SELENIUM AND GOOGLE_IMAGES_DOWNLOAD

Web Scraping Images from Google

Anand Suresh
Mar 19 · 4 min read

. . .

Hello Readers! I am Anand, a very enthusiastic developer and an undergrad student.

Recently, i've been trying to figure out different ways to web scrape images from google.com based on search query using python and I've stumbled upon more than one method to do so. Not too sure to say which method is actually better, but to my usage I've incurred that It depends on the query or the kind of images we want to download.

So for this Blog I've decided to split the content into different snippets for ease of use later and add the final codes in the end for those who want to skip the process.

. . .

Method 1: Using "google_images_download" library

Installation:

```
Using PyPi - $ pip3 install google_images_download
--------------------------------------------------------------------
Using CLI - (cd into working directory)
$ git clone https://github.com/hardikvasa/google-images-download.git
$ cd google-images-download
$ python setup.py install
```

Usage:

Given below is a code snippet as a function call whose argument will be the query as a string. So all we've got to do when embedding this function into a program would be to import the library and call the function with a set of queries.

```
# importing google_images_download module
from google_images_download import google_images_download
```

```
def downloadimages(query):
        response = google_images_download.googleimagesdownload()
        # keywords is the search query
        # format is the image file format
        # limit is the number of images to be downloaded
        # print urs is to print the image file url
        # size is the image size which can
        # be specified manually ("large, medium, icon")
        # aspect ratio denotes the height width ratio
        # of images to download. ("tall, square, wide, panoramic")
        arguments = {"keywords": query,
                                #"format": "jpg",
                                "limit":100,
                                "print_urls":True}
                                #"size": "large"}
                                #"aspect_ratio": "panoramic" }
      '''
      I've played around with different parameters
      some yeild resuts and others don't. It can also depend on the searched query
      '''
        try:
                response.download(arguments)

        # Handling File NotFound Error
        except FileNotFoundError:
                arguments = {"keywords": query,
                                        "format": "jpg",
                                        "limit":4,
                                        "print_urls":True,
                                        "size": "medium"}

                # Providing arguments for the searched query
                try:
                        # Downloading the photos based
                        # on the given arguments
                        response.download(arguments)
```

The above code snippet automatically saves the downloaded images within it's respective folder.

. . .

Method 2: Using Selenium

Installation:

To install Selenium using PyPi

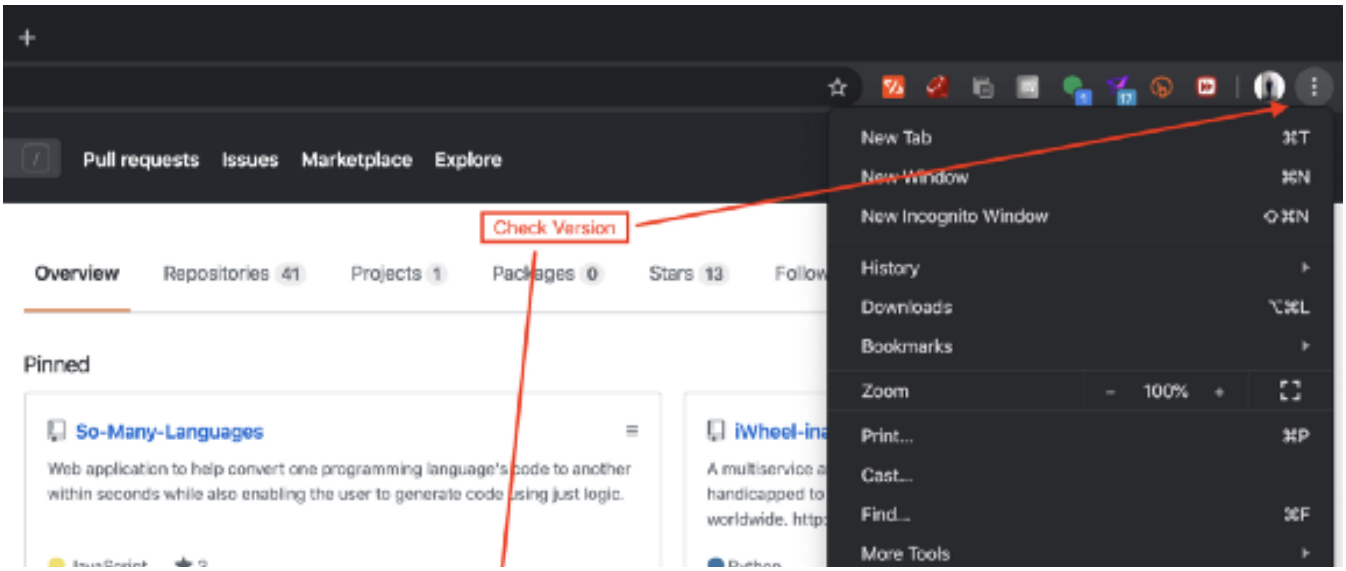
```
$ pip3 install selenium
```

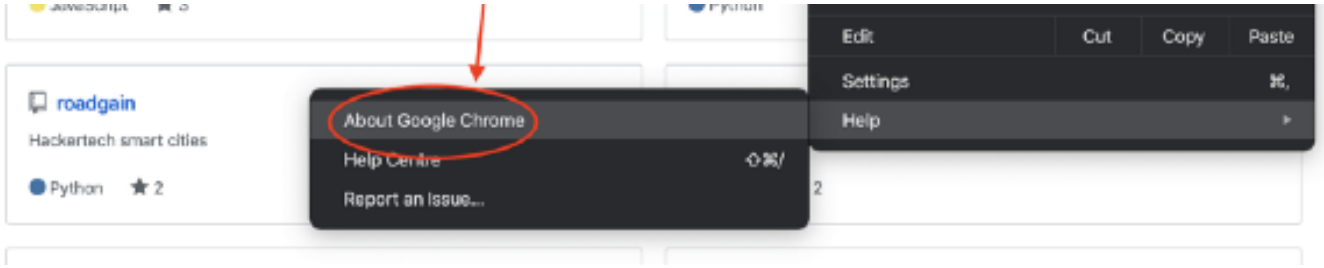
Usage:

Consider Selenium to be this package that lets your program do what a human user would normally do on the browser, like searching for something, going through videos on youtube, downloading images from xyz website and such can all be possible by using selenium to simulate a user's actions.

For this tutorial we are going to pair up Selenium with Google Chrome and automate it to fetch the images we require. To do that we are going to need the Chrome driver. There is a specific chrome-driver to every version of chrome, My chrome version is V80, hence i use the ChromeDriver version 80.

