Ministry of Education of Republic of Moldova Technical University of

Moldova Faculty of Computers, Informatics and Microelectronics

Software Engineering Department

Web Programming

Laboratory work #2

Self made browser

Author:
Victor Caragiu
std. gr. FAF-191

Supervisor:
Alexei Şerşun

Chișinău 2022

1 Task

1. You have to write a command line program, using [go2web](go2web) as a starting point;
2. The program should implement the following CLI:
   ```

   go2web -u <URL>         # make an HTTP request to URL and print the response
   go2web -s <search-term> # search the term using your favorite search engine and print top 10 results
   go2web -h              # show help
   ```
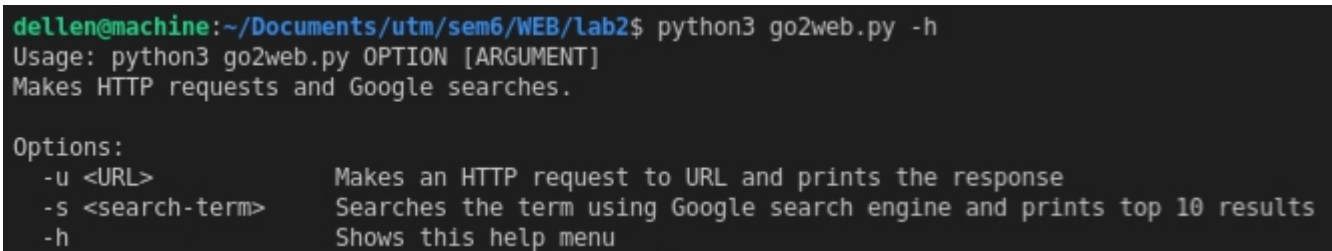
3. The responses from request should be human-readable (e.g. no HTML tags in the output)
4. Add another option to the program.

   You can use any programming language, but not the built-in/third-party libraries for making HTTP requests. You can't build a GUI application.

2 Results

Task 1:

   The help functionality is trivial, implemented by displaying a string with the necessary information (figure 1).


```
dellen@machine:~/Documents/utm/sem6/WEB/lab2$ python3 go2web.py -h
Usage: python3 go2web.py OPTION [ARGUMENT]
Makes HTTP requests and Google searches.

Options:
  -u <URL>              Makes an HTTP request to URL and prints the response
  -s <search-term>      Searches the term using Google search engine and prints top 10 results
  -h                    Shows this help menu
```

Figure 1. The help option

Task 2:

The search functionality was implemented using the *sockets* package, which provided the functionality to establish a TCP connection, through the specified port 80 for HTTP protocol. As a search engine I used Google. I used the *find* function for strings and cut only the HTML part of the response. I loaded it into a *BeautifulSoup* object, which provides parsing functionalities. I extracted the results as title and link by the class names of the HTML elements. The output contains the first page results (figure 2).

Some links contained encoded characters, for example the links to YouTube, so I added one more step for decoding the URLs with UTF-8 format, using *unquote* function from *urlib.parse* module.

```
dellen@machine:~/Documents/utm/sem6/WEB/lab2$ python3 go2web.py -s dog
Here are the first page results:
1. DOG | Official Trailer | MGM Studios - YouTube
Link: https://www.youtube.com/watch?v=V4tAtp-TyzQ

2. Dog (2022) - IMDb
Link: https://www.imdb.com/title/tt11252248/

3. Dog - Wikipedia
Link: https://en.wikipedia.org/wiki/Dog

4. Dog - Rotten Tomatoes
Link: https://www.rottentomatoes.com/m/dog_2022

5. Dog, facts and photos - National Geographic
Link: https://www.nationalgeographic.com/animals/mammals/facts/domestic-dog

6. DOG (@dogthefilm) • Instagram photos and videos
Link: https://www.instagram.com/dogthefilm/

7. Dog Breeds - Types Of Dogs - American Kennel Club
Link: https://www.akc.org/dog-breeds/

8. Dog Supplies | Dog Products & Accessories - Dog.com
Link: https://www.dog.com/

9. Dog Care Advice, Tips & Health Information | RSPCA
Link: https://www.rspca.org.uk/adviceandwelfare/pets/dogs
```

