It pairs work stages with chapters for planning, design, development, marketing, operations, and research roles.

If you have not installed `Claude Code` or `Codex`, start with "Getting started with Claude Code and Codex." To learn the concepts of agents and skills first, read "What is an agent, and where do skills fit?" and "What are agent skills? Claude skills and Codex skills." These two chapters use illustrations to explain what agents and skills are and how they differ.



A large library has three entrances. A person with a notebook, one with a ruler and pencil, and one with a laptop enter through different doors and meet at the same shelves inside.

Figure 00-1-1 Different entrances for different readers, the same shelves

Type	Where to read next
A Professionals (unfamiliar with the terminal)	00-1-1, the Part for your field
B Analysis and design	00-4, Part 07, the Part for your field
C Development and operations	00-4 through 00-7, Part 07, Part 05

The foundations chapters have the following numbers and titles. Chapters 00-4 through 00-7 are references on the rules (specifications) the two products follow when reading skill files. You can skip them at first.

Number	Title
00-0	Skill combination roadmap by role
00-0-1	53 recommended Claude skills and Codex skills
00-1-1	Getting started with Claude Code and Codex
00-2	What is an agent, and where do skills fit?
00-3	What are agent skills? Claude skills and Codex skills
00-4, 00-5	How skills work in the two products
00-6, 00-7	Specification comparison, using one folder for both
00-8	Glossary

Look up unfamiliar words in “Glossary.” Paste the installation commands from skill pages into a terminal and run them. If you are new to the terminal, first choose a skill using the “Who it fits” table on its page. Install it after following “Getting started with Claude Code and Codex” to install the two products.

> Tip: Part 07, “Skills for working with agents,” collects skills that change how an agent works, regardless of field. Once you are comfortable with the terminal, I recommend reading it before the Parts for specific fields.

Reference versions

The first edition is based on `Claude Code` v2.1.273 and `Codex` v0.153.4. The concept chapters in the foundations Part were checked with `Claude Code` v2.1.274. Both products read `SKILL.md` using the agentskills.io open standard (a skill file specification shared by multiple products). When the specification changes, I update the comparison table first.

Notation

Item	Notation
Product names, file names	English inline code
Prompts	text code blocks, direct commands
Callouts	Three types: Watch out, Tip, Check

Item	Notation
Cross-references	See “Title”

Skill introduction pages are based on each repository’s documentation. I then add screenshots from actual runs in both products and accounts of trial and error.

What this book does not cover

“Getting started with Claude Code and Codex” covers installation and the first conversation. For detailed plan comparisons and installation troubleshooting, use the official documentation. This book also does not cover general prompt writing or hands-on marketplace installation exercises.

> Tip: If you have not installed the two products yet, start with “Getting started with Claude Code and Codex.”

Update commitment

I do not hide failures, and I attach URLs or file paths to factual claims.

What is included and excluded?

First, here are the criteria you can use to choose. This is not a collection of skills I created. It is a book in which I check skills that have attracted attention elsewhere in both products and explain them for people who do not write code every day.

Criterion	Details
First priority	Public skills that have attracted attention on GitHub
Star threshold	10,000 for development and data analysis; 2,000 for other fields
Updates	A commit within the last 90 days
License	External skills must use MIT or Apache-2.0
Author’s skills	A few frequently used skills in each Part. A license may not be stated
Excluded	Curated lists, unlicensed skills

Candidates are collected again automatically each week. Skills that pass these criteria may be added in future editions.

Which Part should you read first?

You can read only the Parts that interest you. The first page of each Part has a table listing its skills, one-line summaries, and main users. If you are building a service with vibe coding, start with Part 08.

Part	Skill count	What it covers
Part 08 Vibe coding skills	6	

Part	Skill count	What it covers
		Skills used from building a web service in natural language through launch
Part 01 Business skills	7	Skills for assigning business judgments and decisions to an agent
Part 02 Marketing skills	7	Skills for assigning recurring marketing work to an agent
Part 03 Data analysis skills	5	Skills for collecting and reading material and reporting results in data analysis work
Part 04 Design skills	7	Skills for assigning design outputs to an agent
Part 05 Development skills	6	Skills that make coding agents follow development procedures, plus n8n workflow skills
Part 06 Operations automation skills	7	Practical operations skills for email, scheduled tasks, folders, PDFs, and browsers
Part 07 Skills for working with agents	8	Skills that change how agents work, including planning, questions, memory, and wrapping up

How to read the ratings

The stars beside a skill's title reduce the GitHub star count of its repository to a three-star scale. For repositories containing multiple skills, this is the star count of the whole repository. At least 50,000 stars earns three, at least 10,000 earns two, and anything below earns one.

The rating indicates how much interest people have shown, not a quality guarantee. The author's skills without public repositories are not rated, so their titles have no stars.

Frequently asked questions

What does "How to read this book" cover? It explains the book's reading order, reference versions, notation, and scope.

Which versions is this book based on? The first edition is based on `Claude Code` v2.1.273 and `Codex` v0.153.4.

Does it cover installing Claude Code? "Getting started with Claude Code and Codex" covers installation, login, and the first conversation. For detailed plan comparisons and installation troubleshooting, use the official documentation.

