MODULE 03 · AI EDUCATION SERIES

# Mode 2: API + Terminal / CLI

From zero coding to your first AI automation script.

**WHAT YOU GET**

- What API + CLI mean in plain English (with simple analogies)
- Install Python on Windows + Mac, step-by-step and screenshot-friendly
- Terminal basics: cd, ls, pwd, mkdir, how to navigate folders
- Get an API key from Anthropic (Claude) or Google (Gemini)
- Your first script: call AI from Python
- Alex's project: auto-summarize 10 academic PDFs
- Cron / scheduling: run your script automatically every day
- Common errors + beginner debugging mindset

**Nicky Pandelaki**

AI Engineer & Creator · Tangerang, 2026

*Series: Module 01 · 02 (Mode 1) > 03 (Mode 2) · 04 (Mode 3)*

# Introduction

Welcome to Module 03. This is the heaviest module in the series, but also the most transformative. In Module 02 you were just an AI user. In Module 03 you become an AI operator. You can make AI do repetitive tasks for you without manually chatting each time.

Honest note: the first 30 minutes will feel overwhelming. Lots of new terms. But once you run your first script and it works, everything clicks. Promise.

## Recap from Module 02

In Module 02, Alex got good at using claude.ai. They can summarize, brainstorm, and decode papers. But Alex starts noticing Mode 1 limits.

- Summarizing 10 papers = 10 uploads + 10 copy-paste rounds
- Daily writing practice = has to remember to open claude.ai every day
- Wants to share results with a study group = sends files one by one over chat

Mode 2 removes these limits with one idea: API. You can call AI from your own code. Loop over 10 PDFs? 5 minutes with a script. Want it to run every morning? Drop it in cron. Share the tool with friends? Share the script.

## Guide persona: Alex, continued

**Alex, two weeks after finishing Module 02:**

Alex is now comfortable using Claude. But the new semester starts and they have to read 15 research methodology papers for a seminar. Doing it manually through the UI would take 3-4 hours. Alex decides to invest one day learning Mode 2 so they can save dozens of hours later.

## Skills you will gain

- Install Python on your laptop (Windows or Mac)
- Use the terminal / command line to navigate and run programs
- Get an API key and protect it as an environment variable
- Write your first Python script that calls AI
- Build a program that reads files, processes them with AI, and writes output
- Set up cron so your script runs automatically every day

# Section 1: Core Concepts

Before installing anything, understand three core ideas: API, CLI, and Terminal. With simple analogies.

## 1.1 What is an API?

API stands for Application Programming Interface. Forget the long name and focus on the function: an API is how your computer talks to another computer.

Analogy: imagine ordering coffee at Starbucks. Normal way: stand in line, order at the counter, hand over money, get coffee. API way: there is a drive-thru. You send the order via intercom (request), the barista prepares it (process), and they hand back coffee through the window (response). Fast, no need to see anyone's face, and anyone can do it.

AI APIs work the same way. You send text (request) to Claude or Gemini's servers, they process it, they send back text (response). The difference from claude.ai: no UI, everything is via code.

## 1.2 What is a CLI?

CLI stands for Command Line Interface. How it works: you type a text command, hit Enter, and the command executes.

Analogy: SMS vs WhatsApp. WhatsApp has UI (buttons, stickers, emojis). SMS is just text. Simpler and faster once you know the commands. CLI is the SMS of the computing world.

CLI tools you will use: Claude Code, the Python interpreter, gemini-cli. All called from the terminal by typing the tool name plus arguments.

## 1.3 What is a Terminal?

The terminal is a text-based application on your laptop where you type commands and see the output.

- On Mac: search 'Terminal' in Spotlight (Cmd + Space, type 'terminal')
- On Windows: search 'PowerShell' or 'Command Prompt' in the Start menu
- On Linux: already installed, usually Ctrl+Alt+T

Open it now. You will see a black screen with text. That is the prompt where you type commands. Try: echo 'hello' then press Enter. The terminal replies 'hello'. Congrats, you just ran your first CLI command.

# Section 2: Install Python

Python is the beginner-favorite programming language. Syntax is close to English, the AI library ecosystem is huge, and the community is welcoming. Mode 2 and Mode 3 both run on Python.

