BSD2513 ARTIFICIAL INTELLIGENCE

LAB REPORT 6

NAME : TEAN JIN HE

MATRIC ID : SD21063

SECTION : 02G

Questions 1 : General Knowledge

Explain the applications of speech recognition in real world problems. Give reference/s.

```
Speech Recognition in Language Learning:
https://en.oxforddictionaries.com/definition/speech_recognitionSpeech recognition
technology has numerous applications across various industries and real-world
problems. For example:-

Voice Assistants: Speech recognition is at the core of voice assistants like
Apple's Siri, Amazon's Alexa, and Google Assistant. These intelligent virtual
assistants understand and respond to spoken commands, enabling users to perform
tasks like setting reminders, playing music, searching the internet, and
controlling smart home devices.

Reference: Apple Siri: https://www.apple.com/siri/

Transcription Services: Speech recognition technology is widely used in
transcription services, converting spoken language into written text. It
simplifies the process of transcribing interviews, meetings, lectures, and other
audio recordings, saving time and effort.

Reference: Transcription Services: https://www.rev.com/

Call Center Automation: Speech recognition is employed in call centers to automate
certain tasks, such as routing calls to the appropriate department, generating
automatic responses to frequently asked questions, and analyzing customer
sentiment during conversations. It enhances efficiency and improves the overall
customer experience.

Reference: Call Center Automation:
https://www.genesys.com/platform/solutions/ai/customer-experience-ai

Voice Biometrics: Speech recognition is utilized in voice biometric systems for
speaker verification and identification. It analyzes unique vocal characteristics
like pitch, tone, and pronunciation to authenticate individuals for secure access
to systems or as a part of anti-fraud measures.

Reference: Voice Biometrics: https://www.nuance.com/omni-channel-customer-
engagement/security/biometrics.html

```

```
Dictation Software: Speech recognition technology is employed in dictation software, allowing users to dictate text instead of typing. It finds applications in fields like medical transcription, legal documentation, and content creation, enhancing productivity and reducing typing-related injuries.

Reference: Dragon Dictation: https://www.nuance.com/dragon.html

Accessibility Tools: Speech recognition plays a crucial role in accessibility tools for individuals with disabilities. It enables people with motor impairments or conditions like dyslexia to interact with computers and mobile devices using their voice, improving their access to technology and digital content.

Reference: Accessibility Tools: https://www.microsoft.com/en-us/accessibility

Language Learning: Speech recognition can be used in language learning applications to provide feedback on pronunciation and intonation. It allows learners to practice speaking and receive instant evaluations, helping them improve their language skills.

Reference: Speech Recognition in Language Learning: https://en.oxforddictionaries.com/definition/speech_recognition
```