Sources

- Open skill standard: https://agentskills.io/specification
- `Claude Code` skill documentation: https://code.claude.com/docs/en/skills
- `Codex` skill documentation: https://learn.chatgpt.com/docs/build-skills

Recap and the next chapter

The next chapter, “Getting started with Claude Code and Codex,” covers installing the two products and having your first conversation. If you already use them, move on to “What is an agent, and where do skills fit?”

I continue to publish hands-on test records and example files for this book’s skills on the blog: https://buildnwrite.com/

00-0-1 53 recommended Claude skills and Codex skills

This list brings together 53 skills you can use in both `Claude Code` and `Codex` because they follow the open Agent Skills specification.

Each skill has a one-line purpose and a list of suitable roles. Click a skill name to open its chapter, which covers installation, a first run, and things to watch out for. To choose skills in workflow order, also see 00-0 Skill combination roadmap by role.

53 skills in this book

The first two digits identify the Part: 01 Business, 02 Marketing, 03 Data analysis, 04 Design, 05 Development, 06 Operations automation, 07 Working with agents, and 08 Vibe coding. Part 08 (08-01 through 08-07) covers skills that help with design, interface design, code understanding and review, before launch when you build a service through vibe coding with Claude Code or Codex.

Number	Skill	One-line purpose	Suitable roles
01-02	pm-skills	A pack of product planning methods	Product planners, product managers
01-03	distilly	Turn a person's approach into a skill	Operations leads, product planners
01-04	wondelai/skills	A skill pack of book frameworks	Entrepreneurs, product planners, marketers
01-05	yc-office-hours	A business mentor who asks before you build	Solo founders, entrepreneurs, product planners
01-06	research	Turn research into a document with sources	Researchers, instructors, writers
01-07	flow-map	Accumulate flowcharts as dated snapshots	Solo founders, product planners, marketers
01-08	meeting-agenda	An agenda that requires options and deadlines	Entrepreneurs, product planners, operations leads
02-01	marketingskills	A collection of practical marketing skills	Marketers, entrepreneurs, solo founders
02-02	humanizer	Edit machine-like prose	Marketers, writers, solo founders
02-03	claude-ads	Help auditing ad accounts	Marketers, operations leads
02-04	blog-post	A production cycle for one blog post	Marketers, solo founders, writers
02-05	blog-topic-judge	Evaluate blog topics along five axes	Marketers, solo founders, writers
02-06	technical-writer	Writing steps and checks by document type	Writers, marketers, developers

Number	Skill	One-line purpose	Suitable roles
02-08	card-news	A card-news deck from one JSON file	Marketers, solo founders, product planners
03-01	last30days-skill	Summarize research on recent reactions	Marketers, product planners, product managers
03-02	nature-skills	Check figures and statistical reporting for submission	Researchers, data analysts
03-03	notebooklm-py	Control NotebookLM with commands	Researchers, data analysts
03-04	insane-search	Read blocked public pages again	Researchers, data analysts
03-05	apps-script-collector	Load market prices into a sheet automatically each day	Solo founders, operations leads
04-02	archify	Generate validated architecture diagram HTML	Product planners, developers
04-03	diagram-design	44 diagram types in brand colors	Product planners, marketers
04-04	huashu-design	Create design outputs in HTML	Designers, product planners
04-05	architecture-drawio-diagram	Create architecture diagrams with large logos using code	Developers, product planners, writers
04-06	lecture-slide-make	Generate lecture pptx files from JSON	Instructors, product planners, solo founders
04-07	lecture-slides	Create lecture pptx files from JSON	Instructors, product planners, solo founders
04-08	notebooklm-slides	Draw image decks from specifications	Instructors, product planners, solo founders
05-01	superpowers	A development procedure from design through merge	Developers, product managers
05-02	agent-skills	Procedures by development stage	Developers, product managers
05-03	n8n-skills	Knowledge for building n8n workflows	Developers, operations leads
05-04	n8n-route	Confirm the instance and enforce build rules	Developers, operations leads
05-05	n8n-workflow-builder	Workflow design and completion checks	Developers, operations leads, product planners
05-06	n8n-debugger	Trace failure boundaries and prevent recurrence	Operations leads, developers, solo founders
06-01	obsidian-skills	Support work in an Obsidian vault	Writers, operations leads
06-02	gmail-search	Find email in four directions	Operations leads, solo founders, product planners
06-03	email-draft	Generate Gmail drafts with your signature	Operations leads, solo founders
06-04	add-launchd-job		Developers, operations leads

Number	Skill	One-line purpose	Suitable roles
		Six steps to create a launchd job	
06-05	new-project	Check naming rules, then create a folder	Operations leads, developers
06-06	markdown-to-pdf	Convert Markdown to PDF	Operations leads, developers, writers
06-07	aside-browser	An entry point to Aside CLI instructions	Operations leads, product planners
07-01	i-have-adhd	Act first and number the steps	Developers, data analysts
07-02	grilling	Validate plans with questions	Developers, product planners
07-03	plan	Plan, cross-review, and wait for approval	Developers, product planners, product managers
07-04	planning-with-files	Work plans that survive interruptions	Developers, operations leads
07-05	session-wrap	Commit and hand off before ending a session	Developers, data analysts
07-06	skill-creator	Repeat skill writing and evaluation	Developers, product planners
07-07	skill-recorder	Turn recorded work into a skill	Operations leads, product planners
07-08	SkillSpector	Check skill security before installation	Developers, operations leads
08-01	gstack	A pack of development commands by role	Entrepreneurs, developers, product planners
08-02	taste-skill	Design rules that prevent generic interfaces	Developers, designers
08-04	ponytail	A pack of rules for minimizing code	Developers, solo founders
08-05	ui-ux-pro-max	Search industry-specific design rules	Developers, designers
08-06	Understand-Anything	Generate a codebase knowledge graph	Solo founders, product managers
08-07	open-code-review	Automatically review changed code	Developers, operations leads

