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()
```