SOFTWARE NOW
S224 HIT137

Assignment 1
Group SYD 301
Semester 2, 2024

Lecturer: Lucia Carrion Gordon

Submitted By:

Rupesh Pun :s378248@students.cdu.edu.au
Susmita Khadka :s378032@students.cdu.edu.au
Surat Adhikari : s376778@students.cdu.edu.au
Nabin Oli :s377309@students.cdu.edu.au
Shiva Raj Bhurtel :s374134@students.cdu.edu.au

Question 1

This question consists of multiple CSV files (In the Zipped Folder) with 'large texts' in one of the columns in each file. Your job is to use the open-source NLP (Natural Language Processing) libraries and perform various tasks.

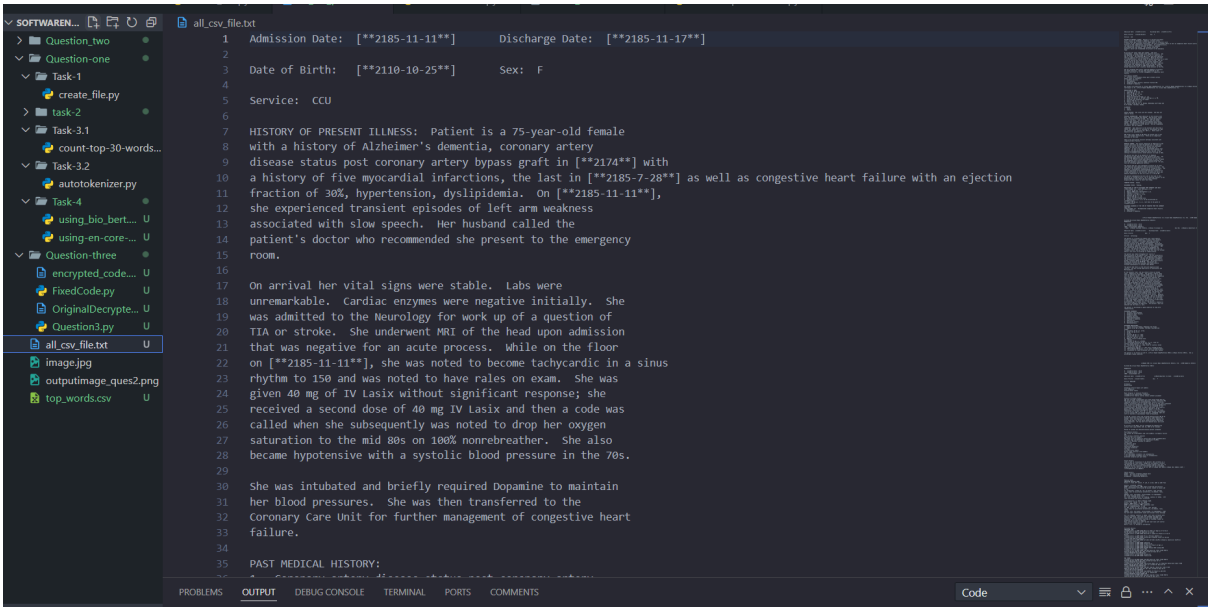
Task 1: Extract the 'text' in all the CSV files and store them into a single '.txt file'.

Answer:

Code:

```
import pandas as pd

# List of CSV file names and thier text columns
csv_files = [('CSV1.csv', 'SHORT-TEXT'), ('CSV2.csv', 'TEXT'),
('CSV3.csv','TEXT'), ('CSV4.csv','TEXT')]

# List to store text information from all files
all_texts = []

# Read each CSV file and extract text information
for (file, text_column) in csv_files:
    # Read CSV file into a DataFrame
    df = pd.read_csv(file)

    # Assuming 'text' is the column containing text information
    if text_column in df.columns:
        # Extract text information and append to the list
        all_texts.extend(df[text_column].astype(str).tolist())

# Write text information to a new text file
output_file = 'all_csv_file.txt'
with open(output_file, 'w', encoding='utf-8') as f:
    for text in all_texts:
        f.write(text + '\n')

print(f'Text information written to {output_file}')
```

<u>Output:</u>



Task 2: Research Install the libraries(SpaCy – scispaCy – ‘en_core_sci_sm’/’en_ner_bc5cdr_md’). Install the libraries (Transformers (Hugging Face) - and any biomedical model (BioBert) that can detect drugs, diseases, etc from the text).

