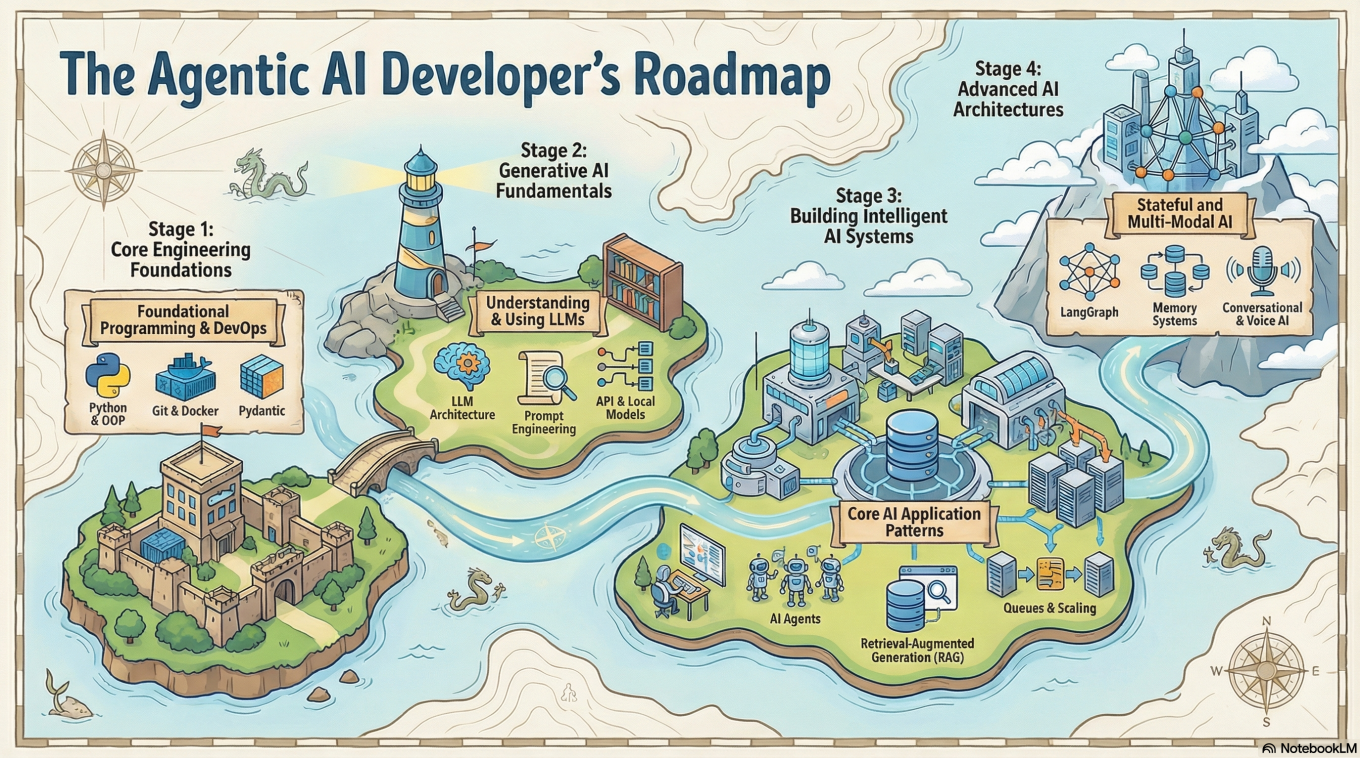
kintsugi-stack-generative-agentic-ai-python-fullstack

> "The problem with automation is that it only works until it doesn't." — Martin Fowler

- Author: Kintsugi-Programmer



> Disclaimer: The content presented here is a curated blend of my personal learning journey, experiences, open-source documentation, and invaluable knowledge gained from diverse sources. I do not claim sole ownership over all the material; this is a community-driven effort to learn, share, and grow together.

