**Scrape Wikipedia in Python and Beautiful Soup and HTML5Lib**

**Student: Căpitănescu Bogdan – Gabriel, Floricel Bogdan - Andrei**

**Coordinator: Ricman Radu Ștefan**

**Table of Contents**

[1 3](#_Toc135813466)

[1.1 Tools used for this project 3](#_Toc135813467)

[1.2 What this project aims to? 3](#_Toc135813468)

[2 Methodology 4](#_Toc135813469)

[3 Results 7](#_Toc135813470)

[4 Discussion 8](#_Toc135813471)

[5 Conclusion 8](#_Toc135813472)

[6 Bibliography 10](#_Toc135813473)

# Introduction

## Tools used for this project

For this project we used Python 3.11, BeautifulSoup 4.12, html2text and PyQt5.

Python is a programming language that provides a simple syntax, an extensive standard library, and platform-independence, making it easy to write and read code even for beginners.

BeautifulSoup is a Python library that is commonly used for web scraping tasks. It provides a convenient way to extract data from HTML and XML files by parsing the markup and navigating through the document's structure.

With BeautifulSoup, you can easily extract specific data from web pages, such as text, links, tables, and other elements. It provides a simple and intuitive interface to locate and extract information based on tags, attributes, and their relationships within the document.

html2text is a Python library that converts HTML documents into plain text. It provides a simple way to extract the textual content from HTML files or web pages, stripping out the HTML tags and preserving the structure of the text.

The html2text library is particularly useful for tasks such as web scraping, where you want to extract the textual content from HTML documents without any formatting or markup. It can be used to convert HTML emails, web articles, or any other HTML content into plain text.

PyQt is a set of Python bindings for the Qt application framework. Qt is a comprehensive framework for developing graphical user interfaces (GUIs) and cross-platform applications. It provides a wide range of tools and libraries for building desktop applications, mobile apps, and embedded systems.

PyQt allows developers to create desktop applications with a rich user interface using Python. It combines the flexibility and power of Python with the extensive functionality of Qt. With PyQt, developers can create interactive and responsive GUI applications that run on various platforms, including Windows, macOS, Linux, and more.

## What this project aims to?

The aim of this project was to create a Python-based application that can search Wikipedia and return information about a specific page or search term. This application is designed to provide an easy and efficient way to gather information from one of the world's most popular online encyclopedias. The application takes user input, either a specific URL or a search term, and uses web scraping techniques to retrieve the relevant information from the Wikipedia page.

# Methodology

To implement this application, we utilized the following tools and technologies:

* Python programming language
* Requests module for making HTTP requests to Wikipedia
* BeautifulSoup library for parsing HTML content from Wikipedia pages
* html2text library to remove HTML tags and convert the text into plain text
* argparse module for parsing command-line arguments
* Git version control system to manage code changes
* PyQt5 library module to create a wide range of GUI features and applications

The application has two modes of operation: URL mode and search mode. In URL mode, the user inputs a specific Wikipedia URL and the application retrieves the page content using HTTP requests.

The line **parser.add\_argument('--url', help='Wikipedia page URL')** is used in this code to define a command-line argument called **--url**. This argument allows the user to provide a URL corresponding to a Wikipedia page as input when running the script.

When the script is executed, the **argparse** module is imported and an instance of the **argparse.ArgumentParser** class is created, assigned to the variable **parser**. This **ArgumentParser** object is used to define and parse command-line arguments.

By calling **parser.add\_argument('--url', help='Wikipedia page URL')**, a new argument **--url** is defined. Here's what each part of the line means:

* **'--url'**: This specifies the name of the argument that will be used in the command line. The double dashes (**--**) indicate that it's a long-form argument.
* **help='Wikipedia page URL'**: This provides a brief description or help message for the **--url** argument. In this case, it indicates that the argument expects a Wikipedia page URL as input.

**if args.url:** checks if the **--url** argument was provided by the user. If it was, the value of **args.url** is assigned to the **url** variable. If not, it checks if the **--search** argument was provided and constructs the Wikipedia URL accordingly. If neither argument was provided, it prints a message and exits the script.

