**Generating audio signals**

Now that we know how audio signals work, let's see how we can generate one such signal. We can use the NumPy package to generate various audio signals. Since audio signals are mixtures of **sinusoids**, we can use this to generate an audio signal with some predefined parameters.

**Create a new Python file and import the following packages:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.io.wavfile import write

**Define the output audio file's name:**

*# Output file where the audio will be saved*

output\_file = 'generated\_audio.wav'

Specify the audio parameters, such as duration, sampling frequency, tone frequency, minimum value, and maximum value:

***# Specify audio parameters***

duration = 4 *# in seconds*

sampling\_freq = 44100 *# in Hz*

tone\_freq = 784

min\_val = -4 \* np.pi

max\_val = 4 \* np.pi

**Generate the audio signal using the defined parameters:**

*# Generate the audio signal*

t = np.linspace(min\_val, max\_val, duration \* sampling\_freq)

signal = np.sin(2 \* np.pi \* tone\_freq \* t)

**Add some noise to the signal:**

*# Add some noise to the signal*

noise = 0.5 \* np.random.rand(duration \* sampling\_freq)

signal += noise

**Normalize and scale the signal:**

# Scale it to 16-bit integer values

scaling\_factor = np.power(2, 15) - 1

signal\_normalized = signal / np.max(np.abs(signal))

signal\_scaled = np.int16(signal\_normalized \* scaling\_factor)

**Save the generated audio signal in the output file:**

*# Save the audio signal in the output file*

write(output\_file, sampling\_freq, signal\_scaled)

**Extract the first 200 values for plotting:**

*# Extract the first 200 values from the audio signal*

signal = signal[:200]

**Construct the time axis in milliseconds:**

# Construct the time axis in milliseconds

time\_axis = 1000 \* np.arange(0, len(signal), 1) / float(sampling\_freq)

**Plot the audio signal:**

# Plot the audio signal

plt.plot(time\_axis, signal, color='black')

plt.xlabel('Time (milliseconds)')

plt.ylabel('Amplitude')

plt.title('Generated audio signal')

plt.show()

The full code is given in the file audio\_generator.py. If you run the code, you will see the following screenshot:

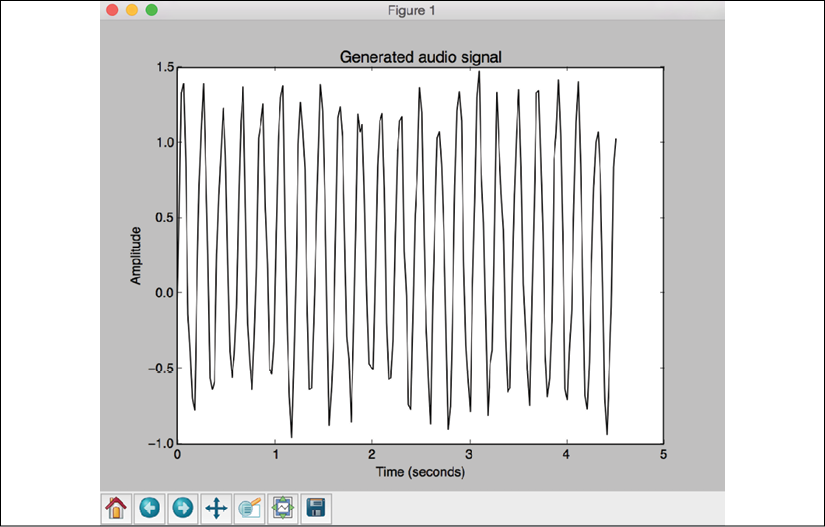


Figure 4: Visualization of audio signal generation

Play the file generated\_audio.wav using your media player to see what it sounds like. It will be a signal that's a mixture of a *784 Hz* signal and the noise signal.