**Speech to Insights: Telemedicine Conversation Analysis (A Beginner's Guide)**

Welcome! This guide will walk you, step-by-step, through a fun and practical coding project. We'll build a program that can "listen" to a telemedicine conversation and figure out the sentiment. Don't worry if you've never written a line of code before – we'll explain everything as we go. This guide will teach you how to install and setup all the required tools, how to use LLM (Large Language Models such as ChatGPT or Gemini) to help you write code using simple English prompts. We will then show you how to run the code, and how to setup version control for your code using git (sort of like a Time Machine for your code, so if something breaks, you can revert back to an older version). We will also cover debugging in this document.

**Part 0: Your Toolkit Explained**

Before we start building, let's get familiar with our tools. Think of this as looking at the instructions before you build a LEGO set.

* **Programming Language (Python):** Imagine you're writing a recipe. You list ingredients and steps in a specific order. **Python** is our recipe language—it's a set of instructions we give the computer. It's known for being readable and beginner-friendly.
* **Code Editor (VS Code):** This is your special notebook for writing code. We'll use **VS Code** because it's like a smart notebook that color-codes your instructions, helps catch typos, and keeps your project files organized.
* **The Terminal:** This might look like a throwback to old computers, but it's the most direct way to give your computer commands. Instead of clicking icons, you type simple commands. We'll tell you exactly what to type.
* **AI Coding Buddy (LLM):** A Large Language Model (like ChatGPT, Gemini, DeepSeek, Llama, Grok, etc) is an AI you can chat with. It understands plain English and can write code, explain concepts, and help you fix errors.

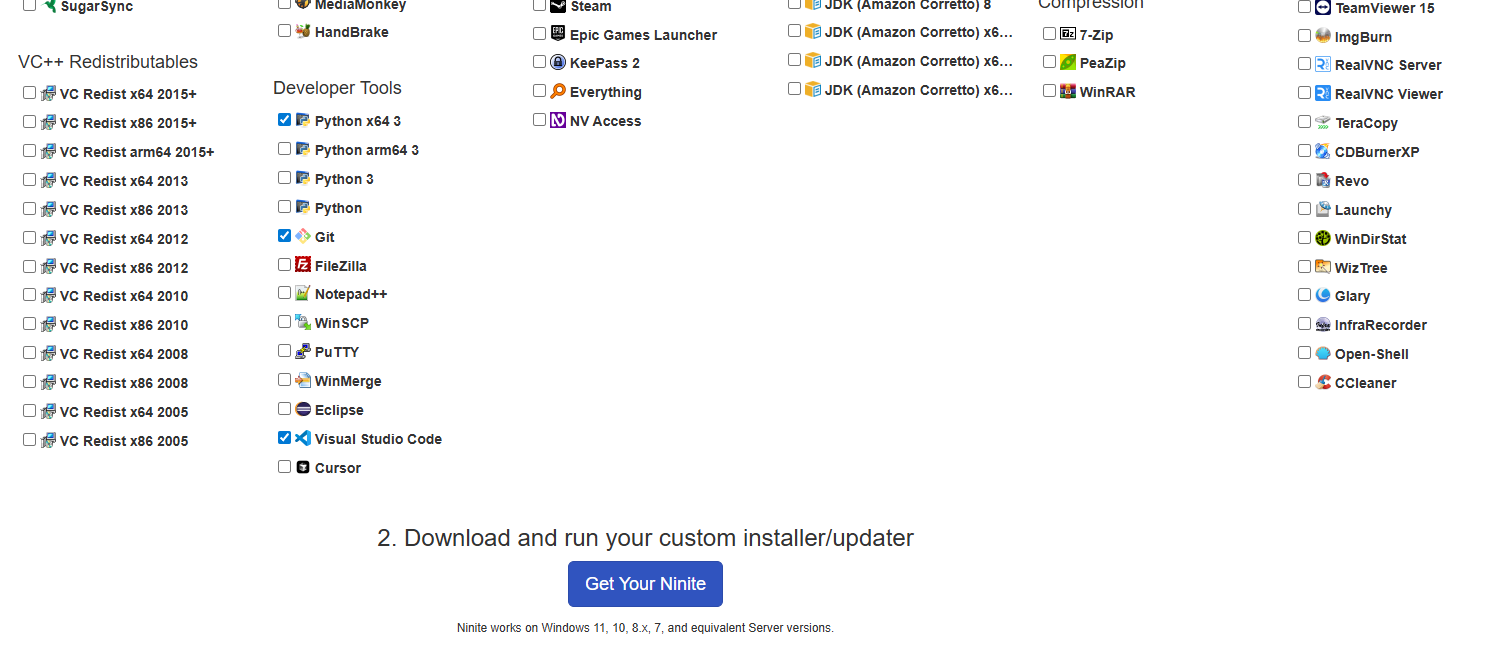
**Part 1: Setting Up Your Workshop**

First, we need to install the necessary tools and set up our project folder.

**Step 1.1: Installing the Essentials**

You'll need three key pieces of software, VS Code, Python and Git.

* **For Windows Users:**
  1. Go to [**ninite.com**](https://ninite.com/).
  2. Check the boxes for **VS Code**, **Python**, and **Git**.
  3. Click "Get Your Ninite" and run the downloaded installer. It will handle everything automatically.



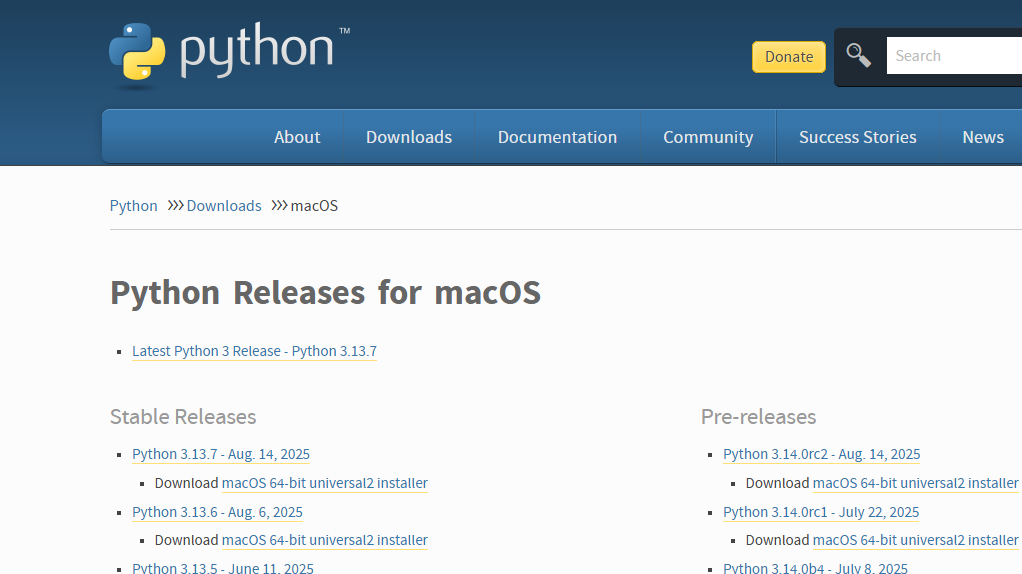
[Screenshot: Ninite website with VS Code, Python, and Git selected]

* **For Mac Users:**
  1. **VS Code:** Download from <https://code.visualstudio.com/>.

