Red Points

Python developer technical task

Autor: Baldelomar Garcia, Abel Santiago

Data: 6/10/2019

Contents

1 Introduction

In the sections **Exercise: Github Crawler** and **Exercise: Extra information** I explain how I have solved each exercise at a theoretical level. In the section **Documentation of the code** I explain how to execute the main program. All the code is inside the **code** directory.

2 Exercise: Github Crawler

2.1 Exercise title

We want you to code a GitHub crawler that implements the GitHub search and returns all the links from the search result, requisites are:

- Python 3
- The crawler should be as efficient as possible (fast, low memory usage, low CPU usage,...)
- Input:
  - Search keywords: a list of keywords to be used as search terms (unicode characters must be supported)
  - List of proxies: one of them should be selected and used randomly to perform all the HTTP requests (you can get a free list of proxies to work with at https://free-proxy-list.net/)
  - Type: the type of object we are searching for (Repositories, Issues and Wikis should be supported)
- Documentation about how to use it should be included
- Output: URLS for each of the results of the search
- The code should also include tests for the crawler, with a minimum coverage of 90
- For the purpose of this task you only have to process first page results
- For the purpose of this task we want to work with raw HTML and API can't be used
- You can use any library you consider useful for the task

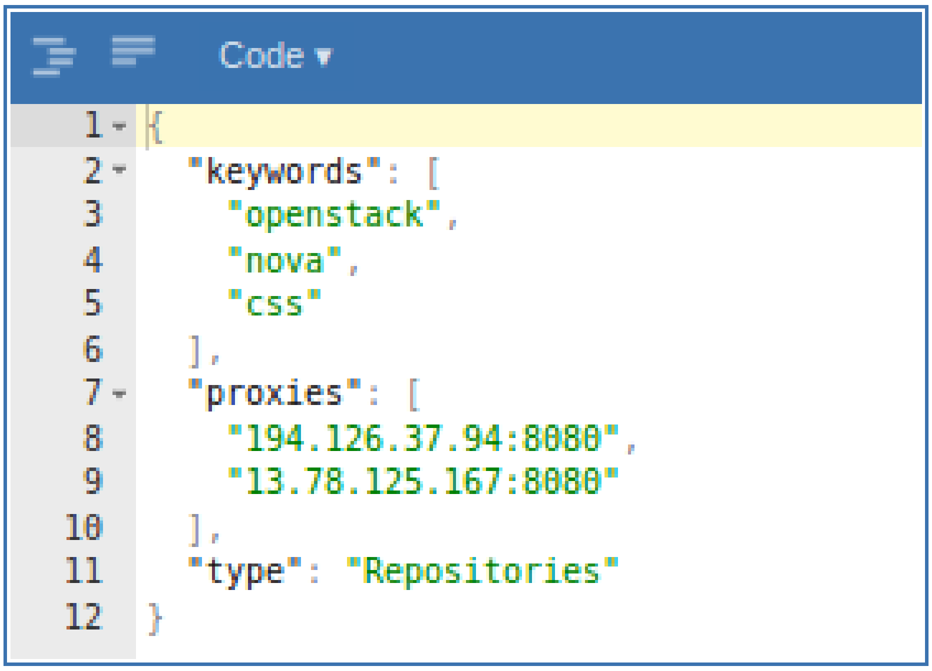
Example

Keywords: "openstack", "nova" and "css"

Proxies: "194.126.37.94:8080" and "13.78.125.167:8080"

Type: "Repositories"

