With so many paid AI-powered services around these days, has anyone ever wondered if one can do these things locally on their own machine without the use of paid services to do this deal? Well, I thought I'd give it a go on my own setup. A few months ago I did the same thing with SoVITS, which is a standalone program for cloning your voice or other voices, locally on one's own machine with no paid services needed. While I got mixed results due to my own hardware limitations and age of that hardware I did get it to work and that was the mission of those experiments. Just to see if I could get it working based on my current hardware setup and its limitations. For this experiment we are work with transcribing audio to text, I thought this process would be much easier for my system to handle and I wasn't wrong. However, I did run into several challenges that hindered my computer's ability to properly install the software and all its dependencies properly without errors. I did several workarounds and I did eventually get it working to the level I wanted it to work at. For me it was about quality, speed and accuracy. I now feel confident that if in the future I need an audio file that has the majority of that audio file as spoken word dialog/conversation and have my computer transcribe that and dump it into a very easy to manipulate text file that I can copy/paste/edit said text. Below is my explanation along with the methodologies I used to get it working with all the proper commands and explanation of those commands.

First let's talk about Whisper and what it is…

Whisper is an advanced automatic speech recognition (ASR) system developed by OpenAI. OpenAI is the same company that develops ChatGPT. It is designed to transcribe spoken language into written text. The program is capable of processing a variety of audio files containing speech. Examples would be MP3, WMA, ACC, AIFF, PCM, WAV, Ogg, FLAC and many others. Whisper's practicality comes from its ability to transcribe audio files into text, making it highly useful for a range of tasks, such as:

1. **Transcribing Conversations or Meetings**:
   - If you recorded a conversation or a meeting, Whisper can take that audio and turn it into a readable text formatted document. This could be helpful for keeping a written record of important discussions, or for reference in the future.
2. **Content Creation**:
   - For creators, podcasts, YouTube videos, or interviews, Whisper can convert spoken words into text, which could then be used for captions, subtitles, or blog posts.
3. **Accessibility**:
   - It provides a way for individuals who are deaf or hard of hearing to have access to content that is spoken. This is especially useful for video content, meetings, or any form of multimedia.
4. **Research or Note-Taking**:
   - Researchers or students can use Whisper to transcribe lectures, interviews, or audio notes, making it easier to analyze and reference the material in written form.

Whisper doesn't come with a typical graphical interface (GUI) for easier usage that one might expect from most software. Instead, it uses a command-line interface (CLI), where you type in text-based commands in programs like CMD, PowerShell, or Python. While this might sound complicated, there are a few practical reasons why Whisper is designed this way, especially for its intended audience. A GUI requires more system resources, like memory and processing power, because it needs to display visuals and handle user interactions. The basic functionality in a command-line tool is much lighter on resources, meaning Whisper can focus all its energy on doing the heavy work of transcribing or processing audio. This is especially important for Whisper, which can handle long audio files or large amounts of data—something that could slow down if it had to display a graphical interface as well.

Another reason is that Whisper is mainly aimed at developers, researchers, and people with some technical background who need a lot of control over how the program works. With a command-line tool, users can easily adjust things, like the model it uses, whether to process the audio with a computer's CPU or in some cases a more powerful GPU, and what kind of output file they want. These things might not be easy to do with a GUI, and people who are comfortable with coding often prefer to have this level of control. Whisper also needs to be able to run on a variety of systems, from personal computers to powerful cloud servers. A command-line interface is more flexible and can be used in many different environments, whether on a personal desktop or remotely via the internet. GUIs are harder to set up and maintain across different systems, making the CLI a better choice for Whisper's developers, who want to keep it simple and efficient.

Since Whisper is open-source software, it's also built to be transparent and modifiable. Advanced users can dive into the code and customize how it works or integrate it into larger projects. This wouldn't be as easy to do if there were a GUI, which would hide much of the underlying functionality. The people who use Whisper are usually looking to get fast results, not spend time navigating through menus, settings and buttons. The command-line interface allows them to quickly type commands and get the task done, whether they are transcribing an audio file or running batch processes for many files at once. While a graphical interface might make Whisper look more user-friendly, the command-line design actually gives users more power, speed, and control. It's a tool made for people who know how to work with it on a deeper level, and it allows Whisper to run efficiently and be used in a wide range of setups and environments.

