FREE GUIDE · 2026

BEGINNER'S GUIDE TO AI SKILLS

How to Build *AI Skills* That Actually Work

A practical, no-fluff guide to creating custom AI skills using Claude and Codex. No experience needed.

The Problem It Solves

If you have ever typed the same prompt more than once, "write this email in my tone," "summarise this meeting," "turn my voice note into a post," you were doing the work a skill should be doing for you.

An AI skill is a reusable set of instructions you give an AI so it performs a specific task **consistently, every time,** without you re-explaining yourself. Build it once. Use it forever.

Without a Skill vs With a Skill

WITHOUT A SKILL	WITH A SKILL
You rewrite the prompt every time	One input. Consistent output. Done.
Different results every session	Same quality output every single time
Only you know how to run it	Your whole team can use it
Knowledge lives in your head	Knowledge lives in a file, forever
Each task takes manual effort	Tasks run automatically in workflows

Why Skills Are the Building Blocks of AI

Skills are to AI agents what functions are to code. Build small, focused skills, then chain them into workflows that run your business automatically. A meeting ends, a skill summarises it, another skill drafts the follow-up email, another creates the proposal. All without touching a keyboard.

Every automated AI system you have seen, a sales coach, a client brief agent, a podcast summariser, is just a collection of well-written skills talking to each other.

Three Fields. Every Skill.

Every skill, whether you are building it in Claude, storing it in a file, or sharing it with your team, requires exactly three components. Get these right and your skill is usable, readable, and improvable.

Skill Structure — REQUIRED FIELDS

Name `string · short`	A short, clear label for what this skill does. Like a job title for a task. Specific enough that anyone reading it immediately knows its purpose without opening the file. `skill-linkedin-post skill-meeting-summary skill-client-prep-brief`
Description `string · 1 to 3 lines`	When to use this skill and what it produces. Write it as if telling a new team member which tool to reach for. Answer: what goes in, and what comes out? `Use when given a rough voice note transcript. Produces a LinkedIn post in the founder's tone. Output: 150 to 250 words, hook plus body plus CTA.`
Instructions `string · full prompt`	The actual system prompt. Everything Claude needs to do the job correctly: the AI's role, the task, the exact output format, and what to never do. This is where the real work lives. `You are a ghostwriter for a NZ entrepreneur... [Role, Task, Format, Rules in full]`

■ **Save skills as .md files.** Name the file after the skill, e.g. `skill-meeting-summary.md`, then use three headings: Name, Description, Instructions. Simple to share. Structured enough to build with.

What a Saved Skill Looks Like

SKILL-MEETING-SUMMARY.MD

```
---
name: skill-meeting-summary
description: >
  Use when given a meeting transcript or rough notes.
  Produces a structured summary with decisions, actions
  and open questions. Readable in under 2 minutes.
---

# Instructions
You are a professional meeting summariser for a NZ AI
consultancy. Output exactly these four sections:
  1. WHAT WAS DECIDED  (bullet points, max 5)
  2. ACTION ITEMS      (who does what, by when)
  3. OPEN QUESTIONS    (unresolved, needs follow-up)
  4. ONE-LINE SUMMARY  (what this meeting was about)
FORMAT: Bullets. Under 20 words each. Active voice.
NEVER: exceed 300 words · add opinions · summarise
things not actually decided
```

The 4-Part Instructions Formula

Every great set of instructions follows the same four-part structure, whether you are writing for Claude, ChatGPT, or Codex.

1. **Role: Who the AI should be**
   Give the AI a specific job title. "You are an expert NZ business consultant" vastly outperforms "you are a helpful assistant."

2. **Task: What it needs to do**
   Be specific. "Write a follow-up email after a sales call, max 150 words" beats "write a follow-up email." Precision = consistency.

3. **Format: How the output should look**
   Do not describe the format. Show it. Paste the actual template into the instructions. If you do not specify, the AI will guess.

