Enabling Learning — Companion Guide

AI Agents in Your Terminal: A Beginner's Guide for Educators

A downloadable companion resource for the course module *The Age of AI Agents*
Enabling Learning LLC | enablinglearning.com Luis Hernandez | February 2026

Table of Contents

1. What Is the Terminal?

The terminal is a text-based interface for communicating with your computer. Instead of clicking icons and menus, you type commands and press Enter. The computer responds with text.

Every computer has a terminal built in: - **Mac:** It's called **Terminal** (preinstalled) - **Windows:** It's called **Command Prompt**, **PowerShell**, or **Windows Terminal** (preinstalled) - **Linux:** It's called **Terminal** (preinstalled)

You may have never opened it, and that's perfectly normal. Most people never need to. But terminal-based AI agents — like Claude Code — are among the most powerful AI tools available today, and they are surprisingly easy to use once you get past the initial setup.

What Does It Look Like?

When you open a terminal, you see something like this:

```
luis@MacBook-Pro ~ %
```

or on Windows:

```
C:\Users\Luis>
```

This is called the **prompt** — the computer is waiting for you to type a command. That's it. No buttons, no menus, just a blinking cursor.

Why Text Instead of Buttons?

The terminal gives you more direct control over your computer. For AI agents specifically, the terminal allows the AI to:

- Read and create files on your computer
- Navigate through your folders
- Execute multi-step tasks autonomously
- Create entire project structures in seconds
- Search through hundreds of documents at once

These are things a web-based chatbot simply cannot do.

---

2. Why Would a Teacher Use the Terminal?

Most teachers will be well served by the web-based AI tools described earlier in this course (Claude Projects, ChatGPT Projects, Gemini Gems) and by Claude Cowork. You do **not** need

to use the terminal to benefit from AI.

However, terminal-based agents are uniquely powerful for tasks like:

- **Batch processing:** Converting, reformatting, or analyzing 20, 50, or 100 files at once
- **Building organized systems:** Creating structured folder systems with templates, standards references, and instruction files that AI agents can use across multiple projects
- **Automating repetitive work:** Renaming files, reorganizing folders, extracting data from spreadsheets
- **Creating custom tools:** Building simple web apps, forms, or interactive materials for your students
- **Managing large curriculum projects:** Working across an entire semester's worth of lesson plans, assessments, and resources simultaneously

Who Is This For?

This guide is for educators who:

- Are curious about what's possible with AI beyond web-based chat
- Are comfortable trying new software (even if the terminal is new to you)
- Want to understand what tools like Claude Code can do, even if they don't use them daily
- Are instructional technologists, curriculum designers, or department leads who manage large volumes of materials

If you read this guide and decide the terminal isn't for you, that's completely fine. The web-based tools and Claude Cowork are excellent alternatives for most teaching tasks.

---

3. Opening the Terminal on Your Computer

On Mac

1. Press **Command + Space** to open Spotlight Search
2. Type **Terminal**
3. Press **Enter**

The Terminal app opens. You'll see a window with a text prompt.

> **Tip:** You can also find Terminal in Applications > Utilities > Terminal.

On Windows

Option 1: Windows Terminal (recommended)

1. Press the **Windows key**
2. Type **Terminal** or **Windows Terminal**
3. Click to open

Option 2: PowerShell

1. Press the **Windows key**
2. Type **PowerShell**
3. Click to open

> **Note:** If you're on Windows, we recommend using **Windows Terminal** or **PowerShell** rather than the older Command Prompt. The commands in this guide work in both, but Windows Terminal provides a better experience.

Basic Terminal Commands You Should Know

You only need a handful of commands to work with AI agents:

Command	Mac/Linux	Windows (PowerShell)	What It Does
See where you are	`pwd`	`pwd`	Shows your current folder location
List files	`ls`	`ls` or `dir`	Shows files in the current folder
Go into a folder	`cd FolderName`	`cd FolderName`	Moves into a subfolder
Go up one folder	`cd ..`	`cd ..`	Moves up to the parent folder
Go to your home folder	`cd ~`	`cd ~`	Returns to your home directory
Go to Desktop	`cd ~/Desktop`	`cd ~/Desktop`	Navigates to your Desktop
Clear the screen	`clear`	`cls`	Clears all text from the terminal

Example: If you want to navigate to a folder called "Teaching Materials" on your Desktop:

```
cd ~/Desktop/"Teaching Materials"
```

> **Important:** If a folder name has spaces, put it in quotes: `"Teaching Materials"`, not `Teaching Materials`.