****

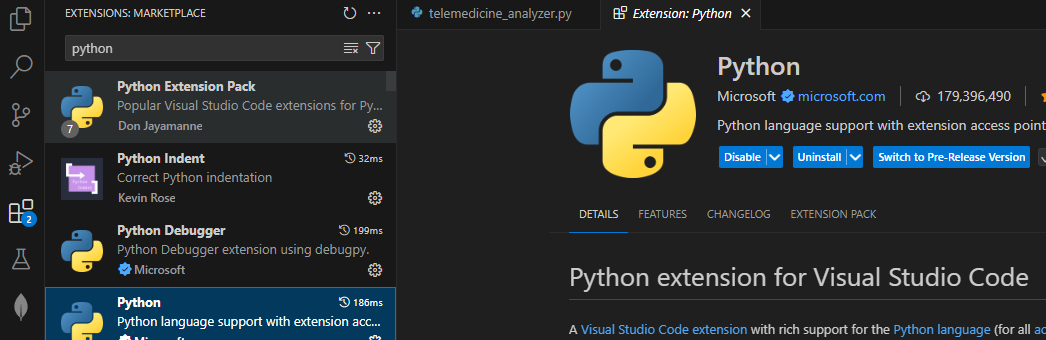
[Screenshot: VS Code download page for Mac]

* 1. **Python:** Download from <https://www.python.org/downloads/>.



[Screenshot: Python download page for Mac]

* 1. **Git:** Open your Terminal app and type git --version. If it's not installed, it will prompt you to install Apple's Command Line Tools. Follow the instructions.



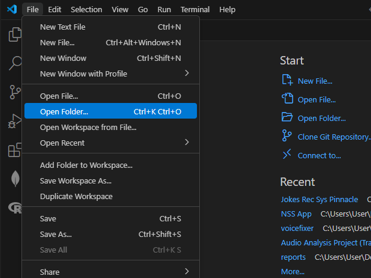


You will also require the Python extension from the VS Code Extension Marketplace.

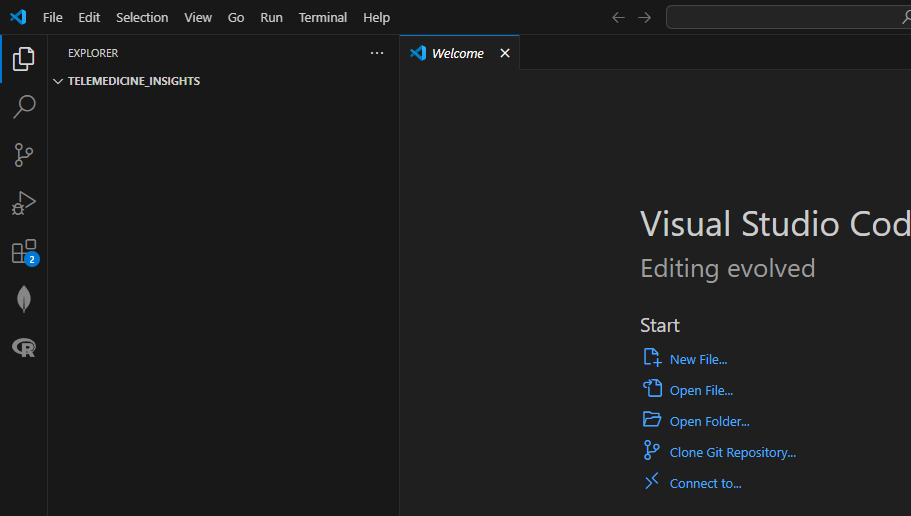
**Step 1.2: Creating Your Project Space in VS Code**

Now, let's create a dedicated, clean workspace for our project.

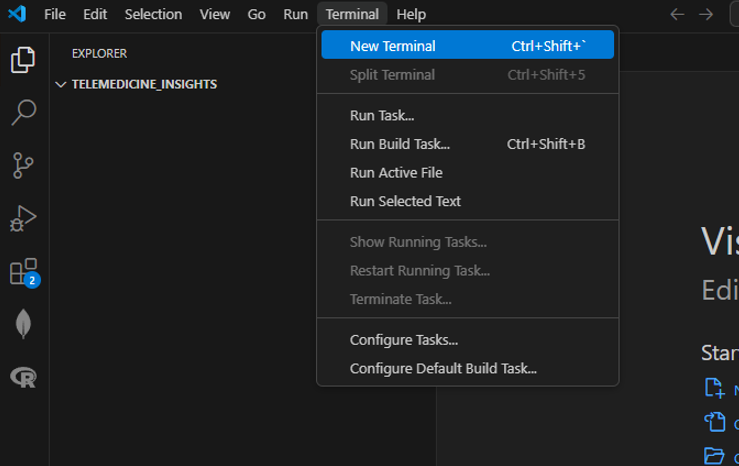
1. **Create a Project Folder:** Create a folder on your computer where you want to keep your project. Let's call it telemedicine\_insights.
2. **Open in VS Code:** Open VS Code, go to File > Open Folder, and select the telemedicine\_insights folder you just created.



[Screenshot: VS Code interface showing the 'File > Open Folder' menu option]



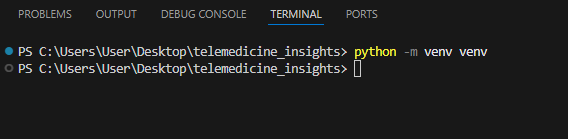
1. **Open the VS Code Terminal:** At the top menu, click Terminal > New Terminal. A command-line panel will appear at the bottom of the screen.



[Screenshot: VS Code interface with an arrow pointing to the 'Terminal > New Terminal' option]

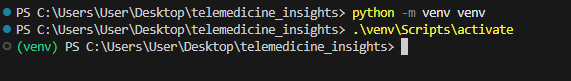
1. **Create a Virtual Environment:** This is like creating a separate, clean toolbox just for this one project. In the terminal, type this command and press Enter:

python -m venv venv



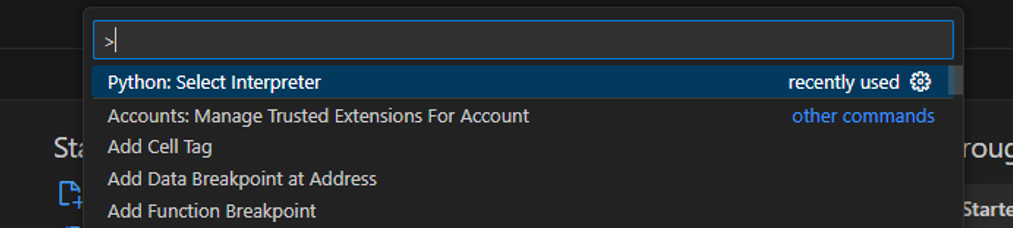
[Screenshot: VS Code terminal after running the 'venv' command]

1. **Activate the Environment:** Now, let's "open" our special toolbox. In the terminal, type the command for your system:
   * **On Windows:** .\venv\Scripts\activate
   * **On macOS/Linux:** source venv/bin/activate You'll know it's working because you'll see **(venv)** at the start of your terminal prompt.