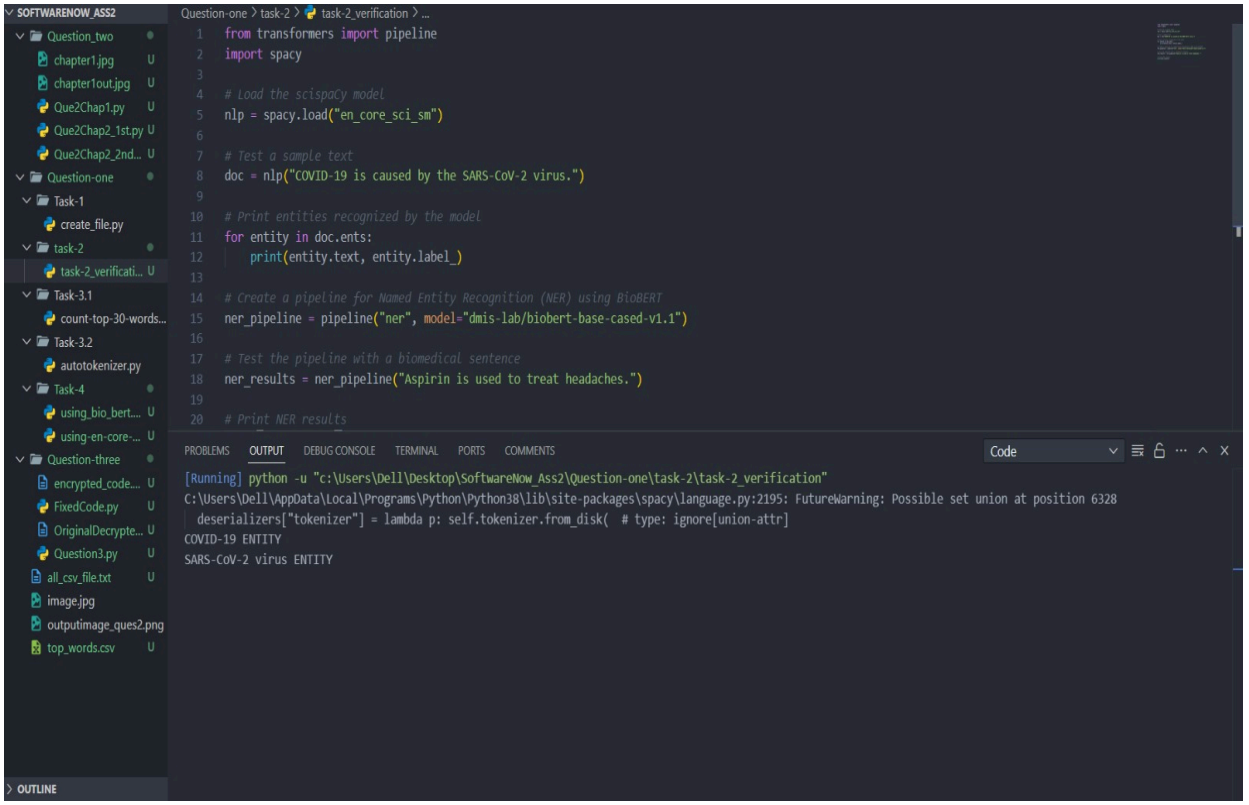
Answer:

<u>Installation Verification Code</u>

```
from transformers import pipeline
import spacy

# Load the scispaCy model
nlp = spacy.load("en_core_sci_sm")

# Test a sample text
doc = nlp("COVID-19 is caused by the SARS-CoV-2 virus.")

# Print entities recognized by the model
for entity in doc.ents:
    print(entity.text, entity.label_)

# Create a pipeline for Named Entity Recognition (NER) using BioBERT
ner_pipeline = pipeline("ner", model="dmis-lab/biobert-base-cased-v1.1")

# Test the pipeline with a biomedical sentence
ner_results = ner_pipeline("Aspirin is used to treat headaches.")
```

```
# Print NER results
print(ner_results)
```

Output:



Task 3: Programming and Research

3.1: Using any in-built library present in Python, count the occurrences of the words in the text (.txt) and give the 'Top 30' most common words. And store the 'Top 30' common words and their counts into a CSV file.