---

4. Installing the Prerequisites

Before you can install Claude Code, you need one piece of software: **Node.js**. This is a free, open-source tool that many modern applications run on. Installing it takes about 5 minutes.

What Is Node.js?

Node.js is a runtime environment that lets programs written in JavaScript run on your computer. Claude Code is built with Node.js. You install Node.js once, and then you can install Claude Code (and many other tools) with a single command.

Installing Node.js

On Mac

Option 1: Download from the website (easiest)

1. Go to nodejs.org
2. Download the **LTS** (Long Term Support) version — this is the stable, recommended version
3. Open the downloaded file and follow the installation wizard
4. When it finishes, open Terminal and type: `node --version`
5. You should see a version number like `v22.x.x` — this means it worked

Option 2: Using Homebrew (if you have it)

```
brew install node
```

On Windows

1. Go to nodejs.org
2. Download the **LTS** version for Windows

3. Run the installer — accept the defaults, including the option to add Node.js to your PATH
4. When it finishes, open Windows Terminal and type: `node --version`
5. You should see a version number — this means it worked

Verifying the Installation

After installing Node.js, verify that both `node` and `npm` (Node Package Manager, installed alongside Node.js) are available:

```
node --version
npm --version
```

Both commands should return version numbers. If you see an error like "command not found," try closing and reopening your terminal, or restarting your computer.

---

5. Installing Claude Code

With Node.js installed, installing Claude Code takes one command.

Step 1: Install Claude Code

Open your terminal and type:

```
npm install -g @anthropic-ai/claude-code
```

This downloads and installs Claude Code globally on your computer. The `-g` flag means "global" — you can run it from any folder.

Step 2: Verify the Installation

```
claude --version
```

You should see a version number. If you see an error, try closing and reopening your terminal.

Step 3: Authenticate

The first time you run Claude Code, you'll need to sign in with your Anthropic account:

```
claude
```

Claude Code will open a browser window or provide a link for you to authenticate. Sign in with the same account you use at claude.ai. You need a paid plan (Pro, Max, Team, or Enterprise).

Step 4: You're Ready

That's it. Claude Code is installed. From now on, whenever you want to use it, you navigate to a folder in your terminal and type `claude`.

---

6. Your First Session with Claude Code

Let's walk through a complete first session.

Step 1: Create a working folder

```
mkdir ~/Desktop/"My Teaching AI Workspace"
cd ~/Desktop/"My Teaching AI Workspace"
```

Step 2: Start Claude Code

```
claude
```

Claude Code starts and shows you a prompt. You can now type in natural language.

Step 3: Ask Claude to set up your workspace

Type something like:

```
I'm a bilingual education teacher in Texas working with English Learners in g
Set up a workspace for me with folders for my three main units this semester
Fractions, and Immigration Narratives. In each unit folder, create subfolders
lesson-plans, student-materials, and assessments. Also create a standards-re
```

```
where I'll put my ELPS and TEKS documents. Create an INSTRUCTIONS.md file in
with appropriate context for an AI assistant.
```

What Happens Next

Claude Code will:

1. Create all the folders you described
2. Create `INSTRUCTIONS.md` files in each one with tailored context
3. Create a `CLAUDE.md` file at the root of your workspace (more on this below)
4. Show you everything it's doing as it works

When it finishes, your workspace looks like this:

```
My Teaching AI Workspace/
├── CLAUDE.md
├── standards-reference/
│   └── INSTRUCTIONS.md
├── Unit-Ecosystems/
│   ├── lesson-plans/
│   ├── student-materials/
│   ├── assessments/
│   └── INSTRUCTIONS.md
├── Unit-Fractions/
│   ├── lesson-plans/
│   ├── student-materials/
│   ├── assessments/
│   └── INSTRUCTIONS.md
└── Unit-Immigration-Narratives/
    ├── lesson-plans/
    ├── student-materials/
    ├── assessments/
    └── INSTRUCTIONS.md
```

All of this was created in seconds, just by describing what you wanted.