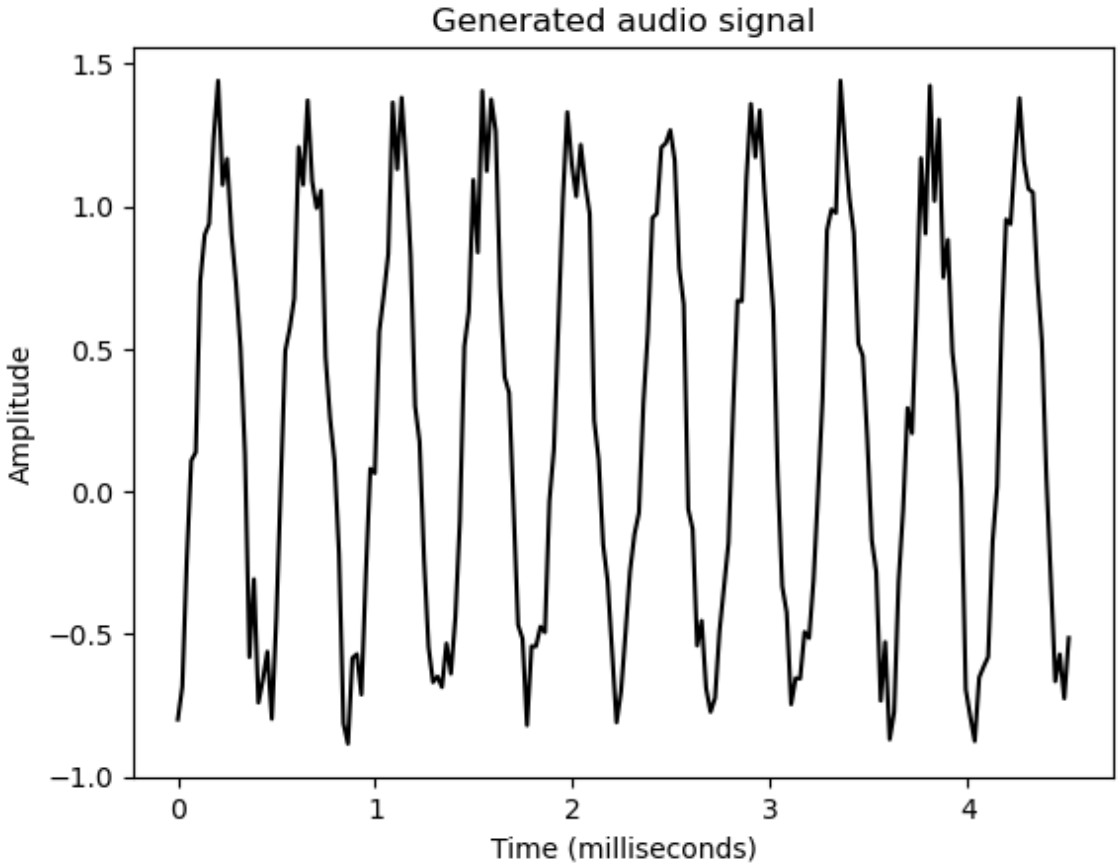
Question 2 Python: Speech Recognition

Most commonly, speech signals are sampled at 44,100 Hz. This means that each second of the speech signal is broken down into 44,100 parts and the values at each of these timestamps is stored in an output file.

Create an audio generator visualisation code.

In [1]:

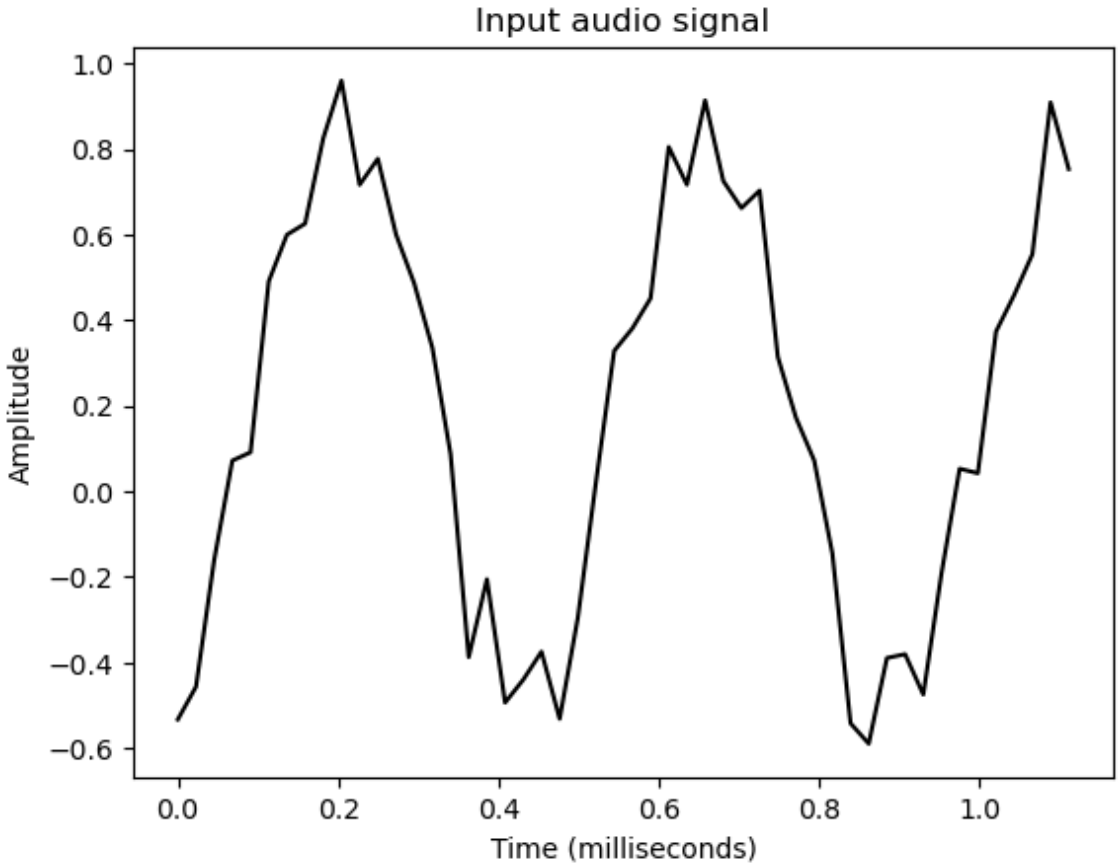
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import write

# Output file where the audio will be saved
output_file = 'SD21063_generated_audio.wav'

# Specify audio parameters
duration = 5  # in seconds
sampling_freq = 44100  # in Hz
tone_freq = 440
min_val = -4 * np.pi
max_val = 4 * np.pi

# 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
noise = 0.5 * np.random.rand(duration * sampling_freq)
signal += noise

# 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 audio signal in the output file
write(output_file, sampling_freq, signal_scaled)

# Extract the first 200 values from the audio signal
signal = signal[:200]

# Construct the time axis in milliseconds
time_axis = 1000 * np.arange(0, len(signal), 1) / float(sampling_freq)

# 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()

```


Generated audio signal
Amplitude
Time (milliseconds)
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

# Speech signals are sampled at 44,100 Hz
# Each seccond of the speech signal is broken down into 44,100 Hz (1/44,100 sec)

# Read the audio file
sampling_freq, signal = wavfile.read('SD21063_generated_audio.wav')

# Display the params
print('\nSignal shape:', signal.shape)
print('Datatype:', signal.dtype)
print('Signal duration:', round(signal.shape[0] / float(sampling_freq), 2), 'seconds

# Normalize the signal
signal = signal / np.power(2, 15)

# Extract the first 50 values
signal = signal[:50]

# Construct the time axis in milliseconds
time_axis = 1000 * np.arange(0, len(signal), 1) / float(sampling_freq)

# Plot the audio signal
plt.plot(time_axis, signal, color='black')
plt.xlabel('Time (milliseconds)')
plt.ylabel('Amplitude')
plt.title('Input audio signal')
plt.show()
```