## 2.1 Install Python on Mac

1. Open a browser and go to python.org
2. Click 'Downloads' in the top menu, choose 'macOS'
3. Download the latest installer (version 3.12 or newer)
4. Open the .pkg file, follow the steps (Next, Next, Install)
5. Open Terminal, type: python3 --version, press Enter
6. If you see 'Python 3.12.x' or similar, install succeeded

## 2.2 Install Python on Windows

1. Open a browser and go to python.org
2. Click 'Downloads' in the top menu, choose 'Windows'
3. Download the latest installer (version 3.12 or newer)
4. Open the .exe file you downloaded
5. IMPORTANT: check 'Add Python to PATH' at the bottom, then click Install Now
6. Wait until it finishes, click Close
7. Open PowerShell, type: python --version, press Enter
8. If you see 'Python 3.12.x' or similar, install succeeded

## 2.3 Install pip (Package Manager)

pip is how you install additional libraries. It usually ships with Python, but verify it with this command in the terminal.

**terminal**

```
python3 -m pip --version
```

If you see a pip version, you are good. If you get an error, run: python3 -m ensurepip --upgrade

## 2.4 Install the Anthropic library

This library connects your Python code to the Claude API. Install with this command.

**terminal**

```
pip install anthropic python-dotenv pypdf
```

Wait a few seconds. If you see 'Successfully installed ...', you are ready.

# Section 3: Terminal Basics

Before writing code, you need to be comfortable in the terminal. These 5 commands cover 90 percent of your needs.

## 3.1 pwd (Print Working Directory)

Shows which folder you are in right now.

```terminal
pwd
```

Output: /Users/alex/Documents (or a similar path)

## 3.2 ls (List)

Lists all files and folders at the current location.

```terminal
ls
```

Output: a list of file and folder names, separated by spaces or lines.
Variant: ls -la (more detail: size, date, permissions)

## 3.3 cd (Change Directory)

Move into a different folder.

```terminal
cd Documents
cd Documents/School
cd ..  # go up one level
```

Note: '..' means parent folder. '~' means your home folder.

## 3.4 mkdir (Make Directory)

Create a new folder.

```terminal
mkdir alex-paper-summaries
```

Now there is a new 'alex-paper-summaries' folder at your current location.

## 3.5 python3 (Run Python)

Runs a Python file you have written.

```terminal
python3 filename.py
```

Or launch the Python interpreter (REPL) for quick experimenting:

```terminal
python3
```

Once inside the REPL, type exit() to quit.

## 3.6 Your cheat sheet

```
pwd            - where am I
ls             - list folder contents
cd folder/     - move into folder
cd ..          - go up one level
mkdir name     - make a folder
python3 file.py - run a Python script
clear          - clear the terminal screen
Ctrl + C       - stop the running program
```

# Section 4: Get Your API Key

An API key is the 'password' for accessing an API. Each AI company has a similar flow to get one. We will use Claude (Anthropic) as the example because it fits Alex's task best.

## 4.1 Register at the Anthropic Console

1. Open a browser and go to console.anthropic.com
2. Click 'Sign Up' if you do not have an account, or 'Sign In' if you do
3. Verify your email via the link sent to your inbox
4. Once logged in, click 'API Keys' in the left sidebar
5. Click 'Create Key', give it a name (for example: 'alex-module-03')
6. Anthropic shows the key once. COPY it and store it safely
7. Top up your balance: click 'Billing' in the sidebar, add minimum 5 USD via card

After topping up 5 USD, each Claude call costs roughly 0.003 to 0.015 USD depending on input length. 5 USD is plenty for hundreds of experiments.

## 4.2 Store your API key safely

An API key is like a password. Do not share it, do not commit it to GitHub, do not paste it in chat. Standard practice: store it in a .env file that does not get shared.

In the terminal, cd into your project folder and create a .env file

**terminal**

```
cd alex-paper-summaries
nano .env
```

The file editor opens. Type this line, replacing xxxxxx with your API key.

**.env**

```
ANTHROPIC_API_KEY=sk-ant-xxxxxxxxxxxxxxx
```

Press Ctrl+O then Enter to save, Ctrl+X to exit nano.

Note: files starting with a dot (.) are hidden. Check with 'ls -la'. This file is not shared if you zip the folder and send it. Standard industry practice.

# Section 5: Your First Script

Now the moment you have been waiting for: write your first Python script that calls the Claude API.