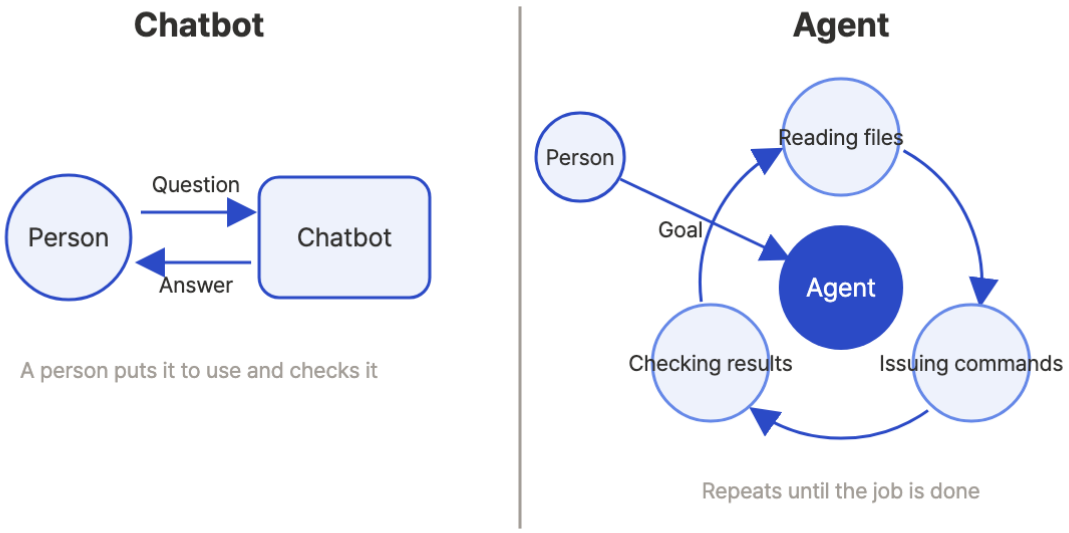
00-2 What is an agent, and where do skills fit?

An agent is a program that takes a goal and works through it on its own, repeatedly reading files, issuing commands to the computer, and checking results until the job is done. A skill is a work manual the agent opens when it needs one during a task. This chapter uses illustrations to explain their relationship and how skills differ from the other five things that accompany an agent.

How is an agent different from a chatbot?

A chatbot takes one question and gives one answer. A person copies the code or text from that answer, puts it to use, and checks whether it worked.

An agent handles those follow-up steps too. It reads files directly from a folder, issues commands to create files or run programs, and fixes and reruns them if the results differ from what it expected. `Claude Code` and `Codex` are both agents of this kind. They work inside a working folder on your computer.



On the left, a person and a chatbot exchange one question and answer. On the right, an agent repeats reading, execution, and checking in a circular loop.

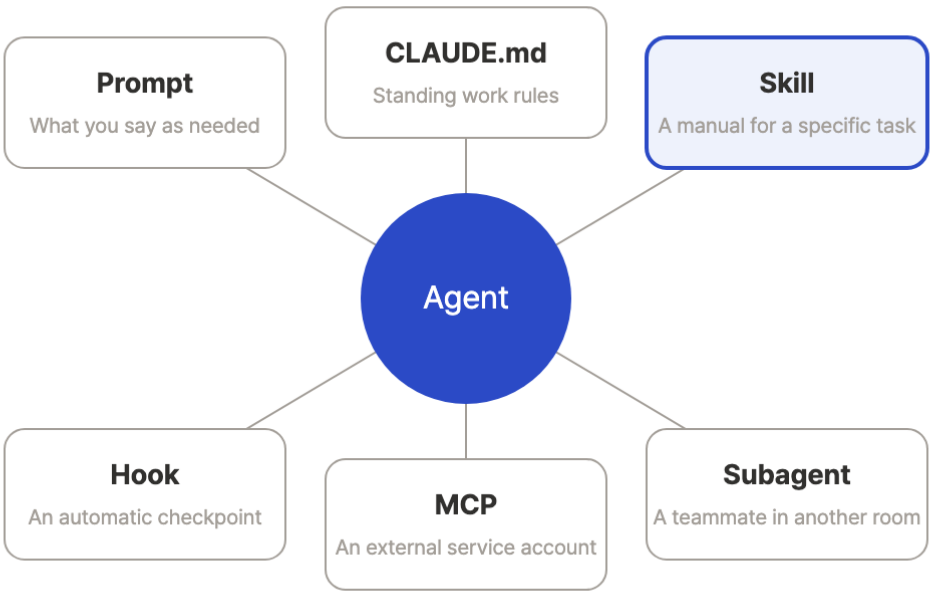
Figure 00-2-1 A chatbot answers once; an agent repeats until the result is verified

This book compares an agent to **a new colleague**. The colleague is smart but does not know how your company works. Anthropic's official documentation also describes skill folders as being organized like onboarding materials for a new team member (Source 1).

