20PW39

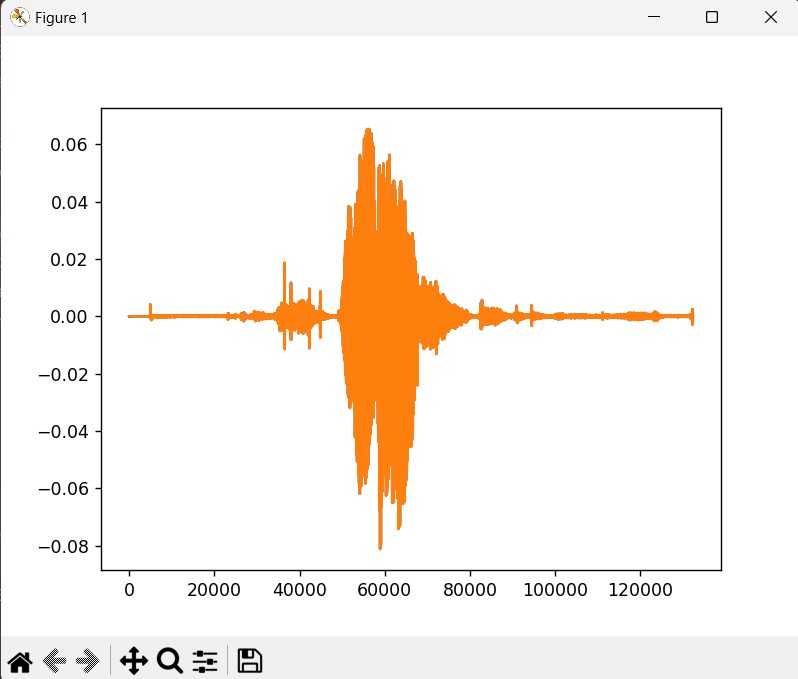
**Task 1: Record your voice by saying "Hello" and save it as .wav file. Read it and display the voice.**

Recording and saving the audio file

import sounddevice as sd  
from scipy.io.wavfile import write  
  
duration = 3   
freq = 44100   
  
print("Speak now...")  
recording = sd.rec(int(duration \* freq), samplerate=freq, channels=2)  
sd.wait()   
  
write("Hello.wav", freq, recording)  
  
print("Recording saved as Hello.wav")

Reading and displaying the voice

import sounddevice as sd  
from scipy.io.wavfile import read  
import matplotlib.pyplot as plt  
  
fs, x = read('E:\SEM\_9\pythonProject\Hello.wav')   
  
plt.plot(x)  
plt.show()   
  
sd.play(x, fs)  
sd.wait()

The audio file is attached separately and the plot I got is:  


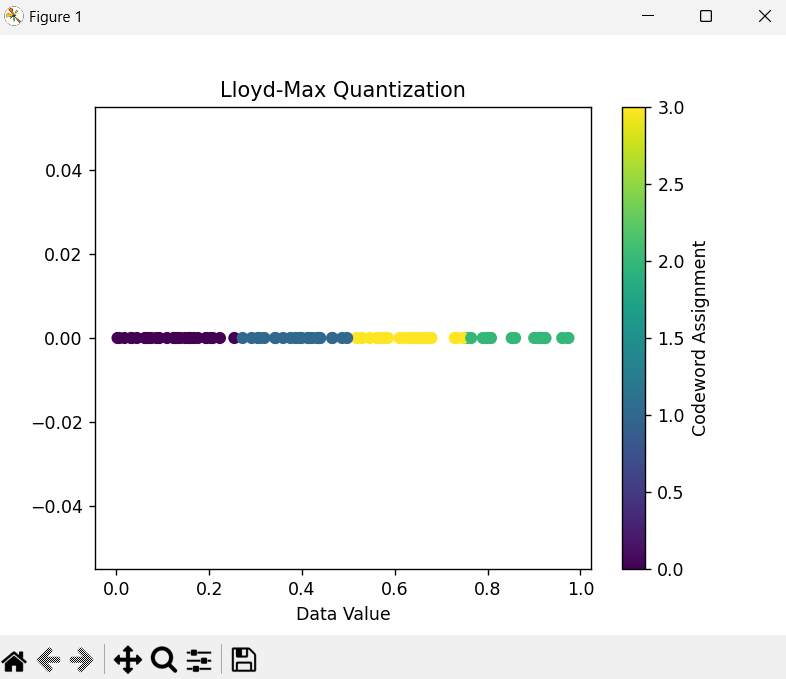
**Task 2: Read Lloyd-Max quantization algorithm.**

Lloyd-Max quantization (LMQ) is a technique used to compress data by reducing the number of bits used to represent it. It works by finding a set of representative values (codewords) that minimize the distortion introduced when replacing the original data points with the nearest codeword.