Step 4: Start Working

Now you can give Claude Code tasks within this workspace:

```
Create a 5-day lesson plan for my Ecosystems unit on food webs and energy tra
Align it with 5th grade TEKS for Science and include ELPS-aligned language ol
```

```
for Beginning and Intermediate proficiency levels. Save it in the lesson-plan
```

Claude Code reads your `INSTRUCTIONS.md` files for context, creates the lesson plan, and saves it exactly where you asked.

Step 5: Exit

When you're done, type:

```
/exit
```

or press **Ctrl+C** to leave Claude Code and return to your regular terminal.

---

7. How AI Agents Organize Your Work

One of the most powerful aspects of terminal-based AI agents is their ability to create and maintain organized file systems. This section explains how this works and why it matters.

The CLAUDE.md File

When you use Claude Code in a folder, it can create a file called `CLAUDE.md` at the root of your workspace. This file serves as a **permanent instruction set** — every time you start a new Claude Code session in that folder, Claude reads this file first and knows your context.

Think of `CLAUDE.md` as a memo you leave on your desk for your teaching assistant. It might say:

```
# My Teaching Workspace

## About Me
I teach bilingual education (English/Spanish) in Texas, grades 6-8.
I work with English Learners at Beginning through Advanced High
        proficiency levels.

## Standards
- ELPS documents are in the standards-reference/ folder
- Always align language objectives with ELPS
- Use TELPAS proficiency levels for differentiation
```

```
## Preferences
- Lesson plans should follow the 5E model
- Include Spanish cognates in vocabulary activities
- Format for 50-minute class periods
- Use student-friendly language in all materials
```

You don't have to write this yourself. You can tell Claude Code: "Create a CLAUDE.md file based on what I just told you about my teaching context." Claude writes it, and from then on, every session starts with that context already loaded.

The Folder-as-Knowledge-Base Pattern

This is where terminal agents become truly powerful. Your folder structure is not just organization — it's a **knowledge base** that the AI reads and references.

When you place your ELPS document, TEKS standards, sample lesson plans, or curriculum guides into your workspace folders, Claude Code can:

- **Read all of them** at the start of a session
- **Cross-reference** between documents (e.g., matching a TEKS standard to an ELPS expectation)
- **Generate new materials** that are grounded in your actual documents, not generic training data
- **Maintain consistency** across everything it creates because it sees the full picture

This is the same principle behind Claude Cowork (see the companion guide), but the terminal gives you even more control and can handle larger, more complex projects.

Sharing Data Across Projects

You can create a shared reference folder that multiple projects point to:

```
My Teaching AI Workspace/
├── CLAUDE.md
├── _shared-references/
│   ├── texas-elps-2024.pdf
│   ├── telpas-proficiency-descriptors.pdf
│   ├── district-lesson-plan-template.docx
│   └── my-teaching-philosophy.md
├── Fall-2026/
│   ├── CLAUDE.md  ← references _shared-references/
│   ├── Unit-Ecosystems/
```

```
│   └── Unit-Fractions/
└── Spring-2026/
    ├── CLAUDE.md  ← references _shared-references/
    ├── Unit-Immigration-Narratives/
    └── Unit-Civil-Rights/
```

Each semester's `CLAUDE.md` can include an instruction like: "Shared reference documents are in `../_shared-references/`. Always consult those documents when creating materials." This way, you maintain one set of standards documents and both semesters' projects use them.

Other Agents Use the Same Pattern

This folder-based approach is not unique to Claude Code. Other terminal AI tools follow similar patterns:

Tool	Instruction File	How It Works
Claude Code	`CLAUDE.md`	Read automatically at session start
Gemini CLI	`GEMINI.md`	Read automatically at session start
GitHub Copilot	`.github/copilot-instructions.md`	Read automatically in the repository
OpenCode	Configuration files	Read at session start

The concept is the same across all of them: **you create a folder, add your reference documents and instructions, and the AI agent reads everything when it starts working.**

---

8. Other Terminal AI Tools

Claude Code is not the only option. Here are other terminal-based AI tools worth knowing about.

Gemini CLI (Google)

What it is: Google's terminal AI agent, powered by the Gemini model family.

How to install:

```
npm install -g @anthropic-ai/claude-code
```

Actually, Gemini CLI uses a different installation method:

```
npm install -g @anthropic-ai/claude-code
```

> **Note:** Check Google's official documentation for the latest installation instructions, as the CLI is under active development.