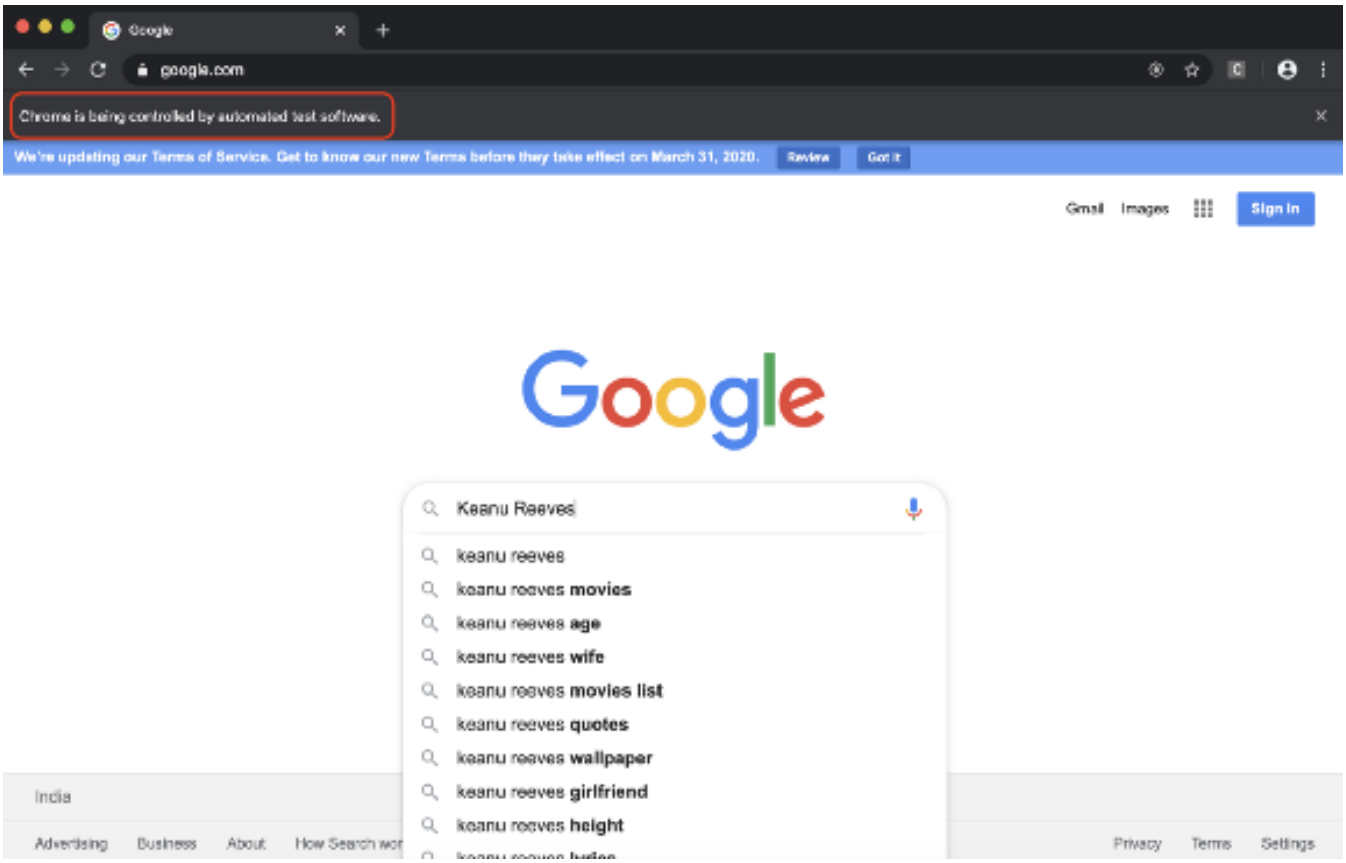
You can find the version of your chrome in the "About Google Chrome" section and download the respective version of chrome-driver from HERE .





Check Chrome Version

First, we will start by running a simple program which will open up the chrome window and automate a search of the query of our choice from the code.



Automated Searching

```
import selenium
from selenium import webdriver
# This is the path I use
# DRIVER_PATH = '/Users/anand/Desktop/chromedriver'
# Put the path for your ChromeDriver here
DRIVER_PATH = 'Enter/your/path/here'
wd = webdriver.Chrome(executable_path=DRIVER_PATH)
wd.get('https://google.com')
search_box = wd.find_element_by_css_selector('input.gLFyf') #input box selector
search_box.send_keys('Any query')
#wd.quit()
```

creates a window

Second phase of this program would be to search for the query, move into the images section and get the image links. To achieve the following just like the previous code, we'll be using the chrome-driver to navigate using selectors and pages.

```
fetch_image_urls(query:str, max_links_to_fetch:int, wd:webdriver,
sleep_between_interactions:int=1)
```

The function fetch_image_urls takes 4 input arguments but the 4th argument has a default value of 1(sleep interval) so, using only 3 arguments will suffice. The function returns the image-urls.

1. 'query' is the string input in the textbox from the previous snippet.
2. 'max_links_to_fetch' is an integer input that defines the number of images required after scraping.
3. 'wd' is the path to the chromedriver installed on your PC or mac.

```
def fetch_image_urls(query:str, max_links_to_fetch:int, wd:webdriver, sleep_between_interactior
    def scroll_to_end(wd):
        wd.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(sleep_between_interactions)

    # build the google query
    search_url = "https://www.google.com/search?safe=off&site=&tbm=isch&source=hp&q={q}&oq={q}&

    # load the page
    wd.get(search_url.format(q=query))

    image_urls = set()
    image_count = 0
    results_start = 0
    while image_count < max_links_to_fetch:
        scroll_to_end(wd)

        # get all image thumbnail results
        thumbnail_results = wd.find_elements_by_css_selector("img.Q4LuWd")
        number_results = len(thumbnail_results)

        print(f"Found: {number_results} search results. Extracting links from {results_start}:

        for img in thumbnail_results[results_start:number_results]:
```

```
            # try to click every thumbnail such that we can get the real image behind it
            try:
                img.click()
                time.sleep(sleep_between_interactions)
            except Exception:
                continue

            # extract image urls
            actual_images = wd.find_elements_by_css_selector('img.n3VNCb')
            for actual_image in actual_images:
                if actual_image.get_attribute('src') and 'http' in actual_image.get_attribute('
                    image_urls.add(actual_image.get_attribute('src'))

            image_count = len(image_urls)

            if len(image_urls) >= max_links_to_fetch:
                print(f"Found: {len(image_urls)} image links, done!")
                break
        else:
            print("Found:", len(image_urls), "image links, looking for more ...")
            time.sleep(30)
            return
            load_more_button = wd.find_element_by_css_selector(".mye4qd")
            if load_more_button:
                wd.execute_script("document.querySelector('.mye4qd').click();")

        # move the result startpoint further down
```