Fear not though. One can and probably should, as I did, use ChatGPT to help with making the setup more logical and help with the actual commands. I know next to nothing on how Python, Python's venv *(virtual workspace on your computer),* PowerShell actually work; and I only have moderate knowledge on how CMD works. At least it isn't a foreign language for me. You do not have to be an advanced programmer for this. My attempt here is to make this essay-style tutorial be as easy as possible to follow along. I am not saying this will work for **ALL** novice Windows-based computer users, but it should work for most of you. If I can get this working on a 15-year old, low-end gaming machine, built for video editing, then it's likely one can make it work on their newer system that was more/less built for advanced office-style duties. The hardest parts here is installing Whisper and all its support software correctly, in the correct order, paths, with the correct versions of those dependences, depending on one's computer hardware specifications. That is the key here; because everyone's computer setup, whether or not it's a Dell or an HP or whatever, is all configured, setup and has different types of hardware profiles inside the chassis. It is all uniquely different. Just like human-beings…

So, how exactly does Whisper work?

Once the Whisper program is properly set up on your computer it can process an audio file, *(like an MP3 or WAV),* that contains speech. Whisper then transcribes the speech into text and outputs it as a .txt file, which is one of the most basic and universally supported text file formats in the digital world. This text file can be opened and used across virtually all devices, operating systems, and software programs in the world. A real-world practical example; imagine you recorded a conversation between you, and say, your ex, and for legal reasons you need to have a written record of that conversation for your lawyer. Whisper can transcribe the spoken words into a simple text file, which you can then edit, share, or use as needed. Whisper is a tool that allows you to turn speech into text, making audio content more accessible and usable for various purposes. Once set up, it becomes a simple and practical tool for converting spoken word into a written form and it's on your computer, free to use. So after all that sale's pitch and definition nonsense let's try to install this puppy…

Tutorial 1: Installing Whisper Without Virtual Environment on Windows 10:

Our first tutorial here will be on trying to install everything just using Windows 10 Pro's CMD and the basic Python setup as the installation bed. Most recommendations had said it is way easier to install/setup/configure Whisper inside a Python venv *(virtual environment)*. This will be explained a little later, but this was a good lesson on at least learning the programming language, commands and philosophy as to how this stuff works. If one is on a much newer machine than mine, perhaps a brand new one, one may not have to go through all the virtual setups for this as I did. Although this method did not work on my system, it can work on modern machines with s more up-to-date hardware package.

First we may need to download and install the Python system on the machine that is going to use Whisper. One might already have it, but if it is a brand new machine it is likely one would have to download and install it. Now before we get started it should be noted that there are multiple ways to download, install, configure programming language software on your computer. The method I used here was more/less due to my inexperience using CLI software like this and my older, out-of-date, hardware profile.

Download and install Python from the official Python.org/downloads/windows/. Download the newest version. At the time of this writing the newest version is 3.13.2. It should be noted that due to my machine's age I needed a far older version **(3.9.6),** which will be explained later in this essay. While installing if there is an option to check a box to add Python to the system PATH during installation, do so. If there isn't, once Python is installed we will need to ad it manually. I had to.

