Real Time Pitch Estimation:

What is pitch?
The pitch of a sound is a perceptual quality that refers to how high or low a sound is perceived to be. It's closely related to the frequency of the sound wave. Higher-frequency sound waves are perceived as having higher pitches, while lower-frequency sound waves are perceived as having lower pitches.

In technical terms, pitch corresponds to the fundamental frequency of a sound wave. The fundamental frequency is the lowest frequency component of a complex sound, and it determines the perceived pitch. However, the perceived pitch can also be influenced by the presence of harmonics and overtones, which are higher-frequency components that accompany the fundamental frequency and give a sound its characteristic timbre or tone quality.

Steps in making a real-time pitch estimator:

1. **Capture Audio Input:** Use a library like PyAudio to access the computer's microphone and capture audio input in real-time. Set up a stream to continuously read chunks of audio data.
```
p = pyaudio.PyAudio()
stream = p.open(format=pyaudio.paFloat32,
                channels=1,
                rate=sample_rate,
                input=True,
                frames_per_buffer=chunk_size)
```
2. **Preprocess the Audio Data:** Real-world audio signals often contain noise and other artifacts. Apply pre-processing techniques like windowing and normalization to the captured audio data to improve the accuracy of pitch estimation.
```
def preprocess_audio(audio_chunk):
    # Apply a windowing function (e.g., Hamming window) to the audio chunk
    windowed_chunk = audio_chunk * np.hamming(len(audio_chunk))
    # Normalize the audio data
    normalized_chunk = windowed_chunk / np.max(np.abs(windowed_chunk))
    return normalized_chunk
```
3. **Perform Pitch Estimation:** Pitch estimation involves finding the fundamental frequency of the audio signal. We can use methods like autocorrelation or Fast Fourier Transform (FFT) to achieve this. I will be using the FFT method.
4. **FFT:** Compute the FFT of the audio chunk and find the peak frequency in the spectrum. This peak frequency corresponds to the fundamental frequency (pitch).
```
def fft_pitch_estimation(audio_chunk, sample_rate):
    spectrum = np.fft.fft(audio_chunk)
    freqs = np.fft.fftfreq(len(spectrum), d=1/sample_rate)
    peak_freq_index = np.argmax(np.abs(spectrum))
    fundamental_frequency = abs(freqs[peak_freq_index])
    return fundamental_frequency
```
5. **Combine Everything:** Now, continuously capture audio chunks from the microphone, preprocess them, and estimate the pitch using one of the methods mentioned above.

Additional Remarks: In the implementation, I have averaged the pitch over a time interval (2 seconds) to test it using a sine wave generator. This helps in stabilizing the output of the estimated pitch.

You can find my implementation at (https://github.com/Vineet-the-git/EE798/blob/main/assignment1/pitch_estimator.py)