Six things that accompany an agent

Verbal instructions alone are not enough for a new colleague to do well on day one. They need standing rules, manuals for specific tasks, accounts for external services, and teammates to share the work. The same is true of an agent.

Name	New-colleague analogy	Purpose
Prompt	What you say as needed	A request for this one task
`CLAUDE.md`	Standing work rules	Read at the start of every session
Skill	A manual for a specific task	Opened only when needed
Subagent	A teammate in another room	Works separately and reports only a summary
MCP	An external service account	Connects services such as Notion
Hook	An automatic checkpoint	Runs unconditionally at a specified moment



A map with an agent at the center, surrounded by six things: prompts, work rules, skills, subagents, MCP, and hooks.

Figure 00-2-2 Six things that accompany an agent

A session is one continuous conversation from the time you start an agent until you close it. MCP (Model Context Protocol) is a connection protocol that attaches external services, such as Notion or a browser, as tools the agent can use. Here, a tool means a single capability the agent can execute directly, such as reading a file.

In `Codex`, the work rules file is named `AGENTS.md`. The other five exist in both products and have the same names (Sources 5, 6, 7).

When do they step in, and who calls them?

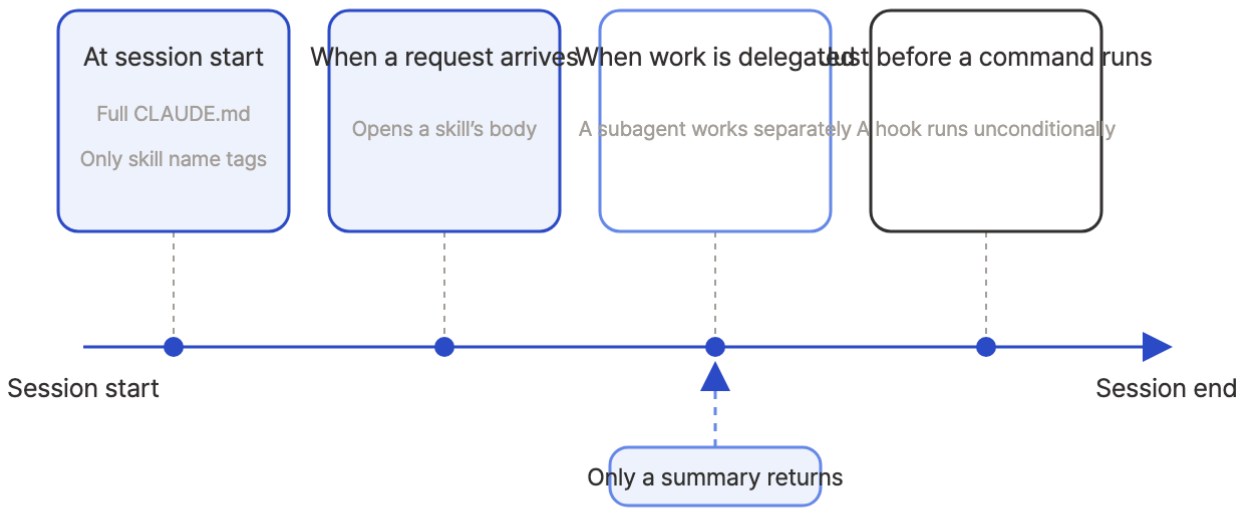
Instead of memorizing all six, distinguish them by when they step in and who calls them. I organize the five other than the human-entered prompt into three groups (Source 9, the author's private notes).

Group	Includes	Who calls it?
Entered by a person as needed	Prompt	When a person types it
Always present	`CLAUDE.md`	Automatically at session start
Chosen by the agent	Skills, subagents, MCP	The agent or a person
Runs unconditionally	Hooks	The program at a specified moment

The first group, work rules, is read at the start of every session. It always takes up space in the context window, the workspace an agent can hold in memory at one time, so keep it short. The official documentation recommends no more than 200 lines per file (Source 3).

The second group is called when the agent looks at a request and decides it is needed now. You can also invoke it by naming it directly. At session start, only a skill's name and one-line description are read. This book calls them its "name tag." The body is opened when the skill is called.

The third group, hooks, runs at specified moments regardless of the agent's judgment. The official documentation explains that work rules provide reference context rather than enforced settings. It advises using hooks to block actions that must be prevented (Source 3).



A session timeline showing work rules and skill name tags being read at the start, a skill's body opening when a request arrives, a subagent running separately when work is delegated, and a hook stepping in just before a command runs.

