Week 4: Agent Skills - Teaching Claude New Capabilities

01. Session Goals

- Understand the Agent Skills specification
- Create effective SKILL.md files
- Build skills for data analytics and GTM use cases
- Learn progressive disclosure and skill organization

02. Block 1: Theory - What Are Agent Skills? (30 min)

The Problem Skills Solve
Claude is powerful but generic. Skills let you:

- Teach Claude your specific workflows
- Encode team knowledge and standards
- Create reusable capabilities
- Ensure consistent behavior

Why Skills Matter: SOPs for Your Agent
Think of skills as SOPs (Standard Operating Procedures) for Claude.

In any organization, you don't train every new hire by having them shadow you forever. You write down the process: "Here's how we score leads. Here's our data quality checklist. Here's the format for weekly reports." Then they follow the SOP.

Skills work the same way. You have knowledge Claude doesn't have:

- How your team scores leads
- What makes a good data quality check at your company
- The specific steps for your weekly reporting workflow
- Your industry's benchmarks and red flags

Skills let you write that expertise down once and have Claude use it automatically, forever.

Real examples:

Your Expertise	Without a Skill	With a Skill
Lead scoring criteria	Explain rubric every time	Claude scores leads your way automatically
Data quality standards	Manually check each dataset	Claude applies your standards consistently
Email writing style	Edit every draft Claude writes	Claude writes in your voice from the start
Report format	Copy-paste template, fix formatting	Claude generates reports in your exact format

This is the core skill of this bootcamp: turning what's in your head into something an agent can use.

How to Build Skills: Start With Your Source of Truth

The most important principle for building skills: start with what already works.

Source of Truth	Example
How you already do the work	Your personal process for scoring leads, your email templates
The best person on your team	How your top sales rep qualifies deals, how your best analyst profiles data
A best practices doc	Your company's style guide, your team's SOP for customer research
A trusted external source	A methodology from a respected practitioner, an industry framework you've vetted

The right workflow:

1. Find or create your source of truth (the "golden" version)
2. Write it down as clear, step-by-step instructions
3. Format it as a SKILL.md with the right structure
4. Test it with Claude and iterate

You can use AI to help build the skill. Once you have your source of truth, Claude can help you structure it, identify gaps, and format it correctly. But AI assists the process, it doesn't replace the expertise.

What NOT to do:

Anti-Pattern	Why It Fails
Ask AI to "create a lead scoring skill" from scratch	AI will generate generic content that doesn't reflect your actual criteria
Copy skills from "awesome-claude-code" repos online	Some influencer's workflow isn't your workflow. Their scoring rubric isn't your rubric.
Start with AI and hope it matches your process	You end up with a skill that looks professional but produces wrong outputs

The best skills encode real expertise. They capture what the best person on your team actually does, not what a random AI thinks they should do. Start with your truth, then use AI to help structure and improve it.

The Architecture: Incremental File Exploration

Skills are part of an emerging pattern in agent design: incremental file exploration.

The naive approach is to dump all knowledge into the prompt upfront. "Here's everything you might need to know." This hits context limits fast and wastes tokens on irrelevant information.

The better approach: let the agent explore and load knowledge on-demand.

```
Traditional approach:
+---------------------------------------------+
| System prompt with ALL instructions         |  <- Context bloat
| + ALL examples + ALL rubrics + ALL formats  |
+---------------------------------------------+

Skills approach:
+---------------------------------------------+
| Lean system prompt                          |
| + Skill names and descriptions (lightweight)|
+---------------------------------------------+
              v (when needed)
+---------------------------------------------+
| Load specific SKILL.md                      |  <- On-demand
| -> Load references/ as needed                |
+---------------------------------------------+
```

How it works:

1. At startup, Claude only loads skill `name` and `description` (a few lines each)
2. When your request matches a skill, Claude loads the full SKILL.md
3. If that skill references detailed docs, Claude loads those only when needed
4. The agent manages its own context window by exploring incrementally

This is the same pattern we saw in Week 2 with filesystem + bash. Instead of stuffing everything into the prompt, the agent retrieves context as it needs it. Skills extend this pattern to domain knowledge.