4. **Rules: What NOT to do**
   A NEVER section is your most powerful tool. "Never exceed 200 words." "Never add a disclaimer." These constraints make a skill reliable.

Think of it like hiring someone. You would write a job description, explain the task, show the format, and tell them what to avoid. Writing a skill is exactly that.

If you find yourself rewriting the same prompt more than *3 times,* it does not need editing. It needs to be a skill.

Claude Is the Best Tool for Business Skills

Claude is exceptional at following complex instructions, maintaining tone, and producing consistent outputs. Here is how to build skills with it, no code required to start.

Method 1: Claude Projects (No Code)

Claude Projects lets you give Claude a permanent set of instructions that apply to every conversation in that project. That instruction is your skill.

1. **Go to claude.ai and create a Project**
   Name it after the skill, e.g. "Meeting Summariser" or "LinkedIn Post Writer."
2. **Paste your Instructions into Project Instructions**
   This runs automatically on every conversation in that project.
3. **Test with real, messy input**
   Use the worst real example you have: a rambling voice note, an incomplete transcript, or a rough email thread.
4. **Save the Instructions as a .md file**
   Back it up using the Name / Description / Instructions format. Now it is shareable and improvable.

Automate with the Claude API

When you want a skill to fire automatically, triggered by a webhook, form, or voice note, you use the Claude API. The `system` field is your permanent skill Instructions. The `messages` field is what changes each call.

JAVASCRIPT · CLAUDE API SKILL CALL

```
const response = await fetch(
  'https://api.anthropic.com/v1/messages', {
    method: 'POST',
    headers: { 'x-api-key': process.env.ANTHROPIC_API_KEY },
    body: JSON.stringify({
      model: 'claude-sonnet-4-20250514',
      max_tokens: 1000,
      system: `[paste your skill Instructions field here]`,
      messages: [{ role: 'user', content: inputText }]
    })
  }
);
const output = (await response.json()).content[0].text;
```

A Live Skill: The Client Prep Brief

This is the actual skill running inside the MissAI pipeline. The moment someone books on Calendly it fires automatically, scrapes their website, generates a 9-section prep brief, and sends it to Telegram within 60 seconds.

SKILL-CLIENT-PREP-BRIEF · INSTRUCTIONS

```
name: skill-client-prep-brief
description: Returns a 9-section prep brief on Calendly booking.

# Instructions
You are an AI business consultant preparing Keira
for a discovery call. Brief must be readable in 3 min.

Output exactly these 9 sections:
1. WHO I'M MEETING     6. BUYING SIGNALS
2. COMPANY DEEP DIVE   7. OBJECTIONS + REFRAMES
3. LIKELY PAIN POINTS  8. RECOMMENDED TIER
4. WHERE AI CAN HELP   9. TELEGRAM SUMMARY
5. DISCOVERY QUESTIONS

NEVER: use jargon · pad sections · write generically
```

Key rule: `system` = your permanent skill. `messages` = what changes each call. Keep them separate and your skill works on any input, forever.

Same Format, Different Tools

Codex (OpenAI / GPT-4o) uses the exact same Name, Description, Instructions format. The API call is nearly identical: just a different URL and model name. Here is when to use each.

USE CASE	BEST TOOL	REASON
Writing code, debugging, refactoring	**Codex / GPT-4o**	Trained heavily on code; GitHub Copilot integration
Business writing, client comms, prep briefs	**Claude**	Better nuance, tone, and long-form reasoning
Following complex multi-step instructions	**Claude**	More reliable with detailed system prompts
Agents needing code execution or file search	**Codex / Assistants**	Built-in tools: code interpreter, file search
Real-time coaching, voice note routing	**Claude**	Better contextual understanding
Shareable custom GPT with a no-code UI	**GPT Builder**	Easy to build and share without code

The only Codex difference: swap the URL to `api.openai.com/v1/chat/completions` and the model to `gpt-4o`. The skill format, system prompt, and messages structure are identical.