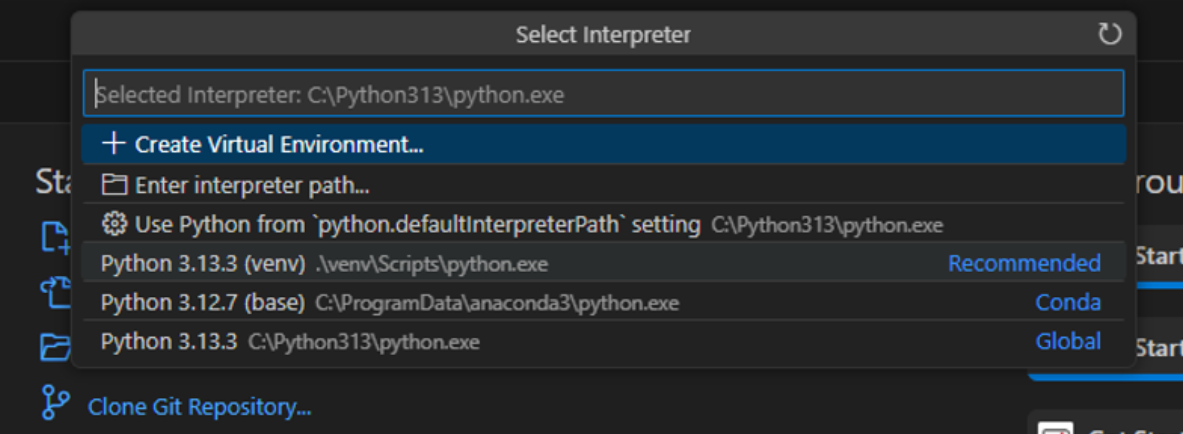


[Screenshot: VS Code terminal showing the (venv) prefix after activation]

1. **Select the Interpreter:** Tell VS Code to use the Python from your new toolbox. Press Ctrl+Shift+P, type "**Python: Select Interpreter**", and choose the option that includes ('venv').







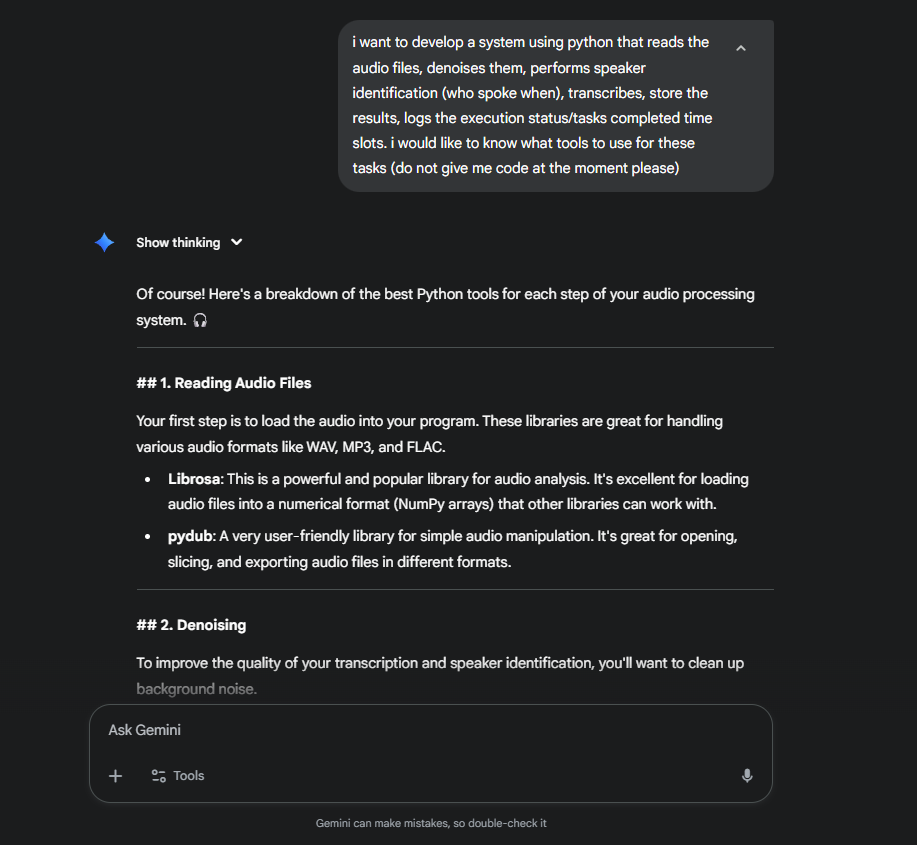


[Screenshot: VS Code Command Palette showing the selection of the venv interpreter]

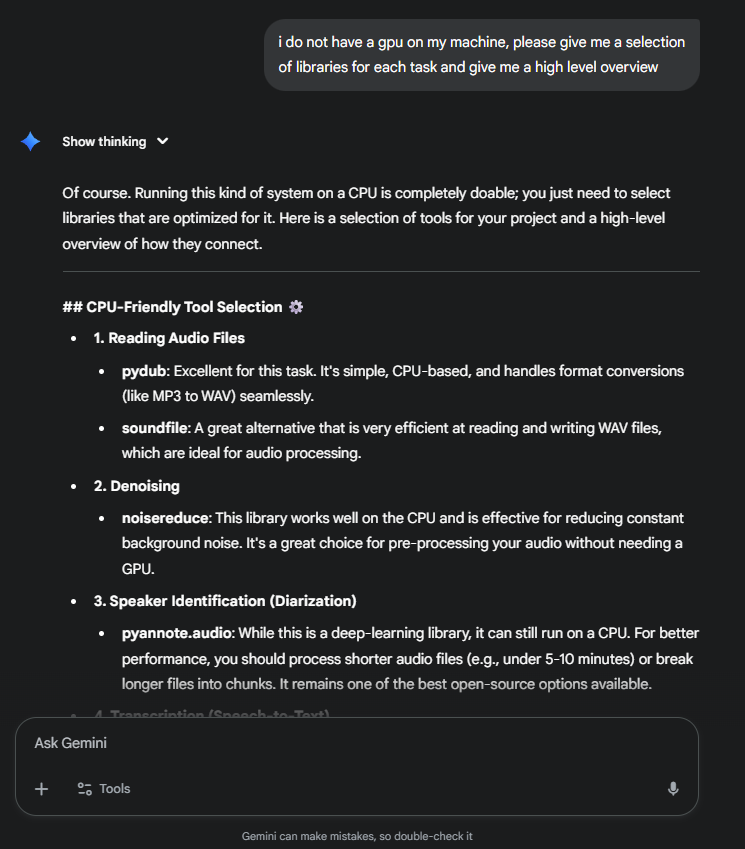
**Part 2: Planning with Your AI Co-Pilot**

A good plan is the key to any successful project. Let's use our AI buddy to create a roadmap.

Go to your favorite LLM (Gemini, ChatGPT, etc.) and describe your goal in plain English. This is your first prompt!

