SI 506 Week 10

Objectives

- Understand the value of caching
- Create your own caching mechanism
- Understand how to store your cache in a file

Materials

- The `get_from_itunes()` function you created last week
- You may find it useful to refer to the caching section of the textbook for this week's exercise

Step 1: Recall the function `get_from_itunes()` you created in last week's section

- **`get_from_itunes()`** makes a request to the iTunes API and returns the list of matching song titles

```
import json
import requests

def get_from_itunes(name, mtype="song"):
      baseurl = "https://itunes.apple.com/search"
      parameters = {}
      parameters["term"] = name
      parameters["entity"] = mtype
      print("Making request to iTunes API...")
      response = requests.get(baseurl, params=parameters)
      python_obj = json.loads(response.text)
      return python_obj
```

Step 2: Write code to ask the user for input which you use to make a request to `get_from_itunes()`

- Use a while loop to take user input for the name of the artist/band name and type of media (song or musicVideo) and use those values to make a request to the iTunes API and print results using `get_from_itunes()`
    - Stop the while loop when the user enters "quit"

- If the user enters ‘The Beatles’ and ‘song’ twice, your code will make two separate requests to the iTunes API, returning the same results each time. This is not an efficient use of network resources.
- This is where caching comes to the rescue!

Step 3: Come up with a good scheme for caching responses that `get_from_itunes()` receives from the iTunes API

- **Plan:**
- How could you use a dictionary to keep track of requests and responses?
    - What should be the key? What should be the value?
- What would make a good unique key for each request?
- **Hint:** Think about the parameters that make each request unique.
- **Hint:** Remember the process that you looked at in the textbook and in lecture? How can you modify this code to implement caching?

Step 4: Modify your program to cache responses using a dictionary and save your cache in a file

- If this is the first request with a particular artist/band name and type of media, add the unique key (from step 3) to your dictionary.
- If a request has already been made with these particular parameters, your code should get the response data from the cache dictionary, rather than from iTunes
- Modify your code so that every time you update your dictionary, you save it to a file called `itunes_cache.json`.
- Modify your code so that every time you run your python file, your cache dictionary is populated with the cached data in `itunes_cache.json`.
- Basically, implement the caching pattern!

Challenges if you are done:

What would happen if a request to iTunes fails? How would you handle this in your code so that the user running the code understands what happened and can easily try again?