```
#TEXT_PROCESSING


# import the necessary libraries
import nltk
import string
import re
def text_lowercase(text):
    return text.lower()

input_str = "Hey, did you know that the summer break is coming? Amazing right !! It's only
text_lowercase(input_str)
# Remove numbers
def remove_numbers(text):
    result = re.sub(r'\d+', '', text)
    return result

input_str = "There are 3 balls in this bag, and 12 in the other one."
remove_numbers(input_str)
# import the inflect library
import inflect
p = inflect.engine()

# convert number into words
def convert_number(text):
    # split string into list of words
    temp_str = text.split()
    # initialise empty list
    new_string = []

    for word in temp_str:
        # if word is a digit, convert the digit
        # to numbers and append into the new_string list
        if word.isdigit():
            temp = p.number_to_words(word)
            new_string.append(temp)

        # append the word as it is
        else:
            new_string.append(word)

    # join the words of new_string to form a string
    temp_str = ' '.join(new_string)
    return temp_str

input_str = 'There are 3 balls in this bag, and 12 in the other one.'
convert_number(input_str)
# remove punctuation
def remove_punctuation(text):
    return text.translate(translator)

input_str = "Hey, did you know that the summer break is coming? Amazing right !! It's only
remove_punctuation(input_str)
# remove whitespace from text
def remove_whitespace(text):
    return " ".join(text.split())

input_str = " we don't need the given questions"
remove_whitespace(input_str)
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# remove stopwords function
def remove_stopwords(text):
    stop_words = set(stopwords.words("english"))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word not in stop_words]
    return filtered_text

example_text = "This is a sample sentence and we are going to remove the stopwords from th
remove_stopwords(example_text)
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()

# stem words in the list of tokenized words
def stem_words(text):
    word_tokens = word_tokenize(text)
    stems = [stemmer.stem(word) for word in word_tokens]
    return stems

text = 'data science uses scientific methods algorithms and many types of processes'
stem_words(text)
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer = WordNetLemmatizer()
# lemmatize string
def lemmatize_word(text):
    word_tokens = word_tokenize(text)
    # provide context i.e. part-of-speech
    lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in word_tokens]
    return lemmas

text = 'data science uses scientific methods algorithms and many types of processes'
lemmatize_word(text)
```

```
['data',
 'science',
 'use',
 'scientific',
 'methods',
 'algorithms',
 'and',
 'many',
 'type',
 'of',
 'process']
```

Ψ

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
True
```

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

TEXT PROCESSING OF LARGE FILES

```
pip install PyPDF2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting PyPDF2
  Downloading PyPDF2-2.2.0-py3-none-any.whl (189 kB)
     |████████████████████████████████| 189 kB 5.2 MB/s
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-pac
Installing collected packages: PyPDF2
Successfully installed PyPDF2-2.2.0
```

```
import PyPDF2
```

```
from PyPDF2 import PdfReader

a = PdfReader("processing.pdf")
print(a.documentInfo)
```

```
     {'/Author': 'siemens', '/CreationDate': "D:20220610103712+05'30'", '/ModDate': "D:202
```

```
from PyPDF2 import PdfReader

a = PdfReader("processing.pdf")
print(a.getNumPages())
```

```
     18
```

```
from PyPDF2 import PdfReader

a = PdfReader("progit.pdf")
str=""
for i in range(1,11):
  str += a.getPage(i).extractText()

with open("text.txt","w",encoding='utf-8') as f:
  f.write(str)
```

```
     /usr/local/lib/python3.7/dist-packages/PyPDF2/_page.py:1278: PdfReadWarning:  impossi
       PdfReadWarning,
     /usr/local/lib/python3.7/dist-packages/PyPDF2/_page.py:1278: PdfReadWarning:  impossi
       PdfReadWarning,
```

```
from PyPDF2 import PdfReader, PdfWriter

reader = PdfReader("processing.pdf")
writer = PdfWriter()

# Add all pages to the writer
for page in reader.pages:
    writer.add_page(page)

# Add a password to the new PDF
writer.encrypt("123456")

# Save the new PDF to a file
with open("encrypted-pdf.pdf", "wb") as f:
    writer.write(f)
```

```
#decrypt pdf file
from PyPDF2 import PdfReader, PdfWriter

reader = PdfReader("encrypted-pdf.pdf")
writer = PdfWriter()

if reader.is_encrypted:
    reader.decrypt("my-secret-password")