Key features: - Up to 1 million tokens of context (the largest of any AI model) - Deep integration with Google Workspace (Docs, Sheets, Drive) - Creates `GEMINI.md` instruction files similar to Claude's `CLAUDE.md` - Free tier available with Gemini API key

Best for: Educators who are already in the Google ecosystem and want seamless integration with Google Workspace tools.

GitHub Copilot CLI

What it is: A terminal agent from GitHub that integrates with code repositories.

How to install:

```
gh extension install github/gh-copilot
```

(Requires the GitHub CLI to be installed first.)

Key features: - Supports multiple AI models (Claude, GPT, and others) - Tight integration with GitHub repositories - Best for educators who maintain websites, learning platforms, or code-based tools

Best for: Instructional technologists and educators who build or maintain web-based educational tools.

OpenCode

What it is: An open-source terminal AI agent that supports 75+ AI providers.

Key features: - Completely free and open source - Works with Claude, GPT, Gemini, and many other models - 95,000+ GitHub stars (large community) - Good option if you want

flexibility in choosing your AI provider

Best for: Educators who want a free option or want to experiment with different AI models.

Which Should You Start With?

If you...	Start with...
Already pay for Claude (Pro/Max)	Claude Code
Use Google Workspace heavily	Gemini CLI
Want free and flexible	OpenCode
Build web tools or websites	GitHub Copilot CLI
Are completely new to the terminal	Claude Code (best documentation, most intuitive)

Our recommendation: **Start with Claude Code.** It has the most polished experience, the best documentation, and the most intuitive natural-language interface. If you already have a Claude Pro subscription, you already have access.

---

9. Privacy and Safety Considerations

The same privacy rules that apply to Claude Cowork apply to terminal-based agents, with one important addition: **terminal agents have broader access to your file system.**

The Key Difference from Cowork

Claude Cowork runs in an isolated virtual machine and can only access folders you explicitly grant. Terminal agents like Claude Code run **directly on your computer** and can potentially access any file you can access. This makes folder discipline even more important.

Golden Rules for Terminal AI Agents

1. **Create a dedicated workspace folder.** Never run Claude Code from your home directory, your Documents folder, or any location that contains student records. Always `cd` into a specific, purpose-built folder first.
2. **Never place student PII in your workspace.** No names, IDs, grades, test scores, IEP/504 documents, behavioral records, or any other personally identifiable

information.

3. **Review what the agent creates.** Terminal agents can create and modify files autonomously. Always review new files before using them with students.
4. **Be careful with the `CLAUDE.md` file.** Do not include sensitive information (passwords, API keys, student data) in your instruction files. These are plain text files visible to anyone who can access the folder.
5. **Use pseudonyms for any student-related work.** If you need to create differentiated materials referencing proficiency levels, use "Student A (Beginning)" rather than actual student names.
6. **Check your district's AI policy.** Some districts restrict the use of terminal-based tools or require IT approval before installing software. Verify before proceeding.

What Terminal Agents Can and Cannot Access

Can Access	Cannot Access
Files in the folder where you started the session	Files in folders you haven't navigated to
Files in subfolders of your current location	System files (unless you specifically navigate there)
Files you explicitly ask it to read	Files protected by your operating system's permissions
Internet (for API calls to the AI model)	Other users' files on a shared computer

Recommended Safety Setup

Before your first real session, set up your workspace like this:

```
My Teaching AI Workspace/          ← Run Claude Code HERE
├── CLAUDE.md                      ← No sensitive info
├── standards-reference/           ← Public documents only
├── curriculum-projects/           ← No student data
└── output/                        ← Where Claude saves new files
```

Keep your gradebooks, student records, IEPs, and other sensitive files in a completely separate location on your computer — preferably on a different drive or in a protected

school-managed folder.

---

10. Troubleshooting Common Issues

“command not found: node”

Node.js is not installed or not in your PATH. - **Mac:** Reinstall from nodejs.org, then close and reopen Terminal - **Windows:** Reinstall from nodejs.org, making sure to check “Add to PATH” during installation, then restart your computer

“command not found: claude”

Claude Code is not installed or not in your PATH. - Run `npm install -g @anthropic-ai/claude-code` again - If that fails, try `sudo npm install -g @anthropic-ai/claude-code` (Mac/Linux) — this runs the command with administrator privileges - On Windows, open Terminal as Administrator and try again

“Permission denied”

