Final Project Fall - 2023

This project essentially combines various assignments we have completed throughout the semester. While some code and functionalities have already been implemented, others remain for you to finalize as part of your final project and presentation. To streamline the code and simplify the process of cleaning HTML and JavaScript, I am utilizing the BeautifulSoup library. It is not required for you to learn this module in detail; rather, your task is to utilize and analyze the output generated by BeautifulSoup.

requests: A powerful HTTP library for sending HTTP requests to web servers, allowing you to fetch web pages.

BeautifulSoup (from bs4): A library for parsing HTML and XML documents. It is widely used for web scraping purposes to extract data easily and efficiently from web pages.

urljoin (from urllib.parse): A utility for constructing absolute URLs from relative fragments. It's particularly useful for handling the links found during web scraping, ensuring they are complete URLs that can be accessed directly.

```
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin


def get_internal_links(url):
    try:
        # Send a GET request to the URL
        response = requests.get(url)

        # Check if the request was successful
        if response.status_code != 200:
            return f"Failed to retrieve content from {url}"

        # Parse the content of the request with BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all the anchor tags in the HTML
        anchors = soup.find_all('a')

        internal_links = set()

        # Filter and add the internal links to the set
        for anchor in anchors:
            link = anchor.get('href')
            # Check if the link is internal
            if link and (link.startswith('/') or link.startswith(url)):
                full_link = urljoin(url, link)
```

```
                internal_links.add(full_link)

        return internal_links
    except Exception as e:
        return str(e)
```

+ Code + Text

0) Please enter the website URL you wish to analyze in the space provided.

Note: Ensure that the URL is neither a social media link nor from a large website encompassing 100 or more pages.

```
url = "https://playvalorant.com/en-us/"
internal_links = get_internal_links(url)

for link in internal_links:
    print(link)
```

```
https://playvalorant.com/en-us/news/announcements/beginners-guide/
https://playvalorant.com/en-us/news/community/community-roundup-november-2023/
https://playvalorant.com/en-us/agents/
https://playvalorant.com/en-us/news/game-updates/valorant-patch-notes-7-12/
https://playvalorant.com/en-us/news/
https://playvalorant.com/en-us/maps/
```

1) internal_links is a set that contains all the internal link on the passing URL. In next cell, write one line of code to print number of URL

```
#your code here:
print(len(internal_links))
```

```
6
```

2) Write a script to crawl through each URL in the `internal_links` set. The script should identify and print the URLs with the highest number of internal links.

```
#Your code here

def get_highest_internal_link(link_list):
  max_count = 0
  max_url = ''

  for link in link_list:
    internal_link_set = get_internal_links(link)
    if len(internal_link_set) > max_count:
      max_count = len(internal_link_set)
      max_url = link

  return max_url
```

```
get_highest_internal_link(internal_links)
```

```
'https://playvalorant.com/en-us/news/announcements/beginners-guide/'
```

The function below is designed to extract purely the clean, human-readable text from a webpage, eliminating any HTML tags or scripts.

```
def return_clean_content_of_url(url):
    try:
        # Send a GET request to the URL
        response = requests.get(url)

        # Check if the request was successful
        if response.status_code != 200:
            return f"Failed to retrieve content from {url}"

        # Parse the content of the response with BeautifulSoup
        soup = BeautifulSoup(response.text, 'html.parser')

        # Extract and print the text content from the parsed HTML
        text_content = soup.get_text(separator='\n', strip=True)
        return text_content

    except Exception as e:
        print(f"An error occurred: {e}")

print(return_clean_content_of_url(url))
```

```
VALORANT: Riot Games’ competitive 5v5 character-based tactical shooter
A 5v5 character-based tactical shooter
PLAY FREE
We are
VALORANT
THE LATEST
GO TO NEWS PAGE.
12/05/23
Game Updates
VALORANT Patch Notes 7.12
12/03/23
Game Updates
DRIFT // Team Deathmatch Map Trailer - VALORANT
11/30/23
Community
Community Roundup: November 2023
EVOLUTION
EPISODE_07 // ACT III / YR 3
Watch Now
WE ARE VALORANT
DEFY THE LIMITS
Blend your style and experience on a global, competitive stage. You have 13 rounds to attack and defend your side using sharp gunplay and tacti
```

```
LEARN THE GAME
Gameplay
YOUR AGENTS
CREATIVITY IS YOUR GREATEST WEAPON.
More than guns and bullets, you'll choose an Agent armed with adaptive, swift, and lethal abilities that create opportunities to let your gunpl
VIEW ALL AGENTS
Place
YOUR MAPS
FIGHT AROUND THE WORLD
Each map is a playground to showcase your creative thinking. Purpose-built for team strategies, spectacular plays, and clutch moments. Make the
VIEW ALL MAPS
```