# Add all pages to the writer
for page in reader.pages:
    writer.add_page(page)

# Save the new PDF to a file
with open("decrypted-pdf.pdf", "wb") as f:
    writer.write(f)
```

```
from PyPDF2 import PdfMerger

merger = PdfMerger()

for pdf in ["progit.pdf", "gfg.pdf"]:
    merger.append(pdf)

merger.write("merged-pdf.pdf")
merger.close()
```

```
from PyPDF2 import PdfWriter, PdfReader

reader = PdfReader("processing.pdf")
writer = PdfWriter()


writer.add_page(reader.pages[0])
writer.pages[0].rotate(90)

with open("rotated_page.pdf", "wb") as fp:
    writer.write(fp)
```

```
#reduce pdf size
#remove images
import PyPDF2
```

```
reader = PyPDF2.PdfReader("progit.pdf")
writer = PyPDF2.PdfWriter()

for page in reader.pages:
    writer.add_page(page)

writer.remove_images()

with open("out.pdf", "wb") as f:
    writer.write(f)


#compression
import PyPDF2

reader = PyPDF2.PdfReader("processing.pdf")
writer = PyPDF2.PdfWriter()

for page in reader.pages:
    page.compress_content_streams()
    writer.add_page(page)

with open("out2.pdf", "wb") as f:
    writer.write(f)



#Reading pdf annotaion
from PyPDF2 import PdfReader

reader = PdfReader("processing.pdf")

for page in reader.pages:
    if "/Annots" in page:
        for annot in page["/Annots"]:
            subtype = annot.get_object()["/Subtype"]
            if subtype == "/Text":
                print(annot.get_object()["/Contents"])


from PyPDF2 import PdfReader

reader = PdfReader("processing.pdf")

for page in reader.pages:
    if "/Annots" in page:
        for annot in page["/Annots"]:
            subtype = annot.get_object()["/Subtype"]
            if subtype == "/Highlight":
                coords = annot.get_object()["/QuadPoints"]
                x1, y1, x2, y2, x3, y3, x4, y4 = coords
                x1, y1, x2, y2, x3, y3, x4, y4 = coords


#cropping and transformming pdfs
from PyPDF2 import PdfWriter, PdfReader

reader = PdfReader("processing.pdf")
writer = PdfWriter()

# add page 1 from reader to output document, unchanged:
writer.add_page(reader.pages[0])

# add page 2 from reader, but rotated clockwise 90 degrees:
writer.add_page(reader.pages[1].rotate(90))

# add page 3 from reader, but crop it to half size:
page3 = reader.pages[2]
page3.mediabox.upper_right = (
    page3.mediabox.right / 2,
    page3.mediabox.top / 2,
)
writer.add_page(page3)

# add some Javascript to launch the print window on opening this PDF.
# the password dialog may prevent the print dialog from being shown,
# comment the the encription lines, if that's the case, to try this out:
writer.add_js("this.print({bUI:true,bSilent:false,bShrinkToFit:true});")

# write to document-output.pdf
with open("PyPDF2-output.pdf", "wb") as fp:
    writer.write(fp)


from google.colab import drive
drive.mount('/content/drive')


#AUDIO PROCESSING

#Downloading  audio files & installing required libraries

!wget -nc https://vvestman.github.io/summerschool19/sounds/Im_Superman.wav
!wget -nc https://vvestman.github.io/summerschool19/sounds/Count_Of_Three-8khz.wav

!pip install pysoundfile
!pip install bitstring
```

```
--2022-06-03 05:30:29--  https://vvestman.github.io/summerschool19/sounds/Im_Supermar
Resolving vvestman.github.io (vvestman.github.io)... 185.199.108.153, 185.199.109.153
Connecting to vvestman.github.io (vvestman.github.io)|185.199.108.153|:443... connect
HTTP request sent, awaiting response... 200 OK
Length: 823996 (805K) [audio/wav]
Saving to: 'Im_Superman.wav'
```

```
    Saving to: ‘Im_Superman.wav’

    Im_Superman.wav     100%[===================>] 804.68K  --.-KB/s    in 0.05s

    2022-06-03 05:30:29 (14.6 MB/s) - ‘Im_Superman.wav’ saved [823996/823996]

