Python+Selenium来定位网页元素和下载图片

上一部分介绍了HTML的构成和对应的标签，这部分将用python结合selenium库完成对元素的定位和下载。

一．元素定位的八种常见方法：

1. 通过id定位元素

find\_element\_by\_id(“id\_vaule”)

2. 通过name定位元素

find\_element\_by\_name(“name\_vaule”)

3. 通过tag\_name定位元素

find\_element\_by\_tag\_name(“tag\_name\_vaule”)

4. 通过class\_name定位元素

find\_element\_by\_class\_name(“class\_name”)

5. 通过css定位元素

find\_element\_by\_css\_selector()

6. 通过xpath定位元素

find\_element\_by\_xpath(“xpath”)

7. 通过link定位

find\_element\_by\_link\_text(“text\_vaule”)

8. 通过partial\_link定位

find\_element\_by\_partial\_link\_text(“text\_vaule”)

通过这些方法可以快速的定位到想要的元素并进行操作。

为了结合RF进行测试，可以将相关操作编写出一个库进行导入。

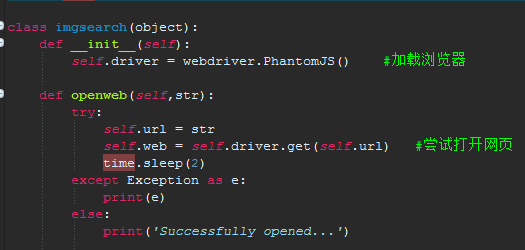
目标：

输入关键字在百度图片中搜索，并将搜索结果的图片数目统计并获取相应的下载地址，最后可以下载到指定的目录中。

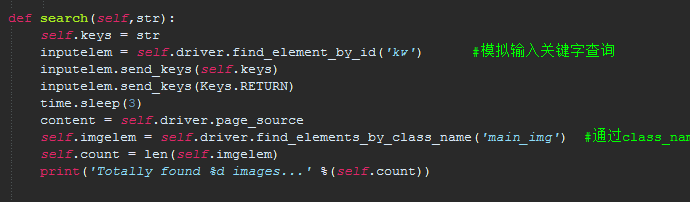
新建一个类，名为imgsearch

其包含3个函数：

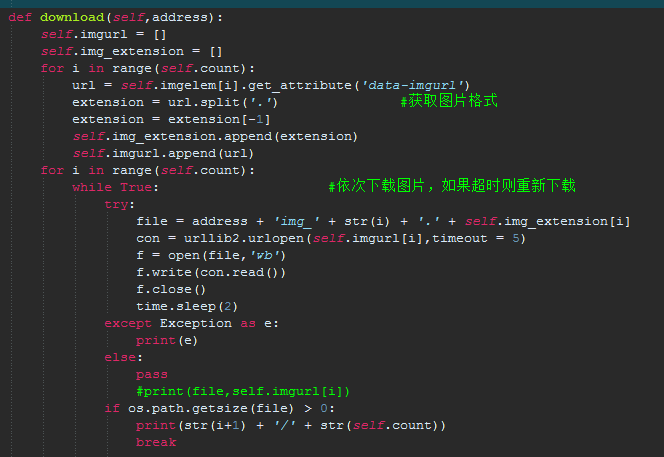
1. openweb(self,str)：输入百度图片的网址；”http://image.baidu.com”，尝试打开网页，成功返回指定消息，失败则返回错误信息。



2. search(self,str)：输入搜索的关键字进行搜索，获得搜索结果，统计图片数目并获取图片的url地址。



3. download(self,address)：输入图片保存的目录，将搜索到的图片存入指定的目录下。 如果下载过程中某个图片下载失败，则将返回错误信息，并尝试重新下载该图片。



4. close(self)：关闭浏览器进程，避免占用系统资源。

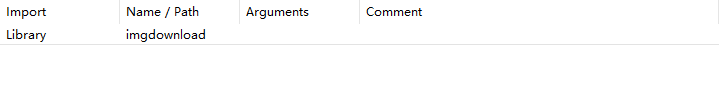
之后再写相对应的\_\_init\_\_.py

from imgsearch import imgsearch

class imgdownload(imgsearch):

ROBOT\_LIBRARY\_SCOPE = 'GLOBAL'

然后打开RF，新建new suite，将做好的库导入



然后新建new test case



运行后将在指定目录下找到下载的图片。