3) Compose a line of code to display the total number of words present in the content of the URL.

```
#your code here:
print(len(return_clean_content_of_url(url).replace('\n', ' ').split(' ')))
```

```
['VALORANT:', 'Riot', 'Games'', 'competitive', '5v5', 'character-based', 'tactical', 'shooter\nA', '5v5', 'character-based', 'tactical', 'shoot
215
```

4) Create a Python function that calculates the count of unique words in the previously mentioned content. This function should also tally the frequency of each word and identify the top 19 most commonly used words, along with their respective occurrence counts.

```
#your code here
def num_of_unique_words(text):
    word_dict = {}

    text_arr = text.replace('\n', ' ').split(' ')

    for word in text_arr:
      word = word.lower()
      if word not in word_dict:
        word_dict[word] = 0

      word_dict[word] += 1

    sorted_dict = sorted(word_dict, key=word_dict.get, reverse=True)

    print('Unique word count: ' + str(len(word_dict)))

    i = 0
    for w in sorted_dict:
      if i == 19:
        break
      print(i + 1, w, word_dict[w])

      i+=1
```

```
    return word_dict


num_of_unique_words(return_clean_content_of_url(url))
```

```
Unique word count: 149
1 your 8
2 and 8
3 to 7
4 the 6
5 valorant 4
6 competitive 3
7 tactical 3
8 a 3
9 play 3
10 game 3
11 as 3
12 agents 3
13 5v5 2
14 character-based 2
15 shooter 2
16 we 2
17 are 2
18 updates 2
19 // 2
{'valorant:': 1,
 'riot': 1,
 'games’': 1,
 'competitive': 3,
 '5v5': 2,
 'character-based': 2,
 'tactical': 3,
 'shooter': 2,
 'a': 3,
 'play': 3,
 'free': 1,
 'we': 2,
 'are': 2,
 'valorant': 4,
 'the': 6,
 'latest': 1,
 'go': 1,
 'to': 7,
 'news': 1,
 'page.': 1,
 '12/05/23': 1,
 'game': 3,
 'updates': 2,
 'patch': 1,
 'notes': 1,
```

```
 '7.12': 1,
 '12/03/23': 1,
 'drift': 1,
 '//': 2,
 'team': 2,
 'deathmatch': 2,
 'map': 2,
 'trailer': 1,
 '-': 1,
 '11/30/23': 1,
 'community': 2,
 'roundup:': 1,
```

5. Write a script to output two specific pages from the main URL: one with the highest word count and another with the lowest word count.

```
#Your code here
def min_max_word_count(url):
  internal_links = get_internal_links(url)

  min_count = 1000000000000000000
  max_count = 0

  min_url = ''
  max_url = ''

  for link in internal_links:
    text = return_clean_content_of_url(link)
    text = text.replace('\n', ' ').split(' ')
    word_count = len(text)

    if word_count > max_count:
      max_count = word_count
      max_url = link

    if word_count < min_count:
      min_count = word_count
      min_url = link
  print(f"Lowest word count is {min_count} from {min_url}.")
  print(f"Highest word count is {max_count} from {max_url}.")
  return (min_url, min_count, max_url, max_count)

min_max_word_count(url)
```

```
Lowest word count is 67 from https://playvalorant.com/en-us/news/.
Highest word count is 561 from https://playvalorant.com/en-us/news/game-updates/valorant-patch-notes-7-12/.
('https://playvalorant.com/en-us/news/',
 67,
 'https://playvalorant.com/en-us/news/game-updates/valorant-patch-notes-7-12/',
 561)
```

Extra credit:

Develop a function that identifies and presents the 19 most commonly used words throughout the entire website.

```
#your code here
```