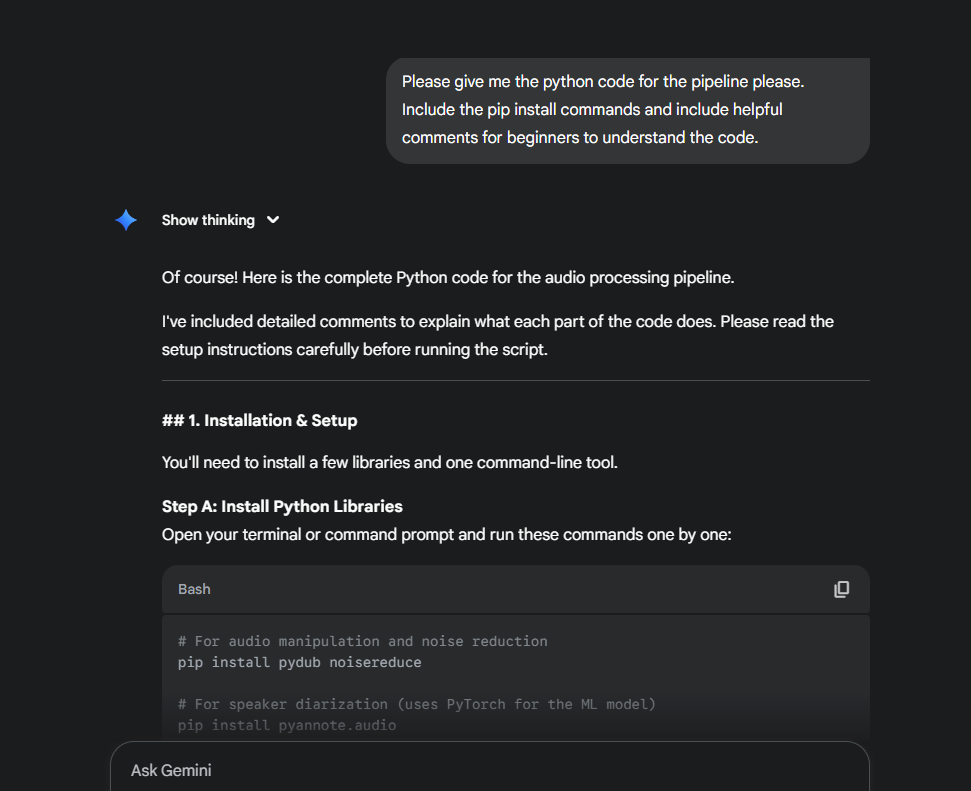


The AI will act as your technical consultant, breaking down the problem and suggesting the right tools (libraries) for each step. You can continue interacting with it to plan out the project.



**Part 3: Generating and Setting Up the Code**

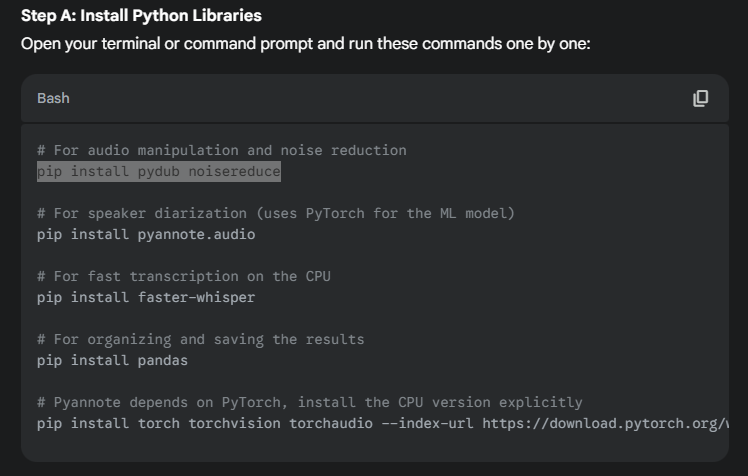
Now that we have a plan, let's ask the AI to write the code and tell us what tools to install.

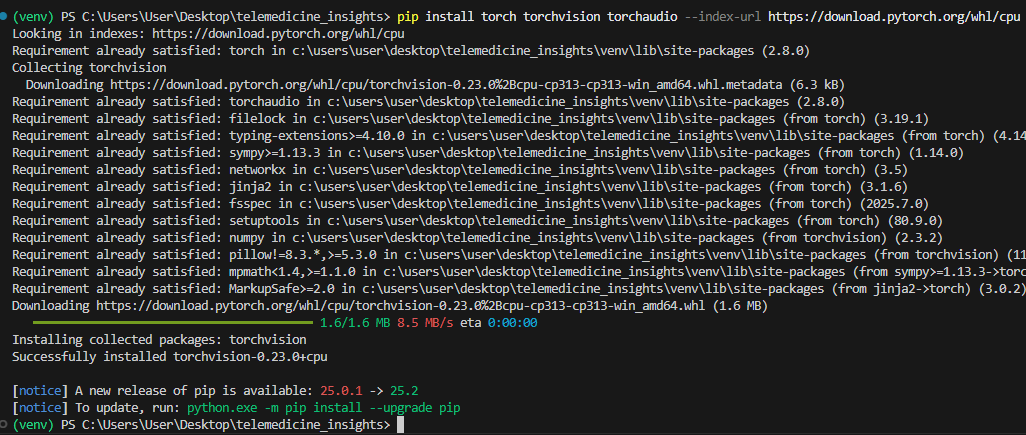


[Screenshot: LLM chat showing the AI's response with both the pip command and the Python code block]

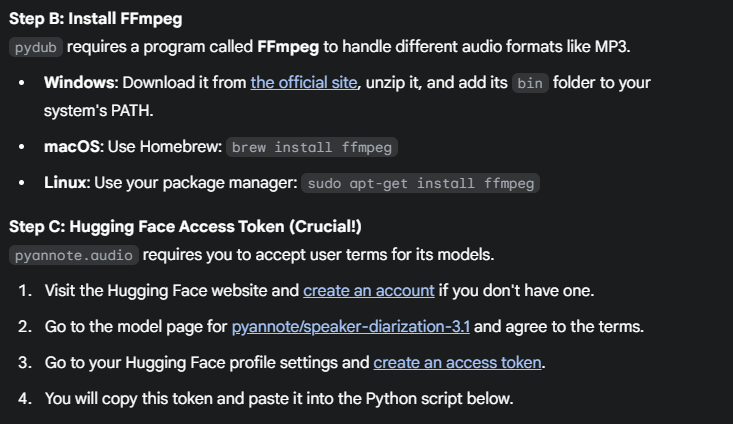
**Step 3.1: Installing the Tools (Libraries)**

Copy the pip install ... command from the AI's response. In your VS Code terminal (make sure it still says (venv)), paste the command and press Enter. This will download and install all the necessary toolboxes for your project.