## 5.1 Create hello.py

terminal

```
nano hello.py
```

Type these lines in the editor:

hello.py

```
import os
from dotenv import load_dotenv
from anthropic import Anthropic

load_dotenv()
client = Anthropic(api_key=os.environ["ANTHROPIC_API_KEY"])

response = client.messages.create(
    model="claude-sonnet-4-5",
    max_tokens=300,
    messages=[
        {"role": "user", "content": "Hi Claude, introduce yourself in 3 sentences."}
    ]
)

print(response.content[0].text)
```

Save with Ctrl+O, Enter, Ctrl+X.

## 5.2 Run the script

terminal

```
python3 hello.py
```

After a few seconds, Claude responds. Output looks something like this:

output

```
Hi! I'm Claude, an AI assistant built by Anthropic.
I'm designed to help you with all sorts of tasks:
document analysis, writing, coding, and much more.
Nice to meet you.
```

## 5.3 What just happened under the hood

- Python reads the .env file and pulls your API key
- The Anthropic library sends an HTTP request to api.anthropic.com
- Claude's server processes the prompt and generates a response
- The response goes back to Python, which prints it to the terminal
- Each call is charged to your balance (about 0.005 USD for this call)

## 5.4 Play with the prompt

Try swapping the 'content' string with a different question. Run again. Every tweak is one test cycle. This is how developers learn.

- Ask for a poem on a specific topic
- Ask it to explain a concept using an analogy
- Ask for 10 cafe name ideas

# Section 6: Alex's Project

**Auto-Summarize 15 Academic PDFs**

Now Alex combines everything they have learned into one real project. They have 15 research PDFs for a seminar. Goal: one script that reads every PDF, summarizes them one by one, and writes each summary to a markdown file.

## 6.1 Set up folder + dependencies

terminal

```
mkdir -p ~/paper-summaries/pdfs
mkdir -p ~/paper-summaries/notes
cd ~/paper-summaries
pip install pypdf python-dotenv anthropic
```

Drop all 15 of Alex's PDF papers into the pdfs/ folder. Summary notes will go into notes/.

## 6.2 Create the summarize.py script

**summarize.py**

```
import os
from pathlib import Path
from dotenv import load_dotenv
from anthropic import Anthropic
from pypdf import PdfReader

load_dotenv()
client = Anthropic(api_key=os.environ["ANTHROPIC_API_KEY"])

PDF_DIR = Path("pdfs")
NOTES_DIR = Path("notes")
NOTES_DIR.mkdir(exist_ok=True)

PROMPT = '''Summarize the following research paper in English, with this structure:

1. Title + authors (1 line)
2. Research goal (max 3 sentences)
3. Method (max 3 sentences)
4. Key findings (bullet points)
5. Practical implications (max 3 sentences)

Use academic style but keep it easy to read.

Paper:
{text}'''

for pdf_path in PDF_DIR.glob("*.pdf"):
    print(f"Processing: {pdf_path.name}")
    reader = PdfReader(str(pdf_path))
    text = "\n\n".join(page.extract_text() for page in reader.pages)
    response = client.messages.create(
        model="claude-sonnet-4-5",
        max_tokens=1500,
        messages=[
            {"role": "user", "content": PROMPT.format(text=text[:50000])}
        ]
    )
    summary = response.content[0].text
    out_path = NOTES_DIR / (pdf_path.stem + ".md")
    out_path.write_text(summary, encoding="utf-8")
    print(f"  Saved: {out_path}")

print("Done. All papers have been summarized.")
```

Save (Ctrl+O, Enter, Ctrl+X), then run it.

## 6.3 Run the script

```
terminal
python3 summarize.py
```

The script loops through every PDF, calls Claude per PDF, and writes a summary into notes/. The terminal output looks like this:

```
output
Processing: paper-01-methodology.pdf
  Saved: notes/paper-01-methodology.md
Processing: paper-02-data-analysis.pdf
  Saved: notes/paper-02-data-analysis.md
...
Processing: paper-15-conclusion.pdf
  Saved: notes/paper-15-conclusion.md
Done. All papers have been summarized.
```

Total time: 5 minutes (3 minutes Claude working, 2 minutes setup). Cost: about 0.10 USD. Compare that to the 3-4 hours Alex would have spent summarizing manually through the UI.

