𝟏 𝐌𝐚𝐫𝐤𝐝𝐨𝐰𝐧  
 Helps structure your prompts clearly for better AI understanding.  
 🧩 Example: Create a login function with username and password  
1.1 Markdown (Standard)  
1.2 XML  
1.3 JSON  
  
𝟐 𝐖𝐒𝐋 (Windows Subsystem for Linux)  
 A universal coding environment that allows developers to run Linux tools directly on Windows no need for dual booting! 🖥️  
  
𝟑 𝐀𝐈 𝐂𝐋𝐈  
 Command-line tools that integrate AI models into your workflow:  
 💻 Claude Code ($20/month)  
 💻 Codex CLI ($20/month)  
 💻 Gemini CLI (1000 requests free)  
 💻 QWEN CLI (2000 requests free)  
  
𝟒 𝐀𝐈 𝐂𝐋𝐈'𝐬 𝐌𝐂𝐏 𝐒𝐞𝐫𝐯𝐞𝐫  
 Connects AI CLIs with local tools making automation and model management seamless. 🤖  
  
𝟓 𝐓𝐞𝐬𝐭 𝐃𝐫𝐢𝐯𝐞𝐧 𝐃𝐞𝐯𝐞𝐥𝐨𝐩𝐦𝐞𝐧𝐭 𝐖𝐨𝐫𝐤𝐟𝐥𝐨𝐰  
 Write tests before writing code to ensure reliability and faster debugging. ✅  
  
𝟔 𝐒𝐩𝐞𝐜 𝐃𝐫𝐢𝐯𝐞𝐧 𝐃𝐞𝐯𝐞𝐥𝐨𝐩𝐦𝐞𝐧𝐭  
 Define clear business requirements before development — plan smart, build faster. 🧠  
  
𝟕 𝐃𝐞𝐩𝐥𝐨𝐲𝐦𝐞𝐧𝐭  
Bring your AI and apps to life in the cloud! 🌐  
  
𝐂𝐥𝐮𝐬𝐭𝐞𝐫: Group of computers  
🌍 Digital Ocean | ☁️ Azure  
7.1 Docker  
7.2 Kubernetes  
7.3 Dapr  
7.4 Ray (only for Linux)

What is Markdown in Prompt Engineering?

Markdown is a lightweight text formatting syntax that helps you make your prompts structured, readable, and clear for both *you* and the *AI model*.

When writing prompts for large language models (like GPT-5), Markdown is used to:

* Highlight important parts (like bold, italics, headings)
* Organize text (with bullet points, numbered lists, or tables)
* Separate sections (with dividers or code blocks)

This improves clarity, structure, and control in your prompt.

Why Markdown is Useful in Prompt Engineering

AI models (like GPT) read Markdown formatting easily.  
It helps the model differentiate between:

* Instruction
* Context
* Example Input/Output
* Desired format of response

That means Markdown = structured context engineering inside your prompt.

Prompt (with Markdown)

# Task

You are an expert data analyst.

## Input

A CSV file containing customer data with the columns: `Name`, `Age`, `City`, and `Purchases`.

## Goal

1. Summarize the data by city.

2. Find which city has the most purchases.

3. Present your result in a \*\*table\*\* format.

🤖 Model Output

| City | Total Purchases |
| --- | --- |
| Karachi | 320 |
| Lahore | 280 |
| Islamabad | 150 |

𝟐 𝐖𝐒𝐋 (Windows Subsystem for Linux)

is a compatibility layer developed by Microsoft that allows you to run a full Linux environment directly on Windows — without needing a virtual machine or dual-boot setup.

In Simple Terms

WSL lets you:

“Run Linux commands, tools, and applications *inside* Windows — as if Linux and Windows were working together.”

You can use bash, apt, python, gcc, git, node, and more — all inside Windows.

How It Works

* It provides a Linux kernel interface that runs on Windows.
* You can install Linux distributions (like Ubuntu, Debian, Kali, etc.) from the Microsoft Store.
* It runs in a lightweight virtual environment but shares the same file system and resources as Windows.

| Version | Description |
| --- | --- |
| WSL 1 | Translates Linux system calls into Windows system calls. Faster for small tasks but limited compatibility. |
| WSL 2 | Uses a real Linux kernel inside a lightweight virtual machine. Offers full system call compatibility and better performance for development. |

Common Uses

* Running Linux commands on Windows (bash, grep, ls, etc.)
* Using package managers like apt or yum
* Running Python, Node.js, or other dev environments in Linux mode
* Using Docker and containers (especially with WSL 2)
* Accessing both Windows and Linux files easily (/mnt/c/ for Windows drives)

Example: Installing Ubuntu via WSL

wsl --install -d Ubuntu

✅ You just installed and ran Python — *natively inside Windows via Linux!*

Key Benefit

You get the power of Linux (tools, scripts, servers)  
with the convenience of Windows (apps, UI, IDEs).