system. Hence, for that we are going to download images using PILLOW module.

```
$ pip3 install pillow
-----------------------------------------------------------------
Program call:
import PIL
```

once we download the images we need to save the downloaded images in an organised folder of our choice and for that we will be using simple OS commands from the python file.

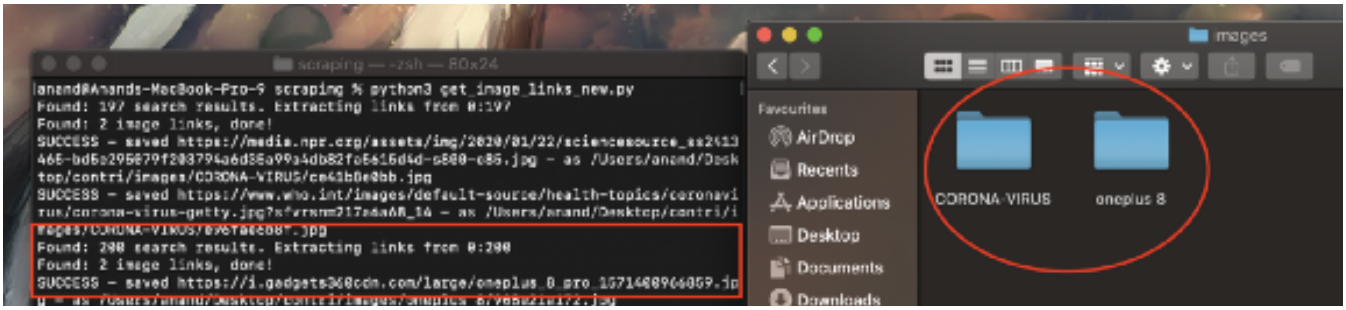
```
def persist_image(folder_path:str,file_name:str,url:str):
```

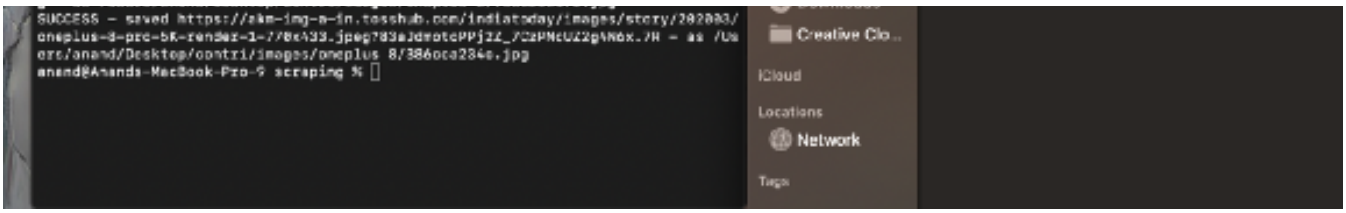
```
    try:
        image_content = requests.get(url).content

    except Exception as e:
        print(f"ERROR - Could not download {url} - {e}")

    try:
        image_file = io.BytesIO(image_content)
        image = Image.open(image_file).convert('RGB')
        folder_path = os.path.join(folder_path,file_name)
        if os.path.exists(folder_path):
            file_path = os.path.join(folder_path,hashlib.sha1(image_content).hexdigest()[:10] +
        else:
            os.mkdir(folder_path)
            file_path = os.path.join(folder_path,hashlib.sha1(image_content).hexdigest()[:10] +
        with open(file_path, 'wb') as f:
            image.save(f, "JPEG", quality=85)
        print(f"SUCCESS - saved {url} - as {file_path}")
    except Exception as e:
        print(f"ERROR - Could not save {url} - {e}")
```

`persist_image(folder_path:str,file_name:str,url:str)`

persist_image function has 3 arguments which are all mandatory for the function call.
1. ‘folder_path’ will be the path to a common folder to where you want to save all the images, mine is “/Users/anand/Desktop/contri/images”.
2. ‘file_name’ this is a str value which is of your choice, but personally i prefer passing file_name = query, because it’ll create separate folders with the query as title.
3. ‘url’ is a string input which we get from the returned values of ‘fetch_image_urls’ function.