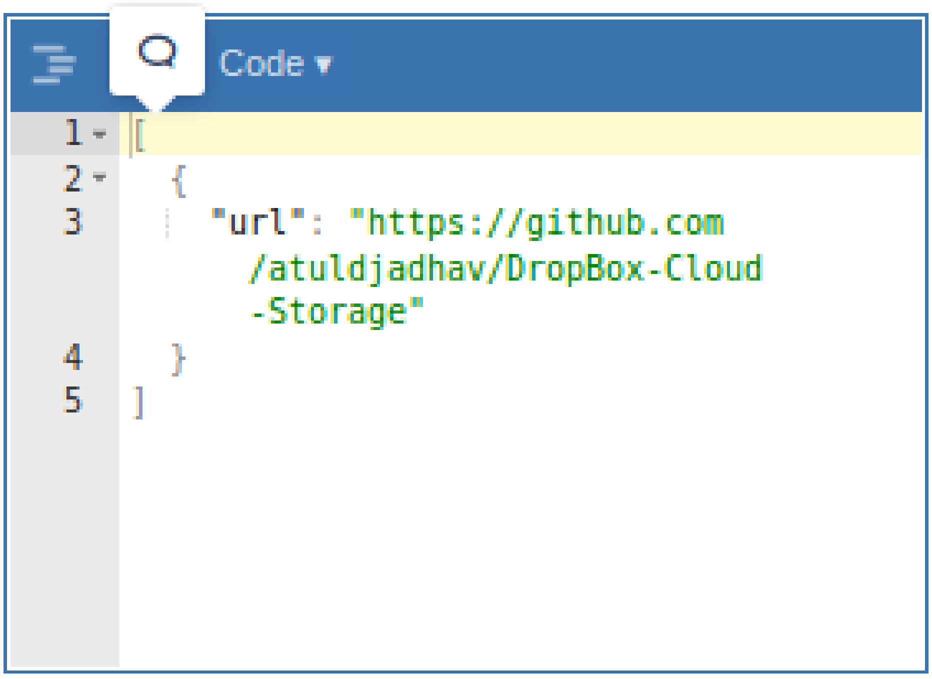
So the input will be a JSON containing:



```
{
  "keywords": [
    "openstack",
    "nova",
    "css"
  ],
  "proxies": [
    "194.126.37.94:8080",
    "13.78.125.167:8080"
  ],
  "type": "Repositories"
}
```

Expected results:

JSON object containing:



```
[
  {
    "url": "https://github.com/atuldjadhav/DropBox-Cloud-Storage"
  }
]
```

2.2 Solution

I considered two steps to solve the problem:

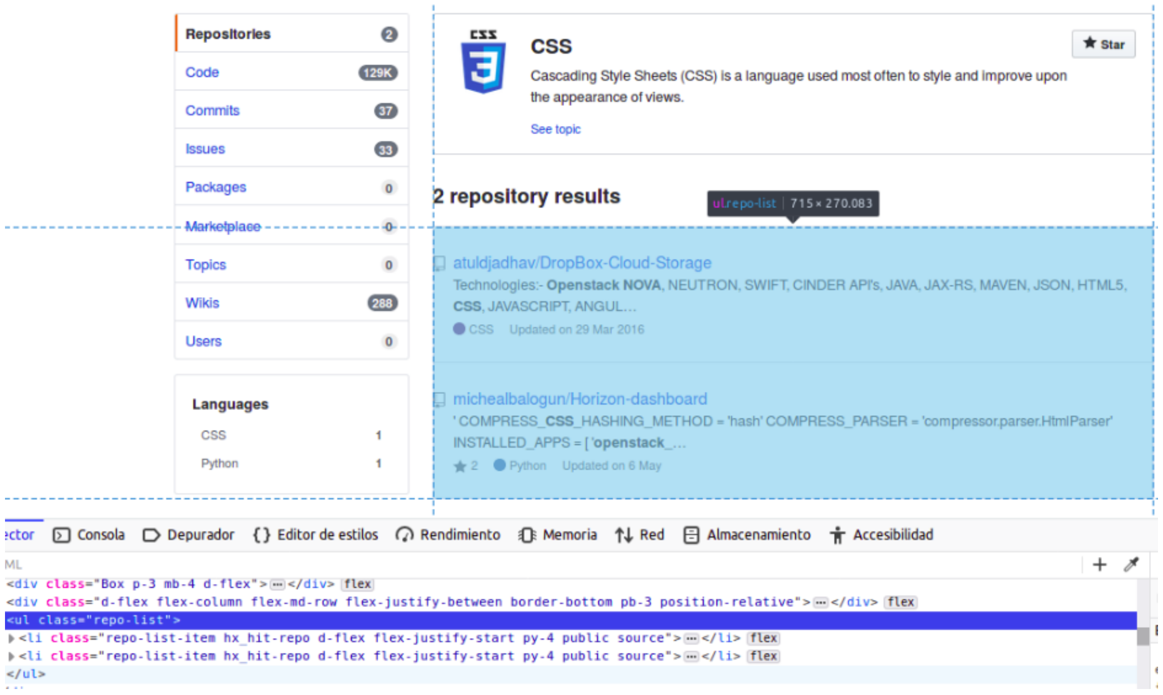
1. Be able to make a request using proxies to Github for a particular keywords and then obtain the HTML result.
2. Analize the HTML result and find out where are located the links, that is to say, identify the piece of HTML where the links are located, and therefore, extract the information and save it.

For the first step, I used the **requests** library of Python to get the HTML of Github. I found out the URL direction of Github to make a search for determinate keywords and a type, I found out this URL analysing the web page on Github. The URL direction is:

```
https://github.com/search?q=openstack+nova+css&type=Repositories
```

Where the variable **type** in the query parameters can be **Repositories, Code, Commits, Issues, Packages, Marketplace, Topics, Wikis and Users**. The variable **q** in the query parameters is destined for the keywords of the search, in this case **openstack, nova and css**.