The Python executables or the actual file that opens the program is not installed in the normal places one installs software for their computer, where Windows will put a shortcut to it either on your desktop top or in your Windows Taskbar or Windows Start Menu. It stores it in a folder that tends to be hidden in Windows by Default in your main Libraries folder. This is the folder where your main folders for your profile on the Operating System are stored: Documents, Pictures, Videos, etc. This folder usually has the name you labeled it as when you setup your computer initially. The folder we need to access is also here, but might be hidden. So we need to tell Windows to show these folders in the Librarie's directory. Navigate to your main Library's folder and click view at the top. There should be a checkbox that says **hidden items.** Make sure that is checked. There should be a folder there named **APPDATA.** Open it. Open the folder named **Local.** Open the folder named **Programs.** Open the folder named **Python.** If you have a single version of Python you should see the folder there. For this installation attempt we are using **Python313.** Open the folder named **Python313.** What we want to do here is copy the file path of this folder. Go into the address, highlight and copy the entire address string for this directory. It should look something like this: **C:\Users\Your-Library-Folder-Name\AppData\Local\Programs\Python\Python313\** - We are going to want to copy this. We are then going to paste this path into our System Properties – Environmental Variables Settings. Go into your Windows search and type in **Environmental Variables** and click the option that says: **Edit environment variables for your account.** This should open a little Window that says: **System Properties.** At the bottom select button that says: **Environmental Variables.** Under System Variables scroll through the variables to **Path.** Click **Edit.** Click **New.** Scroll to the bottom of the list and paste your Python installation here: **C:\Users\Your-Library-Folder-Name\AppData\Local\Programs\Python\Python313\ -** Once that is done you will need to click the OK button three times to get out of the System Environmental Settings. We now want to confirm Python's installation and access. We are going to want to open your Command Prompt or CMD, but we are going to want to open this in Administrator Mode. This will by-pass any permission issues we might run into while downloading, installing and setting up the proper software packages into the Windows OS Environment. In the Windows Search type **CMD,** you should see it on the left but on the right it should say **Run As Administrator.** You can also right click on the CMD and choose to **Run As Administrator** from the right-click menu. Once CMD is open type: **python --version** – This will show you the version number of Python. Mine says Python 3.13.2. I never ran into an error here. So if you did either try the process again after a reboot, check Google and/or ChatGPT or consult a friend who has advanced IT skills.

Next we need to start the process of downloading and installing the required dependencies. In your CMD install pip, numpy, and other dependencies:

In CMD type/copy/paste these commands separately so that we know each one is installed correctly. Doing batch-style commands here gave me issues.

First Command:

```
Python -m pip install --upgrade pip
```

Second Command:

```
Python -m pip install numpy
```

Third Command:

```
Python -m pip install torch torchvision torchaudio
```

Now it is time to install Whisper itself. Type this command:

```
pip install openai-whisper
```

If everything worked correctly one could try to test it with a file to see if it is working. For this tutorial we are just going to use a generic audio file that has dialog on it that we popped into the directory.

This would be the command one would use to do this:

```
whisper "C:\path\to\your\audiofile.mp3"
```

This is where we ran into issues. I never could get past the installation step of Whisper on my system under this type of configuration. It was recommended to attempt to download an older version of Whisper, which I tried, that also failed. Whisper did not work in this setup due to hardware limitations and conflicts between software versions *(such as NumPy version issues)*. **Why It Failed** is probably closer to the lack of a virtual environment resulting in version conflicts between the System, Python, and Packages, especially with NumPy. The solution here was to move to the next method using a virtual environment (venv) to isolate the setup. On a modern system the steps should work without modification. Newer hardware can follow the standard installation process without these issues.

Tutorial 2: Installing Whisper Using a Virtual Environment (venv) with CPU:

This method involved setting up a **virtual environment (venv)** to avoid conflicts with system-level Python packages and dependency conflicts. A Python **virtual environment (venv)** is like creating a separate, contained workspace on your computer where you can install and manage specific tools and libraries for a particular project without affecting the rest of your system. Imagine it like a special, isolated folder that has everything needed for a specific project. Inside this folder, you can install Python packages, like Whisper, and only those packages will be available for that project. Other projects or programs on your computer won't be impacted by these installations and configurations. This is especially useful when you're working on different projects that might need different versions of Python or specific libraries that could be in conflict with each other. By using a virtual environment, you ensure that each project has its own set of tools, rules for those tools and your main system stays clean and organized. A **venv** keeps things tidy, avoids conflicts, and makes it easier to manage dependencies for projects like Whisper or any other Python-based projects one is using.