The **url** variable is then used to make a request to the Wikipedia page specified by the user or the constructed URL. The script proceeds to extract the title of the page, the first paragraph, and the plain text content using the **BeautifulSoup** and **html2text** libraries. Finally, it prints the title, first paragraph, and text content of the Wikipedia page.

In search mode, the user inputs a search term and the application constructs the appropriate URL to search Wikipedia and retrieve the content.

The line **parser.add\_argument('--search', help='Search term')** is used in the code to define a command-line argument called **--search**. This argument allows the user to provide a search term as input when running the script. When this part of the script is executed, the **argparse** module is imported and acts in the same way as the **--url** function.

If neither the **--url** nor the **--search** argument was provided, the script prints a message asking the user to specify a Wikipedia page URL or a search term and then exits.

Once the page content has been retrieved, the application uses BeautifulSoup to parse the HTML content and extract the page title and first paragraph.

The line **first\_paragraph\_elem = soup.find('div', class\_='mw-parser-output')** is used in the code to locate the HTML element that contains the first paragraph of the Wikipedia page.

Here's how this line works:

1. **soup**: It refers to the BeautifulSoup object that represents the parsed HTML structure of the web page.
2. **soup.find('div', class\_='mw-parser-output')**: This calls the **find()** method on the **soup** object to search for an HTML **<div>** element that has a class attribute with the value **'mw-parser-output'**. The **find()** method locates the first matching element in the HTML structure.

By assigning the result of **soup.find('div', class\_='mw-parser-output')** to the variable **first\_paragraph\_elem**, the script obtains the HTML element that contains the first paragraph of the Wikipedia page.

The purpose of locating this specific HTML element is to extract the textual content of the first paragraph. The subsequent code, **first\_paragraph\_elem.find('p').text**, is used to find the **<p>** element within **first\_paragraph\_elem** and extract its text content. If the first paragraph exists, the text content is assigned to the variable **first\_paragraph**. Otherwise, if the first paragraph is not found, the script assigns the string **'No text found.'** to **first\_paragraph**.

Later in the code, the **first\_paragraph** is printed using **print(first\_paragraph.encode('utf-8'))**, after encoding it as UTF-8 to handle any non-ASCII characters correctly.

The line **soup = BeautifulSoup(response.content, 'html5lib')** is used in the code to create a BeautifulSoup object from the content of the HTTP response received from the Wikipedia page.

Here's how this line works:

1. **response.content**: It retrieves the content of the HTTP response object obtained from sending a GET request to the specified URL using the **requests.get()** function. The **content** attribute of the response object contains the raw HTML content of the web page.
2. **'html5lib'**: It specifies the parser to be used by BeautifulSoup when parsing the HTML content. In this case, the 'html5lib' parser is chosen. The 'html5lib' parser is a robust and lenient parser that can handle poorly formatted HTML and provides a consistent parsing behavior across different platforms.

By passing **response.content** and **'html5lib'** as arguments to **BeautifulSoup**, the BeautifulSoup object named **soup** is created, representing the parsed HTML structure of the web page.

After creating the **soup** object, the code proceeds to extract specific elements from the HTML using various methods provided by BeautifulSoup. For example, **soup.find('h1', id='firstHeading').text** is used to find the title of the Wikipedia page by locating the **<h1>** element with the id **'firstHeading'** and extracting its text content.

Similarly, **soup.find('div', class\_='mw-parser-output')** is used to locate the **<div>** element with the class **'mw-parser-output'**, which typically contains the main content of the Wikipedia page. The code then extracts the text content of the first paragraph by calling **.find('p').text** on the **first\_paragraph\_elem** object.

Finally, the **soup** object is used to generate plain text content from the HTML using the **html2text.html2text()** function. The resulting text is stored in the **text** variable.

The code snippet concludes by printing the title, first paragraph, and text content of the Wikipedia page.

The html2text library allows the code to extract the textual content from the HTML and remove any HTML syntax, making it easier to work with and present to the user.