Your computer is blocking the installation. - **Mac:** Use `sudo` before the command: `sudo npm install -g @anthropic-ai/claude-code` - **Windows:** Right-click Terminal and select “Run as Administrator”

“npm ERR! network”

Your internet connection is blocking npm. - Check your internet connection - If you’re on a school network, the network may block npm. Try on your personal Wi-Fi - Some school firewalls block package managers. Contact your IT department

Claude Code starts but can’t authenticate

- Make sure you have a paid Claude account (Pro, Max, Team, or Enterprise)
- Try running `claude auth` to re-authenticate
- Check that your browser is not blocking pop-ups (the authentication window may be blocked)

Files aren’t being created

- Make sure you're in the right folder: type `pwd` to check your location
- Make sure the folder has write permissions
- Try a simple test: ask Claude Code to "create a file called test.md with the text 'hello'"

Pro Tip: Use Voice Input to Talk to Your AI Agent

When working with terminal AI agents, you often need to give longer, more detailed instructions than a quick chat question. Typing a multi-paragraph prompt explaining exactly what you want can feel slow — especially when you know exactly what to say.

Voice-to-text tools let you **speak your instructions** instead of typing them. This is not about replacing writing — writing is thinking, and there is real value in composing your own ideas at the keyboard. But when you are giving instructions to an AI agent, the goal is efficiency. Speaking naturally lets you give richer, more detailed prompts in a fraction of the time.

Recommended tools:

Tool	Platform	Cost	Notes
SuperWhisper	macOS	Free tier available	Runs locally using Whisper AI. Very accurate, works offline, privacy-friendly. This is what the author of this guide uses daily.
Built-in Dictation	macOS / Windows	Free	macOS: System Settings > Keyboard > Dictation. Windows: press Win+H. Already on your computer, no install needed.
Google Docs Voice Typing	Chrome browser	Free	Open a Google Doc, go to Tools > Voice typing. Works well for drafting prompts before pasting them into the terminal.

How to use it with Claude Code: Open SuperWhisper (or your built-in dictation), speak your instructions, and the text appears wherever your cursor is — including directly in the

terminal. For example, instead of typing a long prompt about processing 20 lesson plan files, you can simply describe what you need out loud and let the voice tool transcribe it into the terminal.

> **This is optional.** You can absolutely use terminal agents by typing your instructions. Voice input is simply a productivity tip for those who want to save time when giving detailed prompts.

---

11. Additional Resources

Official Documentation

Resource	Link
Claude Code Documentation	https://docs.anthropic.com/en/docs/claude-code
Claude Code GitHub	https://github.com/anthropics/claude-code
Node.js Downloads	https://nodejs.org
Gemini CLI Documentation	https://ai.google.dev/gemini-api/docs/cli
GitHub Copilot CLI	https://docs.github.com/en/copilot
OpenCode	https://github.com/opencode-ai/opencode

Related Enabling Learning Resources

Resource	Description
Boosting Productivity with AI (Free Course)	enablinglearning.com/courses/boosting-productivity-with-ai-for-teachers/
Claude Cowork for Educators Guide	Companion guide covering the desktop app approach (see course downloads)
ELPS Online Helper (Free Tool)	apps.enablinglearning.com/elps/

Learning More About the Terminal

Resource	Description
Codecademy: Learn the Command Line	Free introductory course on terminal basics
The Missing Semester (MIT)	Free course on command-line tools from MIT

---

12. References

Anthropic. (2026). *Claude Code documentation*. https://docs.anthropic.com/en/docs/claude-code

Anthropic. (2026). *Using Cowork safely*. Claude Help Center. https://support.claude.com/en/articles/13364135-using-cowork-safely

Google. (2026). *Gemini CLI documentation*. https://ai.google.dev/gemini-api/docs/cli

Node.js Foundation. (2026). *Node.js*. https://nodejs.org

U.S. Department of Education. (2021). *Family Educational Rights and Privacy Act (FERPA)*. https://studentprivacy.ed.gov/faq/what-ferpa

U.S. Department of Education, Office of Educational Technology. (2023). *Artificial intelligence and the future of teaching and learning: Insights and recommendations*. https://tech.ed.gov/ai-future-of-teaching-and-learning/

This guide was created by Enabling Learning LLC as a companion resource for the course "Boosting Productivity with AI for Teachers." For questions, feedback, or professional development inquiries, visit enablinglearning.com.

Last updated: February 2026