We will skipped a lot of the specifics within the details that I have already outlined above. Much of these commands are the same. It is just the initial settings where things tend to be different. After trying much of the same processes rom above I ran into more conflicts and system setup errors. This is where we downloaded an older version of Python for my setup. Your setup could literally be the same as the one above except one would be using the **venv** for that setup over this one. This is one is identical except we are using an older version of Python. I will talk about this because it is still likely enough one might have to use an older version of Python, but not all or even most, just some.

To download, install, and ad path to the System Properties Environmental Settings follow the same process above except download and install Python 3.9.6 using this link:
python.org/downloads/release/python-396/

In the **C:\Users\Your-Library-Folder-Name\AppData\Local\Programs** directory you will now see multiple Python folders. Open the folder Python39 and copy the address:
C:\Users\Your-Library-Folder-Name\AppData\Local\Programs\Python\Python39\

When you go into the Environmental Settings to edit the Path you will ad this path beneath the path that say:
C:\Users\Your-Library-Folder-Name\AppData\Local\Programs\Python\Python313\

This essential tells Windows that your primary version is 3.13.2 but if you designate it to use 3.9.6 that Windows will recognize it and use it. We have to tell our OS that in the CMD, but this is the first leg of those steps.

We are now going to create the venv using this older version Python. For users that want to simply use the venv setup using the newest version of Python, both setups for that venv will be listed here. For this to work it is always easier to make Windows do less. So the placement of your venv should be in the most easiest, basic spot on your computer. The C-Drive. In theory you can place it anywhere but the more jumps the more resource-intensive the process becomes. I created a folder on my C-drive and just named it 'whisperabc.' No caps, no quotes just whisperabc – the directory should be: C:\whisperabc

Open your CMD in Run As Administrator mode.

We want to navigate to this directory:
cd C:\whisperabc

This will make sure you are in the C-drive in the whisperabc folder. We now want to run this command in CMD:

C:\Users\Your-Library-Folder-Name\AppData\Local\Programs\Python\Python39\python.exe -m venv whisperabc-env

What this means is you are telling CMD to use the Python executable from that specific location on your computer. The Usual way would be just to type the command:

python3.9 -m venv whisper-env but that command did not work for my system. I had to include the entire file path in order for Windows to recognize that I wanted to specifically use Python 3.9.6 for this venv installation. It should also be noted here we are using the CPU for the processing and not the GPU. We will have a tutorial for swapping out the CPU for the GPU in our final tutorial on Whisper. Once we have our venv created we want to navigate back to the folder in Windows. Go to your C:\whisperabc and open the folder whisperabc-env. Then open the folder Scripts. Copy the file address and go back to your CMD to navigate to this directory with the command:

cd C:\whisperabc\whisperabc-env\Scripts

This will tell CMD to go to this folder. Once there type the command:
activate and press enter.

At this point you should be inside your virtual environment by seeing the prompt with parenthesis around it followed by the directory you are currently in: **(whisperabc-env) C:\whisperabc\whisperabc-env\Scripts>** and now it is time to start installing dependencies.

First Command:
Python -m pip install --upgrade pip

Second Command:
Python -m pip install numpy

Third Command:
Python -m pip install torch torchvision torchaudio

Now it is time to install Whisper itself. This time we are going to download an older version for this configuration because others also failed on my system even in the venv. One would use the command in the above tutorial over this one below.

Type this command:

```
pip install openai-whisper==20230918
```

This tells our pip configuration tool to download and install Open Ai's Whisper, Version released on 09/18/2023. On a newer machine one would not need this version but it may be noteworthy that one can download earlier versions if the newest version is not working with your hardware configuration. A test is in order with the generic audio file we had from earlier. Let's verify the installation first to make sure we have it installed properly.

Verify Installed Version:

Usually one would use the standard command of **whisper --version** but because we are using an older version we may want to run the **pip show openai-whisper** command. This will also display the version number.

