Steps to Build your AI Project

Step 1: Ideation and Conceptualization
Define the project goal, such as building A **python Tutor bot**

Step 2: Test in OpenAI Playground
Experiment and validate the project's functionality using the OpenAI Playground.

Step 3: First Code Version
Converting OpenAI API Playground output into an executable Python script.

Create a basic API call using the OpenAI Python library with essential parameters.

Step 4: Enhancement with User Input
Enable dynamic user input via Python's input() function for interactivity.

Step 5: Integration with Gradio
Implement a Gradio-based UI for user-friendly web interaction.

Step 6: Customization for Local Use
Secure API keys using a .env file and integrate with VS Code for local deployment.

- **Step 7:** Environment Setup

Prepare and install required libraries (e.g., OpenAI, Gradio, python-dotenv).

- **Step 8:** Create the .env File Securely store the API key.
- **Step 9:** Run the Script

Execute the finalized script in the local environment.

- **Step 10:** Test Gradio UI

Validate the interface by interacting with the bot through the browser.

- **Step 11:** Deploy on Hugging Face Spaces

Prepare files and deploy the application to make it publicly accessible.

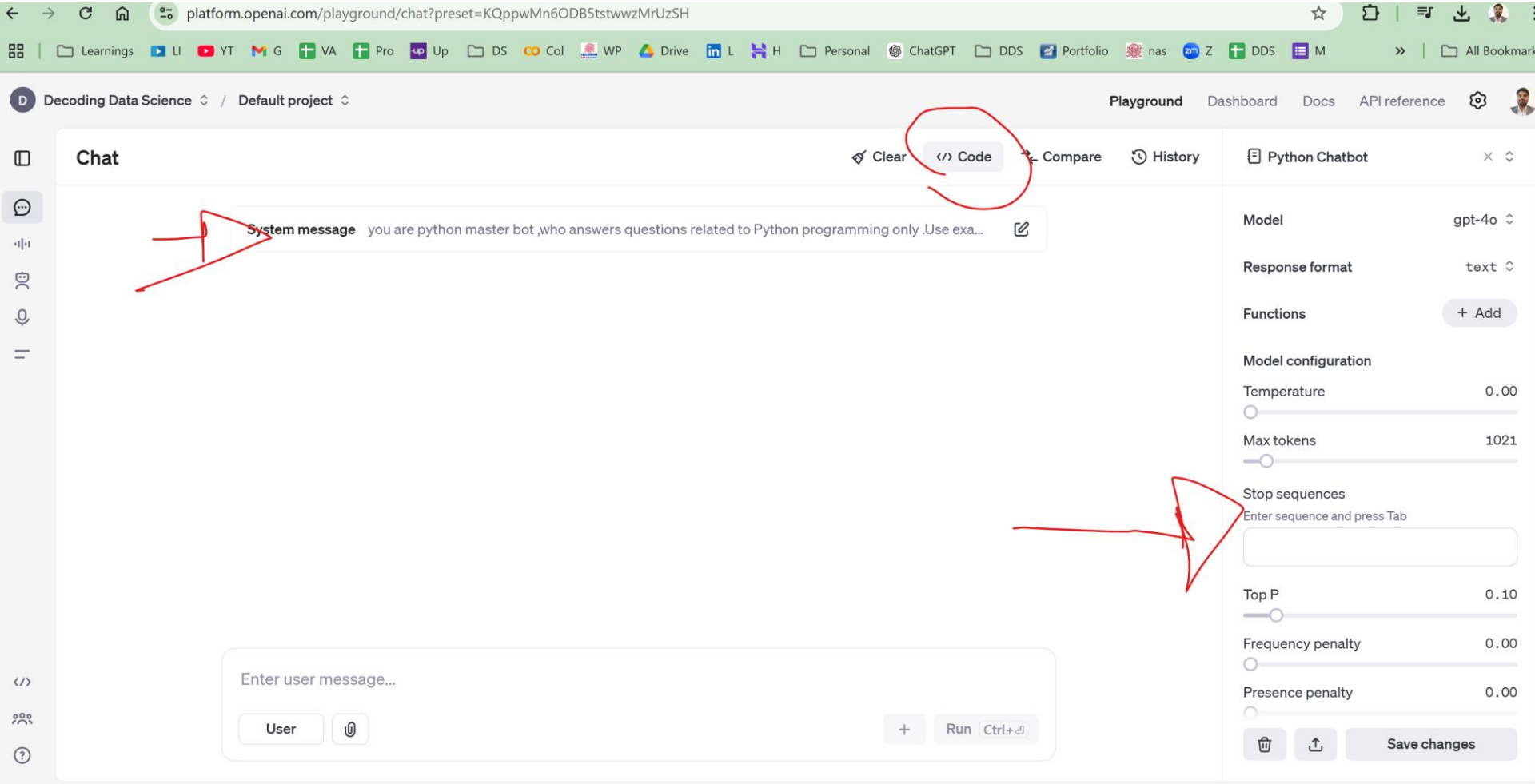
- **Step 12:** Outcome

Achieve a fully functional Python tutoring bot accessible locally and online.

Step 1 : Ideation and Conceptualization

Define the project goal, such as building A python Tutor bot

Step2 : Test your Project in OPENAI Playground



Step 3: First Code Version

You wanted to convert a sample OpenAI API Playground output into a Python script that can be executed locally.

Created a basic API call using the OpenAI Python library (openai.ChatCompletion.create). Included parameters like model='gpt-4', temperature, and a system prompt for a Python tutoring bot.

Step 4: Enhancement - User Input

Updated the code to accept dynamic user input using Python's input() function for interactive chatbot functionality.

Step 5: Integration with Gradio

Converted the code into a Gradio-based UI for interaction via a web interface. Simplified the interface to exclude chat history.

Step 6: Customization for Local Use in VS Code

Adjusted the script to use an .env file for securely storing the OpenAI API key and utilized python-dotenv to load the key.

Step 7: Setting Up the Environment

Making the requirement file

Installed required libraries using pip: openai, gradio, and python-dotenv.

Step 8: Created the .env File

Added the OpenAI API key to a .env file for secure access: OPENAI_API_KEY=your_open ai_api_key.

Step 9: Saved and Ran the Script

Saved the final script (python_tutor_bot.py) in the same directory as the .env file and ran it locally via VS Code.

Step 10: Tested the Gradio UI

Opened the auto-launched Gradio web interface in a browser and interacted with the Python tutoring bot.

Step 11: Deployed on Hugging Face Spaces

Prepared files (app.py and requirements.txt), created a Hugging Face Space, set the API key in Secrets, and pushed the files to deploy the app.

Step 12:Outcome

A fully functional Python tutoring bot runs locally via VS Code with Gradio UI and is deployed on Hugging Face Spaces, accessible via a public URL.

This can be ideal final project for submission.