    --2022-06-03 05:30:29--  https://vvestman.github.io/summerschool19/sounds/Count_Of_Th
    Resolving vvestman.github.io (vvestman.github.io)... 185.199.108.153, 185.199.109.153
    Connecting to vvestman.github.io (vvestman.github.io)|185.199.108.153|:443... connect
    HTTP request sent, awaiting response... 200 OK
    Length: 50384 (49K) [audio/wav]
    Saving to: ‘Count_Of_Three-8khz.wav’

    Count_Of_Three-8khz 100%[===================>]  49.20K  --.-KB/s    in 0.01s

    2022-06-03 05:30:30 (3.72 MB/s) - ‘Count_Of_Three-8khz.wav’ saved [50384/50384]

    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
    Collecting pysoundfile
      Downloading PySoundFile-0.9.0.post1-py2.py3-none-any.whl (24 kB)
    Requirement already satisfied: cffi>=0.6 in /usr/local/lib/python3.7/dist-packages (f
    Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (f
    Installing collected packages: pysoundfile
    Successfully installed pysoundfile-0.9.0.post1
    Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
    Collecting bitstring
      Downloading bitstring-3.1.9-py3-none-any.whl (38 kB)
    Installing collected packages: bitstring
    Successfully installed bitstring-3.1.9
```

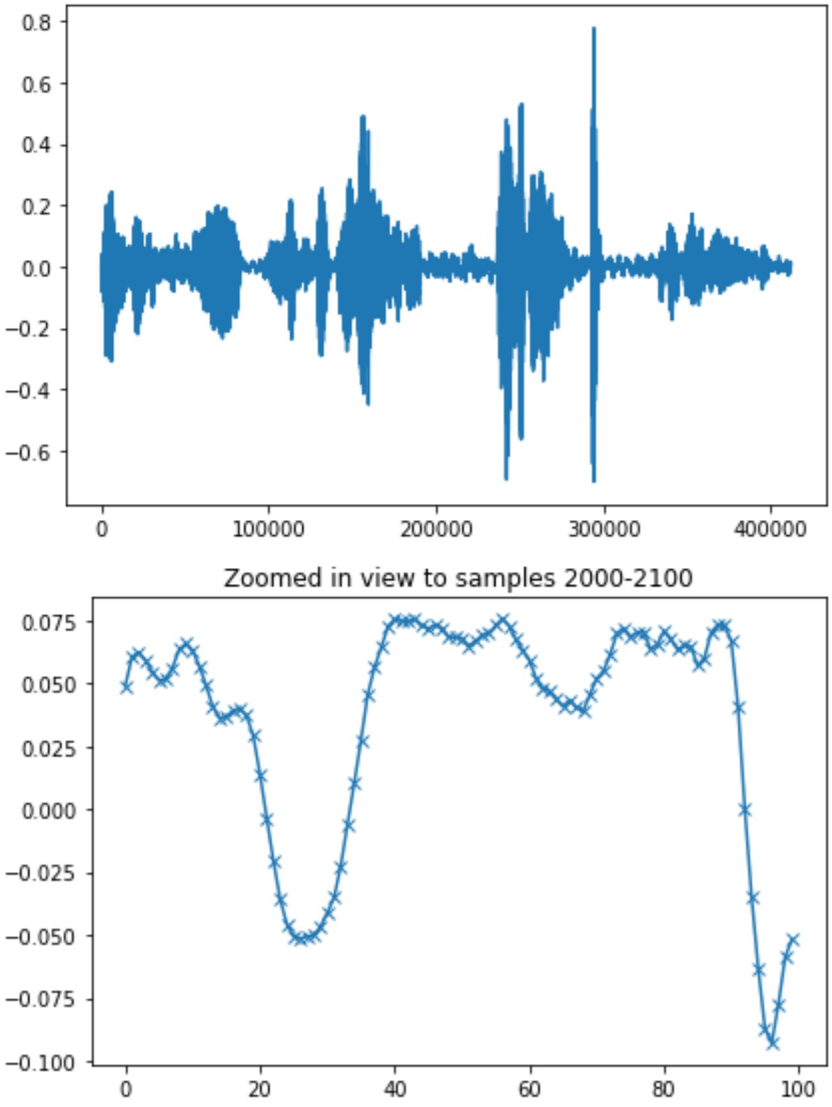
```
#PLAYING AUDIO
import IPython
IPython.display.Audio('Im_Superman.wav')
```

○ 0:00 / 0:09 ○

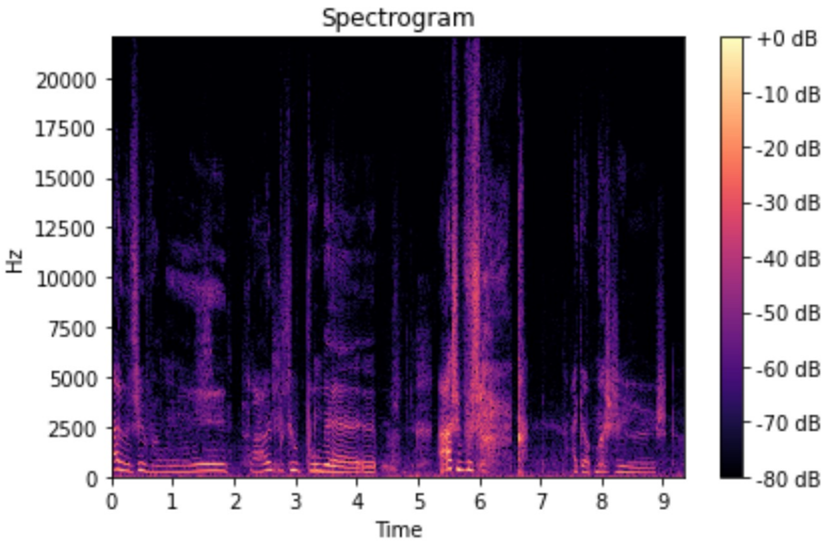
```
#Plotting the audio signal
import soundfile
import matplotlib.pyplot as plt
audio_signal, sampling_rate = soundfile.read('Im_Superman.wav')
print('Sampling rate: {} samples/second'.format(sampling_rate))
print('Signal size: {} samples'.format(audio_signal.shape[0]))
print('Signal duration: {:.3f} seconds'.format(audio_signal.shape[0] / sampling_rate))
plt.plot(audio_signal)
plt.tight_layout()
plt.figure()
plt.plot(audio_signal[2000:2100], marker='x')
plt.title('Zoomed in view to samples 2000-2100')
plt.tight_layout()
```