Figure 2. The search option with a "dog" search term

Task 3:

For implementing the URL accessing functionality, after getting the response and extracting the HTML, I removed the *head* element and all the *style* and *script* elements. Then I printed all the elements contents and the hyperlinks' *href* attributes values.
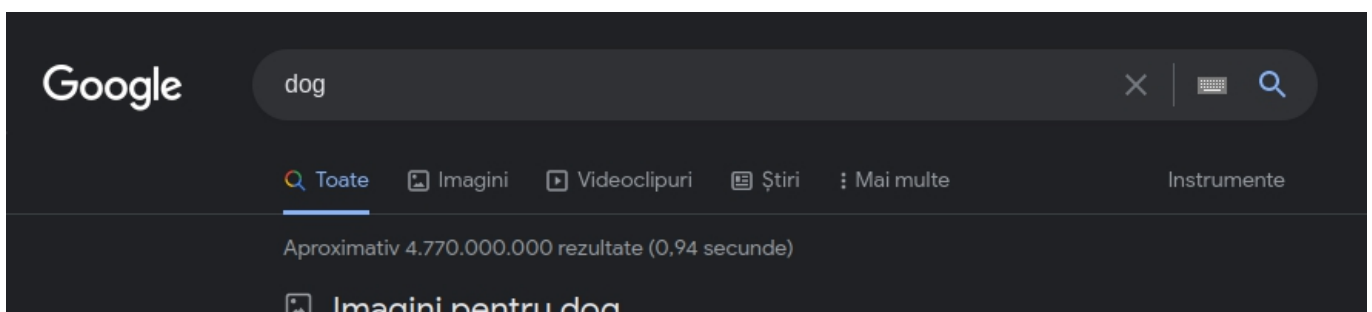
```
dellen@machine:~/Documents/utm/sem6/WEB/lab2$ python3 go2web.py -u https://www.google.com/search?q=dog
Making request to www.google.com at /search?q=dog


https://www.google.com/?sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQQOwgC
G
o
o
g
l
e
https://www.google.com/?output=search&ie=UTF-8&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQPAgE
G
o
o
g
l
e
×

Faceți clic
https://www.google.com/search?q=dog&ie=UTF-8&gbv=1&sei=RK4jYunkHNSL-Qb2kKfABg
aici
 dacă nu sunteți redirecționat în câteva secunde.

11fe

Toate
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICCgB
Imagini
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=vid&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICSgC
Videoclipuri
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=nws&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICigD
Știri
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=bks&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICygE
Cărți
https://www.google.com/advanced_search
Instrumente de căutare


Orice limbă

Orice limbă
https://www.google.com/search?q=dog&ie=UTF-8&source=lnt&tbs=lr:lang_1ro&lr=lang_ro&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXb
```

Figure 3. Accessing the *https://www.google.com/search?q=dog* URL.

At the top, the web page has tabs with links for accessing other sections, such as "Toate", "Imagini", "Videoclipuri" and others (figure 4). These can be observed in the output (figure 5).

Figure 4. The top section of the page.

```
dellen@machine:~/Documents/utm/sem6/WEB/lab2$ python3 go2web.py -u https://www.google.com/search?q=dog
Making request to www.google.com at /search?q=dog


https://www.google.com/?sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQQwgC
G
o
o
g
l
e
https://www.google.com/?output=search&ie=UTF-8&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQPAgE
G
o
o
g
l
e
×

Faceți clic
https://www.google.com/search?q=dog&ie=UTF-8&gbv=1&sei=RK4jYunkHNSL-Qb2kKfABg
aici
 dacă nu sunteți redirecționat în câteva secunde.

11fe

Toate
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICCgB
Imagini
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=vid&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICSgC
Videoclipuri
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=nws&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICigD
Știri
https://www.google.com/search?q=dog&ie=UTF-8&source=lnms&tbm=bks&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQ_AUICygE
Cărți
https://www.google.com/advanced_search
Instrumente de căutare


Orice limbă

Orice limbă
https://www.google.com/search?q=dog&ie=UTF-8&source=lnt&tbs=lr:lang_1ro&lr=lang_ro&sa=X&ved=0ahUKEwjplvGOz6_2AhXURd4KHXbICWgQpwUIDQ
```

Figure 5. The way the app prints the tabs from the top section.