Figure 00-2-3 When the six things step in during a session

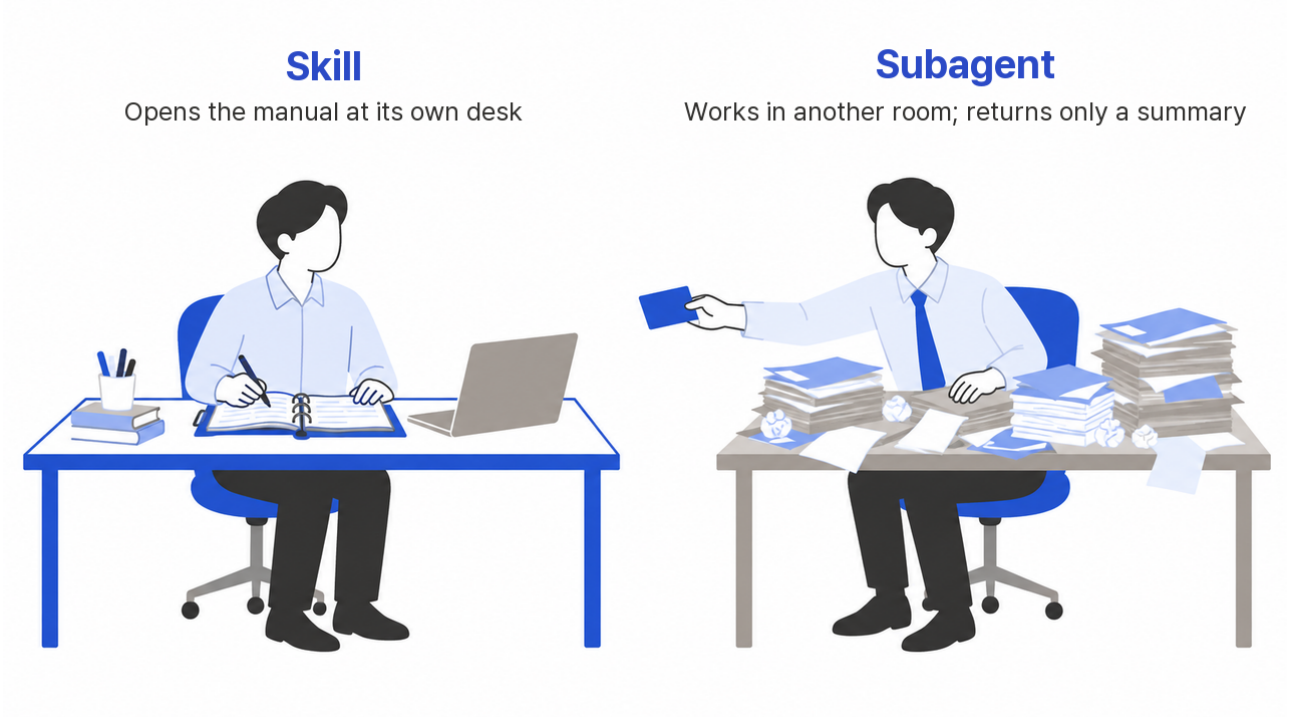
How is a skill different from a subagent?

These two are easy to confuse. The agent decides when to call either one, and each is created with a single configuration file. The difference is **whose desk the work happens at**.

With a skill, the agent you are talking to opens the manual at its own desk and does the work itself. It therefore works with full knowledge of the conversation so far.

A subagent is a teammate working in another room. It cannot hear your current conversation. It receives and returns work only through notes. According to the official documentation, a subagent works in its own context window, cannot see the original conversation history, and returns only a summary when it finishes (Source 2).

Criterion	Skill	Subagent
Who does the work?	The agent you are talking to	A separately launched agent
Conversation history	Sees it as is	Cannot see it
What remains?	The full process in the conversation	Only a summary in the conversation
When to choose it	When you need to follow the process	When you need only a summary of the result



On the left, a person opens a manual and works at their own desk. On the right, a teammate works through a stack of documents in another room, then hands over a single summary note.

Figure 00-2-4 A skill works at your desk; a subagent works in another room

For example, writing a weekly report in the same format calls for a skill because you need to watch and adjust the process. Searching dozens of web documents and extracting only the key points calls for a subagent because the intermediate steps clutter the conversation.

Questions to help you choose

At first, it is enough to find your situation on the left and choose the option on the right.

If this is your situation	Use this
One or two lines are always needed for every task	`CLAUDE.md`
You have an occasional procedure with multiple steps	Skill
A long investigation clutters the conversation	Subagent
You need an external service such as Notion or a browser	MCP
You have a check that must not be forgotten	Hook
It is just for this one time	Prompt

> Tip: A skill is a “recommendation” to the agent. It does not run unless the agent decides to call it or a person invokes it by name. Move checks whose omission could cause an incident into hooks.

Where do they go in each product?

The locations and file formats differ slightly, but the roles are the same. `Claude Code` subagents are created with Markdown files, while `Codex` subagents use TOML configuration files (Source 7). Detailed specifications are covered in “How to use Claude Code skills and how they work” and “How to use Codex skills and how they work.”

Role	Claude Code	Codex
Work rules	`CLAUDE.md`	`AGENTS.md`
Skills	`.claude/skills/`	`.agents/skills/`
Subagents	`.claude/agents/`	`.codex/agents/` (TOML)

Both products connect MCP through registration commands in the terminal (`claude mcp`, `codex mcp`). Both support hooks, but the types of moments at which hooks can step in differ. According to the official documentation as of 2026-09-17, `Claude Code` has more than 30, and `Codex` has 12 (Sources 8, 10).

What went wrong?

I once assigned file edits to a subagent and received a report that they were done, even though none of the actual files had changed (recorded in the author’s private notes on 2026-06-13). Because a subagent returns only a summary, you cannot see the process.