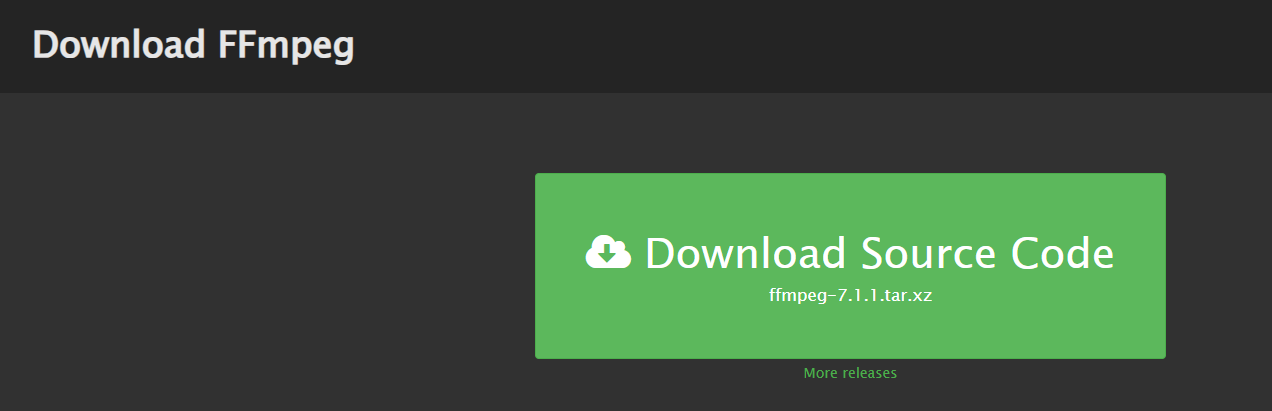




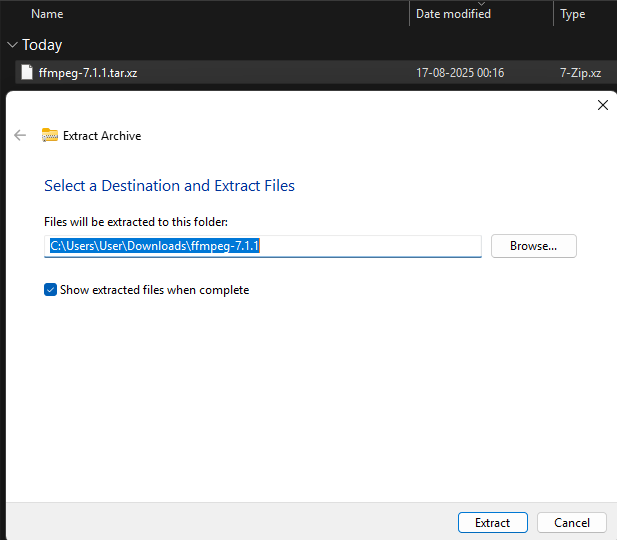
In this manner, copy and paste each of the pip install … commands and execute them individually. If you encounter any errors, copy and paste the output back into the LLM and ask for help.



Use the provided links for any additional programs needed. In this case we require FFMPEG and a Hugging Face Access Token, along with agreeing to the terms for the pyannote.speaker-diarization-3.1.



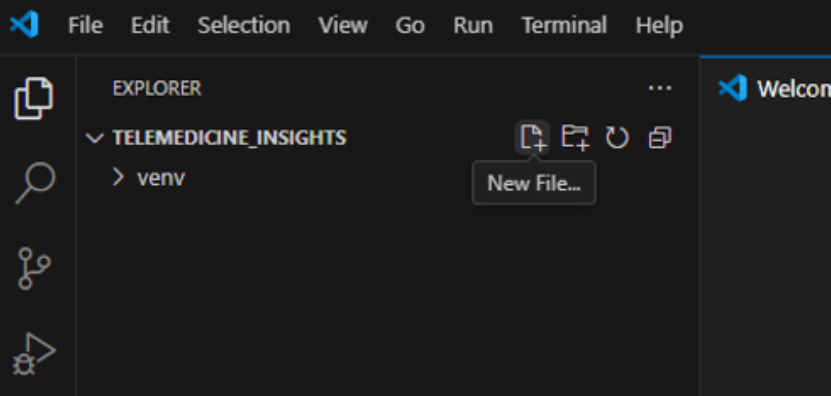
Downloading FFMPEG



Once downloaded we can extract the archive by right clicking and selecting ‘Extract All’

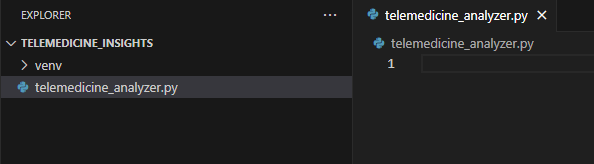
**Step 3.2: Saving the Code**

Copy the Python code block from the AI's response. In VS Code, create a new file named telemedicine\_analyzer.py and paste the code into it.

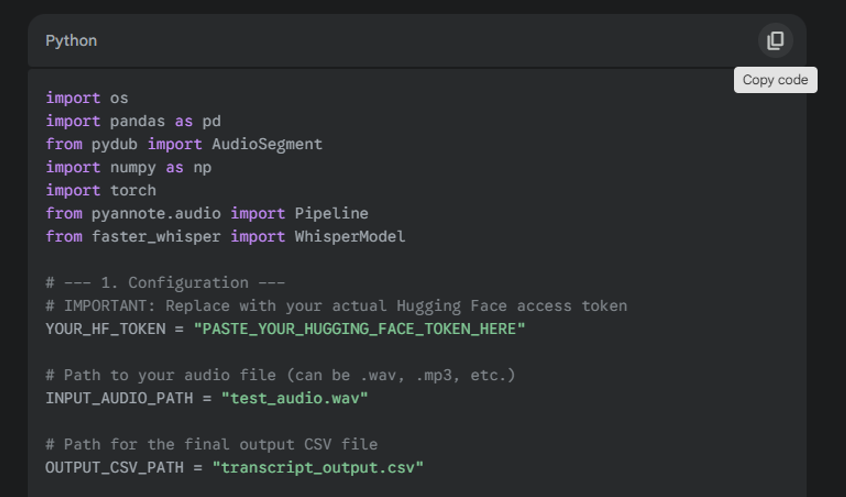


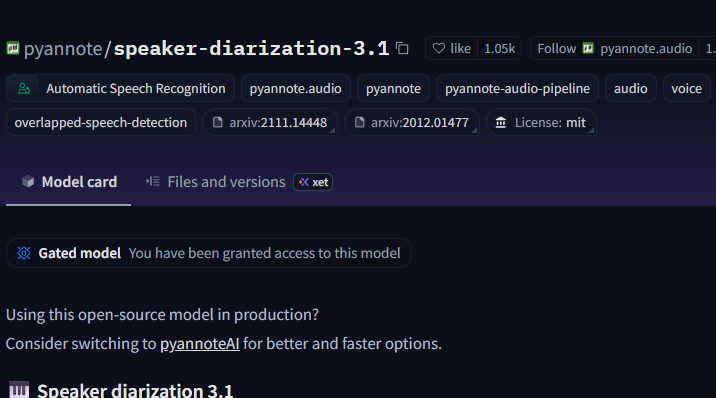


Creating a new file in VS Code



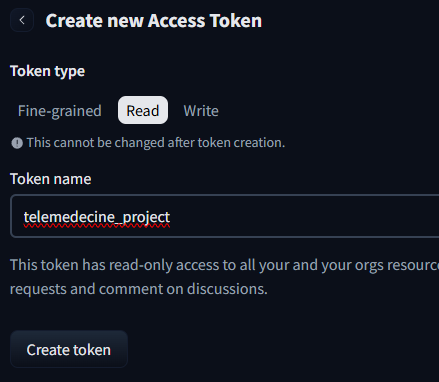
We can now paste in the code that the LLM generated for us.