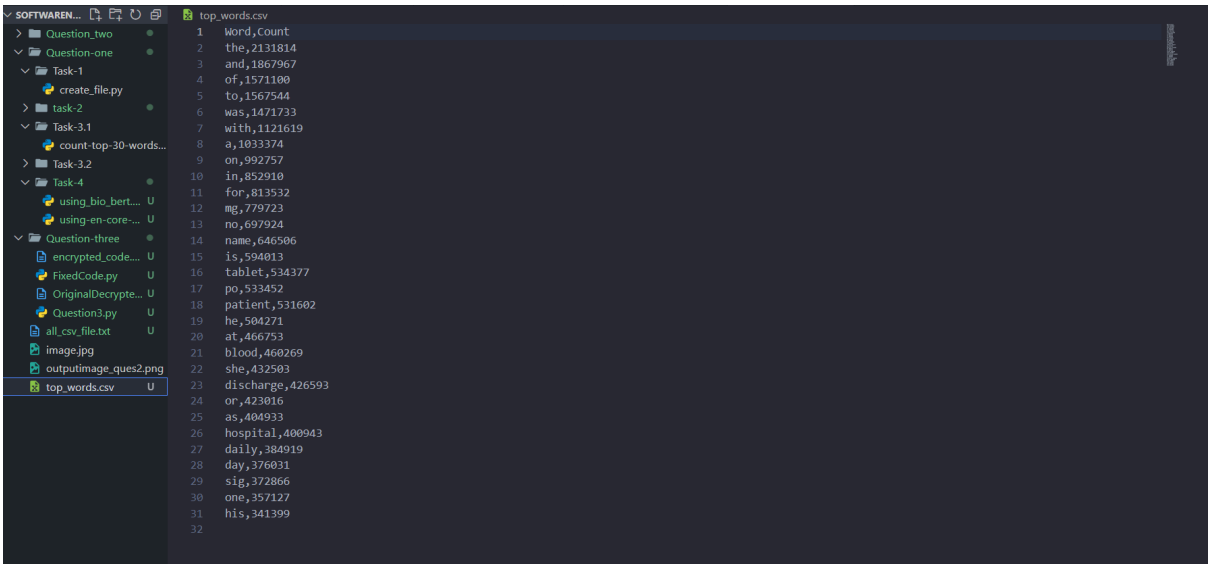
Answer:

Code

```
import csv
from collections import Counter
import re

def extract_top_words(file_path, output_csv_path, top_n=30):
    # Read the text file
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()

    # Preprocess the text (remove non-alphanumeric characters and
convert to lowercase)
```

```
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    words = text.lower().split()

    # Count word occurrences using counter class
    word_counts = Counter(words)

    # Get the top 30 words
    top_words = word_counts.most_common(top_n)

    # Save the top words to a CSV file
    with open(output_csv_path, 'w', newline='', encoding='utf-8') as
csv_file:
        csv_writer = csv.writer(csv_file)
        csv_writer.writerow(['Word', 'Count'])  # Header row

        for word, count in top_words:
            csv_writer.writerow([word, count])

# Example usage
file_path = 'all_csv_file.txt'
output_csv_path = 'top_words.csv'
extract_top_words(file_path, output_csv_path)
```

Output:



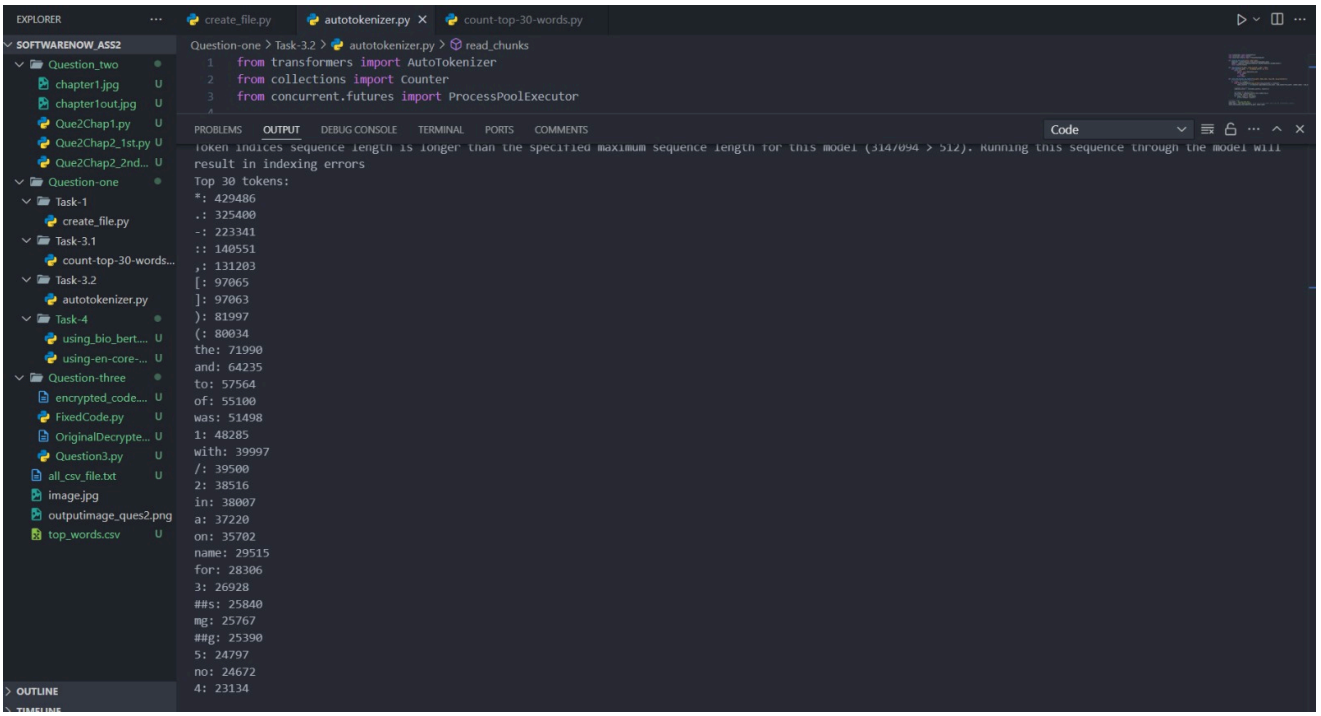
3.2: Using the ‘Auto Tokenizer’ function in the ‘Transformers’ library, write a ‘function’ to count unique tokens in the text (.txt) and give the ‘Top 30’ words. Answer:

Code

```
from transformers import AutoTokenizer
from collections import Counter
from concurrent.futures import ProcessPoolExecutor

def tokenize_and_count(chunk, model_name):
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    tokens =
tokenizer.tokenize(tokenizer.decode(tokenizer.encode(chunk)))
    return Counter(tokens)

def read_chunks(file_path, chunk_size=10 * 1024 * 1024):
    with open(file_path, 'r', encoding='utf-8') as file:
        while True:
            chunk = file.read(chunk_size)
            if not chunk:
                break
            yield chunk

def count_and_display_top_tokens(file_path, model_name, top_n=30,
num_processes=2):
    # Tokenize and count in parallel
    if __name__ == '__main__':
        with ProcessPoolExecutor(max_workers=num_processes) as
executor:
```

```python
            token_counters = list(executor.map(tokenize_and_count,
read_chunks(file_path), [model_name] * num_processes))

        # Combine the results
        combined_counter = sum(token_counters, Counter())

        # Display the top N tokens
        top_tokens = combined_counter.most_common(top_n)
        print(f"Top {top_n} tokens:")
        for token, count in top_tokens:
            print(f"{token}: {count}")

# Example usage:
file_path = 'all_csv_file.txt'
model_name = 'bert-base-uncased'  # You can use any model name from the
transformers library
count_and_display_top_tokens(file_path, model_name)
```

Output



Task 4: Named-Entity Recognition (NER) Extract the ‘diseases’, and ‘drugs’ entities in the ‘.txt file’ separately using ‘en_core_sci_sm’/’en_ner_bc5cdr_md’ and biobert. And compare the differences between the two models (Example: Total entities

detected by both of them, what's the difference, check for most common words, and check the difference.)

Answer:

Code

Using bio_bert

```
from transformers import BertTokenizer, BertForTokenClassification
import torch

# Load BioBERT model and tokenizer
tokenizer =
BertTokenizer.from_pretrained('monologg/biobert_v1.1_pubmed',
do_lower_case=False)
model =
BertForTokenClassification.from_pretrained('monologg/biobert_v1.1_pubme
d', num_labels=2)  # Assuming 2 labels for 'DISEASE' and 'DRUG'
print(tokenizer.convert_ids_to_tokens(range(model.config.num_labels)))
# Function to extract entities from a text
def extract_entities(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True)
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=2)

    # Map token predictions to entities
    entities = []
    for token, prediction in zip(inputs["input_ids"][0],
predictions[0]):
        token_str = tokenizer.convert_ids_to_tokens(token.item())
        label = 'DISEASE' if torch.eq(prediction, torch.tensor(1)) else
'DRUG' if torch.eq(prediction, torch.tensor(0)) else 'O'

        if label != 'O':
            entities.append((token_str, label))

    return entities

# Process the text file
file_path = 'all_csv_file.txt'  # Replace with the actual path to your
text file

with open(file_path, 'r', encoding='utf-8') as file:
    text = file.read()
    entities = extract_entities(text)
```

```
# Separate 'diseases' and 'drugs'
diseases = [entity[0] for entity in entities if entity[1] == 'DISEASE']
drugs = [entity[0] for entity in entities if entity[1] == 'DRUG']

# Print the results
print("Diseases:", diseases)
print("Drugs:", drugs)
```