The command you are going to want to use is:

```
"C:\whisperabc\whisperabc-env\Scripts\python.exe" -m whisper "C:\whisperabc\audio123.mp3" --model large --output_dir "C:\whisperabc" --output_format txt --language English
```

This tells your venv that you want to use your old python 3.9.6 by pointing to it in your venv scripts. You want to use whisper here and to transcribe the mp3 in that directory using the large model, which is the more accurate than the small model, just takes a little bit longer to process. The command then is saying to export the transcribed text into text format with the .txt file extension in the English language. This skips the system checks for these output procedures which quickens the process.

Now you can go back to your C:\whisperabc folder and there should be a text file there called audio123.txt. If you open it –it will have your transcribed text in a simple basic format. You can then copy/paste this text into a document for whatever purpose you need it for.

Tutorial 3: Installing Whisper Using Virtual Environment (venv) with GPU:

Next we are going to try install this again but instead of using our computer's CPU we are going to make it use the GPU. The GPU is the CPU for the Graphics Card. If one has an exterior graphics card independent from the motherboard it is called a GPU. The CPU is like a single smart, multi-tasking tool that handles a variety of tasks efficiently, but only has a few resources. It's great for general work, but can get overwhelmed with heavy, repetitive calculations. The GPU is like a massive team of these specialized tools, plural, each handling a tiny part of a big job at the same time. Since transcribing audio with Whisper involves a ton of complex mathematical calculations, the GPU can break it down into smaller pieces and process them all at once—**making it noticeably faster than a CPU** for this kind of workload. While a CPU can do the job, a GPU **does it much faster and efficiently** because it's built for parallel processing. That's why we want to set Whisper up to use your graphics card. In the case it is a **NVIDIA GeForce GTX 750Ti.** Since we are using an older system this becomes even more problematic because we still have to install this puppy using our older versions. It is very doable, but a little more complex a setup:

The GPU (GTX 750 Ti) does not support CUDA 11+, which is required for the latest versions of PyTorch with GPU acceleration. OpenAI Whisper defaults to CPU unless explicitly told to use the system's GPU.

We need to build another venv. Open your CMD in Run As Administrator mode.

We want to navigate to this directory again:

cd C:\whisperabc

This will make sure you are in the C-drive in the whisperabc folder. We now want to run this command in CMD:

C:\Users\David-PC\AppData\Local\Programs\Python\Python39\python.exe -m venv whisperabc-env

Like before; CMD uses Python from older the version's location. Once we have our venv created we want to navigate back to the folder in Windows. Go to your C:\whisperabc and open the folder. Open the folder whisperabc-env. Then open the folder Scripts. Copy the file address and go back to your CMD to navigate to this directory with the command:

cd C:\whisperabc\whisperabc-env\Scripts

Once there type the command:

activate and press enter.

First Command:

At this point you should be inside your virtual environment and now it is time to start installing dependencies. Since Whisper uses PyTorch under the hood, you need to install the **CUDA-compatible version of PyTorch**. For **NVIDIA GPUs**, install a CUDA-supported version of PyTorch with this commend: **pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu117**

We are using an older GPU (**GTX750Ti)**, CUDA 11+ is **not supported**, so installing an older CUDA version was recommended with this command but failed during the installation process. You want to use the version above this one: pip install torch==1.10.0+cu102 torchvision==0.11.1+cu102 torchaudio==0.10.1 -f https://download.pytorch.org/whl/torch_stable.html in the case one wants to try this hardware profile.

We have to do the same for numpy here. We have to download an older version that supports this older GPU. Through trial and error this version worked for me.

Second Command:

Python -m pip install numpy==1.23.5

We can verify the install with this command:

python -c "import numpy; print(numpy.__version__)"

Now it is time to install Whisper itself. Same as before; we are going to download the same older version for this configuration. For a regular installation on a newer machine, one would use the command in the first tutorial above over this one. The only real difference between this installation and the last one is we are forcing our system to use the GPU over the CPU, but other than that it is basically the same setup.

Type this command:

```
pip install openai-whisper==20230918
```

This tells our pip configuration tool to download and install that same version of Whisper that worked before.