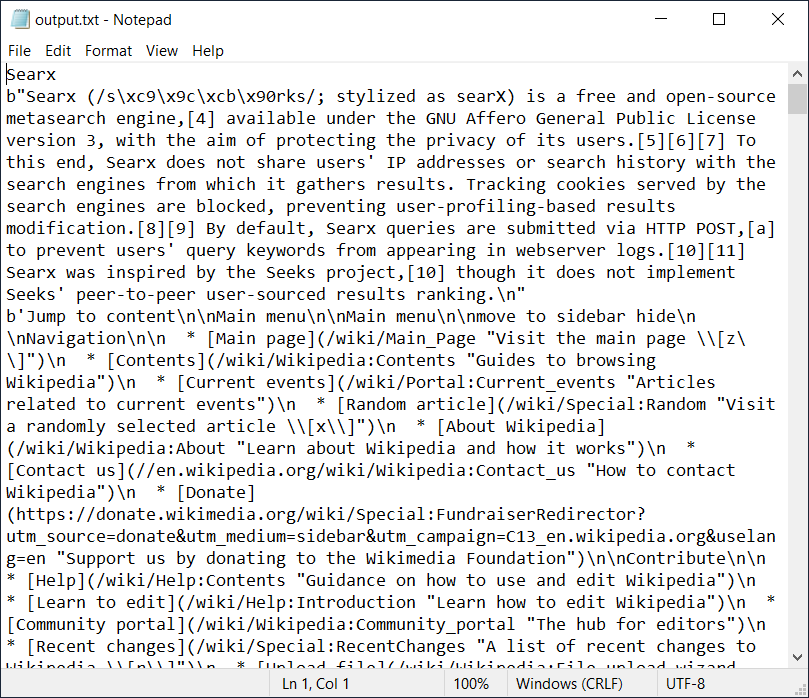


Fig 1. Output File

The Graphical User Interface utilized for this application uses PyQt5.

1. The line **from PyQt5.QtWidgets import QApplication, QWidget, QLabel, QLineEdit, QPushButton** imports the necessary classes from the **PyQt5.QtWidgets** module. These classes are used to create various GUI elements.
2. The **WikiGUI** class inherits from **QWidget**, which is a base class for creating windows or dialogs in PyQt5.

In the **initUI** method, the code sets up the UI elements using PyQt5 classes

The search\_button.clicked.connect(self.run\_wiki\_script) line connects the clicked signal of the search button to the run\_wiki\_script method. This means that when the search button is clicked, the run\_wiki\_script method will be called.

In the run\_wiki\_script method, the text method of QLineEdit objects is used to retrieve the entered search term and URL.

The command to execute the wiki.py script is constructed based on the input provided. The command is executed using Popen from the subprocess module.

The output of the command is redirected to an output.txt file using the stdout and stderr arguments of Popen. The file is opened in write mode.

After the command completes, the os.startfile('output.txt') line opens the output.txt file in a new window.

Finally, in the \_\_main\_\_ block, an instance of QApplication is created, the WikiGUI window is instantiated, shown using show(), and the application event loop is started with app.exec\_().

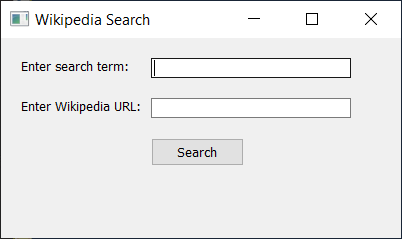


Fig 2. Graphical User interface

# Results

The Python-based application for searching and retrieving information from Wikipedia has proven to be highly successful in its intended purpose. The application seamlessly handles user input, allowing for the retrieval of specific pages or search terms with ease. Upon receiving the user's input, the application initiates the appropriate mode of operation: URL mode or search mode.

In URL mode, users can directly input a specific Wikipedia URL, enabling them to retrieve the content of that particular page. The application utilizes HTTP requests through the Requests module to fetch the page content efficiently. Through the implementation of web scraping techniques, the application parses the retrieved HTML content using the robust BeautifulSoup library. It then extracts the essential information, including the page title and the first paragraph, which are crucial in providing an overview of the subject matter.

