Claude Code Tutorial

A Beginner's Guide

Claude Code Tutorial

Learn to work effectively with your AI assistant

> Important Notice
>
> The company has not yet released an AI policy permitting the use of Claude Code for work purposes.
>
> Please use personal time and resources only for any experimentation with this tool.

Topics covered:

- Basic commands & navigation
- Understanding permissions
- Planning mode & workflows
- Tips for best results

Starting Claude Code

Open your terminal and type:

```
claude
```

You'll see a prompt like this:

```
╭──────────────────────────────────────╮
│ Claude Code                   v1.0   │
╰──────────────────────────────────────╯

  What would you like to do?

> 
```

Your First Conversation

▓▓ Talk to Claude naturally

```
> Can you help me create a Python script that
  converts temperatures from Celsius to
Fahrenheit?
```

▓▓ Claude will:

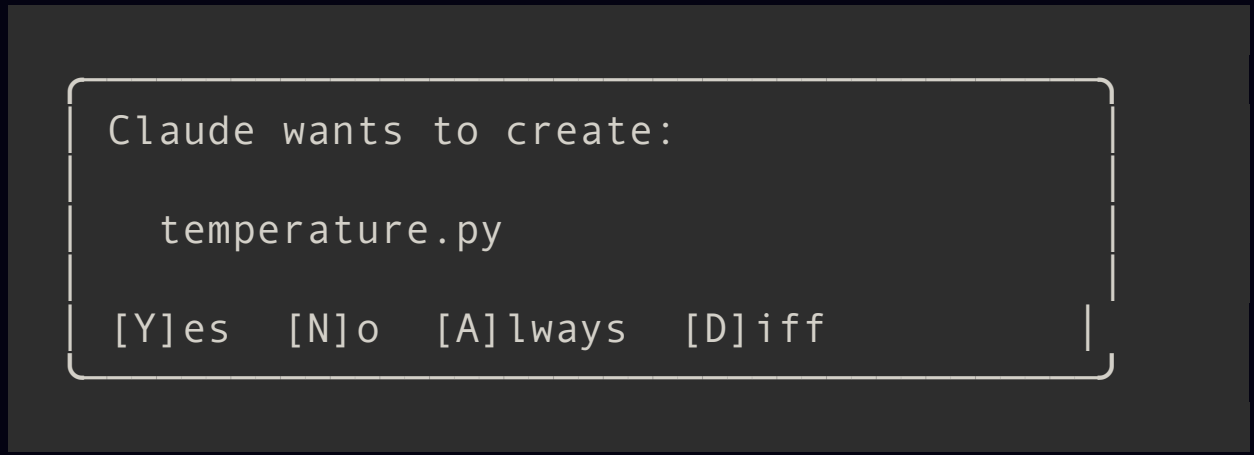
1. Understand your request
2. Write the code
3. Ask permission to save the file
4. Explain what it did

It's like texting with a programmer friend!

Understanding Permissions

Claude always asks before making changes

When Claude wants to create or edit a file:



Option	What it does
Y	Allow this one time
N	Reject the change

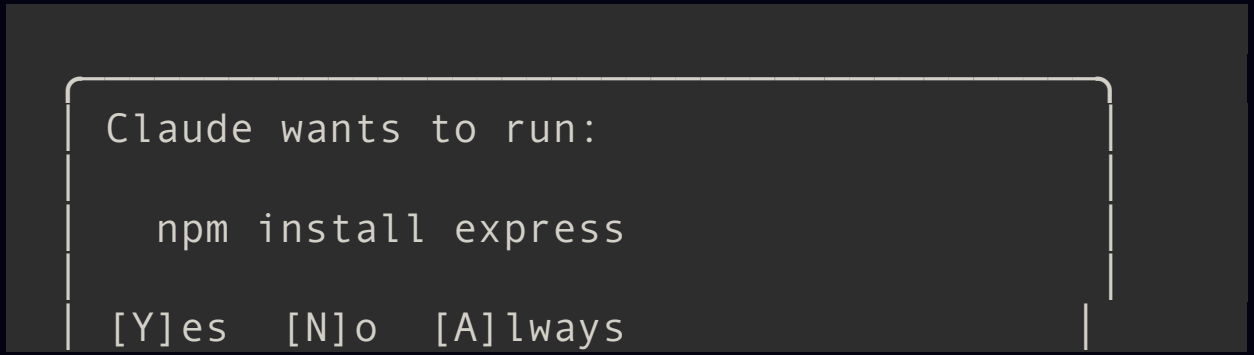
Permission Types

File Operations

- Read files - Claude looks at your code
- Write files - Claude creates new files
- Edit files - Claude modifies existing files