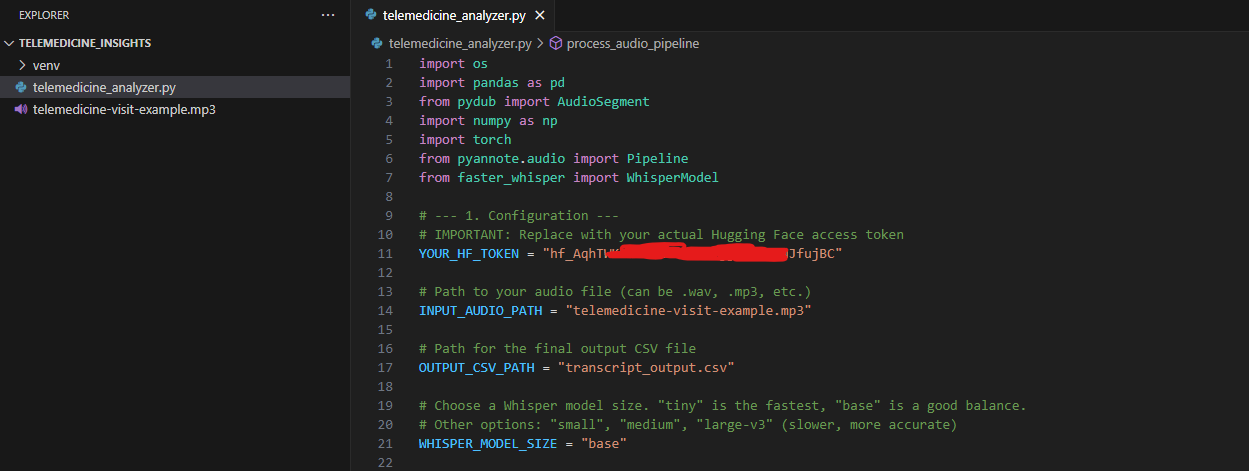
We require to accept the Terms and Conditions of the model (as highlights by the LLM response)

* **Heads-up!** Remember to get your free **Hugging Face** access token as mentioned in the code's comments and paste it where indicated.



[Screenshot: Hugging Face website showing where to create an access token]

Keep this token somewhere safe, since it cannot be seen again after exiting the page.



I have pasted in the Hugging Face token, as well as the input path for my audio file.

**Part 4: Running and Debugging Your Code**

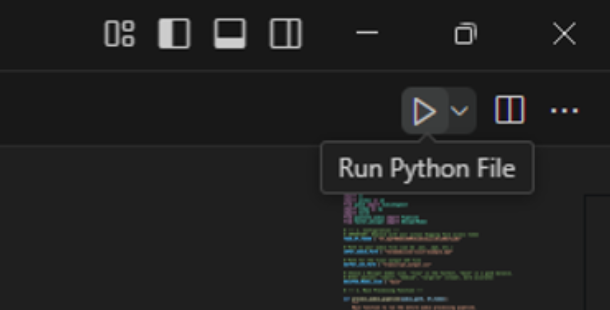
This is the moment of truth! Let's run our script and learn how to fix things if they go wrong.

**Step 4.1: How to Run the Script**

In the VS Code terminal, simply type the following command and press Enter:

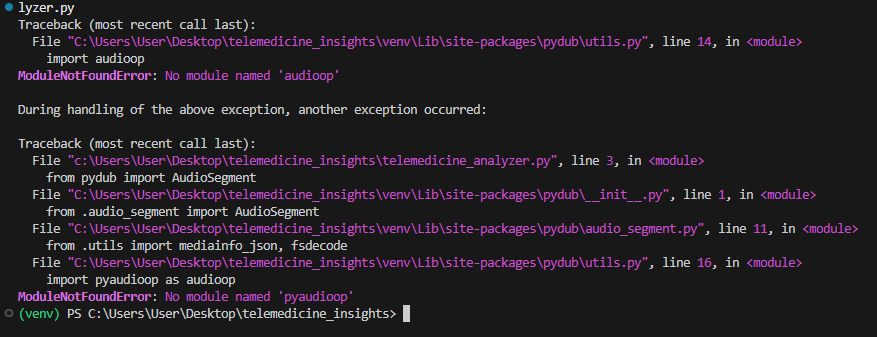
python telemedicine\_analyzer.py

The script will start running, and you'll see log messages appear in the terminal as it completes each task. You can also run the file by pressing the play button as shown from below.



**Step 4.2: The Art of Debugging 🐞**

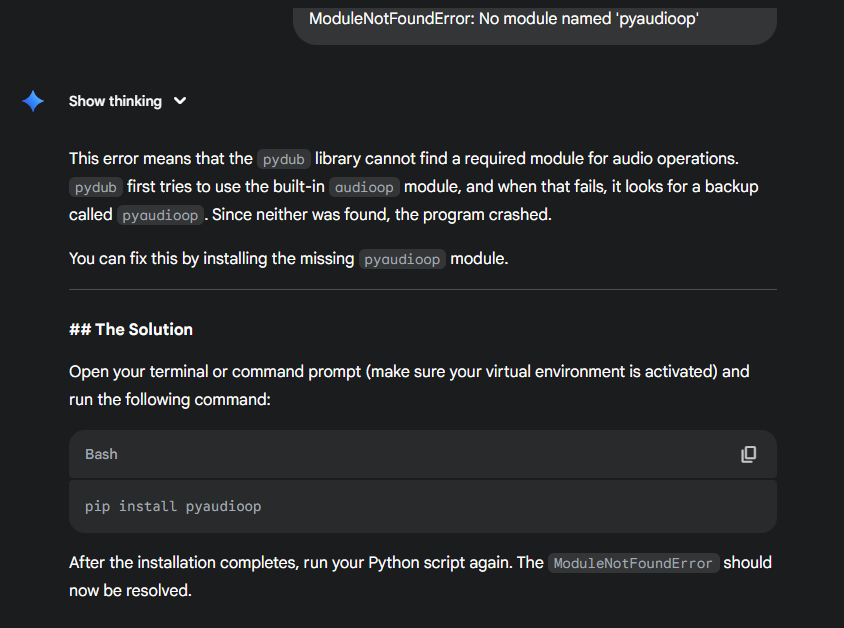
Errors are a normal and expected part of programming. Let's say you see an error like this:



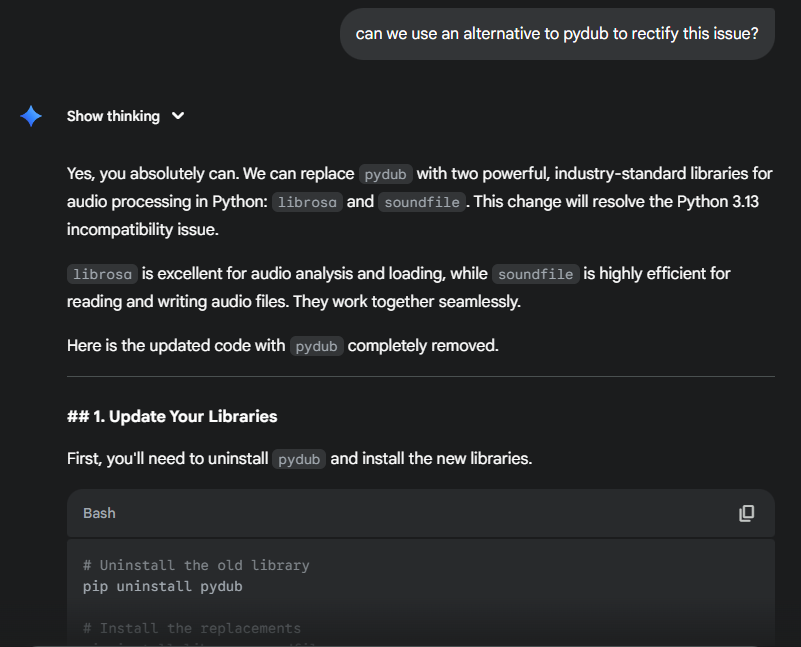
[Screenshot: A Python Error in the VS Code terminal]

