Conversation with Claude

February 04, 2026 at 02:42 AM

You:

what are skills for Claude

Claude:

# Skills in Claude

Skills are specialized knowledge packages that help me achieve higher-quality results for specific types of tasks. They're essentially folders containing best practices and detailed instructions that I consult before starting certain work.

## How Skills Work

When you ask me to do something that involves creating or working with certain file types, I automatically read the relevant skill documentation first. This helps me:

- Follow proven workflows that have been refined through extensive testing
- Produce more professional, polished outputs
- Avoid common pitfalls and mistakes
- Apply domain-specific best practices

## Currently Available Skills

Based on what I can see, you have access to these skills:

Document Creation:
- **docx** - Creating and editing Word documents with professional formatting
- **pdf** - Reading, creating, merging, and manipulating PDF files
- **pptx** - Building and modifying PowerPoint presentations
- **xlsx** - Working with Excel spreadsheets and tabular data

Development & Design:
- **frontend-design** - Creating distinctive, production-grade web interfaces and UI components
- **product-self-knowledge** - Accurate information about Anthropic's products and APIs

Meta Skills:
- **skill-creator** - Guide for creating your own custom skills

## Custom Skills

You can also create your own skills tailored to your specific workflows! These would live in `/mnt/skills/user/` and can contain specialized knowledge for:
- Your company's coding standards
- Specific documentation templates you use frequently
- Domain-specific workflows
- Integration patterns you commonly implement

The skill-creator skill can guide you through building custom skills that extend my capabilities with your specialized knowledge and preferences.

Would you like me to help you create a custom skill for any of your specific workflows or projects?

You:

do skills work with claude.ai, claude code, and claude agents

Claude:

Great question! Skills work differently across Claude's different products:

## Claude.ai (Web/Mobile/Desktop Chat)
■ Skills are fully supported
- All the skills I mentioned are available here
- I automatically consult relevant skills when you ask me to create documents, write code, etc.
- You can also upload custom skills to personalize my capabilities for your workflows

## Claude Code (Terminal/CLI Tool)
■ Needs verification
I'm not certain about skills support in Claude Code. Let me search for the current information since this is a newer product and capabilities may have evolved.

## Claude API / Agents (via Anthropic API)
■■ Skills work differently
When you build agents using the Anthropic Python API (like your Software Development Agent and Rust Crate Analyzer), skills aren't automatically available in the same way. However, you can:
- Include skill-like instructions directly in your agent's system prompts
- Store best practices as prompt templates in your codebase
- Create your own "skill" system by loading instruction files when initializing agents

Given your work building Python agents with the Anthropic API, you might want to consider creating a local "skills" directory structure that your agents can read from - essentially mimicking the skills system for your custom agents.

You:

does the current directory tree become part of claude agent's current context when started

Claude:

When you start a Claude agent using the Anthropic API (like in your Software Development Agent or Rust Crate Analyzer projects), the directory tree does **NOT** automatically become part of the agent's context.

## What Happens by Default

- The agent only knows what you explicitly provide in the API call
- This includes:

- System prompts you define
- Messages you send
- Tool definitions (if you've implemented any)
- Any text/data you explicitly read and include

## What You Need to Do

To give your agent awareness of the directory structure, you need to **explicitly provide that information**. Common approaches:

### 1. **Pass Directory Listings in System Prompt**
```python
import os

def get_directory_tree(path):
# Build a string representation of directory structure
tree = []
for root, dirs, files in os.walk(path):
level = root.replace(path, '').count(os.sep)
indent = ' ' * 2 * level
tree.append(f'{indent}{os.path.basename(root)}/')
subindent = ' ' * 2 * (level + 1)
for file in files:
tree.append(f'{subindent}{file}')
return '\n'.join(tree)

system_prompt = f"""
You are a software development agent.

Current project structure:
{get_directory_tree('.')}

...rest of your instructions...
"""
```

### 2. **Implement File System Tools**
Give your agent tools to explore the filesystem.

### 3. **Hybrid Approach** (Recommended for your use case)
- Provide high-level directory overview in system prompt
- Implement tools for detailed exploration when needed
- This balances context efficiency with capability

## For Your Agents

Looking at your work on agents that analyze Rust crates and generate documentation, you'd want to:

1. Initially provide the crate's basic structure in the system prompt
2. Let the agent request specific file contents as it analyzes
3. This keeps token usage manageable while giving full access

You:

answer the same question for claude code

Claude:

For **Claude Code**, the answer is different - and much more powerful:

## Claude Code Context Awareness

■ YES - Claude Code has automatic awareness of your project structure when started.

Claude Code is designed as an agentic coding tool that operates within your development environment, and it includes:

### Built-in Context
- **Automatic project discovery** - It can see your current directory and explore the file structure
- **File system access** - Can read, write, and navigate your codebase
- **Git awareness** - Understands your repository context
- **Terminal integration** - Can run commands in your working directory

### How It Works

When you start Claude Code from your terminal:
```bash
claude-code
```

It launches with:
- Access to the current working directory
- Ability to explore subdirectories
- Permission to read existing files
- Capability to create/modify files
- Understanding of your project's structure

### Difference from API Agents

Claude Code could potentially **replace or complement** some of your custom agents for tasks like:
- Analyzing codebases
- Generating documentation
- Refactoring code
- Creating example applications

The main advantage: you get filesystem awareness and tool capabilities out-of-the-box without building the infrastructure yourself.

You:

If I use the claude plugin for VS Code does that work the same way as claude code

Claude:

No, they work differently - but the VS Code Claude plugin is actually very closely related to Claude Code.

## The Official Claude Code VS Code Extension (Beta)

■ Works similarly to Claude Code CLI, with enhanced features:

### What It Provides:
- **Same core capabilities as Claude Code** - full project awareness, file system access, terminal commands
- **Graphical interface** - dedicated sidebar panel instead of terminal-only
- **Real-time inline diffs** - see changes side-by-side in VS Code's native diff viewer
- **Checkpoints** - rewind code changes or fork conversations
- **Plan mode** - review Claude's planned changes before execution
- **Auto-accept features** - streamline approval of changes
- **Conversation history** - persistent chat panel in sidebar
- **@-mentions** - reference terminal output, files, and context
- **Same underlying CLI** - shares conversation history and settings

### Key Difference from Standalone Claude Code:

The VS Code extension actually **includes the CLI** and can use it from the integrated terminal. You can:
- Use the graphical extension for visual workflows
- Run `claude` in VS Code's terminal for CLI features
- Switch between them - they share conversation history
- Use `claude --resume` to continue extension conversations in CLI

## For Your Workflow (Windows 11 + PowerShell)

Given your experience building Python agents and working with development automation, the VS Code extension would give you:

1. **No need to build file system tools** - it's all built-in
2. **Visual diff reviews** - easier than terminal output for code changes
3. **Persistent context** - sidebar stays open while you work
4. **Still terminal access** - can drop to CLI when needed

Installation:
1. Open VS Code Extensions (Ctrl+Shift+X)
2. Search for "Claude Code" by Anthropic (official one)
3. Install
4. Start using via sidebar or integrated terminal

This would be a much richer experience than the custom agents you've been building, since the infrastructure is already there!