Using en_core_sm

```
import spacy
from concurrent.futures import ThreadPoolExecutor

# Load the spaCy model
nlp = spacy.load("en_core_sci_sm")
nlp.disable_pipes('parser', 'ner')

# Function to process a batch of text within a chunk
def process_batch(chunk):
    docs = nlp.pipe(chunk, disable=["parser", "ner"])

    # Extract disease and drug entities
    diseases = []
    drugs = []

    for doc in docs:
        diseases.extend([ent.text for ent in doc.ents if ent.label_ ==
"DISEASE"])
        drugs.extend([ent.text for ent in doc.ents if ent.label_ ==
"CHEMICAL"])

    return diseases, drugs

# Function to process a large text file in batches
def process_large_text_file(file_path, chunk_size=10 * 1024 * 1024,
batch_size=10):
    with open(file_path, "r", encoding="utf-8") as file:
        # Read the file in chunks to avoid loading the entire file into
memory
```

```
        chunks = []
        while True:
            chunk = file.read(chunk_size)
            if not chunk:
                break  # End of file
            chunks.append(chunk)
            if len(chunks) == batch_size:
                yield chunks
                chunks = []

        # Yield the remaining chunks
        if chunks:
            yield chunks

# Process each batch of chunks concurrently
def process_batches(batches):
    with ThreadPoolExecutor() as executor:
        results = list(executor.map(process_batch, batches))

    # Aggregate results
    all_diseases = [disease for result in results for disease in
result[0]]
    all_drugs = [drug for result in results for drug in result[1]]

    # Print the extracted diseases and drugs
    print("Diseases:", all_diseases)
    print("Drugs:", all_drugs)

    return all_diseases, all_drugs

# Example usage:
file_path = 'all_csv_file.txt'
chunk_size = 100000  # Adjust as needed
batch_size = 10
batches = process_large_text_file(file_path, chunk_size=chunk_size,
batch_size=batch_size)

# Process batches
process_batches(batches)
```

Question 2 : The Quest for the Hidden Treasure:

Chapter 1: The Gatekeeper :

```
import time

current_time = int(time.time())

generated_number = (current_time % 100) + 50

if generated_number % 2 == 0:
    generated_number += 10

print(generated_number)
```

The above algorithm generates a number (n).
You should use this number to change the pixels (r,g,b) in the provided image (Chapter1.png) by adding the original pixel values (r,g,b) with the generated number (Example: (r+n, g+n, b+n)).
Generate a new image with the converted pixels (upload it as 'chapter1out.png').
Finally, add all the red (r) pixel values in the new_image and provide the sum as output to move to the next chapter.
Answer
Code:

```
from PIL import Image
# Generate the random number as described in the prompt
import time

current_time = int(time.time())
generated_number = (current_time % 100) + 50
if generated_number % 2 == 0:
   generated_number += 10
print("generated Number:",generated_number)

# Open the image
img = Image.open("chapter1.jpg")
# Get the image size
width, height = img.size
# Create a new image with the same size
new_img = Image.new("RGB", (width, height))
#Iterate through each pixel and modify its RGB values
for x in range(width):
```

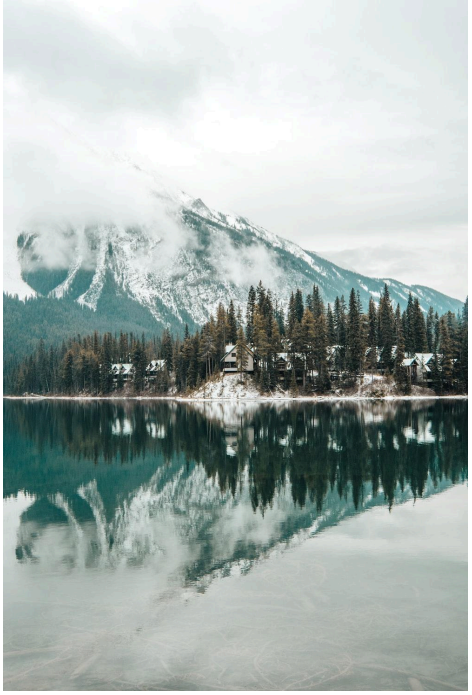
```
    for y in range(height):
        r, g, b = img.getpixel((x, y))
        new_r = min(255, r + generated_number)  # Ensure values don't exceed 255
        new_g = min(255, g + generated_number)
        new_b = min(255, b + generated_number)
        new_img.putpixel((x, y), (new_r, new_g, new_b))

# Save the modified image
new_img.save("chapter1out.jpg")
print("new image is save as the chapter1out.jpg")
# Calculate the sum of red pixel values
red_sum = 0
for x in range(width):
    for y in range(height):
        r, _, _ = img.getpixel((x, y))
        red_sum += r
print("Sum of red pixel values:", red_sum)
```

Output:

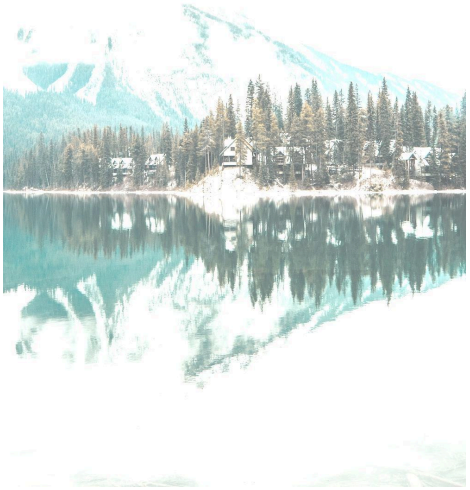
```
PROBLEMS    OUTPUT    TERMINAL    PORTS

TERMINAL
PS E:\Software_Now>  & 'c:\Users\Lotus\AppData\Local\Progra
24.10.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugp
generated Number: 60
new image is save as the chapter1out.jpg
Sum of red pixel values: 248980023
PS E:\Software_Now>
```

Image output:

Original image: “chapter1.jpg”



New image:“chapterout1.jpg”



Chapter 2:

- Assume s is a string. Write a program that separates a long string (at least length of 16) that contains both numbers and letters (upper and lower case) into two substrings of numbers and letters. And then convert the even numbers in the 'number substring' and upper-case letter in the 'letter string' into ASCII Code Decimal Values.
  Example Scenario: String = '56aAww1984sktr235270aYmn145ss785fsq3100'
  Separate them 56198235270145785310 (number string) and aAwwsktraYmnssfsqD (letter string).
  Answer:
  <u>Code:</u>

```
def separate_and_convert(s):

    # Separate numbers and letters

    number_string = ''.join([char for char in s if char.isdigit()])

    letter_string = ''.join([char for char in s if char.isalpha()])



    # Extract even numbers and convert to ASCII code decimal values

    even_numbers = [int(num) for num in number_string if int(num) % 2 == 0]

    even_numbers_ascii = [ord(str(num)) for num in even_numbers]



    # Extract uppercase letters and convert to ASCII code decimal values

    uppercase_letters = [char for char in letter_string if char.isupper()]

    uppercase_ascii = [ord(char) for char in uppercase_letters]



    return number_string, letter_string, even_numbers, even_numbers_ascii,
uppercase_letters, uppercase_ascii



# Example usage
```

```python
s = '56aAww1984sktr235270aYmn145ss785fsq31D0'

number_string, letter_string, even_numbers, even_numbers_ascii,
uppercase_letters, uppercase_ascii = separate_and_convert(s)

print("Original Text:",s)

print(f"Number string: {number_string}")

print(f"Letter string: {letter_string}")

print(f"Even numbers: {even_numbers}")

print(f"ASCII Code of even numbers: {even_numbers_ascii}")

print(f"Uppercase letters: {uppercase_letters}")

print(f"ASCII Code of uppercase letters: {uppercase_ascii}")
```

<u>Output:</u>

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Software_Now>  & 'c:\Users\Lotus\AppData\Local\Programs\Python\Pytho
gpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launche
Original Text: 56aAww1984sktr235270aYmn145ss785fsq31D0
Number string: 561984235270145785310
Letter string: aAwwsktraYmnssfsqD
Even numbers: [6, 8, 4, 2, 2, 0, 4, 8, 0]
ASCII Code of even numbers: [54, 56, 52, 50, 50, 48, 52, 56, 48]
Uppercase letters: ['A', 'Y', 'D']
ASCII Code of uppercase letters: [65, 89, 68]
PS E:\Software_Now>
```

- You are required to create a program that showcases the required output for the following question: Many newspapers publish a cryptogram each day, **for instance:**
  VZ FRYSVFU VZCNGVRAG NAQ N YVGGYR VAFRPHER V ZNXR ZVFGNXRF V NZ BHG BS PBAGEBY NAQNG GVZRE UNEQ GB UNAQYR

OHG VS LBH PNAG UNAQYR ZR NG ZL JBEFG GURA LBH FHER NF URYYQBAG QRFREIR ZR NG ZL ORFGN ZNEVYLA ZBAEBR
The deciphered cryptogram is usually a quote from a famous author or celebrity. To get the original quote, you should replace each character in the ciphered quote using a shift key value (s) condition.
Example 1: If ciphered quote is AB, and 's' is 1, then original quote is ZA
Example 2: If ciphered quote is AB, and 's' is 2, then original quote is YZ
Similarly decrypting the provided cryptogram using a 'certain' shift key value (s) gives the original quote. Find the shift key (s) the gives the original quote

Answer:
Code:

```
def decrypt_caesar_cipher(text, shift):
    decrypted = []

    for char in text:
        if char.isalpha():
            shift_value = 65 if char.isupper() else 97
            decrypted_char = chr((ord(char) - shift_value - shift) % 26 +
shift_value)
            decrypted.append(decrypted_char)
        else:
            decrypted.append(char)  # Keep non-alphabetic characters
unchanged

    return ''.join(decrypted)

def find_correct_shift(ciphered_text):
    for shift in range(1, 26):
        decrypted_text = decrypt_caesar_cipher(ciphered_text, shift)
        print(f"Shift {shift}:")
        print(decrypted_text)
        print()