**Don't panic! Just follow these steps:**

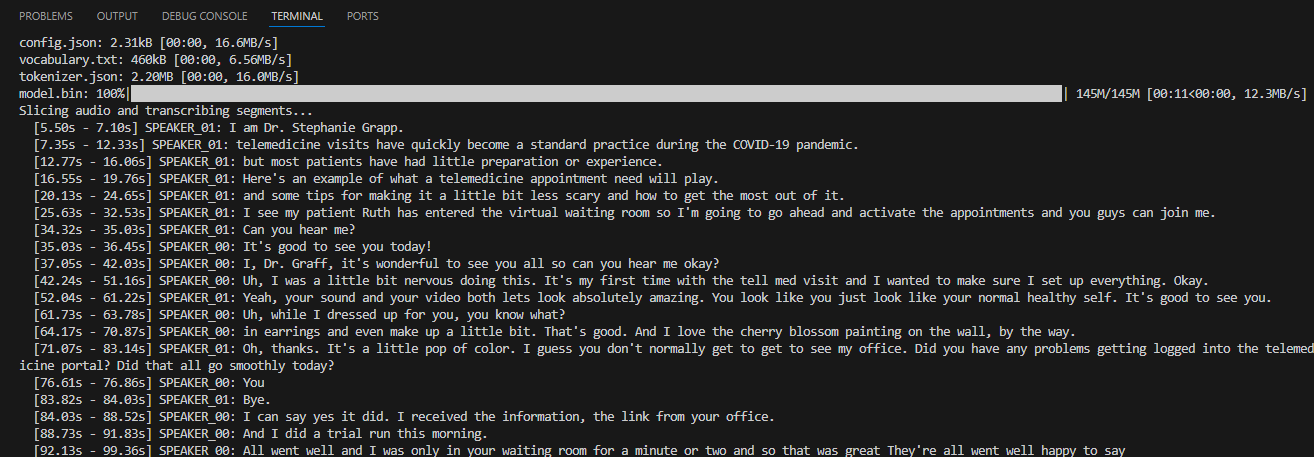
1. **Copy the entire error message** from the terminal.
2. **Paste it into your LLM chat** and ask for help.



After going back and forth with the LLM and getting more errors, it turns out that one of the libraries we were using (pydub) requires some dependencies that are there in an older version of python, but not in the current version that we have. Problems with library versions are common when updates happen. In this case we can either rebuild the venv with an older version of Python, or we can give another library a go.



After getting the updated script from the LLM, we can once again run the code.



This time the code works. Yay!

**Part 5: Saving Your Progress with Git & GitHub**

Now that you have a working script, let's save a snapshot of it using Git. This is like creating a save point in a video game.

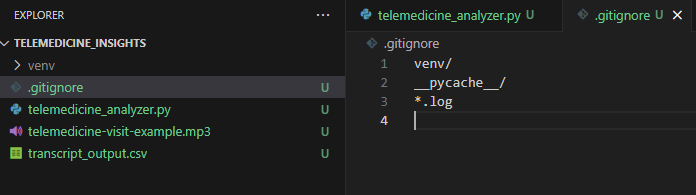
**Step 5.1: Making Your First Commit**

1. **Initialize Git:** In the terminal, run: git init
2. **Create a .gitignore file:** This tells Git to ignore files we don't need to save. Create a new file named .gitignore (the same way we had made the Python file earlier, but do not put .py in the end this time) and add this text:

venv/

\_\_pycache\_\_/

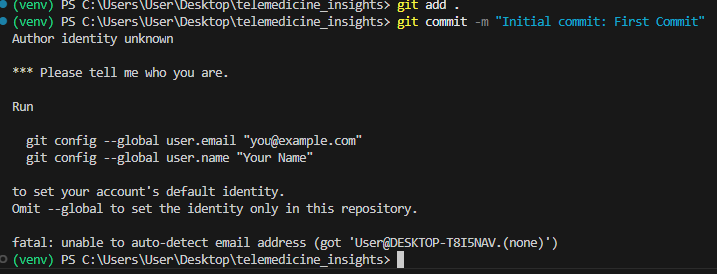
\*.log



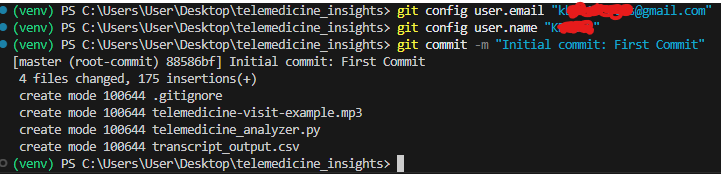
1. **Stage and Commit:** In the terminal, run these commands one by one:

git add .

git commit -m "Initial commit: First Commit"

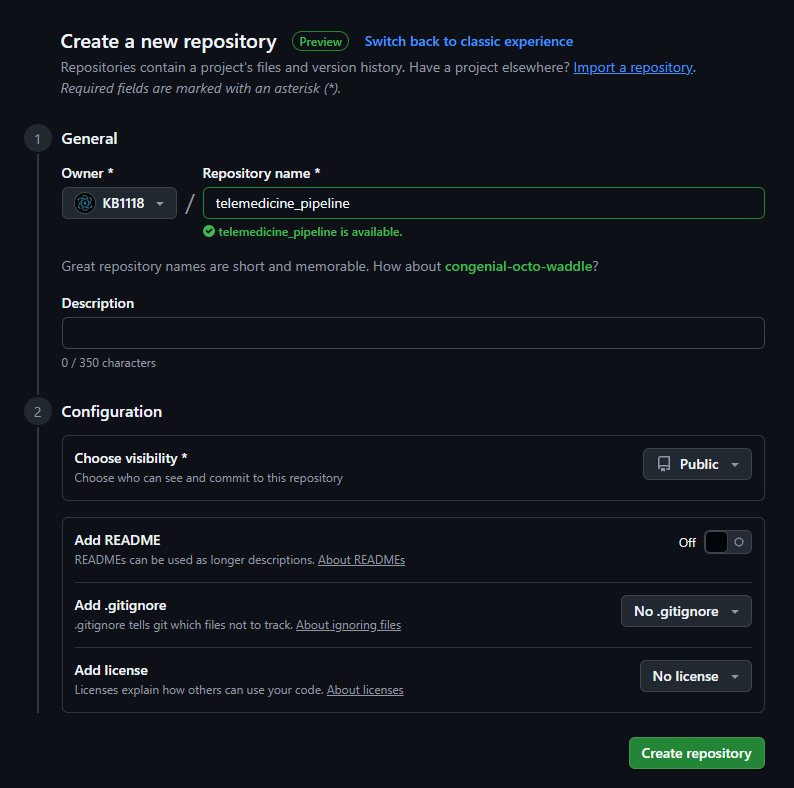


Since this is our first time setting up Git, we need to run the commands given in the output.