```
    Sampling rate: 44100 samples/second
```

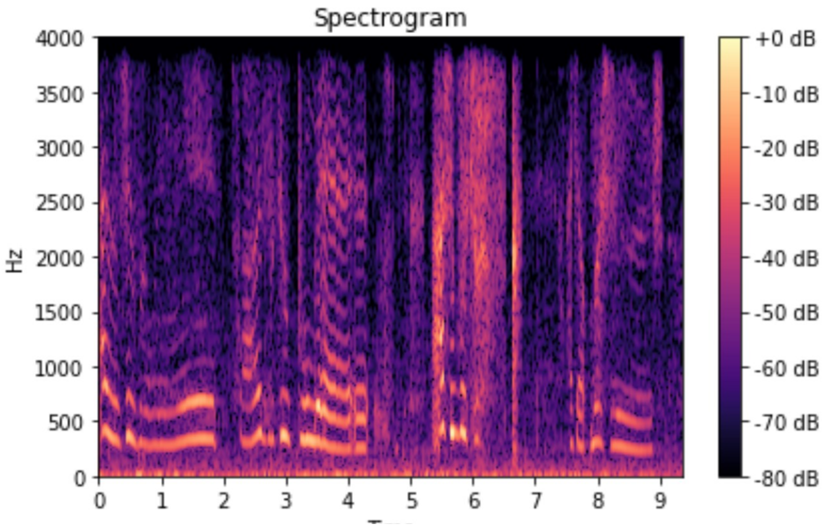
```
Signal size: 411889 samples
Signal duration: 9.340 seconds
```



```
#Using short time Fourier transform to obtain magnitude spectrogram of speech
#Short time Fourier transform (STFT) splits signal into small frames (25ms), so that conse
import numpy as np
import librosa
from librosa.display import specshow

window_length = int(0.025 * sampling_rate)
hop_length = int(0.01 * sampling_rate)

spectrogram = np.abs(librosa.stft(audio_signal, hop_length=hop_length, win_length=window_le

# Plotting the spectrogram:
specshow(librosa.amplitude_to_db(spectrogram, ref=np.max), sr=sampling_rate, hop_length=ho
plt.title('Spectrogram')
plt.colorbar(format='%+2.0f dB')
plt.tight_layout()
```