How Skills Work

A skill is just a folder with a SKILL.md file. Claude reads it like a recipe and follows the steps.

```
.claude/skills/lead-scorer/
+-- SKILL.md              # Your instructions
+-- references/
|   +-- scoring-rubric.md # Detailed criteria (loaded on-demand)
+-- scripts/
    +-- validate.py       # Optional automation
```

When you ask Claude something that matches a skill's description, Claude asks permission to use it, loads the instructions, and works from your playbook instead of improvising.

The key difference from tools:

Tools execute and return results. Skills prepare the agent to solve a problem. When Claude invokes a skill, it loads the SKILL.md as new instructions, adjusts its execution context, and continues with this enriched environment. Skills change how Claude thinks, not just what it can do.

The Story Behind Skills

Agent Skills started inside Anthropic as a way to make Claude Code more useful for specialized tasks. As models got more capable, the team needed a scalable way to equip agents with domain expertise without bloating the base prompt.

At an internal hackathon, teams built skills for everything from code review to customer research. The pattern worked so well that Anthropic productized it.

October 2025: Anthropic launched Agent Skills publicly. They also released a "skill-creator" skill that uses Claude to generate new skills interactively.

December 2025: Anthropic published the Agent Skills specification as an open standard at agentskills.io, following the same playbook as MCP.

Skills as an Open Standard

The format is now supported across major AI tools:

Platform	Status
Claude (Claude.ai, Claude Code, Agent SDK)	Native
GitHub Copilot (VS Code, CLI, coding agent)	Native
OpenAI Codex CLI	Native
Cursor, Gemini CLI, and others	Native or compatible

Skills you create work across this ecosystem. Put a SKILL.md in `.claude/skills/` or `.github/skills/`, and most agents pick it up automatically. Your investment in writing skills pays off regardless of which AI tools you use.

Anthropic also launched a directory with skills from commercial partners: Atlassian, Canva, Cloudflare, Figma, Notion, Ramp, and Sentry. These are production-grade examples of how companies package their workflows as skills.

References:

- Anthropic: Introducing Agent Skills (https://www.anthropic.com/news/skills)
- Anthropic Engineering: Equipping Agents for the Real World (https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills)
- Agent Skills Open Standard (https://agentskills.io)
- GitHub: Agent Skills Repository (https://github.com/anthropics/skills)

Skills vs Slash Commands

Aspect	Slash Commands	Agent Skills
Invocation	Explicit: `/command`	Automatic: Claude detects
Structure	Single `.md` file	Directory with `SKILL.md`
Complexity	Simple prompts	Multi-file workflows
Discovery	Manual	Automatic (by description)

How Skills Work

1. **Discovery**: At startup, Claude loads skill `name` and `description`

2. **Activation**: When your request matches, Claude asks to use the skill

3. **Execution**: Full `SKILL.md` loads, Claude follows instructions

SKILL.md Structure

```
---
name: skill-name
description: What it does and when to use it. Include trigger keywords.
---

# Skill Title

## Instructions
Step-by-step guidance for Claude.

## Examples
Concrete input/output examples.
```

YAML Frontmatter Fields

Field	Required	Description
`name`	Yes	Lowercase, hyphens only (max 64 chars)
`description`	Yes	What + when (max 1024 chars)
`allowed-tools`	No	Restrict tools: `Read, Grep, Bash(python:*)`
`context`	No	`fork` for isolated sub-agent context
`hooks`	No	PreToolUse, PostToolUse, Stop handlers

Demo

Show a skill in action:

1. Create a simple skill
2. Trigger it with a matching request
3. See Claude ask permission and execute

03. Block 2: Lab 1 - Your First Skill (30 min)

Task: Create a Data Profiling Skill
Build a skill that profiles CSV datasets.