1. Initialization: You define the number of codewords (bins) you want to use.
2. Iteration:
   * Assign each data point to the codeword closest to it (encoding).
   * Calculate the average of the data points assigned to each codeword (centroid).
   * Update the codewords with the newly calculated centroids.
3. Repeat step 2: Keep iterating until the codewords no longer change significantly (convergence).

For the demonstration purpose, random numbers are generated and assigned to different bins.

import numpy as np  
import matplotlib.pyplot as plt  
  
def lloyd\_max(data, num\_codewords, iterations):  
 # Evenly spaced initialization  
 codewords = np.linspace(min(data), max(data), num\_codewords + 1)[1:-1]  
  
 for \_ in range(iterations):  
 # Assign data points to closest codewords  
 assignments = np.argmin(np.abs(data[:, np.newaxis] - codewords), axis=1)  
  
 # Update codewords with centroids  
 new\_codewords = []  
 for i in range(num\_codewords):  
 data\_points = data[assignments == i]  
 if len(data\_points) > 0:  
 new\_codewords.append(data\_points.mean())  
 else:  
 # Use a different strategy for the last codeword (or any)  
 if i == num\_codewords - 1:  
 new\_codewords.append(codewords[i - 1])  
 else:  
 new\_codewords.append(np.median(data))  
 codewords = np.array(new\_codewords)  
 codewords = np.sort(codewords)  
  
 print("Original Data:")  
 print(data)  
 print("\nTransformed Data (Codeword Assignments):")  
 print(assignments)  
  
 plt.scatter(data, np.zeros\_like(data), c=assignments)  
 plt.colorbar(label='Codeword Assignment')  
 plt.xlabel('Data Value')  
 # Placeholder for coloring  
 plt.title('Lloyd-Max Quantization')  
 plt.show()  
  
data = np.random.rand(100)  
  
# Perform LMQ with 4 codewords and 10 iterations  
lloyd\_max(data.copy(), 4, 10) # Use a copy to avoid modifying original data



**Task 3**: **Read the "cameraman image" and store it in different file formats like "jpg", "png","bmp","tif". Analyze how much space is required to store the image in different file formats. Plot the storage space against file format as a bar plot.**

from google.colab import drive

import cv2

import matplotlib.pyplot as plt

drive.mount('/content/drive')

image\_path = '/content/drive/My Drive/cameraman.png'

image = cv2.imread(image\_path)

import os

cv2.imwrite('/content/drive/My Drive/cameraman.jpg', image)

cv2.imwrite('/content/drive/My Drive/cameraman.tif', image)

cv2.imwrite('/content/drive/My Drive/cameraman.png', image)

cv2.imwrite('/content/drive/My Drive/cameraman.bmp', image)

file\_sizes = {

    'jpg': os.path.getsize('/content/drive/My Drive/cameraman.jpg'),

    'tif': os.path.getsize('/content/drive/My Drive/cameraman.tif'),

    'png': os.path.getsize('/content/drive/My Drive/cameraman.png'),

    'bmp': os.path.getsize('/content/drive/My Drive/cameraman.bmp')

}

formats = list(file\_sizes.keys())

sizes = list(file\_sizes.values())

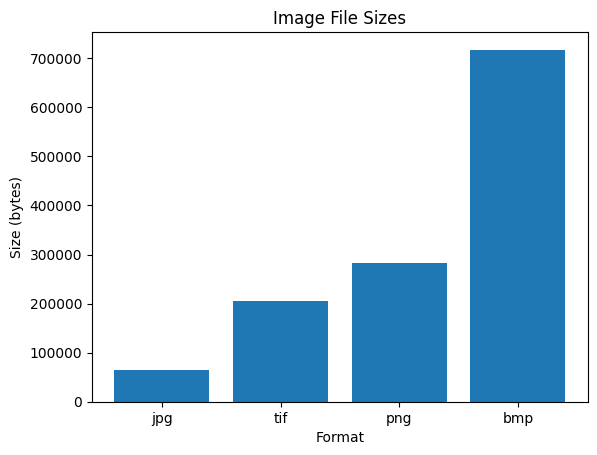
plt.bar(formats, sizes)

plt.xlabel('Format')

plt.ylabel('Size (bytes)')

plt.title('Image File Sizes')

plt.show()



**Task 4**: **Read a gray scale image (cameraman image). Write an algorithm to scramble all the pixels in the image. Display the input image and the scrambled image.**

import cv2

import matplotlib.pyplot as plt

import random

scrambled\_image = image.copy()

for i in range(image.shape[0]):

  for j in range(image.shape[1]):

    scrambled\_image[i, j] = image[random.randint(0, image.shape[0]-1), random.randint(0, image.shape[1]-1)]

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(image, cmap='gray')

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(scrambled\_image, cmap='gray')

plt.title('Scrambled Image')

plt.axis('off')

plt.show()

