# Voice recorder application using Python

Creating a voice recorder application using Python is an excellent way to dive into audio processing and manipulation. This guide will walk you through building a simple voice recorder application that can capture audio from a microphone and save it in various formats such as WAV, MP3, and more.

**Prerequisites**

Before we start, ensure you have the following:

1. **Python**: Make sure Python is installed on your system. You can download it from [python.org](https://www.python.org/).
2. **pip**: This is the package installer for Python. It comes with Python, but you can upgrade it using:

python -m pip install --upgrade pip

1. **pyaudio**: This library is used for capturing audio from the microphone.

pip install pyaudio

1. **wave**: This is a standard Python library for reading and writing WAV files.
2. **pydub**: This library is used for converting audio files to different formats.

pip install pydub

1. **ffmpeg**: Required for converting audio formats with pydub. You can download it from [ffmpeg.org](https://ffmpeg.org/download.html) and ensure it's added to your system's PATH.

**Step 1: Recording Audio**

We'll start by writing a script to capture audio from the microphone and save it as a WAV file using PyAudio and the wave module.

import pyaudio

import wave

def record\_audio(filename, duration, sample\_rate=44100, channels=2):

 p = pyaudio.PyAudio()

 stream = p.open(format=pyaudio.paInt16,

 channels=channels,

 rate=sample\_rate,

 input=True,

 frames\_per\_buffer=1024)

 print("Recording...")

 frames = []

 for \_ in range(0, int(sample\_rate / 1024 \* duration)):

 data = stream.read(1024)

 frames.append(data)

 print("Finished recording.")

 stream.stop\_stream()

 stream.close()

 p.terminate()

 wf = wave.open(filename, 'wb')

 wf.setnchannels(channels)

 wf.setsampwidth(p.get\_sample\_size(pyaudio.paInt16))

 wf.setframerate(sample\_rate)

 wf.writeframes(b''.join(frames))

 wf.close()

record\_audio('output.wav', 10) # Record for 10 seconds

**Step 2: Converting Audio Formats**

To convert the recorded WAV file to other formats like MP3, we'll use the pydub library.

from pydub import AudioSegment

def convert\_audio(input\_file, output\_file):

 audio = AudioSegment.from\_wav(input\_file)

 audio.export(output\_file, format=output\_file.split('.')[-1])

convert\_audio('output.wav', 'output.mp3') # Convert WAV to MP3

**Full Voice Recorder Application**

Here’s how you can combine the recording and conversion functionality into a single application:

import pyaudio

import wave

from pydub import AudioSegment

def record\_audio(filename, duration, sample\_rate=44100, channels=2):

 p = pyaudio.PyAudio()

 stream = p.open(format=pyaudio.paInt16,

 channels=channels,

 rate=sample\_rate,

 input=True,

 frames\_per\_buffer=1024)

 print("Recording...")

 frames = []

 for \_ in range(0, int(sample\_rate / 1024 \* duration)):

 data = stream.read(1024)

 frames.append(data)

 print("Finished recording.")

 stream.stop\_stream()

 stream.close()

 p.terminate()

 wf = wave.open(filename, 'wb')

 wf.setnchannels(channels)

 wf.setsampwidth(p.get\_sample\_size(pyaudio.paInt16))

 wf.setframerate(sample\_rate)

 wf.writeframes(b''.join(frames))

 wf.close()

def convert\_audio(input\_file, output\_file):

 audio = AudioSegment.from\_wav(input\_file)

 audio.export(output\_file, format=output\_file.split('.')[-1])

def main():

 wav\_filename = 'output.wav'

 mp3\_filename = 'output.mp3'

 duration = int(input("Enter the duration of the recording in seconds: "))

 record\_audio(wav\_filename, duration)

 convert\_audio(wav\_filename, mp3\_filename)

 print(f"Audio saved as {wav\_filename} and converted to {mp3\_filename}")

if \_\_name\_\_ == "\_\_main\_\_":

 main()

**Conclusion**

In this tutorial, we've created a simple voice recorder application using Python. We used the PyAudio library to capture audio and the wave module to save it in WAV format. Additionally, we utilized the pydub library and ffmpeg to convert the WAV file to MP3. This application can be further enhanced with a graphical user interface (GUI) using libraries like Tkinter or PyQt, and additional features like pause/resume recording, adjusting sample rates, and more.