Table of Contents

---

1. Introduction

1.1. Installation of Tools (VSCode and Python)

```
winget install --id Microsoft.VisualStudioCode -e
winget install --id Git.Git -e
winget install --id Python.Python.3 -e
code --version
git --version
python --version
code --install-extension zhuangtongfa.Material-theme
code --install-extension GitHub.github-vscode-theme
code --install-extension sdras.night-owl
code --install-extension Equinusocio.vsc-material-theme
code --install-extension GitHub.github-vscode-theme
```

- **Repo Context**
    - This Section is the **first Section** of the **Agentic AI with Python** Repo.
    - The purpose of this Section is to **set up the developer environment** for building **agentic AI systems**.
    - The Section explains **what tools are required** before starting the Repo.
    - All tools discussed are:
        - **Free**
        - **Very easy to set up**
        - **Most likely already installed** on the learner's machine
- **Objective of This Section**
    - To explain the **development environment setup** needed for the Repo.
    - To introduce the **basic tools** required before writing any code.
    - This Section does **not** involve coding yet.
    - It prepares learners to follow upcoming tutorials smoothly.
- **Tool 1: IDE (Integrated Development Environment)**
    - An IDE is required to **write and manage code**.
    - The Author will use **Visual Studio Code (VS Code)**.
    - Learners are **free to use any IDE** they like.
    - However, it is **highly recommended** to use **Visual Studio Code** because:
        - It helps learners **follow the tutorial step by step**
        - Screens and instructions will match the Author's setup
- **Visual Studio Code Details**
    - Visual Studio Code is:

        - **Free**
        - Available for **Windows**
        - Available for **Linux**
        - Available for **other platforms**
    - Installation steps:
        - Click the **Download** button
        - Choose the appropriate platform
        - Follow on-screen instructions
    - VS Code will be used as the **primary IDE** in this Repo.
- **Tool 2: Python Programming Language**
    - Python **must be installed** on the machine.
    - The entire Repo is **based on Python**.
    - The Author **assumes basic Python knowledge**, not advanced expertise.
- **Required Python Knowledge (Basic Level)**
    - Understanding:
        - **Variables**
        - **How to define functions**
        - **How to create variables**
        - **Simple mathematical operations**
        - **Simple classes**
    - Being a **Python expert is not required**.
    - Basic familiarity is **sufficient to continue** with the Repo.
- **Python Installation Check (Example)**
    - On the Author's machine:
        - Python is already installed.
    - To check Python installation:
        - Open the terminal
        - Type:

```
python
```

        - Press **Enter**
    - Output shows:
        - Python version **3.1 / 3.2** installed

    - Conclusion:
        - Python must be installed before proceeding.

- **Python Installation Process**
    - Installation is **very simple**
    - Steps:
        - Click the **Download** button
        - Click **Next** multiple times
        - Follow on-screen instructions
    - Python will be installed successfully after setup.

- **Tool 3: AI and LLM Platforms**
    - This is an **Agentic AI Repo**, so AI platforms are required.
    - The Repo will use:
        - **OpenAI**
        - **Gemini**
        - **Other Large Language Models (LLMs)**

- **OpenAI and ChatGPT Requirement**
    - Learners should:
        - Be aware of **ChatGPT**
        - Have an **OpenAI account**
    - Do not worry if you do not have one yet.

- **Upcoming Dedicated Sections**
    - Separate Sections will cover:
        - How to **set up an OpenAI account**
        - How to **sign up on OpenAI**
        - How to **add credits** to OpenAI
        - How to **use Gemini**
        - How to work with **other LLM tools**

- **Summary of This Section**
    - This Section is a **basic setup overview**.
    - It explains:
        - IDE requirement
        - Python installation
        - AI platform awareness

- Assumption:
  - Most learners already have these tools installed.
- This setup enables learners to:
  - Follow upcoming Sections
  - Start building agentic AI systems smoothly

1.2. VS Code Setup (Extensions and Themes)

```
winget install --id Microsoft.VisualStudioCode -e ;

code --install-extension akshatshrivastava.ao-mirage ;
code --install-extension PKief.material-icon-theme ;
code --install-extension esbenp.prettier-vscode ;

code --install-extension ms-python.python ;
code --install-extension ms-python.vscode-pylance ;
code --install-extension ms-python.debugpy ;

code --install-extension ms-vscode-remote.remote-containers ;
code --install-extension ms-azuretools.vscode-docker ;
code --install-extension docker.docker
```

- **Topic: VS Code Setup, Theme, and Extensions (Optional Section)**
  - **Context**
    - One very interesting comment that I usually get on my Sections is:
      - What is my VS Code setup?
      - What theme am I using?
      - What extensions am I using?
    - Many viewers are curious about how my development environment looks and works.
  - **Purpose of This Section**
    - In this particular Section, we discuss:
      - All the tools I use
      - All the themes I use
      - All the extensions I use in my VS Code
    - This is shared for people who:
      - Are interested in my setup
      - Want to make their development environment look exactly like mine
      - Want to follow this Repo comfortably with the same setup