Verify Installed Version:

You can either try the standard verification of `whisper --version` command but because of the older version the `pip show openai-whisper` command will display the version number a little easier.

We need to verify that PyTorch detects your GPU. We'll need to activate Python inside our venv. Run this command: **python** You will see Python activate inside your venv as **>>>**

Run the following commands in the Python CLI:

First Command:

import torch press enter
The response should be another **>>>**

print(torch.cuda.is_available()) press enter
The response should be **TRUE**

print(torch.cuda.get_device_name(0) if torch.cuda.is_available() else "No GPU detected") press enter.

The response should be **NVIDIA GeForce GTX 750Ti**

If the read outs for each come out as shown above then PyTorch detects your GPU. All we need to do now is Force Whisper to Use GPU. By default Whisper uses yours system's CPU. We have to instruct Whisper to use the GPU inside our command prompt structure. The command will look something like this but we are not done just yet.

```
whisper "C:\whisperabc\audio123.mp3"  --model large --device cuda
```

We are now ready to run our test of Whisper using our GPU with our, sort of, complicated hardware profile. As before, we are using the same mp3 in the C:\whisperabc. The command you are going to want to use is the same as the last tutorial, but with the added –device cuda tag to tell Whisper to use the GPU. We had to do this workaround to tell venv to utilize the GPU for the calculation processing that is why we could not just use the second tutorial and just adding the –device cuda tag to it. We would have gotten errors. Use this command:

```
"C:\whisperabc\whisperabc-env\Scripts\python.exe" -m whisper
"C:\whisperabc\audio123.mp3" --model large --output_dir "C:\whisperabc" --
output_format txt --language English --device cuda
```

We are again explicitly telling Whisper to:

- Use **Python inside your virtual environment**
- Run Whisper as a module
- Use the **large model**
- Process audio123.mp3 from the directory
- Save the output to C:\whisperabc in TXT format (.txt)
- **Specify the language** (English) instead of auto-detecting
- **Force GPU usage** with --device cuda

If PyTorch is properly detecting your GPU (torch.cuda.is_available() returns True), which we verified above, then the command should **run Whisper using the GPU instead of the CPU**. When it's done, go back to your C:\whisperabc and see if audio123.txt is there. Open the file and check the accuracy.

<u>Conclusion:</u>

• **Summary of Methodology**: We first tried installing Whisper without a virtual environment, which caused issues due to package conflicts and our outdated hardware profile. We then used a virtual environment to isolate the installation and ensure proper compatibility with the required dependencies. Finally, we switched to using the GPU to take advantage of hardware acceleration, resolving issues with legacy dependencies.

• **Lessons Learned**: Isolating dependencies with a virtual environment is crucial to avoid conflicts. Additionally, ensuring that the correct versions of libraries (e.g., NumPy, PyTorch) are installed is key to getting Whisper to work on older hardware.

The purpose of this essay on the methodology I used here was to both learn how to use Whisper for my many projects and be able to do something technical, like, installing, configuring it on a machine that is considered vastly outdated by today's standards. Specifically, using your personal computer to do high level, mostly, automated processes. These three tutorials now reflect the specific steps taken and could serve as a guide for others that have older rigs, wants a very basic but detailed version of the steps, what they are, why they exist, and want to do some neat stuff with open source programs like this. Most everyday computers could do this. The high level gaming machines of today would zip through this compared to my current setup here.

It took some trial and error, but that's part of the fun, I guess—learning how to work with what you've got and making it do things it wasn't necessarily built for. If you've got a modern machine, you'll probably breeze through this setup without much hassle. But even if you're working with older hardware like mine, this just proves that with the right tweaks, open-source tools like Whisper can still run just fine. The big takeaway? **You don't always need the latest and greatest tech to do cool stuff.** A little problem-solving, the right software versions, and some patience can go a long way in getting powerful AI tools up and running, locally on your own machine and most important, **FREE...**

Whisper on Older Machines
by David-Angelo Mineo
3/29/2025
5,068 Words