

Search for similar images by text

Author:

Toumi Mustapha Abderrahmane

Under Supervision of :

Mr A.CHENINE



DEPARTMENT OF MATH AND COMPUTER SCIENCE

UNIVERSITY OF IBN KHALDOUNE TIARET

January 2020

Introduction:

Overview:

- This report discusses the work done in the development of [pixabay](#) image downloader and a search for similar images by text using python and [kivy](#) to create the user interface

Background and Motivation:

- Search for images by text can be found in a large variety of applications like search engines because this method gives you quick access to external search functions such as images

Objective:

- Download 50 pictures from [pixabay](#) or [unsplash](#), give each picture a name with the format `studentid_pictureid.jpg` for each picture there is a text file with the same name `studentid_pictureid.txt`, put these pictures in a folder with the name **Pictures**
- Create a search form which enables the user to search for the images in the Pictures Folder by typing a tag in the search form and display the first 10 pictures that have this tag in their text files

Tools:

Programming language:



Figure 1: Python

Framework:

Kivy Open source Python framework for rapid development of applications that make use of innovative user interfaces



Figure 2: Kivy

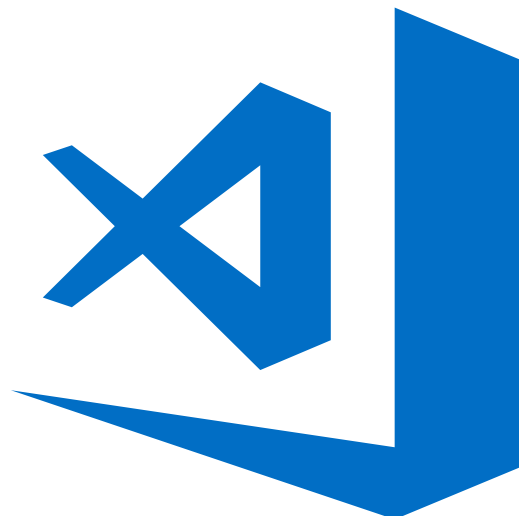
Text editor:

Figure 3: Visual Studio Code

Results :

Pixabay Photos downloader :

- This tool uses a command line interface , implemented on python to communicate with the user .
- Make it easy for any student to download the 50 pictures, give each picture and text file a name , put the pictures tags in the text file, and finally save them in Pictures Folder .
- Inorder to user this tool u should :
 1. install [python](#)
 2. get Pixabay API Key by creating an account on [pixabayApi website](#) Figure4 show this step