Command Execution

Claude may need to run commands:



Essential Slash Commands

Type these anytime during your session

Command	What it does
/help	Show all available commands
/clear	Clear conversation, start fresh
/compact	Summarize long conversations
/cost	Show how much you've used
/quit	Exit Claude Code

More useful commands

Command	What it does
/status	Show current project info
/undo	Undo the last file change
/diff	Show recent changes

`/doctor` | Diagnose common problems

Keyboard Shortcuts

See what Claude is doing

Shortcut	What it does
`Ctrl + T`	Show current task/thinking
`Ctrl + O`	Show tool output details

When to use these:

- Claude seems to be taking a while
- You want to see progress on a long task
- Curious what's happening "under the hood"

> These are great for learning how Claude works!

Background Tasks

Claude can run tasks in the background

Some operations run while you keep chatting:

- Installing packages
- Running tests
- Building projects

View background tasks

```
> /tasks
```

Shows all running and recent tasks.

Control background tasks

Status and Doctor

Check your setup with `/status`

```
> /status
```

Shows:

- Current working directory
- Files in context
- Active project settings
- Session information

Troubleshoot with `/doctor`

```
> /doctor
```

Planning Mode

For bigger tasks, ask Claude to plan first

How to activate:

```
> /plan Create a website for my bakery business
```

Or just ask:

```
> Can you plan out how to build a recipe app?
```

What happens in Planning Mode:

1. Claude analyzes what's needed

2. Creates a step-by-step plan

Working with Plans

▓▓ Claude shows you the plan first

```
┌──────────────────────────────────────┐
│ Implementation Plan                  │
├──────────────────────────────────────┤
│                                      │
│ 1. Set up project structure          │
│ 2. Create database models            │
│ 3. Build API endpoints               │
│ 4. Design user interface             │
│ 5. Add authentication                │
│                                      │
│ Proceed with this plan? [Y/N]        │
└──────────────────────────────────────┘
```

You can:

Accept Edit Mode

▓▓ Review changes before they're saved

When Claude edits a file, you'll see:

```
  def calculate_total(items):
-     total = 0
+     total = 0.0
      for item in items:
-         total = total + item
+         total += item.price * item.quantity
      return total
```

The colors show:

- Red (-) = Lines being removed

- Green (+) = Lines being added

Common Workflows

1. Creating something new

```
> Create a simple website with a contact form
```

Claude will create all necessary files.

2. Fixing a problem

```
> I'm getting an error when I click the submit
  button. Can you help fix it?
```

Claude will find and fix the issue.

Workflow: Step by Step

Best practice for new projects

Step 1: Describe your goal

```
> I want to build a habit tracker app
```

Step 2: Let Claude plan

```
> Can you plan this out before we start?
```

Step 3: Review and approve the plan

Step 4: Let Claude implement

Tips for Best Results

▓▓▓ Be specific about what you want

Less helpful:

```
> Make a website
```

More helpful:

```
> Create a personal portfolio website with
  sections for About Me, My Projects, and
  Contact. Use a modern, clean design.
```

More Tips

▓▓ Ask for explanations

```
> Before you make changes, can you explain
  what you're planning to do?
```

▓▓ Work in small steps

Instead of: "Build me a complete e-commerce site"

Try: "Let's start with a simple product listing page"

▓▓ Use undo when needed

```
> /undo
```


Choosing Your Model

Claude Code can use different AI models

Model	Speed	Token Usage	Best for
Sonnet	Fast	Lower	Most tasks
Opus	Slower	Higher	Complex reasoning

Recommendation for Max plan users:

> Use Sonnet for everyday tasks!
>
> Opus uses significantly more tokens.
> Save Opus for complex architectural decisions
> or very difficult problems.

To switch models:

Understanding Context

What is "context"?

Context = everything Claude remembers about your conversation

- What you've asked
- Files Claude has read
- Changes that were made
- Errors you've encountered

Why it matters

Claude uses context to understand your project. More context = better, more relevant answers!