I ran the whole program with all snippets put together with two queries, “coronavirus” and “OnePlus 8”. The result is shown below





Created Folders with Respective query images

. . .

Thank You for reading!

Just run the whole code and watch the magic happen :)
libraries to import:

```
import selenium from selenium
import webdriver
import time
import requests
import os
from PIL import Image
import io
import hashlib
---------------------------------
execution:
$ python3 complete_program.py
```

```
import selenium
from selenium import webdriver
import time
import requests
import os
from PIL import Image
import io
import hashlib
# This is the path I use
#DRIVER_PATH = '/Users/anand/Desktop/chromedriver'
# Put the path for your ChromeDriver here
DRIVER_PATH = <enter_your_path>


def fetch_image_urls(query:str, max_links_to_fetch:int, wd:webdriver, sleep_between_interactic
    def scroll_to_end(wd):
        wd.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(sleep_between_interactions)
```

```
    # build the google query
    search_url = "https://www.google.com/search?safe=off&site=&tbm=isch&source=hp&q={q}&oq={q}

    # load the page
    wd.get(search_url.format(q=query))

    image_urls = set()
    image_count = 0
    results_start = 0
    while image_count < max_links_to_fetch:
        scroll_to_end(wd)

        # get all image thumbnail results
        thumbnail_results = wd.find_elements_by_css_selector("img.Q4LuWd")
        number_results = len(thumbnail_results)

        print(f"Found: {number_results} search results. Extracting links from {results_start}:

        for img in thumbnail_results[results_start:number_results]:
            # try to click every thumbnail such that we can get the real image behind it
            try:
                img.click()
                time.sleep(sleep_between_interactions)
            except Exception:
                continue

            # extract image urls
            actual_images = wd.find_elements_by_css_selector('img.n3VNCb')
            for actual_image in actual_images:
                if actual_image.get_attribute('src') and 'http' in actual_image.get_attribute(
                    image_urls.add(actual_image.get_attribute('src'))

            image_count = len(image_urls)

            if len(image_urls) >= max_links_to_fetch:
                print(f"Found: {len(image_urls)} image links, done!")
                break
        else:
            print("Found:", len(image_urls), "image links, looking for more ...")
            time.sleep(30)
            return
            load_more_button = wd.find_element_by_css_selector(".mye4qd")
            if load_more_button:
                wd.execute_script("document.querySelector('.mye4qd').click();")

        # move the result startpoint further down
        results_start = len(thumbnail_results)
```

```
67
68      return image_urls
69
70  def persist_image(folder_path:str,file_name:str,url:str):
71      try:
72          image_content = requests.get(url).content
73
74      except Exception as e:
75          print(f"ERROR - Could not download {url} - {e}")
76
77      try:
78          image_file = io.BytesIO(image_content)
79          image = Image.open(image_file).convert('RGB')
80          folder_path = os.path.join(folder_path,file_name)
81          if os.path.exists(folder_path):
82              file_path = os.path.join(folder_path,hashlib.sha1(image_content).hexdigest()[:10]
83          else:
84              os.mkdir(folder_path)
85              file_path = os.path.join(folder_path,hashlib.sha1(image_content).hexdigest()[:10]
86          with open(file_path, 'wb') as f:
87              image.save(f, "JPEG", quality=85)
88          print(f"SUCCESS - saved {url} - as {file_path}")
89      except Exception as e:
90          print(f"ERROR - Could not save {url} - {e}")
91
92  if __name__ == '__main__':
93      wd = webdriver.Chrome(executable_path=DRIVER_PATH)
94      queries = ["CORONA-VIRUS","oneplus 8"]  #change your set of querries here
95      for query in queries:
96          wd.get('https://google.com')
```