```
#In the above code, spectrogram is a 2D numpy array. The size of the array is printed belo
print(spectrogram.shape)

     (1025, 934)


#Resampling audio

#The loaded audio file is sampled at 44.1 kHz. Let's resample the audio to 8 kHz:
audio_signal = librosa.resample(audio_signal, sampling_rate, 8000)
sampling_rate = 8000

window_length = int(0.025 * sampling_rate)
hop_length = int(0.01 * sampling_rate)

spectrogram = np.abs(librosa.stft(audio_signal, hop_length=hop_length, win_length=window_le
librosa.display.specshow(librosa.amplitude_to_db(spectrogram, ref=np.max), sr=sampling_rat
plt.title('Spectrogram')
plt.colorbar(format='%+2.0f dB')
plt.tight_layout()
```



time

```
#Audio steganography with the least significant bit (LSB) coding
#The idea is to embed hidden data (secret message) into a speech file (carrier). After embe

#Let "Im_Superman.wav" be the carrier and let "Count_Of_Three-8khz.wav" be the secret messa
carrier, carrier_sr = soundfile.read('Im_Superman.wav', dtype=np.int16)
message, message_sr = soundfile.read('Count_Of_Three-8khz.wav', dtype=np.int16)

message = np.hstack((message, message, message, message, message))

IPython.display.Audio('Count_Of_Three-8khz.wav')
```

0:00 / 0:03

```
#A function that embeds data to the least significant bits of the carrier signal:
from bitstring import Bits

def lsb_embed(carrier, data, n_bits=1):
  # Assumes that both carrier and data have dtype of int16

  # Convert all integer values of secret message to binary strings:
  secret_bits = []
  for value in np.nditer(data):
    secret_bits.append(np.binary_repr(value, 16))

  # Join all binary strings together
  secret_bits = ''.join(secret_bits)

  # Ensure that the length of binary string is the same as the size of carrier
  secret_bits = secret_bits.ljust(carrier.size * n_bits, '0')[:carrier.size * n_bits]

  # Modify the least significant bits of carrier to contain hidden data
  audio_with_hidden_data = np.zeros(carrier.shape, dtype=carrier.dtype)
  for i in range(len(carrier)):
    # Convert ith value of carrier to binary string:
    binary_string = np.binary_repr(carrier[i], 16)
    # Set the last bit of the binary string to be a bit from the secret message:
    altered_binary = binary_string[:-n_bits] + secret_bits[i*n_bits:i*n_bits+n_bits]
    audio_with_hidden_data[i] = Bits(bin=altered_binary).int # Binary string to int

  return audio_with_hidden_data


#Next, we hide a message using the above function; then save the stego audio (audio with a
audio_with_hidden_data = lsb_embed(carrier, message, 10)
soundfile.write('audio_with_hidden_message.wav', audio_with_hidden_data, carrier_sr)
IPython.display.Audio('audio_with_hidden_message.wav')
```

○ 0:00 / 0:09 ○

```
#Does it sound different than the original file?
IPython.display.Audio('Im_Superman.wav') # Original wav file
```

○ 0:00 / 0:09 ○

```
#A function that retrieves the embedded hidden data:
def lsb_retrieve(signal, n_bits=1):

  # Collect the least significant bits of the 'stego' signal
  secret_bits = []
  for value in np.nditer(signal):
    ls_bit = np.binary_repr(value, 16)[-n_bits:]
    secret_bits.append(ls_bit)

  # Join bits together to form a binary string
  secret_bits = ''.join(secret_bits)

  # Ensure that the length of binary string is divisable by 16
  secret_bits = secret_bits[:-(len(secret_bits) % 16)]

  # Convert chunks of 16 consecutive bits to 16 bit integers to retreive the secret data
  retrieved_audio = np.zeros(len(secret_bits) // 16, dtype=np.int16)
  for i in range(retrieved_audio.size):
    retrieved_audio[i] = Bits(bin=secret_bits[i*16:(i+1)*16]).int

  return retrieved_audio


retrieved_hidden_message = lsb_retrieve(audio_with_hidden_data, 10)
soundfile.write('retrieved_hidden_message.wav', retrieved_hidden_message, message_sr)
IPython.display.Audio('retrieved_hidden_message.wav')
```

○ 0:00 / 0:32 ○

```
#instead of using only the least significant bit to embed data, try using two, three, or mo
```