> But too much context can slow things down

or cause confusion in very long sessions.

Managing Context

Check what Claude knows

```
> /context
```

Shows files and information in current context.

Clear and start fresh

```
> /clear
```

Removes all conversation history. Use when switching to a

different task.

▓▓ Compact long conversations

When to Clear Context

▓▓ Good times to use `/clear`:

- Starting a completely new task
- Claude seems confused about your project
- You want a fresh perspective
- Conversation is getting very long

▓▓ When NOT to clear:

- In the middle of a task
- When Claude needs to remember earlier work
- When debugging a problem step by step

> Tip: If unsure, use `/compact` instead!

It keeps key info while freeing up space.

CLAUDE.md Files

Teach Claude about your preferences

CLAUDE.md files contain instructions that Claude reads automatically when you start.

Two types:

Type	Location	Purpose
Global	~/.claude/CLAUDE.md	Your personal preferences
Project	./CLAUDE.md	Project-specific rules

Project settings override global settings!

What to Put in CLAUDE.md

Global preferences (your home folder)

```
# My Preferences

- Always explain code changes before making them
- Use Python for scripts unless I specify
otherwise
- Keep code simple and well-commented
```

Project-specific rules

```
# Project Guidelines

- This is a React TypeScript project
- Use Tailwind CSS for styling
```


MCP Servers

▓▓ Extend Claude's capabilities

MCP (Model Context Protocol) is an open standard that lets you add extra tools to Claude Code.

▓▓ Think of MCP servers as "plugins"

- They give Claude new abilities
- Many are available from the community
- You can even create your own!

▓▓ Browse available servers:

`https://github.com/modelcontextprotocol/servers`

How MCP Servers Work

▓▓ Claude gains new tools automatically

1. You install and configure an MCP server
2. Restart Claude Code
3. Claude can now use those tools!

▓▓ Example:

With a "weather" MCP server installed:

```
> What's the weather in Paris?
```

Claude calls the weather tool and gives you a real answer - not just training data!

MCP servers connect Claude to live data & services

Recommended: Context7

Up-to-date library documentation

Problem: Claude's training data has a cutoff date. Library docs may be outdated.

Solution: Context7 fetches current documentation!

```
> How do I use the new React 19 features?
```

Claude looks up the latest React docs for you.

Installation

Visit: https://context7.com

Follow the setup instructions for Claude Code.

Recommended: Serena

▓▓▓ Intelligent code navigation

What Serena provides:

- Find where functions are defined
- Find all references to a class
- Rename variables across your project
- Understand complex code structure

▓▓▓ Example:

```
> Find all places where the User class
  is instantiated in this project
```

Serena searches intelligently, not just text matching.

Using MCP Servers

▓▓ They work automatically!

Once configured, just ask naturally:

```
> Look up the latest FastAPI documentation
  for dependency injection
```

```
> What functions call the save_user method?
```

Claude knows which tools to use based on your question.

No special commands needed!

Handling Errors

When something goes wrong

Just tell Claude:

```
> I got this error: "Cannot read property 'map'
  of undefined" - can you fix it?
```

Or paste the error:

```
> Here's the error I'm seeing:
```

```
TypeError: Cannot read property 'map'
at UserList.js:15
```


Ending Your Session

When you're done

Type:

```
> /quit
```

or just press Ctrl + C

Your work is saved

All files Claude created or edited remain in your project folder.

Next time

Just run `claude` again in the same folder to continue where you left off!

Quick Reference Card

Commands to remember

Command	Purpose
`claude`	Start Claude Code
`/help`	Get help
`/plan`	Enter planning mode
`/undo`	Undo last change
`/clear`	Fresh start
`/compact`	Summarize context
`/status`	Check project info
`/doctor`	Troubleshoot issues
`/cost`	Check usage
`/quit`	Exit Claude Code

Permission responses

Key	Meaning

Practice Exercise

Try this on your own!

1. Start Claude Code

```
claude
```

2. Ask for a simple project

```
> Create a webpage that shows the current
  time and updates every second
```

3. Approve the files Claude creates

4. Ask Claude to open it

Need Help?

▓▓ Getting unstuck

If Claude seems confused:

```
> Let's start over. Here's what I want: ...
```

If you're confused:

```
> Can you explain that in simpler terms?
```

If something broke:

```
> /undo
```