So I set up a hook to count the files that actually changed whenever a subagent finishes, using the `git status` command in `git`, a tool that tracks file changes. If the count is zero, it alerts the agent. A checkpoint can operate after the work ends too. I assigned judgment to the agent and verification to the hook.

Conversely, for skills containing actions that are difficult to undo, such as uploading work to an online repository, I blocked the agent from invoking them on its own. They run only when a person calls them by name. In `Claude Code`, you do this by adding `disable-model-invocation: true` to the configuration section at the top of the skill file (Source 4).

Recap and the next chapter

- An agent is a program that repeats reading, execution, and checking on its own. A skill is a work manual the agent opens when it needs it.
- The five things other than prompts fall into three groups: those that are always present, those the agent chooses to call, and those that run unconditionally.
- With a skill, work happens at your desk, where you can follow the process. A subagent works in another room and returns only a summary.

“What are agent skills? Claude skills and Codex skills” examines what one of these manuals actually looks like and when it opens.

Related blog post: Five AI automation terms: MCP, agents, context, skills, and secrets (in Korean)

Frequently asked questions

How is a skill different from an agent? The agent does the work. The skill is a work manual the agent reads when it needs it. A skill does nothing on its own; it takes an agent reading it to put it into action.

How do I choose between a skill and a subagent? Use a skill if you need to follow the process in the conversation. Use a subagent if you need only a summary of the result. A subagent cannot see the original conversation history.

Why not put everything in CLAUDE.md? Work rules are read in full every session and always occupy the context window. The official documentation advises moving occasional procedures with multiple steps into skills, which are read only when needed (Source 3).

Sources

1. https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview
2. https://code.claude.com/docs/en/sub-agents
3. https://code.claude.com/docs/en/memory
4. https://code.claude.com/docs/en/skills
5. https://learn.chatgpt.com/docs/build-skills
6. https://learn.chatgpt.com/codex/agent-configuration/agents-md
7. https://learn.chatgpt.com/codex/agent-configuration/subagents
8. https://code.claude.com/docs/en/hooks
9. Author’s private notes, “Claude Code: Five elements, three trigger mechanisms,” 2026-06-13
10. https://learn.chatgpt.com/docs/hooks

02-02 ★★☆ humanizer: Editing out machine-like prose

humanizer rewrites text that sounds machine-written so it reads like human writing, while preserving its meaning. It fits marketers and writers who want to polish launch announcements or newsletters in their own voice.

Quick link: github.com/blader/humanizer

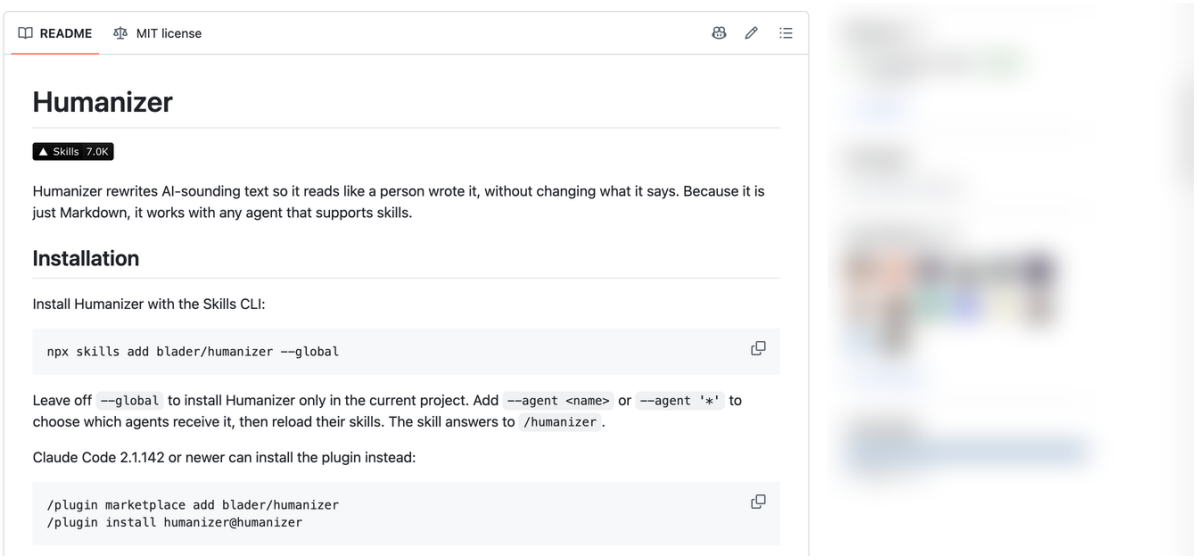
At a glance

Item	Details
One-line summary	Edit machine-like prose
Form	Single skill
Rating	★★☆ (49,444 stars)
Main users	Marketers, writers, solo founders
Main outputs	Edited text, pattern list

Who it fits

Situation	Fit	Reason
Editing a launch announcement	Good fit	Checks exaggerated claims
Adjusting a newsletter's voice	Good fit	Follows a writing sample
Editing a manuscript file	Okay	Edits prose only
Adding examples that do not exist	Poor fit	Does not invent facts

Install


README MIT license

Humanizer

Skills 7.0K

Humanizer rewrites AI-sounding text so it reads like a person wrote it, without changing what it says. Because it is just Markdown, it works with any agent that supports skills.