# given ciphered text
ciphered_text = """VZ FRYSVFU VZCNGVRAG NAQ N YVGGYR
VAFRPHER V ZNXR ZVFGNXRF V NZ BHG BS PBAGEBY
```

```
NAONG GVZRF UNEQ GB UNAQYR OHG VS LBH PNAG UNAQYR ZR NG ZL JBEFG GURA LBH FHER NF URYYQBAG QRFREIR ZR NG ZL ORFG ZNEVYLA ZBAEBR"""

find_correct_shift(ciphered_text)
```

On looping the shift from 1 to 26 the relevant text we observed on 13th shift

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Shift 12:
JN TFMGJTI JNQBUJFOU BOE B MJUUMF JOTFDVSF J NBLF NJTUBLFT J BN PVU PG DPOUSPM
BOCBU UJNFT IBSE UP IBOEMF CVU JG ZPV DBOU IBOEMF NF BU NZ XPSTU UIFO ZPV TVSF BT IFMMEPOU EFTFSWF NF BU NZ CFTU NBSJMZO NPOSPF

Shift 13:
IM SELFISH IMPATIENT AND A LITTLE INSECURE I MAKE MISTAKES I AM OUT OF CONTROL
ANBAT TIMES HARD TO HANDLE BUT IF YOU CANT HANDLE ME AT MY WORST THEN YOU SURE AS HELLDONT DESERVE ME AT MY BEST MARILYN MONROE

Shift 14:
HL RDKEHRG HLOZSHDMS ZMC Z KHSSKD HMRDBTQD H LZJD LHRSZJDR H ZL NTS NE BNMSQNK
ZMAZS SHLDR GZQC SN GZMCKD ATS HE XNT BZMS GZMCKD LD ZS LX VNQRS SGDM XNT RTQD ZR GDKKCNMS CDRDQUD LD ZS LX ADRS LZQHKXM LNMQND

Shift 15:
GK QCJDGQF GKNYRGCLR YLB Y JGRRJC GLQCASPC G KYIC KGQRYICQ G YK MSR MD AMLRPMJ
YLZYR RGKCQ FYPB RM FYLBJC ZSR GD WMS AYLR FYLBJC KC YR KW UMPQR RFCL WMS QSPC YQ FCJJBMLR BCQCPTC KC YR KW ZCQR KYPGJWL KMLPMC

Shift 16:
FJ PBICFPE FJMXQFBKQ XKA X IFQQIB FKPBZROB F JXHB JFPQXHBP F XJ LRQ LC ZLKQOLI
XKYXQ QFJBP EXOA QL EXKAIB YRQ FC VLR ZXKQ EXKAIB JB XQ JV TLOPQ QEBK VLR PROB XP EBIIALKQ ABPBOSB JB XQ JV YBPQ JXOFIVK JLKOLB

Shift 17:
EI OAHBEOD EILWPEAJP WJZ W HEPPHA EJOAYQNA E IWGA IEOPWGAO E WI KQP KB YKJPNKH
WJXWP PEIAO DWNZ PK DWJZHA XQP EB UKQ YWJP DWJZHA IA WP IU SKNOP PDAJ UKQ OQNA WO DAHHZKJP ZAOANRA IA WP IU XAOP IWNEHUJ IKJNKA
```

At 13 th shift we have:
Shift 13:
IM SELFISH IMPATIENT AND A LITTLE INSECURE I MAKE MISTAKES I AM OUT OF CONTROL
ANBAT TIMES HARD TO HANDLE BUT IF YOU CANT HANDLE ME AT MY WORST THEN YOU SURE AS HELLDONT DESERVE ME AT MY BEST MARILYN MONROE