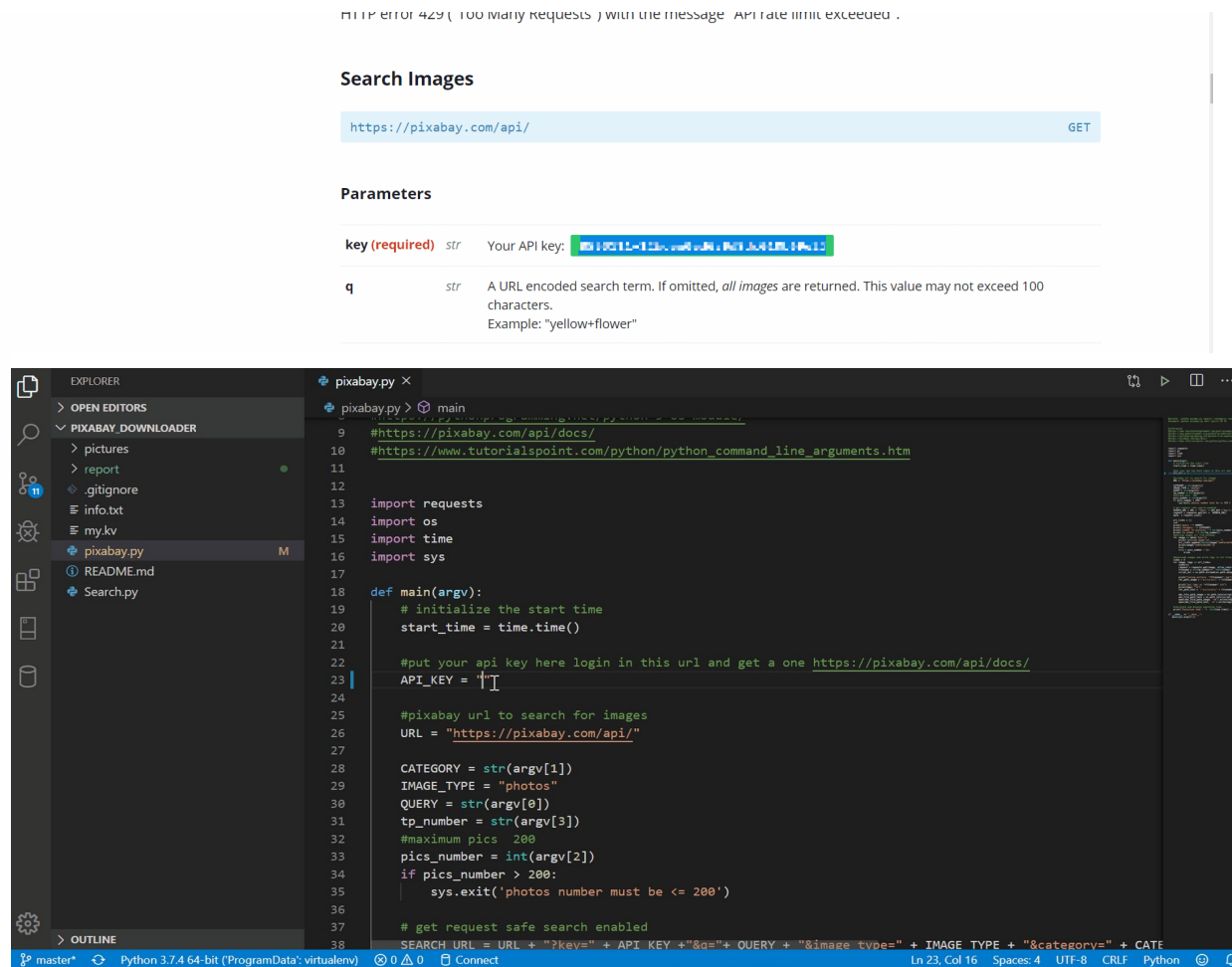


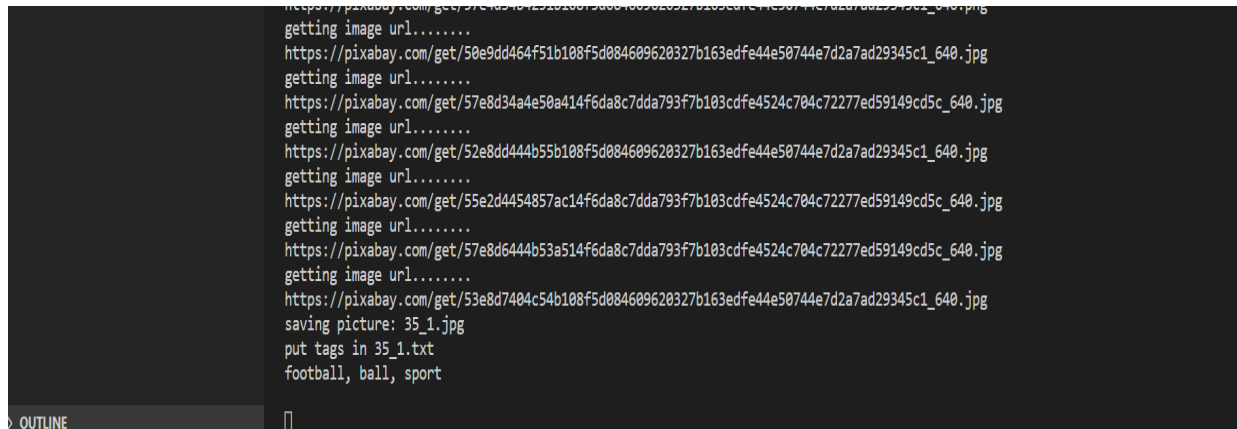
Figure 5: copy your api key in pixabay downloader

3. Copy the api key and put it in `pixaby.py` Figure5 show this step

4. now execute :

```
python pixaby.py <query> <category> <number_of_pics <= 200 > <tp_num>
```