    - **Optional Nature of This Section**
        - This Section is **absolutely optional**
        - If you want, you can skip this Section
        - Skipping this Section will not affect learning the main Repo content

- **Overview of My VS Code Environment**
    - Here is my VS Code
    - These are a few extensions that I currently have installed
    - My primary work area:
        - I work **a lot on TypeScript**
        - So I have TypeScript-related extensions installed

- **Installed Extensions (General)**
    - I have:
        - Some **.NET extensions** installed
        - Some **container-related tools** for Docker work

- **Theme Used**
    - The theme I use is:
        - **AO Mirage**
    - More specifically:
        - **AO Mirage Dark Border Theme**
    - This theme:
        - Is the **overall theme**
        - Will be used **throughout the entire Repo**

- **Docker and Container Tools**
    - I have the following container-related extensions:
        - **Dev Containers**
        - **Docker**
        - **Docker DS**
    - These are used for:
        - Working with Docker
        - Container-based development workflows

- **Icon Theme**
    - The icon theme I use is:

        - **Material Icon Theme**

    - This controls:

        - Folder icons
        - File icons
        - Overall file explorer visuals

- **Code Formatting Tool**

    - **Prettier**

        - Prettier is a very useful extension

        - It is used to:

            - Format code
            - Keep code clean and consistent
            - Automatically manage spacing and structure

- **Python Development Extensions**

    - **Pylance**

        - This is a **main extension**

        - It is created by **Microsoft**

        - I use this extension **a lot**

        - Highly recommended

        - Helps with:

            - Python intelligence
            - Code analysis
            - Better development experience

    - **Python Language Extension**

        - The official **Python extension for VS Code**
        - Also by **Microsoft**
        - Highly recommended to install
        - Essential for Python development

    - **Python Debugger**

        - Python Debugger extension by **Microsoft**

        - I do not use this extension a lot

        - Still:

            - It is good to have

        - Useful for debugging Python applications

- **Summary of Required Extensions for Python Developers**
    - These are the **major extensions** you need as a Python developer:
        - Python
        - Pylance
        - Python Debugger
        - Prettier
        - Material Icon Theme
    - No additional extensions are strictly required
- **Final Confirmation**
    - The theme used is:
        - **AO Mirage**
        - **AO Mirage Dark Border Theme**
    - This setup will remain consistent throughout the Repo
- **Transition to Core Repo Content**
    - From the next Section onwards:
        - We will jump into the **real workings of LLMs**
        - We will **deep dive into Agentic AI**
    - This marks the start of:
        - Practical learning
        - Core AI concepts
        - Hands-on implementations

2. Breaking Into Coding World with Python

2.1. KintsugiStack

- **KintsugiStack**
    - The KintsugiStack repository, authored by Kintsugi-Programmer, is less a comprehensive resource and more an Artifact of Continuous Research and Deep Inquiry into Computer Science and Software Engineering. It serves as a transparent ledger of the author's relentless pursuit of mastery, from the foundational algorithms to modern full-stack implementation.
- **Teaching Approaches Used**
    - **First Principles Learning**
        - Go deep into frameworks and libraries
        - Explore every detail

        - Understand there is **no magic**
        - Build features from the ground up
        - Learn how things actually work internally
    - **Investigative / Explorative Learning**
        - Learn by exploring and experimenting
        - Investigate while writing code
        - No official documentation exists for this term
        - Termed as **Investigative Learning**
        - Works especially well for:
            - Programming languages
            - Ground-level technical understanding
- **Teaching Style in This Repo**
    - Repo focuses mainly on **Investigative Learning**
    - Every output will be questioned
    - Every line of code will be questioned
    - Learning is active, not passive
    - The Repo aims to be:
        - One of the most amazing Python classes
    - Repo preparation took **months**
    - Notes were prepared in extreme detail
    - Teaching is now the main focus of life
    - No rush, no pressure
    - Teaching is done calmly and thoughtfully
- **Repo Philosophy**
    - The goal is to truly understand:
        - What it takes to **master Python**
    - Learning is not rushed
    - Learning is deep and permanent
    - This Repo is **exceptional**
    - This is not a quick tutorial

    - This is a Repo meant to be:
        - Completed fully
        - Committed to completely

