**NOISE POLLUTION PYTHON SCRIPT**

**Program No:1**



import pyaudio

import numpy as np

import sounddevice as sd

import soundfile as sf

# Parameters for audio recording

sample\_rate = 44100 # Sample rate (samples per second)

duration = 10 # Duration of recording (in seconds)

# Parameters for noise level threshold

threshold\_db = 40 # Set your desired threshold in decibels

def record\_audio(duration, sample\_rate): audio\_data = np.array([], dtype=np.int16) print("Recording audio...")

# Initialize audio recording

with sd.InputStream(callback=callback):

sd.sleep(duration \* 1000)

return audio\_data

def callback(indata, frames, time, status):

if status:

print(status, file=sys.stderr)

if any(indata):

* Process audio data here if needed audio\_data = indata
* You can add your noise pollution analysis code here

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

audio\_data = record\_audio(duration, sample\_rate)

# Analyze the audio data and calculate the noise level

noise\_level = np.max(audio\_data) # You can replace this with your own noise analysis algorithm

print(f"Noise level: {noise\_level} dB")

if noise\_level > threshold\_db:

print("Noise pollution detected!")

else:

print("No significant noise pollution detected.")

**Program No:2**



import pyaudio

import numpy as np

* Constants for audio settings FORMAT = pyaudio.paInt16 CHANNELS = 1

RATE = 44100 # Sample rate (samples per second) CHUNK = 1024 # Size of each audio chunk THRESHOLD = 2000 # Adjust this threshold as needed

* Initialize the audio stream

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,

channels=CHANNELS,

rate=RATE,

input=True,

frames\_per\_buffer=CHUNK)

print("Listening...")

try:

while True:

data = stream.read(CHUNK)

audio\_data = np.frombuffer(data, dtype=np.int16)

rms = np.sqrt(np.mean(audio\_data\*\*2))

if rms > THRESHOLD:

print(f"Noise level: {rms:.2f} dB")

except KeyboardInterrupt:

print("Recording stopped.")

finally:

stream.stop\_stream()

stream.close()

p.terminate()

import soundmeter

import time

**Program No:3**



* Initialize the sound meter meter = soundmeter.Meter()
* Create a log file to store noise level data log\_file = 'noise\_log.txt'

try:

while True:

* Measure the noise level noise\_level = meter.get\_level()
* Get the current timestamp

timestamp = time.strftime("%Y-%m-%d %H:%M:%S")

* Print the noise level and save it to the log file print(f"{timestamp} - Noise Level: {noise\_level} dB") with open(log\_file, 'a') as f:

f.write(f"{timestamp} - Noise Level: {noise\_level} dB\n")

* Sleep for a specified interval (e.g., 1 minute) time.sleep(60)

except KeyboardInterrupt:

pass

print("Monitoring stopped.")

**Program No:4**

import sounddevice as sd

import numpy as np

import math

def calculate\_noise\_level(audio\_data, sample\_rate):

* Calculate the FFT of the audio data fft\_data = np.fft.fft(audio\_data) num\_samples = len(audio\_data)
* Calculate the frequency values for each FFT bin frequencies = np.fft.fftfreq(num\_samples, 1.0 / sample\_rate)

* Find the peak frequency and its corresponding amplitude peak\_freq\_index = np.argmax(np.abs(fft\_data)) peak\_freq = abs(frequencies[peak\_freq\_index]) peak\_amplitude = abs(fft\_data[peak\_freq\_index])
* Calculate the noise level in decibels (dB)

noise\_level\_dB = 20 \* math.log10(peak\_amplitude)

return noise\_level\_dB

def main():

duration = 10 # Duration of the recording in seconds sample\_rate = 44100 # Sampling rate in Hz

print("Recording... (Press Ctrl+C to stop)")

try:

audio\_data = sd.rec(int(duration \* sample\_rate), samplerate=sample\_rate, channels=1, dtype='float64')

sd.wait()

noise\_level = calculate\_noise\_level(audio\_data, sample\_rate)

print(f"Noise Level: {noise\_level:.2f} dB")

except KeyboardInterrupt:

print("\nRecording stopped.")

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

main()