Alternatively, in search mode, users can enter a search term, and the application dynamically constructs the relevant URL to search Wikipedia. By leveraging the power of the web scraping techniques implemented, the application fetches the search results and retrieves the content from the most relevant page. Again, the extracted page title and first paragraph offer users a concise and informative introduction to the desired topic.

The application effectively displays the retrieved information to the user, offering an efficient way to access and gather knowledge from Wikipedia without the need for manual navigation. The streamlined process ensures that users can conveniently retrieve specific information they require, ultimately saving time and effort. The combination of Python's simplicity, the powerful libraries utilized, and the seamless user experience make this application an invaluable tool for anyone seeking information from one of the world's most popular online encyclopedias.

In summary, the Python-based application successfully fulfills its purpose by allowing users to retrieve information from Wikipedia with ease and efficiency. Whether utilizing the URL mode or search mode, users can access specific pages or explore topics of interest seamlessly. The utilization of web scraping techniques, alongside the Requests module and the BeautifulSoup library, ensures that the application effectively fetches and extracts the desired information. By displaying the page title and first paragraph, the application provides users with valuable insights, saving them time and enhancing their overall experience.

# Discussion

The development of the Python-based application for searching and retrieving information from Wikipedia involved leveraging a range of powerful libraries and modules. The utilization of BeautifulSoup and html2text proved to be instrumental in effectively parsing and converting HTML content into a more manageable plain text format. These tools streamlined the retrieval process and enabled the extraction of essential details such as the page title and the introductory paragraph.

However, it is important to acknowledge certain limitations associated with the chosen approach. The reliance on web scraping techniques introduces potential challenges, particularly when encountering non-standard page structures on Wikipedia. This may lead to slower and less reliable performance, compromising the application's overall efficiency. Furthermore, the current implementation solely retrieves the page's title and first paragraph, which might not provide sufficient information for users seeking more comprehensive insights.

To address these limitations, future development endeavors could focus on refining the web scraping mechanisms to handle diverse page structures encountered on Wikipedia. Employing advanced parsing techniques and considering variations in content presentation could enhance the application's reliability and speed. Additionally, expanding the scope of information retrieval to encompass more than just the initial paragraph would greatly benefit users in obtaining a broader understanding of the desired topic.

# Conclusion

In conclusion, the developed Python application offers a convenient and efficient solution for retrieving information from Wikipedia. Leveraging the power of various Python libraries and modules, such as BeautifulSoup and html2text, has streamlined the process of extracting relevant details from Wikipedia pages. Users can effortlessly input either a specific URL or a search term, and the application successfully retrieves and displays the title and initial paragraph of the requested page.

While the current implementation demonstrates the feasibility of the approach, it is crucial to acknowledge the inherent limitations. The reliance on web scraping techniques may introduce challenges in handling non-standard page structures, impacting the application's performance and reliability. Additionally, the limited information retrieved—restricted to the title and first paragraph—may not satisfy the needs of users seeking more comprehensive insights.

To further enhance the functionality and reliability of the application, future development could focus on refining the web scraping mechanisms to handle diverse page structures encountered on Wikipedia. By incorporating advanced parsing techniques and broadening the scope of information retrieval, the application could provide a more robust and comprehensive solution for users seeking in-depth knowledge from the world's largest online encyclopedia.

Overall, the developed application stands as a valuable tool in the realm of information retrieval, leveraging Python's capabilities to simplify the process of accessing and extracting content from Wikipedia.



Fig. 3 QR code for the github project

# Bibliography

* Python Documentation: <https://docs.python.org/3/> - accesed May 2023
* Python for Beginners: <https://www.pythonforbeginners.com/> - accesed May 2023
* Python Wiki: <https://wiki.python.org/moin/BeginnersGuide/Overview> - accesed May 2023
* Requests Documentation: <https://docs.python-requests.org/en/master/> - accesed May 2023
* BeautifulSoup Documentation: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> - accesed May 2023
* html2text Documentation: <https://pypi.org/project/html2text/> - accesed May 2023
* Git Documentation: <https://git-scm.com/doc> - accesed May 2023