- **Lecture Style**
    - Teaching is not robotic
    - Teaching includes:
        - Ups and downs in voice
        - Storytelling
    - Lectures are:
        - Laid back
        - Not fast-paced
    - Learners are encouraged **not to rush**
    - Speeding Sections to 1.5x or 2x is discouraged
    - Reason:
        - The brain needs time to process information
        - Information must move to **secondary storage**
        - This helps with **permanent memory**
    - Learning is a **marathon**, not a race
    - Once Python is learned properly:
        - You never need to look back

- **Closing Message**
    - This introduction is enough to begin
    - Learners are invited to start the Python journey
    - Confidence is emphasized
    - The journey ahead is exciting
    - The Repo is designed for true mastery

2.2. What is Programming ?

```
FUNCTION makeChai

    IF kettle does NOT have water THEN
        fill kettle
    END IF

    plug in kettle
    boil water
```

```
    IF cup is NOT clean THEN
        wash cup
    END IF

    add tea leaves to cup
    add sugar to cup

    pour boiled water into cup
    stir cup

    serve chai

END FUNCTION

CALL makeChai
```

- **Teaching Style and Screen Modes**
    - Usually, the Author appears on screen
    - Sometimes switches to **full screen mode**
        - Purpose:
            - 100% focus on the screen
            - No distractions
    - When engagement is needed:
        - Author switches back to camera view
    - Majority of the time:
        - Students should focus only on the screen
        - Full attention is expected on what is shown
- **Primary Code Editor**
    - **VS Code (Visual Studio Code)** is used
    - Students are asked to:
        - Download VS Code
    - Installation will be covered step by step
        - VS Code
        - Python
        - Everything required
- **Diagram and Drawing Tools**
    - **tldraw**

        - Used to draw diagrams
        - Helps people understand concepts better
    - **Excalidraw**
        - Used extensively
        - Especially in YouTube Sections
        - Author has a strong habit of:
            - Drawing diagrams
            - Visual explanations
    - **Repo Preference**
        - For Repos:
            - **Eraser is preferred**
        - Reasons:
            - Easier to work with
            - More interesting
            - Smoother teaching experience
- **Transition to Learning**
    - This overview is complete
    - Author does not want to waste time
    - Repo will start directly
    - Designed even for:
        - People who have **never coded**
        - People who have **never programmed**
- **Fundamental Question**
    - **What is programming?**
    - Programming is:
        - Giving instructions to a computer
    - Most important aspect:
        - Instructions must be in a form the computer understands
- **How Computers Work**
    - Computers:

        - Cannot think on their own
    - About AI:
        - AI does not actually think
        - AI:
            - Repeats patterns seen on the internet
            - Is essentially **fancy word completion**
        - AI is good at this, but:
            - Still cannot think independently
    - Computers and AI:
        - Require **exact instructions**
        - Only follow what is explicitly written
- **Analogy: Making Chai (Tea)**
    - Programming is explained using an analogy
    - Example:
        - Creating a program to make **chai (tea)**
    - Narrative approach:
        - A chai store or cafe
        - All exercises in the Repo relate to chai
        - Chai is a favorite example
- **Challenge**
    - Teaching a computer how to make chai is difficult
    - But the Author will try to give clear instructions
- **Step 1: Gathering Items (Data Collection)**
    - Gather water
    - Gather milk (if milk tea is preferred)
    - Gather tea leaves
    - Gather sugar (if sugar tea is preferred)
    - Gather utensils
    - This stage represents:
        - **Collection of data**

    - This step:
        - Has no exception
        - Must be done first

- **Step 2: Conditions**
    - After gathering, work is not done
    - Conditions must be checked
    - Examples:
        - Enough water
        - Clean cups
    - Programming involves:
        - Multiple moving parts
        - Just like making chai

- **Step 3: Steps (Process)**
    - After:
        - Data collection
        - Condition checking
    - Precise steps must be given
    - These three components are:
        - Exactly what the Repo will teach

- **Three Core Programming Components**
    - Gathering data
    - Checking conditions
    - Writing steps (logic)

- **Is Coding Hard?**
    - Honest answer:
        - **Yes and no**
    - Coding is:
        - Not a walk in the park
        - Not extremely easy
    - Common myth:
        - “Coding is easy”

    - Reality:
        - Requires years of effort
        - Comfort may come in months
        - True mastery takes **a couple of years**