For the step two, I analyzed the HTML result and I observed that the section of the results were identified as **repo-list** located as a value attribute in the case of a search with the query attribute type equal to **Repositories**. With this identifier I delimited and extracted this piece of substrig inside of HTML string.



Cascading Style Sheets (CSS) is a language used most often to style and improve upon the appearance of views.

2 repository results

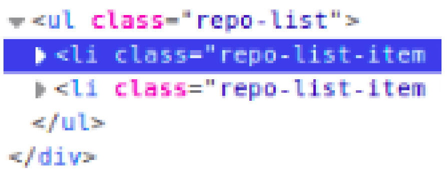
atuldjadhav/DropBox-Cloud-Storage
Technologies:- **Openstack NOVA**, NEUTRON, SWIFT, CINDER API's, JAVA, JAX-RS, MAVEN, JSON, HTML5, **CSS**, JAVASCRIPT, ANGUL...
● CSS Updated on 29 Mar 2016

michealbalogun/Horizon-dashboard
' COMPRESS_**CSS**_HASHING_METHOD = 'hash' COMPRESS_PARSER = 'compressor.parser.HtmlParser' INSTALLED_APPS = ['**openstack**_...
★ 2 ● Python Updated on 6 May

```
<div class="Box p-3 mb-4 d-flex">⋯</div> flex
<div class="d-flex flex-column flex-md-row flex-justify-between border-bottom pb-3 position-relative">⋯</div> flex
<ul class="repo-list">
 ▸<li class="repo-list-item hx_hit-repo d-flex flex-justify-start py-4 public source">⋯</li> flex
 ▸<li class="repo-list-item hx_hit-repo d-flex flex-justify-start py-4 public source">⋯</li> flex
</ul>
```

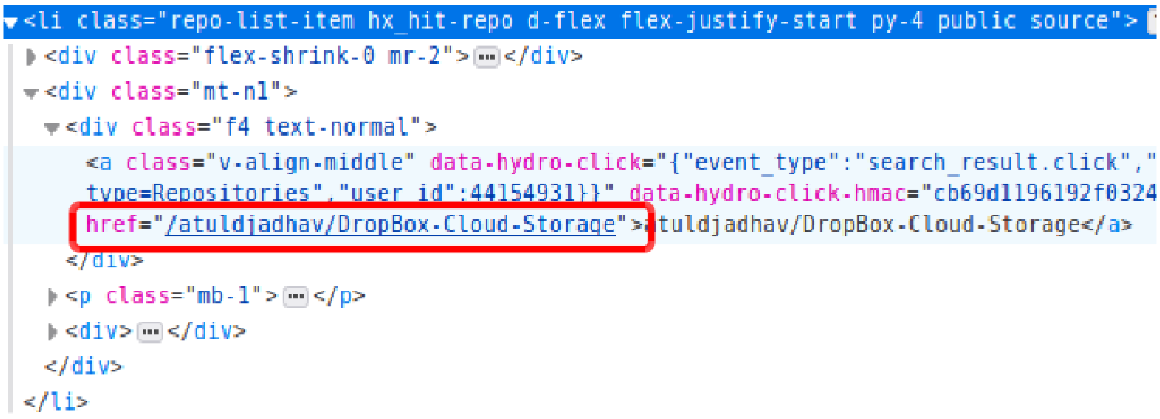
Figure 2: Secction of results

Once, I had the substring of the results, in the case of repositories the piece identified with **repo-list**, each result was identified as **repo-list-item** and inside each item or result the link of the item was identified with an **href=**. In the same way, I delimited and extracted the substring for each item and inside it I delimited and extracted the link.



```
<ul class="repo-list">
  <li class="repo-list-item
  <li class="repo-list-item
</ul>
</div>
```

Figure 3: Each item was identified with repo-list-item



```
<li class="repo-list-item hx_hit-repo d-flex flex-justify-start py-4 public source">
  <div class="flex-shrink-0 mr-2">…</div>
  <div class="mt-n1">
    <div class="f4 text-normal">
      <a class="v-align-middle" data-hydro-click="{"event_type":"search_result.click","
      type=Repositories","user_id":44154931}}" data-hydro-click-hmac="cb69d1196192f0324
      href="/atuldjadhav/DropBox-Cloud-Storage">atuldjadhav/DropBox-Cloud-Storage</a>
    </div>
    <p class="mb-1">…</p>
    <div>…</div>
  </div>
</li>
```

Figure 4: Each link was identified with href

When the request was with the query attribute type equal to **Issues** the results inside the HTML were located in a section called **issue-list** and each item was identified as **issue-list-item**, in the case of query attribute type equal to **Wikis** the identifiers were **wiki_search_results** and **wiki-list-item**. With these notes I have implemented a function to search **hrefs** and its links when the results in the HTML are located in a section that follows the pattern like **ID-List-Items, ID-item, Href=Link**.