Step 1: Create the skill directory:

```
mkdir -p .claude/skills/data-profiler
```

Step 2: Create `SKILL.md`:

```
---
name: data-profiler
description: Profile CSV datasets to understand structure, quality, and statistics. Use when analyzi
---

# Data Profiler

When profiling a dataset, provide:

## 1. Structure Overview
- Number of rows and columns
- Column names and data types
- Sample of first 5 rows

## 2. Quality Assessment
- Missing values per column (count and percentage)
- Duplicate rows
- Obvious data type mismatches

## 3. Statistical Summary
For numeric columns:
- Min, max, mean, median
- Standard deviation
- Outlier candidates (beyond 3 std devs)

For categorical columns:
- Unique value count
- Most common values (top 5)
- Distribution skew

## 4. Recommendations
Based on findings, suggest:
- Columns that need cleaning
- Potential data quality issues
- Next steps for analysis

## Output Format

Use markdown tables for statistics. Be concise but thorough.
```

Step 3: Test the skill:

```
> Profile the sample-leads.csv file
```

Watch for:

- Claude asking to use the skill
- Structured output following the template

Success Criteria

- [ ] Skill created in correct location
- [ ] Claude discovers and asks to use it
- [ ] Output follows the defined structure

04. BREAK (10 min)

05. Block 3: Theory - Advanced Skill Patterns (30 min)

Multi-File Skills

For complex skills, add supporting files:

```
.claude/skills/lead-scorer/
+-- SKILL.md              # Main instructions
+-- references/
|   +-- scoring-rubric.md # Detailed criteria
|   +-- examples.md       # Sample scores
+-- scripts/
    +-- validate.py       # Validation script
```

Reference files load on demand:

```
For detailed scoring criteria, see [scoring-rubric.md](references/scoring-rubric.md).
```

Progressive Disclosure

Keep `SKILL.md` under 500 lines. Structure for efficiency:

```
## Quick Start
[Essential instructions - always loaded]

## Detailed Reference
See [reference.md](references/reference.md) for complete documentation.

## Utility Scripts
Run validation: `python scripts/validate.py input.csv`
```

Description Best Practices

Bad:

```
description: Helps with leads.
```

Good:

```
description: Score leads 1-100 based on firmographic and behavioral signals. Use when prioritizing l
```

Include:

- What it does (capabilities)
- When to use it (trigger scenarios)
- Keywords users might say

Restricting Tools
Use `allowed-tools` for focused skills:

```
---
name: read-only-analyzer
description: Analyze code without making changes
allowed-tools: Read, Grep, Glob
---
```

Patterns:

- `Read` - exact tool name
- `Bash(python:*)` - Bash with python prefix only
- `Bash(git:*)` - Git commands only

Hooks for Skills
Add validation or logging:

```
---
name: secure-operations
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/security-check.sh $TOOL_INPUT"
---
```

---

06. Block 4: Lab 2 - Build a GTM Skill (45 min)

Task: Create a Lead Scoring Skill
Build a comprehensive lead scoring skill with:

1. Scoring rubric
2. Example scores
3. Structured output

Step 1: Create skill structure:

```
mkdir -p .claude/skills/lead-scorer/references
```

Step 2: Create `SKILL.md`:

```
---
name: lead-scorer
description: Score leads 1-100 based on company fit, buying signals, and engagement. Use when priori
allowed-tools: Read, Grep, WebSearch, WebFetch
---

# Lead Scorer

Score leads based on the rubric in [scoring-rubric.md](references/scoring-rubric.md).

## Scoring Process

1. **Gather Information**
   - Read lead data from provided source
   - Research company if URL/name provided
   - Note any missing information

2. **Apply Scoring Rubric**
   - Company Fit (0-40 points)
   - Buying Signals (0-30 points)
   - Engagement Level (0-30 points)

3. **Output Format**

| Lead | Company | Score | Breakdown | Reasoning |
|------|---------|-------|-----------|-----------|
| Name | Company | XX/100 | Fit: X, Signals: X, Engage: X | Brief explanation |

4. **Prioritization**
   - Hot (80-100): Immediate follow-up
   - Warm (60-79): Nurture sequence
   - Cool (40-59): Long-term nurture
   - Cold (<40): Deprioritize

## Important Notes

- If data is missing, note it and score conservatively
- Explain reasoning for non-obvious scores
- Flag any red flags or concerns
```