What Separates Skills That Work From Ones That Don't

Be embarrassingly specific — RULE 01

VAGUE: BREAKS OFTEN

"Write a professional email for me."

SPECIFIC: WORKS EVERY TIME

"Write a follow-up email after a sales call. Max 150 words. Start with one thing you appreciated. Include one next step with a date. Warm but businesslike."

Give it a clear persona — RULE 02

NO PERSONA

"Summarise this podcast episode."

WITH PERSONA

"You are a sharp podcast producer for the MissAI Podcast. Write summaries that make people want to listen. Punchy, energetic, never corporate."

Show the format, do not describe it — RULE 03

DESCRIBING FORMAT: OFTEN IGNORED

"Use a structured format with headers and bullets."

SHOWING THE FORMAT: ALWAYS FOLLOWED

Paste the actual template into your Instructions: `## WHAT WAS DECIDED - [bullet] ## ACTION ITEMS - [who] will [what] by [when]`

Use negative rules: your secret weapon — RULE 04

Add a NEVER section to every skill. AI is trained to be helpful, meaning it adds content you did not ask for. Negative rules stop that.

- → "Never start a sentence with I"
- → "Never use the words delve, leverage, or journey"
- → "Never add a disclaimer" · "Never pad the response"

Separate the skill from the content

RULE 05

MIXED TOGETHER

"Summarise this transcript: [1000 words] and always use bullet points."

CORRECTLY SEPARATED

System = "Always use bullet points." | User = "Summarise: [transcript]"

Test with real, messy input

RULE 06

Skills look great with clean test inputs. Real business content is messy. Test with the worst input you would actually receive. If it still works, ship it.

■ **Those are the 6 rules.** Apply all 6 to every skill you build and your output quality will be consistent, professional, and reliable from day one.

Ready-to-Use Skills for Your Business

Each skill is formatted with Name, Description, and Instructions. Save as a .md file or paste directly into a Claude Project.

SKILL 01

Voice Note to LinkedIn Post

SKILL-LINKEDIN-POST.MD

```
name: skill-linkedin-post
description: Given a rough voice note transcript, produces
  a ready-to-post LinkedIn caption. 150 to 250 words,
  hook plus body plus CTA. Direct tone, no corporate language.

# Instructions
You are a ghostwriter for a NZ entrepreneur posting about
AI, business, and real talk on LinkedIn.

FORMAT: Hook line (no I to start) · 3 to 5 short
paragraphs · one CTA at the end · no hashtags unless asked

NEVER: use "delve", "leverage", "foster", "journey",
"game-changer" · exceed 250 words
```

SKILL 02

Cold Email Personaliser

SKILL-COLD-EMAIL.MD

```
name: skill-cold-email
description: Given company name, website info, and service
  offering. Returns a personalised cold email under 100
  words plus a subject line under 7 words.

# Instructions
You are a B2B sales expert writing cold emails that get
replies because they feel like genuine human research.

STRUCTURE: Line 1: specific observation · Line 2: why
that connects to [service] · Line 3: one proof point
· CTA: one soft ask (15-min call or a question)

NEVER: "I hope this email finds you well" · exceed 100
words · use generic observations
```

SKILL 03

Proposal First Draft

SKILL-PROPOSAL-DRAFT.MD

```
name: skill-proposal-draft
description: Given meeting notes or call transcript.
  Returns a first-draft proposal under 400 words.

# Instructions
You are a proposal writer for an AI consultancy.

OUTPUT: 1. SITUATION (their words, 2 to 3 lines) ·
2. SOLUTION (specific) · 3. DELIVERABLES (bullets) ·
4. INVESTMENT [Discovery $750-1,500 | Agent $3-8k |
Retainer $800-2k/mo | Full OS $12-25k] · 5. NEXT STEP

NEVER: use "AI strategy" as a deliverable · exceed 400
words · recommend a tier without explaining why
```