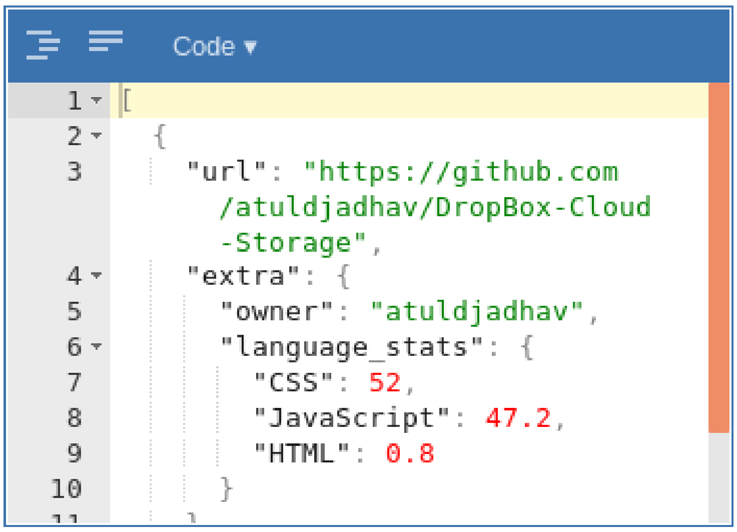
3 Exercise: Extra information

3.1 Exercise title

In the previous task we asked you to implement GitHub search and return just the links, now we want the crawler to be extended so the following information is extracted for each repository link:

- The owner of the repository
- Language stats

You dont have to return any other extra information for the rest of the link types (Wikis and Issues)



Example

For the repository in the first example above expected results should be now a JSON like:

```
[
  {
    "url": "https://github.com
      /atuldjadhav/DropBox-Cloud
      -Storage",
    "extra": {
      "owner": "atuldjadhav",
      "language_stats": {
        "CSS": 52,
        "JavaScript": 47.2,
        "HTML": 0.8
      }
```

Figure 5: Example

3.2 Solution

In the same way as before, once I acces to the repository, the languages stats are located in the section called **repository-lang-stats-numbers** with the identifiers **lang** and **percent** for each language. Therefore, I implemented a function that extract the information when the HTML of a repository follows this pattern.

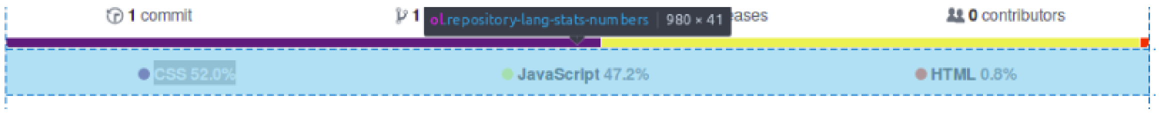
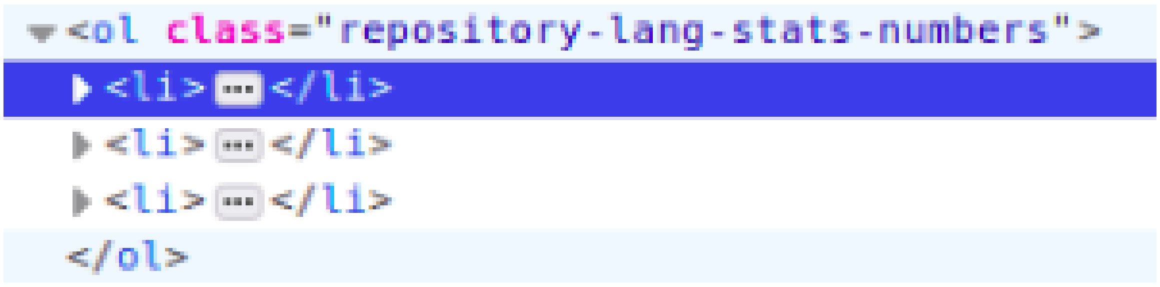
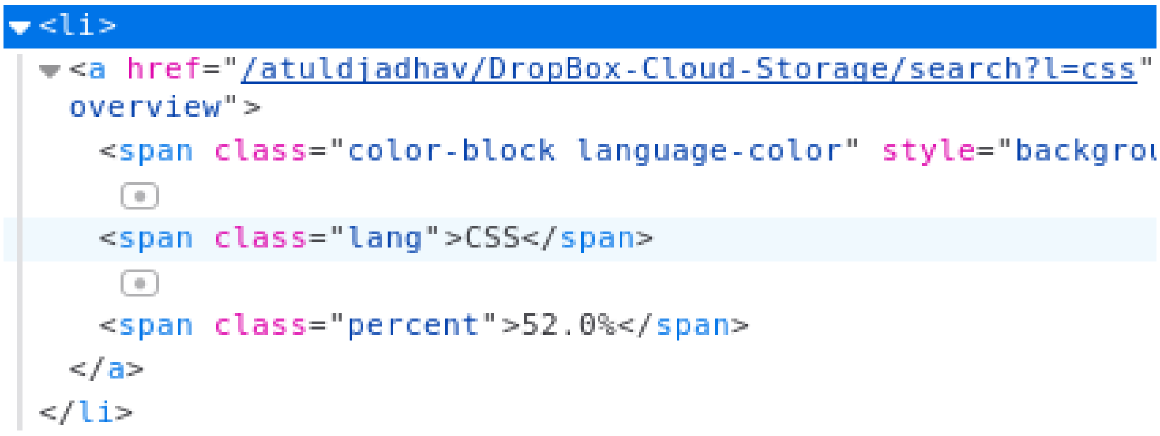