Step 3: Create `references/scoring-rubric.md`:

```
# Lead Scoring Rubric

## Company Fit (0-40 points)

### Industry Match (0-15)
- 15: Perfect target industry
- 10: Adjacent industry
- 5: Tangentially relevant
- 0: Outside target market

### Company Size (0-15)
- 15: Ideal company size (e.g., 100-1000 employees)
- 10: Acceptable range
- 5: Edge of range
- 0: Too small or too large

### Budget Indicators (0-10)
- 10: Clear budget signals (funding, growth)
- 5: Neutral
- 0: Budget concerns

## Buying Signals (0-30 points)

### Intent Signals (0-15)
- 15: Active evaluation (demo request, pricing page)
- 10: Research phase (content downloads)
- 5: Awareness (website visit)
- 0: No signals

### Timing (0-15)
- 15: Immediate need expressed
- 10: Near-term project planned
- 5: Future consideration
- 0: No timeline

## Engagement Level (0-30 points)

### Recency (0-10)
- 10: Activity in last 7 days
- 7: Activity in last 30 days
- 3: Activity in last 90 days
- 0: No recent activity

### Frequency (0-10)
- 10: Multiple touchpoints
- 5: Single touchpoint
- 0: No engagement

### Quality (0-10)
- 10: Decision-maker engaged
- 5: Influencer engaged
- 0: Unknown contact
```

Step 4: Test with sample leads:

```
> Score the leads in data/sample-leads.csv

> Score this lead: John Smith, VP Engineering at Acme Corp (250 employees, Technology industry), dow
```

Deliverable

- Working lead-scorer skill
- Scoring rubric reference
- Screenshot of scored leads with reasoning

---

07. Wrap-Up (15 min)

Key Takeaways

1. Skills = Reusable Knowledge - Encode workflows Claude can discover
2. Description is Critical - Include what, when, and keywords
3. Progressive Disclosure - Keep SKILL.md lean, reference details
4. Test Thoroughly - Verify discovery and output quality

Homework

Create Two Skills for Your Project:

Build two skills that encode expertise from your domain. Each skill should capture a repeatable workflow.

Domain	Example Skills
GTM/Sales	Lead scorer, email writer, company researcher
Developer Tools	Code reviewer, documentation generator, PR summarizer
Content/Marketing	Content brief writer, SEO analyzer, repurposing guide
Customer Support	Ticket classifier, response drafter, escalation checker
Operations	Invoice processor, compliance checker, report formatter
Data Analytics	Data profiler, anomaly detector, visualization generator

For each skill:

- Clear trigger description (when should it activate?)
- Step-by-step instructions Claude can follow
- Defined output format
- Put detailed references in `references/` subdirectory

Submit:

- Skill directories
- Screenshot of each skill in action
- Brief description of when each triggers

Next Week Preview
Week 5: Sub-agents - Orchestrating specialized agents for complex workflows

---

08. Facilitator Notes

Common Issues

1. Skill not triggering: Check description keywords match request
2. Wrong location: Must be `.claude/skills/` not `skills/`
3. YAML errors: Check frontmatter syntax (spaces, not tabs)
4. Reference not loading: Verify relative path is correct

Skill Quality Checklist

- [ ] Name is lowercase with hyphens
- [ ] Description includes what + when + keywords
- [ ] Instructions are step-by-step
- [ ] Output format is defined
- [ ] Examples are included

Timing Adjustments

- Lab 1 is essential - ensure everyone completes
- Lab 2 can be simplified to just SKILL.md if time short
- References can be homework if needed