Installation

Install Humanizer with the Skills CLI:

```
npx skills add blader/humanizer --global
```

Leave off `--global` to install Humanizer only in the current project. Add `--agent <name>` or `--agent '*'` to choose which agents receive it, then reload their skills. The skill answers to `/humanizer`.

Claude Code 2.1.142 or newer can install the plugin instead:

```
/plugin marketplace add blader/humanizer
/plugin install humanizer@humanizer
```

The opening screen of the repository README, showing the title, introduction, and installation commands.

Figure 02-02-1 The opening screen of the humanizer repository README

```
# Install globally with Skills CLI
npx skills add blader/humanizer --global
# Install the Claude Code plugin
/plugin marketplace add blader/humanizer
/plugin install humanizer@humanizer
# Install globally for Codex
npx skills add blader/humanizer --global --agent codex
```

First run

Paste your text, make a request in plain language, or provide a file path and ask for an edit. The README says that two or three paragraphs of your own writing let it follow your sentence length, word choices, and even habitual punctuation such as dashes.

```
[text to edit]
[a sample of my writing style]
```

SKILL.md describes the following sequence.

- **Flag patterns.** Find 26 machine-like writing patterns in order of strength and frequency, and identify what needs editing.
- **Rewrite the draft.** Create a new draft that preserves the meaning without being bound to the structure. When you paste text directly, it shows the first draft and any remaining problems. It reduces monotonous passages and merges or splits paragraphs while keeping the information intact.
- **Check facts.** Check for facts that were dropped or added before producing the final version. It preserves the writer's opinions and personality in personal writing and keeps a neutral tone in

technical documents. During this process, it reads the text aloud to check for passages that still sound machine-written.

What makes it useful

- **Organizes patterns.** It covers six groups of patterns based on a related Wikipedia article. It shows before-and-after examples for each pattern in tables and separately marks patterns that are weak signals when used alone.
- **Does not invent facts.** Its instructions say to ask for missing names, numbers, dates, or quotes rather than fill them in. The example travel essay also omits places and times absent from the writer's notes. This rule applies to factual claims rather than opinions.
- **Preserves file structure.** When given a file path, it edits only prose and leaves code, data, frontmatter, and link URLs untouched. Even when called within another task, it returns only the final text.

What's included

Name	Purpose
`SKILL.md`	Patterns and workflow
`/humanizer`	CLI invocation command
`.claude-plugin/`	Plugin configuration
`agents/openai.yaml`	Display settings
`scripts/`	Validation scripts

Watch out for

- **Explanations focus on English.** The pattern descriptions and examples in the README and SKILL.md are written in English.
- **Invocation names depend on installation.** Plugin installation requires Claude Code 2.1.142 or later, and the invocation command changes to `/humanizer:humanizer`. For earlier versions, install with Skills CLI. For Claude Desktop, download the repository as a ZIP and upload it as a skill.
- **Fact-checking still matters.** The skill is instructed to ask for missing information, so it helps to provide supporting material. WikiProject AI Cleanup maintains the Wikipedia article behind the pattern list.

Frequently asked questions

What is humanizer?

humanizer is a skill that rewrites machine-sounding text to read like human writing while preserving its meaning. It identifies what needs editing using 26 patterns ordered by strength and frequency.

Can it edit files too?

Yes. The README says that when given a file path, it edits only prose and leaves code, data, frontmatter, and links untouched.

Can I use it in Codex?

Yes. The README provides separate instructions for installing it in Codex with `npx skills add blader/humanizer --global --agent codex`.

Sources

- Repository: https://github.com/blader/humanizer
- License: MIT, Copyright (c) 2025 Siqi Chen

05-01 ★★★ How to use superpowers: Development methodology skill pack

superpowers bundles design, planning, testing, and review into skills that make a coding agent follow a development procedure. It fits developers adding features to an existing service or assigning a long implementation to an agent.