Figure 6: Section of languages stats



```
▾<ol class="repository-lang-stats-numbers">
   ▸<li>…</li>
   ▸<li>…</li>
   ▸<li>…</li>
  </ol>
```

Figure 7: Section of languages stats in the HTML



```
▾<li>
  ▾<a href="/atuldjadhav/DropBox-Cloud-Storage/search?l=css"
    overview">
      <span class="color-block language-color" style="backgrou
      
      <span class="lang">CSS</span>
      
      <span class="percent">52.0%</span>
    </a>
  </li>
```

Figure 8: Identifiers of the language and the stat

4 Documentation of the code

All the code is inside the **code** directory.

I have developed a main program that is the resposable to obtain the input from a json file and save the results (the links and extra information) to a json file. Bassically, this program reads the inputs from the input file, that is to say, the **keywords, proxies** and the **type**, and then calls a python class that I have developed which performs a searcher to Github and have methods that return the links or the extra information, finally these results are written to an output json file. The main program is named **main.py** and the class that performs the searcher is named **GitHubCrawler** and is in the python module named **gitHubCrawler.py**.

4.0.1 Execute the main program

To execute the main program the input and output json file must be passed as command line arguments. The output file will be generated in case it does not exist, but its name is required in the commad line. The commad line to execute the program is:

```
python3 main.py <inputFile.json> <outputFile.json>
```

4.0.2 GitHubCrawler class

The class GitHubCrawler which is the searcher to Github is in **gitHubCrawler.py**.

The creation of an instance of the class.

```python
from gitHubCrawler import GitHubCrawler

keywords = ["openstack","nova","css"]
proxy = "5.196.132.118:3128"

instance = GitHubCrawler(keywords,proxy)
```

The attributes **keywords** and **proxy** are requied.

- **keywords**: must be a list of strings with the keywords for the search.
- **proxy**: must be a string that follows the pattern IP:PORT and is the proxy used to make all the requests.

Aviable methods

- **getLinksRepos()** This method returns a list with all the links of the repositories found in the search for specified keywords. In case of an internal error it returns a empty list, it can happen when the proxy connection failed to make a HTTP request.

- **getLinksIssues()** This method returns a list with all the links of the issues found in the search for specified keywords. In case of an internal error it returns a empty list, it can happen when the proxy connection failed to make a HTTP request.

- **getLinksWikis()** This method returns a list with all the links of the wikis found in the search for specified keywords. In case of an internal error it returns a empty list, it can happen when the proxy connection failed to make a HTTP request.

- **getLangStatRepos(repository)** This method returns a dictionary with all the languages stats for the specified repository. The repository parameter must be a string of the URL of the repository desired like `https://github.com/atuldjadhav/DropBox-Cloud-Storage`. The dicctionary will have the pattern **{Language0:stat0, Language1:stat1,Language2:stat2, ...}**. In case of an internal error it returns a empty dictionary, it can happen when the proxy connection failed to make a HTTP request.

4.0.3 Test of GitHubCrawler class

I have realized a python test for the **GitHubCrawler class**. This test is the file **testGitHubCrawler.py** and is implemented with the module **unittest** of python. The test can be executed with the next command:

```
python3 -m unittest -v testGitHubCrawler
```