Emplay Assignment Code & Output

Name : Shanmukeswara Reddy Medapati
Assignment : Text Summarization from Web Scraping
Date : 02-04-2024

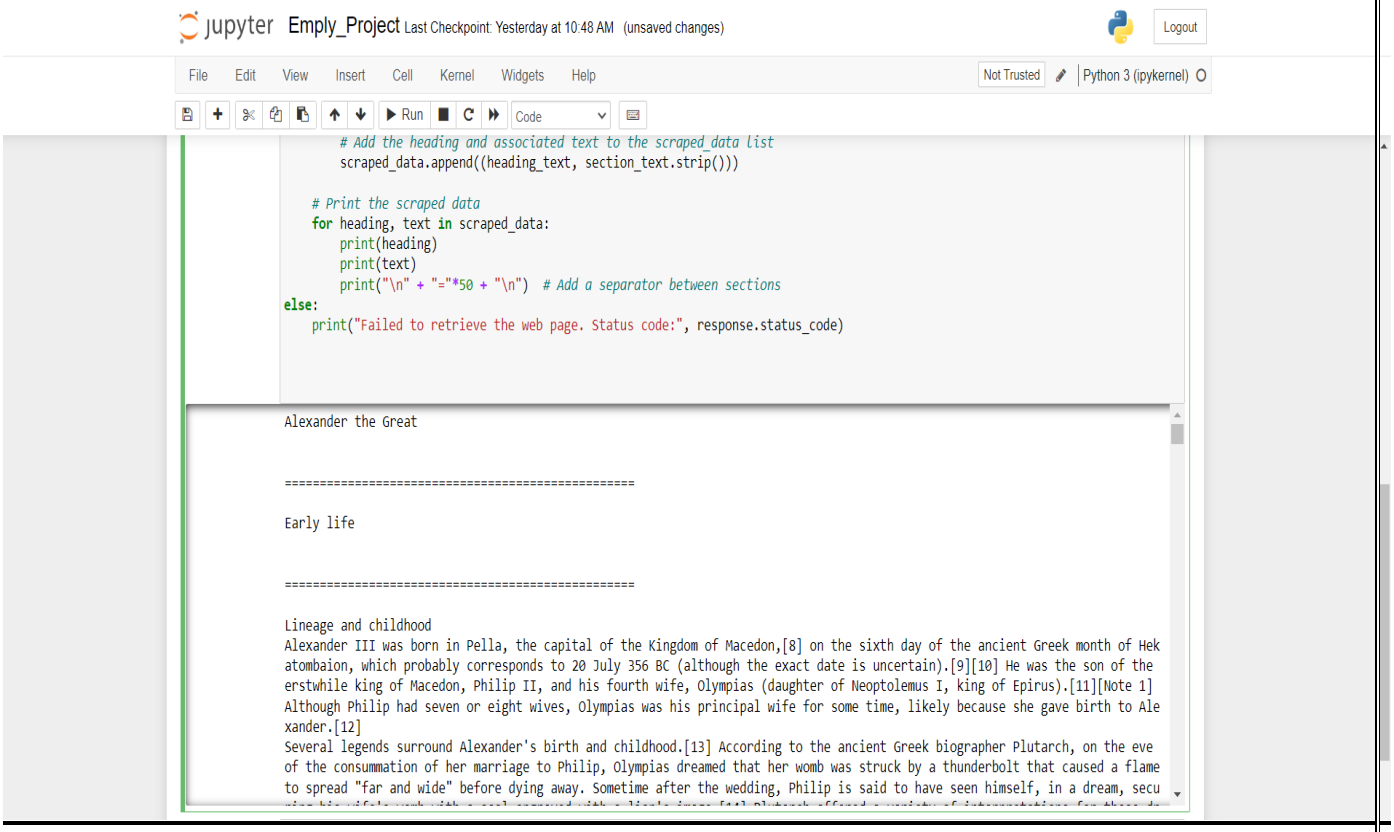
Code :

```
import requests
from bs4 import BeautifulSoup
from gensim.summarization import summarize

# Step 1: Web Scraping
def webscrap(url):
    try:
        # Send an HTTP request to the URL
        a= requests.get(url)
        a.raise_for_status()

        # Parse the HTML content of the page using BeautifulSoup
        b = BeautifulSoup(response.text, 'html.parser')

        return b
    except Exception as e:
        print(f"Error while scraping: {e}")
        return None

# Step 2: Extract Headings and Sections
def extract(b):
    c = []

    # Find all headings (e.g., h1, h2, h3, etc.)
    headings = b.find_all(['h1', 'h2', 'h3', 'h4', 'h5', 'h6'])

    for heading in headings:
        section_title = heading.get_text(strip=True)
        section_content = []

        # Extract text content under the heading until the next heading
        sibling = heading.find_next_sibling()
        while sibling and sibling.name not in ['h1', 'h2', 'h3', 'h4', 'h5', 'h6']:
            if sibling.name == 'p':
                section_content.append(sibling.get_text())
```

