Importing necessary libraries

In [ ]:
```python
import requests  # Importing the requests library to make HTTP requests
from bs4 import BeautifulSoup  # Importing BeautifulSoup for HTML parsing
```

Scrape the data from the link provided and extract the event details and iterate over each and every event description from html file

In [ ]:
```python
url = "https://www.lucernefestival.ch/en/program/summer-festival-24"  # URL of the webpage to scrape provided in the coding chall

# Fetch the HTML content from the URL
response = requests.get(url)
html_content = response.text

# Parse the HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, 'html.parser')


# Function to extract event details
def extract_event_details(soup):
    """
    Extract event details from the BeautifulSoup object.

    Args:
        soup (BeautifulSoup): The BeautifulSoup object containing parsed HTML content.

    Returns:
        list: A list of dictionaries, each containing event details.
    """
    event_details = []  # List to store extracted event details

    # Extracting event titles
    event_titles = soup.find_all('div', class_='event-content')
    for title in event_titles:
        event_details.append(title.text.strip())  # Append event title to the list

    return event_details
```

```
# Extract event details
event_details = extract_event_details(soup)

# Print or process the extracted event details
for event in event_details:
    print(event)  # Print each event title
```

Created a list called all_events which stores dictionaries in which Date Time Venue Title Artist_Name and Works are keys with their corresponding values from the website

In [ ]:

```
import re  # Importing the re module for regular expression operations

all_events = []  # List to store all extracted event details

def extract_event_details(input_text):
    """
    Extract event details from the input text using regular expressions.

    Args:
        input_text (str): The text containing event details.

    Returns:
        dict: A dictionary containing the extracted event details.
    """
    def extract_text_between_keywords(text, keyword1, keyword2):
        """
        Extract text between two keywords using regular expressions.

        Args:
            text (str): The input text.
            keyword1 (str): The starting keyword.
            keyword2 (str): The ending keyword.

        Returns:
            str: The text between the two keywords, or None if not found.
        """
        pattern = re.compile(f'{re.escape(keyword1)}(.*?){re.escape(keyword2)}', re.DOTALL)
        match = pattern.search(text)

        if match:
```

```
            return match.group(1).strip()
        else:
            return None

    # Define start and end keywords for extraction
    start_keyword = "Date and Venue"
    end_keyword = "Program"

    # Extract different parts of the event details using the defined keywords
    result = extract_text_between_keywords(input_text, start_keyword, end_keyword)
    result_artist = extract_text_between_keywords(input_text, '|', 'Date and Venue')
    result_works = extract_text_between_keywords(input_text, 'Program', 'Summer')
    result_title = extract_text_between_keywords(input_text, '\n', '|')  # Extracting title from the first line

    if result:
        # Split the result into tokens using '|' as delimiter
        tokens = result.split('|')

        # Remove leading and trailing whitespaces from each token
        tokens = [token.strip() for token in tokens if token]

        # Create the event dictionary
        event_dict = {
            "Date": tokens[0],
            "Time": tokens[1],
            "Venue": tokens[2],
            "Title": result_title.splitlines()[0],  # Use the extracted title
            "Artist_Name": result_artist,
            "Works": result_works
        }

        return event_dict
    else:

        event_dict = {
            "Date": "",
            "Time": "",
            "Venue": "",
            "Title": "",  # Use the extracted title
            "Artist_Name": "",
            "Works": ""
        }
        return event_dict
```

```
# Iterate over each event detail and extract its details
for event in event_details:
    # Extract event details using the defined function
    event_dict = extract_event_details(event)

    # Append the extracted event details to the list
    all_events.append(event_dict)

    # Print the extracted event details
    print(event_dict)
    print(len((event_dict)))
```

Printing all_events list od dictionaries which contains "Date","Time","Venue","Title", "Artist_Name" and "Works" as key elements and corresponding values are fetched from the webscript

In [ ]:
```
print(all_events)
```

In [ ]:
```
print(len(all_events))
```

In [ ]:
```
# Fetch HTML content from the URL
response = requests.get(url)
html_content = response.text

# Parse HTML content using BeautifulSoup
soup = BeautifulSoup(html_content, 'html.parser')

# Find all <picture> elements with class 'clr-sec'
li_elements = soup.find_all('picture', class_='clr-sec')

# Filter out None values from all_events
filtered_events = [event for event in all_events if event is not None]

# Determine the minimum length to avoid IndexError
min_length = min(len(filtered_events), len(li_elements))

# Iterate over the minimum length and update image links
for i in range(min_length):
    try:
        img_element = li_elements[i].find('source')  # Find <source> element within <picture>
```

```
        if img_element:
            img_path = "https://www.lucernefestival.ch" + img_element['srcset']  # Construct image URL
            filtered_events[i]['ImageLink'] = img_path  # Update image link in event dictionary
        else:
            filtered_events[i]['ImageLink'] = 'NOT AVAILABLE'  # Set image link as 'NOT AVAILABLE' if <source> not found
    except Exception as e:
        print("Error:", e)  # Print any encountered errors



# Update original all_events list with filtered_events
all_events = filtered_events
```

printing all_events list of dictionaries which contains "Date","Time","Venue","Title", "Artist_Name","Works" with added "ImageLink" as a new key element and corresponding values are fetched from the webscript

In [ ]:
```
print(all_events)
```

In [ ]:
```
print(len(all_events))
```

I imported the psycopg2 library in Python to establish a connection with a PostgreSQL database. Then, I created a schema named CodingChallenge and a table named events within it, with columns for date, time, venue, artist_name, works, and image link. Finally, I inserted data from a list of dictionaries(all_events) containing event information into this table.

In [ ]:
```
import psycopg2
from psycopg2 import sql

# Connect to PostgreSQL database
try:
    conn = psycopg2.connect(
        dbname="FutureDemand",
        user="postgres",
        password="Krishna7@",
        host="localhost",
        port="5432"
    )
```

```
    # Create a cursor object using a context manager
    with conn.cursor() as cur:
        # Create CodingChallenge schema
        cur.execute("CREATE SCHEMA IF NOT EXISTS CodingChallenge")

        # Define schema and create events table
        cur.execute("""
            CREATE TABLE IF NOT EXISTS CodingChallenge.events (
                id SERIAL PRIMARY KEY,
                date TEXT,
                time TEXT,
                venue TEXT,
                title TEXT,
                artist_name TEXT,
                works TEXT,
                image_link TEXT
            )
        """)
        # Delete all existing data from the events table
        cur.execute("DELETE FROM CodingChallenge.events")

        # Iterate over all events and insert them into the PostgreSQL database
        for event_data in all_events:
            cur.execute(sql.SQL("""
                INSERT INTO CodingChallenge.events (date, time, venue, title, artist_name, works, image_link)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
            """), (
                event_data["Date"],
                event_data["Time"],
                event_data["Venue"],
                event_data["Title"],
                event_data["Artist_Name"],
                event_data["Works"],
                event_data["ImageLink"]
            ))

    # Commit the transaction
    conn.commit()

except psycopg2.Error as e:
    print("Error occurred:", e)

finally:
```

```
    # Close connection
    conn.close()
```

Checking the Versions:-

In [ ]:
```
import bs4
print(bs4.__version__)
```

In [ ]:
```
import psycopg2
print(psycopg2.__version__)
```

In [ ]:
```
import requests
print(requests.__version__)
```

In [ ]:
```
import psycopg2
from psycopg2 import sql
```