Quick link: github.com/obra/superpowers

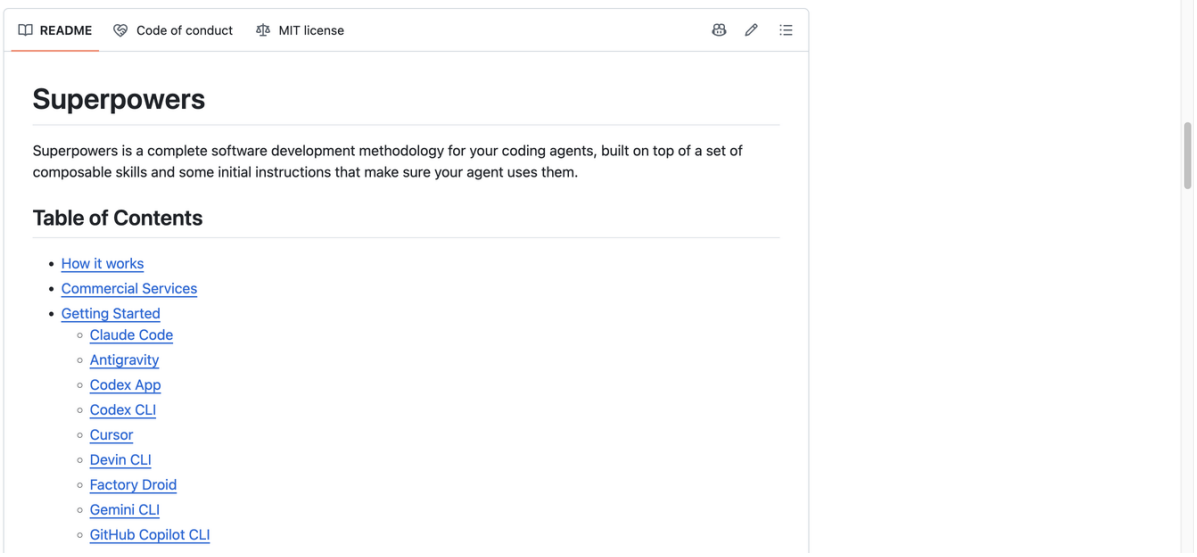
At a glance

Item	Details
One-line summary	Development procedures from design to merge
Form	Plugin (15 skills)
Rating	★★★ (287,921 stars)
Main users	Developers, product managers
Main outputs	Design documents, implementation plans, tests

Who it fits

Situation	Fit	Reason
Adding features to an existing service	Good fit	Design, planning, then TDD
Delegating a long implementation to an agent	Good fit	Subagent review for each task
Reviews in a team with little testing practice	Okay	Takes time to adjust to enforced procedures
A one-line fix or quick prototype	Poor fit	Long questioning and approval stages

Install



The opening screen of the repository README, showing the title, introduction, and installation commands.

Figure 05-01-1 The opening screen of the superpowers repository README

```
# Claude Code: Install from the official marketplace
/plugin install superpowers@claude-plugins-official
# Codex CLI: Open search, find superpowers, and select Install Plugin
/plugins
```

First run

You do not need to invoke a command. After installing, request the feature you want as usual. The agent checks for relevant skills before starting and activates them.

```
Add [the feature I want to build]
```

The README describes the following sequence.

- **Questions come first.** Before writing code, `brainstorming` asks about your goal and briefly summarizes its understanding so you can correct it. It then presents the design in sections. For a large task such as a new project, it saves a design document and asks for approval.
- **Approval prepares the workspace and plan.** After creating a workspace for a new branch, `writing-plans` breaks the work into two- to five-minute tasks and lists file paths and verification steps.
- **Choose how to execute, then implementation begins.** Choose between assigning a fresh subagent to each task and reviewing tasks individually, or a lighter approach that implements everything in one session and reviews once at the end.

What makes it useful

- **No code before design approval.** The `brainstorming` SKILL.md explicitly prohibits calling implementation skills before approval, preventing code from piling up while requirements remain unclear.
- **Tests come first.** `test-driven-development` enforces the sequence of writing a failing test and making it pass. The README says code written before the test is deleted.
- **Long tasks stay on plan.** The README says it is not unusual for agents to work on their own for a couple of hours without straying from the plan they created.

What's included

These are the skills from the README's default workflow, selected from the 15 skills and listed in order.

Skill	Purpose
`brainstorming`	Clarify the design through questions and save a document
`using-git-worktrees`	Prepare a workspace for a new branch
`writing-plans`	Plan implementation in small tasks
`test-driven-development`	Write a failing test first, then minimal implementation
`requesting-code-review`	Review against the plan and report severity
`finishing-a-development-branch`	Check tests and choose a merge approach

Watch out for

- **Install separately for each tool.** The README says to install it in each agent if you use several. In the Codex app, use the Plugins menu in the sidebar.
- **Version information is sent by default.** The optional visual helper in `brainstorming` fetches a logo from the maker's website and sends version information. You can disable this with the `SUPERPOWERS_DISABLE_TELEMETRY` environment variable.
- **The procedure may feel heavy.** The README defines skills as a required workflow rather than suggestions. Check whether this conflicts with your current way of working.

Frequently asked questions

What is superpowers?

It is a plugin that bundles design, planning, testing, and review into skills that make coding agents follow a development procedure. It contains 15 skills and is installed separately in Claude Code, Codex, and other tools.

Do I need to memorize commands?

No. The README says the agent checks for relevant skills before starting and activates the appropriate ones automatically.

Can I use it in Codex?

Yes. In Codex CLI, open search with `/plugins` and install it. In the Codex app, use the Plugins menu in the sidebar.

Sources

- Repository: https://github.com/obra/superpowers
- License: MIT, Copyright (c) 2025 Jesse Vincent

End of the free sample

You have read the front matter, the reading guide, the full list of 53 skills, one foundation chapter, and two of the skill chapters.

The full book adds 51 more skill chapters across business, marketing, data analysis, design, development, operations, agent workflows, and vibe coding, plus the foundation chapters on how `Claude Code` and `Codex` find, load, and invoke skills, and how to share one skill folder between them.

Get the full book (PDF and EPUB, about 280 pages):

- The book ($29): jaylim5.gumroad.com/l/claude-code-codex-skills
- Book + the author's Skill Pack ($59): jaylim5.gumroad.com/l/guidebook-skill-pack
- All packages: buildnwrite.com/en/guidebook/#packages

Not ready to buy? Get the free 10-minute checklist for sharing one SKILL.md across both tools and join the update list: jaylim5.gumroad.com/l/skill-md-checklist

© 2026 BuildnWrite. All rights reserved.