the Figure 6 show the result and the example

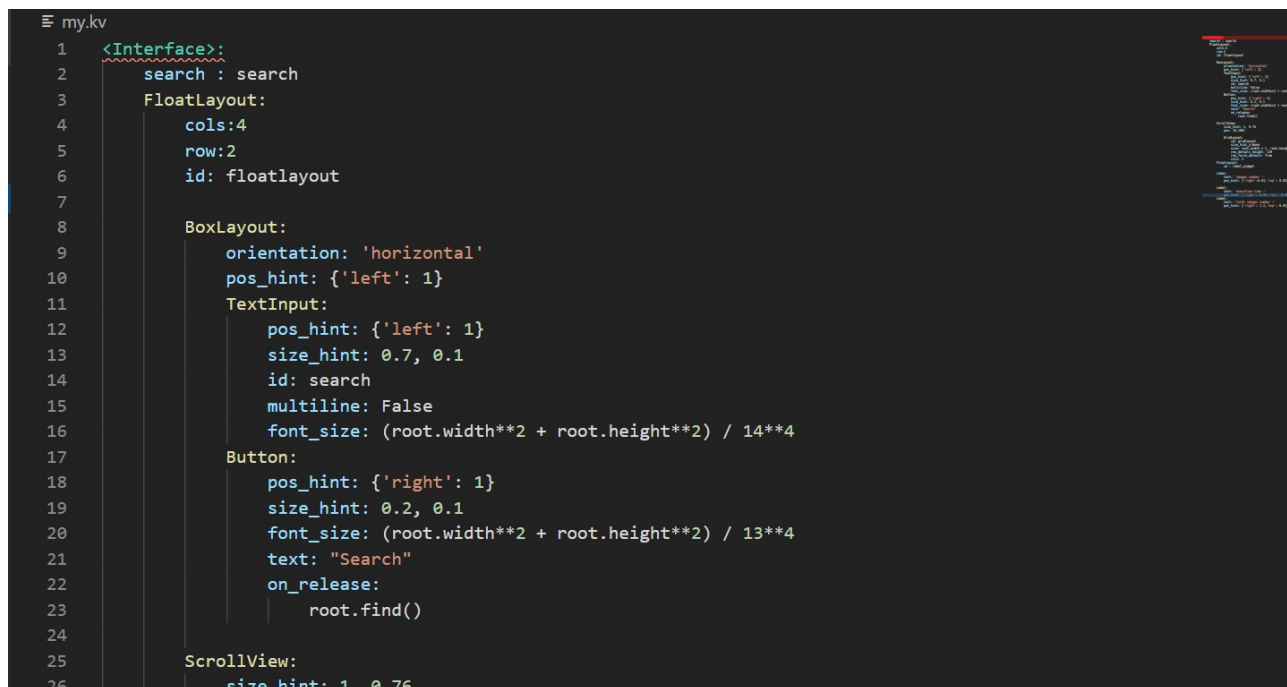


```
https://pixabay.com/get/57e8d7404c54b108f5d084609620327b163edfe44e50744e7d2a7ad29345c1_640.jpg
getting image url.....
https://pixabay.com/get/50e9dd464f51b108f5d084609620327b163edfe44e50744e7d2a7ad29345c1_640.jpg
getting image url.....
https://pixabay.com/get/57e8d34a4e50a414f6da8c7dda793f7b103cdfe4524c704c72277ed59149cd5c_640.jpg
getting image url.....
https://pixabay.com/get/52e8dd444b55b108f5d084609620327b163edfe44e50744e7d2a7ad29345c1_640.jpg
getting image url.....
https://pixabay.com/get/55e2d4454857ac14f6da8c7dda793f7b103cdfe4524c704c72277ed59149cd5c_640.jpg
getting image url.....
https://pixabay.com/get/57e8d6444b53a514f6da8c7dda793f7b103cdfe4524c704c72277ed59149cd5c_640.jpg
getting image url.....
https://pixabay.com/get/53e8d7404c54b108f5d084609620327b163edfe44e50744e7d2a7ad29345c1_640.jpg
saving picture: 35_1.jpg
put tags in 35_1.txt
football, ball, sport
```

Figure 6: How to run and execute pixaby downloader

Images Search:

1. Create an Interface using Kivy



```
1 <Interface>:
2     search : search
3     FloatLayout:
4         cols:4
5         row:2
6         id: floatlayout
7
8     BoxLayout:
9         orientation: 'horizontal'
10        pos_hint: {'left': 1}
11        TextInput:
12            pos_hint: {'left': 1}
13            size_hint: 0.7, 0.1
14            id: search
15            multiline: False
16            font_size: (root.width**2 + root.height**2) / 14**4
17        Button:
18            pos_hint: {'right': 1}
19            size_hint: 0.2, 0.1
20            font_size: (root.width**2 + root.height**2) / 13**4
21            text: "Search"
22            on_release:
23                root.find()
24
25    ScrollView:
26        size hint: 1 0 76
```

Figure 7: Kivy User Interface Code

2. call the Kivy Interface in my Python code

```
        print(abs_image_file_path)
        count+=1
        if count == 10 :
            break;
    self.no_result(count)
    self.add_labels(count, str(time.time() - start_time))
    self.create_info_file(count, str(time.time() - start_time), search_text)

    kv = Builder.load_file("my.kv")

class MainApp(App):
    def build(self):
        return Interface()
```