Question3: Fixing the error prone code:
Answer:
Code:

```
#the encrypted text is saved as encrypted_code.txt
#decrypted code is saved as decryted_code.txt
total = 0
for i in range(5):
    for j in range(3):
        if i + j == 5:
            total += i + j
        else:
            total -= i - j

counter = 0
while counter < 5:
    if total < 13:
        total += 1
    elif total > 13:
        total -= 1
    else:
        counter += 2

print("Counter:", counter)
print("Total:", total)

def encrypt(text, key):
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            shifted = ord(char) + key
            if char.islower():
                if shifted > ord('z'):
                    shifted -= 26
                elif shifted < ord('a'):
                    shifted += 26
            elif char.isupper():
                if shifted > ord('Z'):
```

```
                shifted -= 26
            elif shifted < ord('A'):
                shifted += 26
        encrypted_text += chr(shifted)
    else:
        encrypted_text += char
    return encrypted_text
# decrypting function: Taking the reference on the function of encrption
def decrypt(text, key):
    decrypted_text = ""
    for char in text:
        if char.isalpha():
            shifted = ord(char) - key
            if char.islower():
                if shifted > ord('z'):
                    shifted -= 26
                elif shifted < ord('a'):
                    shifted += 26
            elif char.isupper():
                if shifted > ord('Z'):
                    shifted -= 26
                elif shifted < ord('A'):
                    shifted += 26
            decrypted_text += chr(shifted)
        else:
            decrypted_text += char
    return decrypted_text

# Read the encrypted text from a file
with open("encrypted_code.txt", "r") as file:
    encrypted_code = file.read()

# Key value for decryption (you can set it based on the value you used for
encryption)
key = total # Example key, you can change this

# Decrypt the text
```

```python
decrypted_code = decrypt(encrypted_code, key)

# Save the decrypted text to a new file
with open("OriginalDecryptedCode.txt", "w") as output_file:
    output_file.write(decrypted_code)

print("Decryption complete. Decrypted code saved in 'OriginalDecryptedCode.txt'.")
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Software_Now>  & 'c:\Users\Lotus\AppData\Local\Programs\Python\Python312
10.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '59010' '-
Counter: 6
Total: 13
Decryption complete. Decrypted code saved in 'OriginalDecryptedCode.txt'.
PS E:\Software_Now>
```

OriginalDecryptedCode - Notepad

File Edit Format View Help

```
global_variable
100
my_dict = {'key1': 'value1', 'ke12': 'value2', 'ke13': 'value3']
def process_numbers():
global global_variable
local variable = 5
numbers [1, 2, 3, 4, 5]
while local variable > 0:
if local variable % 2 == 0: numbers.remove(local_variable)
local_variable - 1
return numbers
my_set = {1, 2, 3, 4, 5, 5, 4, 3, 2, 1} result- process_numbers (numbers=my_set)
def modify_dict():
local variable
10
my_dict['ke14'] = local_variable
modify_dict(5)
def update_global():
global global_variable
global variable + 10
for i in range(5):
print(i)
I+ 1
if m1_set is not None and my_dict['ke14'] == 10: print("Condition met!")
if 5 not in my_dict:
print("5 not found in the dictionary!")
print(global_variable)
print(m1_dict)
print(my_set)
```

The above code was fixed which is given below:

Code:

```
global_variable = 100  # Fixed global variable assignment
my_dict = {'key1': 'value1', 'ke12': 'value2', 'ke13': 'value3'}  # Fixed typo in the
dictionary closing bracket

my_set = {1, 2, 3, 4, 5}  # Fixed duplicate entries in the set (set can't have
duplicates)

def process_numbers():
    global global_variable  # Added 'global' to modify the global variable inside the
function
    local_variable = 5  # Fixed local variable assignment syntax (replaced '=' with ':=')
    numbers = [1, 2, 3, 4, 5]  # Added missing assignment operator for 'numbers'
```

```
    while local_variable > 0:  # Corrected the loop structure by adding ':' after the while statement
        if local_variable % 2 == 0:  # Fixed missing colon and condition to check for even numbers
            numbers.remove(local_variable)  # Corrected list removal logic
        local_variable -= 1  # Properly decremented 'local_variable' using '-=' instead of using just '-'

    return numbers  # Return the modified numbers list

# Call process_numbers and assign the result to my_set
my_set = process_numbers()  # Fixed function call syntax and ensured it was assigned to my_set

def modify_dict():
    local_variable = 10  # Fixed local variable assignment
    my_dict['ke14'] = local_variable  # Added a new key-value pair to my_dict

modify_dict()  # Fixed function call; previously the function was called with an argument, which wasn't required

def update_global():
    global global_variable  # Added 'global' keyword to modify the global variable
    global_variable += 10  # Corrected increment logic; used '+=' to add 10 to global_variable

    for i in range(5):  # Added colon for loop syntax
        print(i)  # Print current loop index

# Call update_global
update_global()

# Fixed condition to use 'my_set' instead of 'm1_set', and accessed 'ke14' safely using 'get' method
if my_set is not None and my_dict.get('ke14') == 10:  # Used .get() to safely access dictionary key
```

```
    print("Condition met!")

if 5 not in my_dict:  # Checking if the key '5' is not in my_dict
    print("5 not found in the dictionary!")

# Final print statements
print(global_variable)  # Print the updated global_variable
print(my_dict)  # Print the updated dictionary
print(my_set)  # Print the result of process_numbers
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Software_Now>  & 'c:\Users\Lotus\AppData\Local\Programs\Python\
gpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\la
0
1
2
3
4
Condition met!
5 not found in the dictionary!
110
{'key1': 'value1', 'ke12': 'value2', 'ke13': 'value3', 'ke14': 10}
[1, 3, 5]
PS E:\Software_Now>
```