- **Why Programmers Are Paid Well**
    - Programming is not easy
    - Requires thinking and problem-solving
    - High demand skill

- **Encouragement**
    - Coding is:
        - Hard
        - But **doable**
    - Languages like:
        - **Python**
        - **JavaScript**
    - These languages are:
        - Easier to learn
        - Beginner friendly

- **Python Specifically**
    - Feels almost like English
    - Easy to start
    - Still has:
        - Nuances
        - Complexities
    - Realistic expectation is important

- **Learning Timeline**
    - Writing simple code:
        - Can be done in a few months
    - Mastery:
        - Takes time
    - Coding is more about:
        - Thinking

        - Breaking down problems
    - Writing code:
        - Is the easy part
    - Thinking like a programmer:
        - Is the hardest part
        - Where real magic happens
- **Second Example: Detailed Tea Steps**
    - Step 1:
        - Check if kettle has water
    - Step 2:
        - Plug in kettle
        - Boil water
    - Step 3:
        - Get clean cup or cups
    - Step 4:
        - Add tea leaves for color
        - Add sugar (choice based)
    - Step 5:
        - Pour boiled water into cup
        - Add milk if preferred
    - Step 6:
        - Stir and serve
- **Important Notes About Steps**
    - Number of steps does not matter
    - Everyone’s steps are different
    - That is why:
        - Programming code differs for everyone
- **Key Takeaway**
    - Making chai was converted into:
        - A simple logical process

    - Demonstrates:
        - Programming is approachable
        - Logic comes before code

- **Diagram Summary**
    - Gathering items
    - Checking conditions
    - Writing steps
    - All steps are clearly laid out

- **What's Next**
    - Next Section:
        - Convert these steps into code
    - It will:
        - Mimic Python
        - Not be fully accurate Python
    - Purpose:
        - Show Python is not hard
        - Build confidence

- **Conclusion**
    - Python is:
        - Easy to work with
        - Beginner friendly
    - Moving on to the next Section

2.3. Convert that into Python Code

```
def fill_kettle():
    pass


def plug_in_kettle():
    pass


def boil_water():
    pass


def wash_cup():
    pass
```

```
def add_to_cup(item):
    pass


def pour(item, into):
    pass


def stir(container):
    pass


def serve_chai():
    pass


def make_chai():

    kettle_has_water = False
    cup_is_clean = True

    if not kettle_has_water:
        fill_kettle()

    plug_in_kettle()
    boil_water()

    if not cup_is_clean:
        wash_cup()

    add_to_cup("tea leaves")
    add_to_cup("sugar")

    pour("boiled water", into="cup")
    stir("cup")

    serve_chai()


make_chai()
```

- **Purpose of this section**
    - The goal is to **convert real-world steps into Python-like code**
    - The code is **not perfect Python**
    - The goal is to **get an idea of what coding in Python feels like**
    - This is about **experience**, not correctness
- **Important note for learners**

- You **do not need to follow along**
- You **do not need to write this code**
- This part is only to **experience how Python code looks**
- Even this code **will not actually work**
- It is just a **brief overview and explanation**

- **Opening the code editor (VS Code)**
  - The instructor goes into the **code editor**
  - This is **Visual Studio Code**
  - The editor was opened fresh to show the process
- **Creating and opening a project folder**
  - An **empty folder** is created
  - Location:
    - Inside an **English subfolder**
    - Folder name: **Python Udemy**
  - This folder is where **all course code will be written**
  - The folder is **dragged and dropped into VS Code**
  - That is all that is needed to open a project

- **Fresh VS Code look**
  - This is how VS Code looks when opened fresh
  - Multiple things may already be installed
  - Only the relevant parts are shown

- **Python extensions in VS Code**
  - Search for:
    - **Python**
  - Install the **Python extension**
    - Provides:
      - Color coding
      - Type hinting
  - Install **Pylance**
    - Provides:
      - Color coding

        - Type hinting
    - These extensions:
        - Help write code faster
        - Reduce manual typing
        - Assist while writing code

- **Creating folders and files**
    - Click the **top icons** in VS Code
        - Create folder
        - Create file
- **Folder creation**
    - A folder is created to mimic chapters
    - Folder name:
        - **00_Python**
    - This represents **Chapter 00**
- **File creation**
    - Inside the folder, create a new file
    - File name:
        - **non_python_code.py**
    - Reason:
        - It is **not real Python**
        - But the `.py` extension helps experience Python
    - Extension is important to:
        - Get syntax highlighting
        - Feel like Python coding