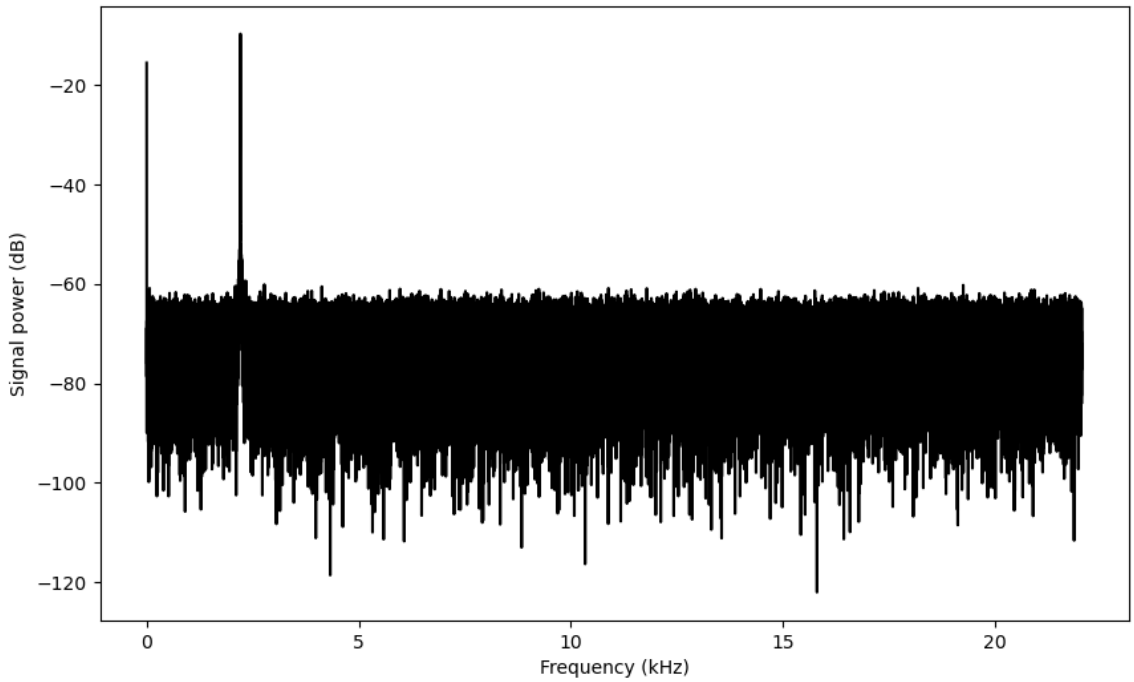
```
Signal shape: (220500,)
Datatype: int16
Signal duration: 5.0 seconds
```


Input audio signal
Amplitude
Time (milliseconds)
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0.0
0.2
0.4
0.6
0.8
1.0

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

# Read the audio file
sampling_freq, signal = wavfile.read('SD21063_generated_audio.wav')

# Normalize the values
signal = signal / np.power(2, 15)

# Extract the length of the audio signal
len_signal = len(signal)

# Extract the half length
len_half = np.ceil((len_signal + 1) / 2.0).astype(np.int)

# Apply Fourier transform
freq_signal = np.fft.fft(signal)

# Normalization
freq_signal = abs(freq_signal[0:len_half]) / len_signal

# Take the square
freq_signal **= 2

# Extract the length of the frequency transformed signal
len_fts = len(freq_signal)

# Adjust the signal for even and odd cases
if len_signal % 2:
    freq_signal[1:len_fts] *= 2
else:
    freq_signal[1:len_fts-1] *= 2

# Extract the power value in dB (unit for signal power)
signal_power = 10 * np.log10(freq_signal)

# Build the X axis
x_axis = np.arange(0, len_half, 1) * (sampling_freq / len_signal) / 1000.0

# Increase the figure size
plt.figure(figsize=(10, 6))

# Plot the figure
plt.plot(x_axis, signal_power, color='black')
plt.xlabel('Frequency (kHz)')
plt.ylabel('Signal power (dB)')
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_19928\3857506365.py:15: Deprecat
ionWarning: `np.int` is a deprecated alias for the builtin `int`. To silen
ce this warning, use `int` by itself. Doing this will not modify any behav
ior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int
64` or `np.int32` to specify the precision. If you wish to review your cu
rrent use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.or
g/devdocs/release/1.20.0-notes.html#deprecations (https://numpy.org/devdoc
s/release/1.20.0-notes.html#deprecations)
  len_half = np.ceil((len_signal + 1) / 2.0).astype(np.int)
```



In [ ]:

```
1
```