Figure 8: call kivy interface in the python code

3. find the 10 first pictures which contain the tag that the user searched for

```
for rel_path_text in rel_path_text_list:
    #get full path of txt document
    abs_file_path = os.path.join(script_dir, rel_path_text)

    #get tags from txt file
    file_tags = open(abs_file_path, 'r').read()
    #search in text file
    if search_text in file_tags:
        abs_image_file_path = self.replace(abs_file_path)

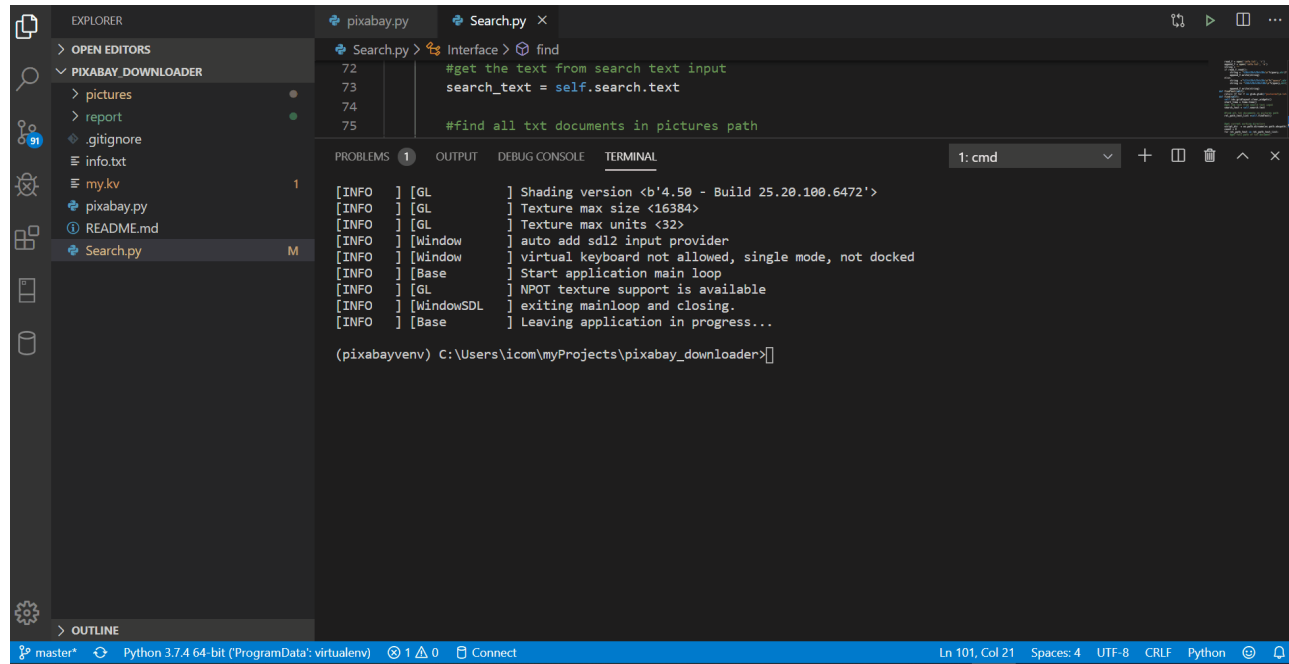
        #add to the grid layout
        self.addtogrid(abs_image_file_path)

        #enable this lane you will see the full path of the image
        #print(abs_image_file_path)

        count+=1
        if count == 10 :
            break;
```