- **Instruction to learners**
    - This part is only for **experience**
    - Do not follow along
    - Do not worry about correctness
    - Just observe and understand

- **Concept: Function**

- In programming, instructions are wrapped in a **function**
- A function is like a **box**
- It contains instructions inside it
- This helps organize steps

- **Creating a function**
  - Function name:
    - **makechai**
  - Syntax elements:
    - Parentheses `()`
    - Colon `:`
  - You do not need to memorize syntax
  - You will learn it naturally over time

```
def makechai():
```

- **Indentation in Python**
  - After pressing Enter:
    - Python automatically adds **four spaces**
  - Python works on **indentation**
  - Removing indentation is **bad**
  - Code will not work without indentation
  - Best practice:
    - Use **four spaces**
    - Do not use tab
  - Internally:
    - Tab also equals four spaces
  - Details will be explained later

- **First condition**
  - Check if the kettle has water

```

```

```
    if not kettle_has_water:
        fill_kettle()
```

- **Explanation**
  - Reads like English:
    - If kettle does not have water
    - Then fill the kettle
  - This is based on **condition**
  - You can understand this without knowing Python

- **Plugging in the kettle**
  - Another function is used
  - Someone else does the work
  - Actual steps might include:
    - Pulling the cord
    - Moving nearby
    - Switching on
    - Plugging in
  - But those details are abstracted

```
    plug_in_kettle()
```

- **Boiling water**
  - Another function
  - Purpose:
    - Boil the water

```
    boil_water()
```

- **Checking if cup is clean**
  - Condition:

        - Is the cup clean?
    - If not clean:
        - Wash the cup
        - Or pick another cup
    - Both are valid in real life

```
if not is_cup_clean():
    wash_cup()
```

- **Adding ingredients to cup**
    - Use the same method repeatedly
    - First ingredient:
        - Tea leaves
    - Second ingredient:
        - Sugar

```
add_to_cup("tea leaves")
add_to_cup("sugar")
```

- **Pouring boiled water**
    - A pouring method is used
    - Specify:
        - What to pour
        - Where to pour

```
pour("boiled water", into="cup")
```

- **Stirring**
    - Stir inside the cup

```
stir("cup")
```

- **Serving chai**
    - Final step

```
    serve_chai()
```

- **Calling the function**
    - Functions must be called to run
    - Move outside the function
    - Call the function

```
makechai()
```

- **Understanding the code**
    - You may not understand how to write it
    - But you can understand:
        - 30%
        - 60%
        - Or even more
    - That is enough

- **Key realization**
    - Python code looks like **English**
    - Example:
        - `if not kettle_has_water`
    - Python is easy to read
    - That is why people say:
        - Python is easy to learn
    - If you can read English:
        - You can read Python

- **Important clarification**

    - This is **not accurate Python**
    - 100% acknowledged
    - But it is:
        - Fairly understandable
        - Conceptually correct
    - Many concepts already introduced:
        - Functions
        - Conditions
        - Methods
        - Strings
        - Flow of logic

- **Core programming idea**
    - Programming means:
        - Take steps
        - Convert steps into code
    - Throughout the course:
        - Steps will be designed first
        - Then converted into Python code

- **Confidence goal**
    - Even **20% confidence** is enough
    - If you feel:
        - “Yes, I can do this”
    - That is the goal

- **Transition**
    - Next video will focus on:
        - Pythonic concepts
    - Move on to the next video

Notes Formatting Nested Lines

Create super depth notes in Markdown (.md) format with 100% information preserved, no loss. Use simple grammar and keep everything clear, direct, and well-structured. using headings, subheadings,paragraphs,

statements and code blocks when needed. Include every detail, definition, example, and step exactly from the source. transform the given content into clean, readable .md format. and no #, just nested - lines plaintext, add bold wherever necessary

---

Video -> Section Course -> Repo Instructor -> Author

End-of-File

The KintsugiStack repository, authored by Kintsugi-Programmer, is less a comprehensive resource and more an Artifact of Continuous Research and Deep Inquiry into Computer Science and Software Engineering. It serves as a transparent ledger of the author's relentless pursuit of mastery, from the foundational algorithms to modern full-stack implementation.

> Made with ♡ Kintsugi-Programmer