24.12. Searching for tags on flickr

Consider another service, the image sharing site flickr. People interact with the site using a web browser or mobile apps. Users who upload photos can assign one or more short "tags" to each photo. Tags are words like "sunset" or "spaghetti" that describe something about the photo. They can be used for searching or finding related photos.

An API is available to make it easier for application programs to fetch data from the site and post data to the site. That allows third parties to make applications that integrate elements of flickr. Flickr provides the API as a way to increase the value of its service, and thus attract more customers. You can explore the official documentation about the site.

Here we will explore some aspects of one endpoint that flickr provides, for searching for photos matching certain criteria. Check out the full documentation for photo search for details.

This API is more complex than the APIs you examined in previous subchapters, because it requires authentication, and because it requires you to specify that you want results to be returned in JSON format. It also provide more good practice at learning to use an API from the documentation. Plus, photo data is pretty interesting to examine.

The structure of a URL for a photo search on flickr is:

- base URL is `https://api.flickr.com/services/rest/`
- `?`
- **key=value pairs, separated by &s:**
  - One pair is `method=flickr.photos.search`. This says to do a photo search, rather than one of the many other operations that the API allows. Don't be confused by the word "method" here– it is not a python method. That's just the name flickr uses to distinguish among the different operations a client application can request.
  - `format=json`. This says to return results in JSON format.
  - `per_page=5`. This says to return 5 results at a time.
  - `tags=mountains,river`. This says to return things that are tagged with "mountains" and "river".
  - `tag_mode=all`. This says to return things that are tagged with *both* mountains and river.
  - `media=photos`. This says to return photos
  - `api_key=...`. Flickr only lets authorized applications access the API. Each request must include a secret code as a value associated with api_key. Anyone can get a key. See the documentation for how to get one. We recommend that you get one so that you can test out the sample code in this chapter creatively. We have included some cached responses, and they are accessible even without an API key.
  - `nojsoncallback=1`. This says to return the raw JSON result without a function wrapper around the JSON response.

Let's put everything together to make a little retrieval tool for flickr images containing particular tags. Of course, in a browser, you can just use flickr's search tool. But doing this through the API opens up other possibilities that you can explore for features not provided on the regular flickr website.

Below is some code that queries the flickr API for images that have a particular tag.

Note

Searching for "mountains" and "rivers" usually produces beautiful images that are "safe for work", so the example below does that search. We have already cached a response for the particular search in the code window below. That allows the code to run even if you don't provide a valid flickr api_key. We've also checked to make sure that the five returned images are indeed safe for work. If you run this code outside of a browser, or if you do other searches, you will need to provide a valid flickr api_key.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
# import statements
import requests_with_caching
import json
# import webbrowser

# apply for a flickr authentication key at http://www.flickr.com/services/apps/cre
# paste the key (not the secret) as the value of the variable flickr_key
flickr_key = 'yourkeyhere'

def get_flickr_data(tags_string):
    baseurl = "https://api.flickr.com/services/rest/"
    params_diction = {}
    params_diction["api_key"] = flickr_key # from the above global variable
    params_diction["tags"] = tags_string # must be a comma separated string to wor
    params_diction["tag_mode"] = "all"
```

```
found in permanent_cache
https://api.flickr.com/services/rest/?api_key=yourkeyhere&tags=river%2Cmountains&tag_mode=all&meth
od=flickr.photos.search&per_page=5&media=photos&format=json&nojsoncallback=1
https://www.flickr.com/photos/45934971@N07/44858440865
https://www.flickr.com/photos/145056248@N07/43953569330
https://www.flickr.com/photos/145056248@N07/43953448610
https://www.flickr.com/photos/131540074@N08/44857602655
https://www.flickr.com/photos/145056248@N07/44857423045
```

Activity: 1 -- ActiveCode (ac27_10_1)

The response sent back by flickr is loaded into a python dictionary using `json.loads()`.

In the end, we loop through the list of photo dictionaries that were returned, extracting two fields, `owner` and `photo_id`. Those are used to create new URLs that are in the format flickr expects for displaying a webpage containing a single image. In the runestone environment, you'll have to copy those URLs to new tabs to see the photos. In a full python environment, you can uncomment the line of code that imports the webbrowser module and the line of code that calls the `webbrowser.open()` function. If all goes well, that should open five browser tabs, each with a picture that some flickr user had tagged with the words

"mountains" and "rivers".

Check your understanding

requests-11-1: If you wanted to search for photos tagged with *either* river or mountains, rather than requiring both, what would you change in the code? (Hint: check the documentation)

- A. Make two calls, ``get_flickr_data('mountains')`` and ``get_flickr_data('river')``
- B. Call ``get_flickr_data('mountains | river')``
- C. Set ``params_diction["tag_mode"] = "any"``
- D. Set ``params_diction["method"] = "any"``
- E. Set ``params_diction["tag_mode"] = "OR"``

Check me | Compare me

✔ You're a careful documentation reader!

Activity: 2 -- Multiple Choice (24_flickr_1)

Data file: `flickr_cache.txt`

```
{
  "https://api.flickr.com/services/rest/format-json_media-photos_method-flickr.photos.search
}
```

You have attempted 3 of 2 activities on this page

24.11. Searching for Media on iTunes">
24.13. Unicode for non-English characters">
earching for Media on iTunes">
✔ Completed. We
24.13. Unicode for non-English characters">Next Section - 24.13. Unicode for non-English characters

© Copyright 2017 bradleymiller. Created using Runestone 4.1.17.

| Back to top