**Step 5.2: Pushing to GitHub**

1. **Create a GitHub Repository:** Go to [GitHub](https://github.com/), sign up, and click "New repository". Give it a name and create it. You can make it Private if you like as well. Remember to remove your Hugging Face Token from the code before pushing it to Git.



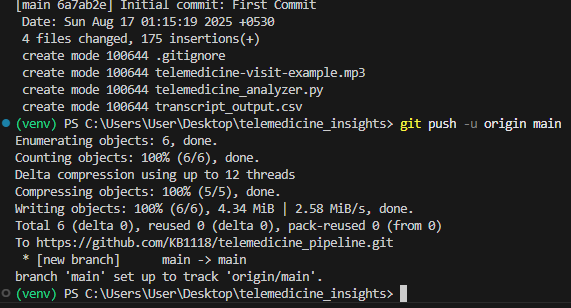
[Screenshot: The 'Create a new repository' page on GitHub]

1. **Link and Push:** GitHub will give you commands to run. Copy them and run them in your terminal. They will look something like this:

git remote add origin <your-repository-url>

git branch -M main

git push -u origin main



[Screenshot: GitHub page showing the commands to link and push a local repository]

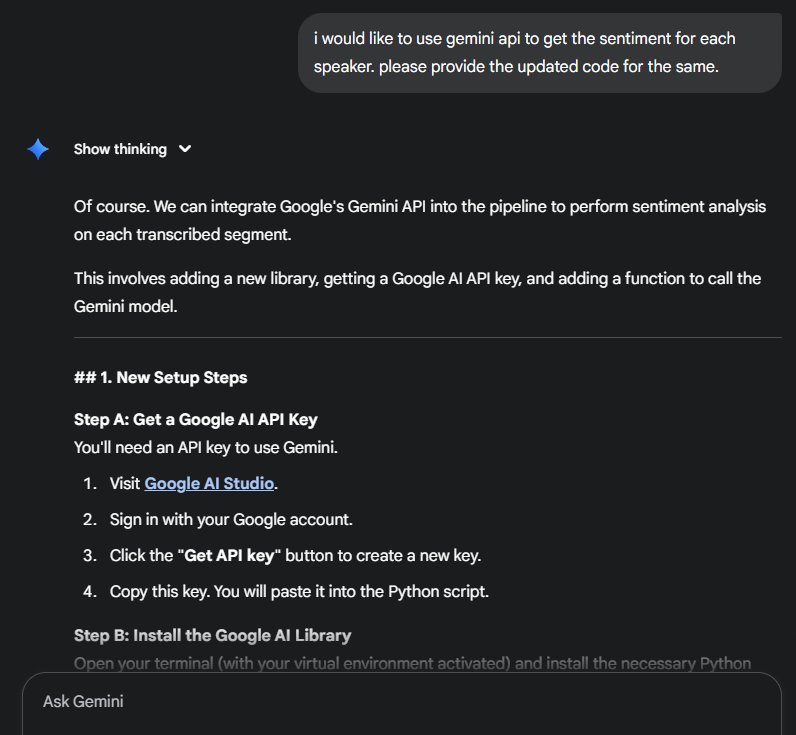
A pop-up will appear for you to sign into GitHub. Once you do that and the files have been pushed, you should be able to view the files in your GitHub repo.



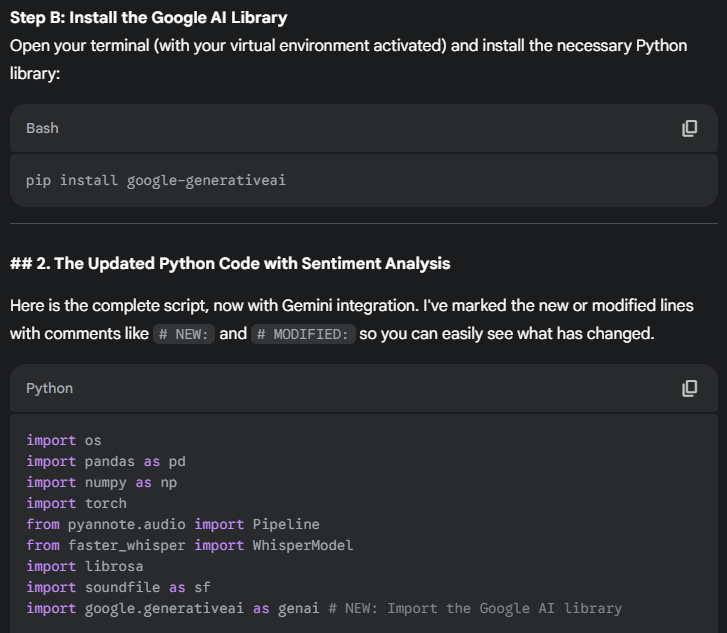
Your code is now safely backed up on GitHub!

**Part 6: Leveling Up - Demo 2**

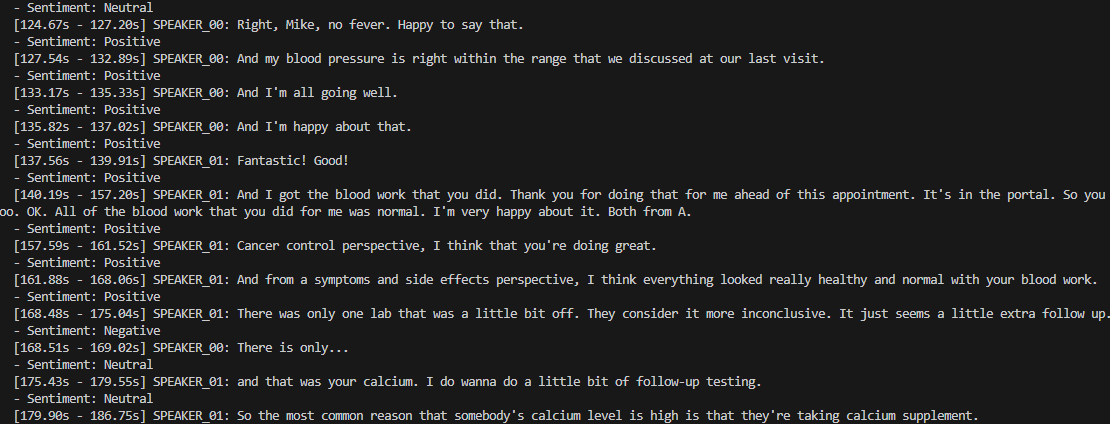
Let's make our project even better by getting the sentiment for *each speaker*. I am going to be using Gemini since it has a free Tier that lets us make prototypes and try out functionality.

****

The AI will provide an updated code block. We need to grab a Gemini API Key from the link given by the LLM, and then we need to install the appropriate library as well. We can then replace the old code in your telemedicine\_analyzer.py file with this new, improved version. Make sure to paste in the Hugging Face Token, the Gemini API Key, and the correct file paths.



[Screenshot: LLM chat showing the prompt for Demo 2 and the AI's modified code response]

****

**Save Your New Version:** After testing the new code, save a new snapshot with Git:

git add .

git commit -m "Feat: Add speaker-specific sentiment analysis"

git push

You've now got a more advanced, version-controlled project. Congratulations on taking your first steps into the world of programming and AI!