Concept Overview

Prompt Engineering involves designing, testing, and refining prompts to get the best output from AI models (like GPT).

When working on AI apps, chatbots, or agentic systems, developers often need Linux tools, Python environments, or open-source SDKs — all of which work more smoothly in Linux than Windows.

Goal

You want to test multiple prompts using Python scripts to evaluate LLM responses.

Without WSL

* You might face installation issues with Linux-only packages.
* Commands like pip, curl, or grep might behave differently.
* Managing virtual environments can get messy on Windows.

Why WSL Helps Prompt Engineers

| Feature | Benefit |
| --- | --- |
| 🐧 Linux environment | Works seamlessly with Python, shell scripts, and APIs |
| 📦 Package management | Use apt, pip, or conda without compatibility issues |
| 🔄 Automation | Run bash scripts to batch-test many prompts |
| 💬 Integration | Combine prompt tests with open-source tools (LangChain, LlamaIndex, etc.) |
| ⚡ Speed | Faster setup than using full virtual machines |

𝟑 𝐀𝐈 𝐂𝐋𝐈

🔹 What is AI CLI?

AI CLI stands for Artificial Intelligence Command Line Interface —  
a terminal-based tool that lets developers interact with AI models directly using commands, instead of using web dashboards or APIs manually.

Think of it as a prompting playground inside your terminal 💻 — where you can send prompts, test model behavior, and automate prompt workflows.

It’s like talking to ChatGPT, but through your command line (Terminal / PowerShell)

How It Helps in Prompt Engineering

Prompt Engineers often need to:

* Experiment with different prompt versions
* Compare model outputs quickly
* Automate prompt testing
* Integrate prompts into development pipelines

With an AI CLI, you can:

1. 🧪 Test prompts locally without opening a browser.
2. ⚙️ Automate prompt tests using scripts.
3. 🔁 Version-control prompts (using Git).
4. 📊 Benchmark outputs from different models.
5. 🚀 Deploy or share prompt templates easily.

| Concept | Description | Example |
| --- | --- | --- |
| AI CLI | Command-line tool for interacting with AI | ai chat "What is WSL?" |
| Use in Prompt Engineering | Test, refine, and automate prompts | ai prompt run prompt.md |
| Advantage | Faster experimentation, scriptable, reproducible | Automate testing prompts daily |
| Common Tools | OpenAI AI CLI, LangChain CLI, Ollama CLI | ollama run llama3 |

What is “AI CLI work” in Prompt Engineering?

AI CLI works as a bridge between you (the Prompt Engineer) and the AI model (like GPT-4o) —

allowing you to run, test, debug, and improve prompts directly from the command line.

It automates the process of sending prompts to models and getting outputs —  
so you can focus on refining the quality of prompts instead of manually copy-pasting them in a web interface.

It performs 4 main roles

1. Takes Your Prompt as Input

➡ The AI CLI reads that text prompt from your command line.  
➡ It can also load a prompt file (e.g., prompt.md or prompt.json) for complex experiments.

2. Sends It to the Model API

AI CLI connects to your chosen model endpoint (like OpenAI’s gpt-4o, Anthropic’s Claude, or a local LLM).  
It sends:

* Your prompt text
* Optional parameters (model, temperature, system message, etc.)

3. Receives and Displays the Model Output

The CLI then prints the model’s reply instantly in your terminal.  
This makes it easy to:

* Observe different outputs
* Compare prompt changes
* Log results into files for analysis

4. Supports Iteration and Automation

In Prompt Engineering, testing multiple versions of a prompt is key.  
AI CLI lets you automate and compare versions quickly:

Why AI CLI Is Powerful for Prompt Engineers

| Feature | Benefit |
| --- | --- |
| 🔤 Direct Prompt Input | Instantly test new ideas |
| 🧩 File + Template Support | Manage structured prompts (system + user) |
| 📊 Automation | Benchmark or A/B-test prompts |
| 🔒 Reproducibility | Same setup, same output |
| 🧰 Integration | Connects with version control, CI/CD, and notebooks |

AI CLI works as a fast testing engine for prompts.  
It automates the *prompt → model → response → evaluation* loop  
— the core cycle of Prompt Engineering.