The section with images(figure 6), is printed by the app as links to those images (figure 7).
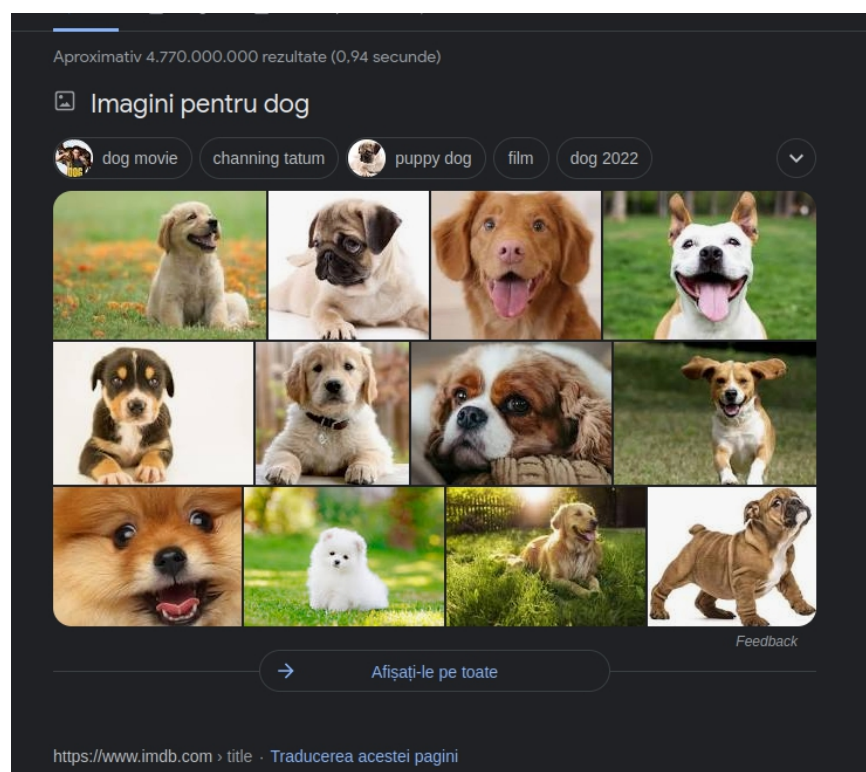
Figure 6. The images section in Firefox.

4

Figure 7. The images section printed by the app, consisting mainly of links to the images.

From the other components, as in the Google search results(figure 8), all the text is extracted, meaning that both the result heading, description, details and the link to access the other page are printed (figure 9).

Figure 8. The section with a Google search result.

5

Figure 9. The section with a Google search result, printed by the app.

The source files are available at: https://github.com/victor0198utm/WEB2

3   Conclusion

When implementing the tasks, I learned how to create an HTTP request by specifying the method, URI, HTTP version and the needed headers, using TCP sockets and receive the bytes from the server.

I exercised working with BeautifulSoup module to navigate the HTML tree by the ids, class names and tags. I also learned how to decode URLs from the UTF-8 encoding.

References

[1] TCP Sockets / How to work with TCP Sockets in Python, *https://steelkiwi.com/blog/working-tcp-sockets/*

Accessed on February 24, 2022.

[2] UTF-8 Decoding / Decoding UTF-8 strings in Python, *https://stackoverflow.com/questions/13110629/decoding-utf-8-strings-in-python#13110762*

Accessed on March 5, 2022

[3] BeautifulSoup / How to find any elements by class in BEautifulSoup, *https://pytutorial.com/find-all-by-class-with-python-beautifulsoup*

Accessed on March 5, 2022