Figure 9: find the 10 pictures

4. Execution:

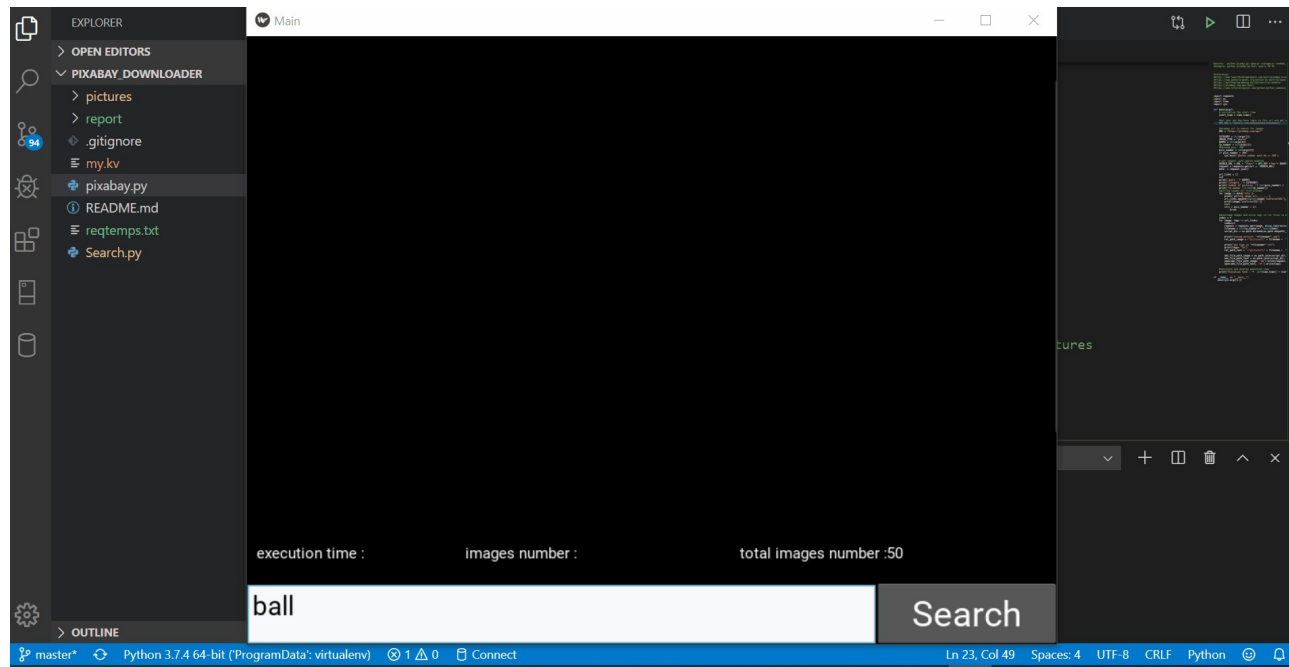


```
72 #get the text from search text input
73 search_text = self.search.text
74
75 #find all txt documents in pictures path
```

```
[INFO ] [GL      ] Shading version <b'4.50 - Build 25.20.100.6472'>
[INFO ] [GL      ] Texture max size <16384>
[INFO ] [GL      ] Texture max units <32>
[INFO ] [Window  ] auto add sdl2 input provider
[INFO ] [Window  ] virtual keyboard not allowed, single mode, not docked
[INFO ] [Base    ] Start application main loop
[INFO ] [GL      ] NPOT texture support is available
[INFO ] [WindowSDL] exiting mainloop and closing.
[INFO ] [Base    ] Leaving application in progress...

(pixabayvenv) C:\Users\icom\myProjects\pixabay_downloader>
```

5. save query info in reqtemps.txt



Complexity:

- In this mini project i prefer to use a Linear search to find words in text file because we have an unsorted files

1. **Find all text files function:** Complexity of $O(n)$ because “f” run through the hole pictures file and add the text files in a new list

```
def findText(self):  
    return [f for f in glob.glob(r"pictures\*.txt")]
```

2. **Search function:** Complexity of $O(n)$

```
def find(self):  
    self.ids.gridlayout.clear_widgets()  $O(1)$   
    start_time = time.time()  
    #get the text from search text input  $O(1)$   
    search_text = self.search.text  
    #find all txt documents in pictures path  $O(n)$   
    rel_path_text_list = self.findText()  
    #get current working directory  $O(1)$   
    script_dir = os.path.dirname(os.path.abspath(__file__))  $O(1)$   
    count = 0  $O(n)$   
    for rel_path_text in rel_path_text_list:  
        #get full path of txt document  $O(1)$   
        abs_file_path = os.path.join(script_dir, rel_path_text)  
  
        #get tags from txt file  
        file_tags = open(abs_file_path, 'r').read()  $O(1)$   
        #search in text file  
        if search_text in file_tags:  
            abs_image_file_path = self.replace(abs_file_path)  $O(1)$   
            #add to the grid layout  
            self.addtogroup(abs_image_file_path)  $O(1)$   
            #enable this lane you will see the full path of the image  
            #print(abs_image_file_path)  
            count+=1  $O(1)$   
            if count == 10 :  $O(1)$   
                break;  
  
    self.no_result(count)  
    self.add_labels(count, str(time.time() - start_time))  
    self.create_info_file(count, str(time.time() - start_time), search_text)
```


Conlusion:

In computer science, analysis of algorithms is a very crucial part. It is important to find the most efficient algorithm for solving a problem. It is possible to have many algorithms to solve a problem, but the challenge here is to choose the most efficient one.