SKILL 04

Podcast Show Notes

SKILL-PODCAST-SHOW-NOTES.MD

```
name: skill-podcast-show-notes
description: Given a transcript or rough episode notes.
  Produces title, hook, 4 to 6 specific bullet points,
  estimated timestamps, and guest bio (2 lines max).

# Instructions
You are the show notes writer for the MissAI Podcast.
Write for someone scrolling Spotify deciding to play.

OUTPUT: Title (max 8 words) · One-line hook · What We
Cover (4 to 6 specific bullets) · Timestamps · Guest
bio (2 lines max if applicable)

NEVER: start with "In this episode..." · use vague
bullets like "tips on AI" · be specific always
```

SKILL 05

Meeting Agenda Builder

SKILL-MEETING-AGENDA.MD

```
name: skill-meeting-agenda
description: Given meeting purpose, attendees, duration.
  Returns a tight timed agenda with owners and decision
  types. Flags if it should be an email instead.

# Instructions
You are an EA who builds focused agendas. If it runs
over time, that is a bad agenda, not a complex topic.

FORMAT:
## [Title] [Date] [Duration]
OUTCOME: one line on what "done" looks like
| Time | Item | Owner | Type: Inform/Discuss/Decide |
PARKING LOT: items to take offline

NEVER: exceed meeting duration · omit an owner · omit
type · forget the parking lot section
```

■ **These five skills are your starting library.** Save them as .md files in a /skills folder. Each one you build from here adds to that library. In six months you will have an arsenal that runs your business on autopilot.

A Skill You Cannot Find Is a Skill You Will Rebuild

- **Markdown files (recommended).** One .md file per skill in a /skills folder. Name, Description, Instructions. Version in GitHub so you can track what changed and roll back.
- **Claude Projects.** Go to claude.ai, create a Project, add instructions. Paste the Instructions field directly into the Project Instructions box. Each project becomes its own specialised assistant.
- **Notion database.** Table with columns: Skill Name, Tool, Description, Instructions, Last Tested, Status. Filter by status to find active skills instantly.

Skill Review Checklist

Run through this every time you build or update a skill:

- Has a Name, Description, and Instructions (all three fields)?
- Instructions include a clear role and persona?
- Task is specific enough that anyone would produce the same output?
- Format is shown in the instructions, not just described?
- Has a NEVER section with at least 3 negative rules?
- Tested with real, messy input, not a clean example?
- Saved somewhere findable with a clear name?

What Goes Wrong and How to Fix It

Most skills fail for the same five reasons. Learn to spot them before they waste your time.

Mistake 1: Too vague. "Be helpful and professional" tells the AI nothing. Every skill needs specifics: word counts, structure, tone examples, and what to avoid. Vague instructions produce vague outputs.

Mistake 2: Skipping the NEVER rules. AI is trained to be helpful, meaning it adds content you did not ask for. Negative constraints are the fix. Add them to every skill, without exception.

Mistake 3: Testing with ideal input. Real business input is messy. Test with incomplete data, rambling voice notes, or badly formatted text. If the skill breaks there, fix it before you rely on it.

Mistake 4: Not saving the skill. You will spend 20 minutes perfecting a prompt and never find it again. Save every skill. Give it a proper name. Treat it like code.

Mistake 5: Doing too much in one skill. A skill that does 10 things does none of them well. Build small, focused skills and chain them together for complex workflows.

■ **The fix for every mistake above is the same:** go back to the three-field format. Does it have a Name? Does the Description say exactly what goes in and what comes out? Do the Instructions have Role, Task, Format, and Rules? If all four are there, the skill will work.

Build it once. ***Let it run forever.***

MISS AI · EDUCATION SERIES

You built it. Now make it work for you.

MissAI runs workshops, courses, and consulting for businesses ready to build with AI. From your first skill to your first full agent.

realmissai.com
aiconsultancy.co.nz