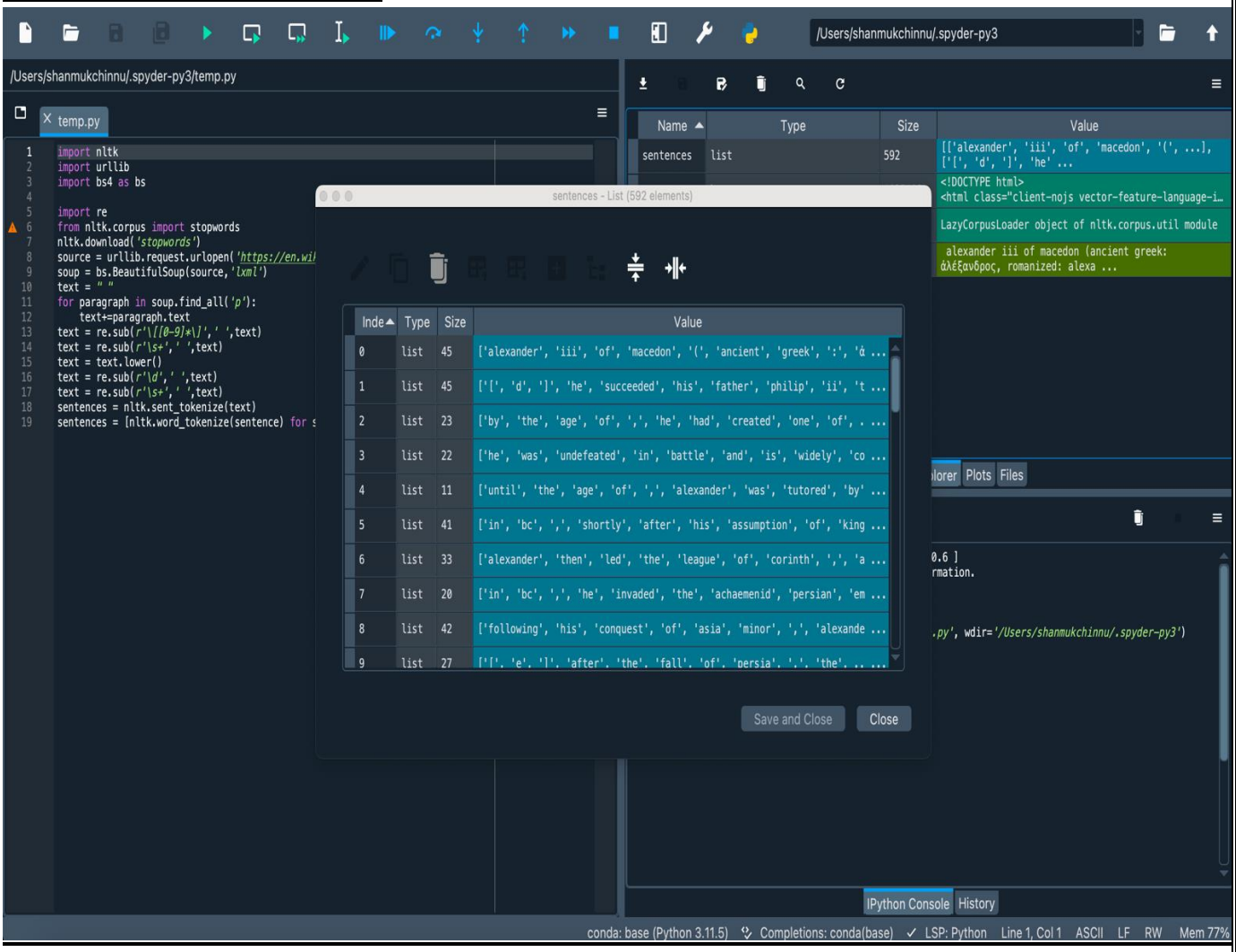
```
            sibling = sibling.find_next_sibling()

        headings_and_sections.append({
            'title': section_title,
            'content': ' '.join(section_content),
        })

    return headings_and_sections

# Step 3: Summarization
def summary(headings_and_sections):
    summarized_sections = []

    for section in headings_and_sections:
        section_title = section['title']
        section_content = section['content']

        # Summarize each section's content
        section_summary = summarize(section_content, ratio=0.3)  # Adjust the ratio as needed

        summarized_sections.append({
            'title': section_title,
            'summary': section_summary,
        })

    return summarized_sections

# Step 4: Main Pipeline
def main(url):
    # Step 1: Web Scraping
    soup = scrape_web_page(url)

    if soup is not None:
        # Step 2: Extract Headings and Sections
        headings_and_sections = extract_headings_and_sections(soup)

        if headings_and_sections:
            # Step 3: Summarization
            summarized_sections = summarize_sections(headings_and_sections)

            # Step 4: Display or further process the summarized sections
            for section in summarized_sections:
                print(f"Section Title: {section['title']}")
                print(f"Section Summary: {section['summary']}\n")
```

```
# Provide the URL of the web page to be scraped and summarized
web_page_url = 'https://en.wikipedia.org/wiki/Alexander_the_Great'
main(web_page_url)
```

Output of the Text Summarization



Output of the Web Scraping



Code for Prompt of Chains in generating summaries

```
import openai

# Initialize the OpenAI API client (you need to provide your API key)
openai.api_key = "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"

# Define the content with sections and headings
content = """
Section: Early Life
Text: Alexander III was born in Pella, the capital of the Kingdom of Macedon, on the sixth day of
the ancient Greek month of Hekatombaion...

Section: Conquests
Text: Alexander's conquests extended his empire across three continents...
```

```
"""

# Initialize the conversation with an instruction
conversation = [
    {"role": "system", "content": "You are a text summarization bot. Please summarize the
following content while preserving the original headings and sections:"},
]

# Split the content into sections
sections = content.strip().split("\n\n")

# Create a prompt chain for each section
for section in sections:
    heading, text = section.strip().split("\n", 1)

    # Provide the heading and text as user input
    conversation.append({"role": "user", "content": f"Section: {heading}\nText: {text}"})

    # Request a summary
    conversation.append({"role": "user", "content": "Please summarize the above section while
preserving the original heading."})

# Generate summaries using GPT-3
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=conversation,
)

# Extract and save the summaries
summaries = [message["content"] for message in response["messages"] if message["role"] ==
"assistant"]
with open("summarized_document.txt", "w", encoding="utf-8") as summarized_file:
    for summary in summaries:
        summarized_file.write(summary + "\n")
```

>>>