## 6.4 Generalize: what else can this pattern do?

- Summarize 20 meeting transcripts
- Classify client emails into buckets (urgent, info, complaint)
- Auto-translate a folder of documents
- Generate Instagram captions from 50 product photos
- Sentiment analysis on customer review comments

The base pattern is the same: read input, send to AI with a clear prompt, save output. You can adapt this pattern for almost any repetitive task.

# Section 7: Automation with Cron

Cron is the built-in scheduler on Mac/Linux that runs commands automatically on a schedule. Example: every morning at 7am, run your script.

### Open the crontab editor

```
terminal
crontab -e
```

Add this line to run the script every day at 7am:

```
crontab
0 7 * * * /usr/bin/python3 /Users/alex/paper-summaries/summarize.py
```

Format: minute hour day month weekday /path/to/python /path/to/script.py

Save and exit the editor. Cron will now run the script automatically each morning without you opening the terminal.

# Section 8: Beginner Debugging

Errors will happen. That is a normal part of coding. Here are the most common ones and how to fix them.

## Error 1: ModuleNotFoundError

```
error
ModuleNotFoundError: No module named 'anthropic'
```

Meaning: the anthropic library is not installed. Fix:

```
fix
pip install anthropic
```

## Error 2: KeyError on the API key

```
error
KeyError: 'ANTHROPIC_API_KEY'
```

Meaning: Python cannot find the environment variable. Check:

- The .env file is in the same folder as your script
- The .env value has no spaces around '=' (ANTHROPIC_API_KEY=sk-xxx)
- load_dotenv() is called before os.environ[]

## Error 3: AuthenticationError

```
error
anthropic.AuthenticationError: invalid x-api-key
```

Meaning: the key is wrong, expired, or has a stray character. Fix:

- Log in to console.anthropic.com and regenerate a new key
- Make sure you copied the entire key without typos
- Check your balance, if depleted, top up

## Error 4: RateLimitError

```
error
anthropic.RateLimitError: rate limit exceeded
```

Meaning: you sent too many requests too fast. Fix:

- Add time.sleep(2) inside the loop between API calls
- Or upgrade your account tier at console.anthropic.com

## Error 5: PDF encoding error

```
error
UnicodeDecodeError: 'utf-8' codec can't decode byte
```

Meaning: the PDF has unusual characters. Fix:

- Use try/except to skip a problem PDF
- Or add an OCR library (pytesseract) for image-based PDFs

## Debugging Mindset

Beginner's best tool: copy your error message into Claude or ChatGPT, share the context (script + error), ask for an explanation and fix. AI is excellent at debugging simple cases. The fastest way to level up.

# Section 9: What's Next?

Congrats. You now have a skill that puts you ahead of 99 percent of regular AI users. You can automate your own tasks. But there is one more level: build a tool that other people can use too.

## Mode 2 Limits Alex Hits

- summarize.py only runs on Alex's laptop. Friends cannot use it
- Alex has to remember to open the terminal, run the script, read the output
- If there is a bug, you have to edit code. Non-programmer friends get stuck
- Output is plain markdown, no friendly UI

## Preview: Module 04 (Mode 3 Custom Bot)

In Module 04, Alex learns to build a personal Telegram bot: 'Study Buddy'. Features:

- Alex can chat 'summarize this paper' on Telegram, attach a PDF, get a reply with the summary
- The bot remembers Alex's conversation history (persistent memory)
- The bot reminds Alex of morning classes via push notifications
- Alex can share the bot with their study group, everyone uses it from their own phone
- The bot is online 24/7 on a cheap VPS (5 USD per month)

Module 04 covers: building a Telegram bot from scratch, integrating Calendar and Sheets, deploying to a cloud VPS, and multi-user support.

## Action Items Before You Continue

1. Reproduce summarize.py on your own laptop, run it against any 5 PDFs
2. Experiment by modifying the prompt: change style, target audience, output format
3. Think: what task do you do manually every day that Mode 2 could automate?
4. Write down 3 custom bot ideas you would want for your own life
5. Sign up for Telegram (if you haven't), get ready for Module 04

---

**Nicky Pandelaki**

AI Engineer & Creator

*Next: Module 04 (Mode 3: Custom Bot Deep Dive)*