Lab 5

Problem definition

This lab requires you to develop a Python script that retrieves a video feed from YouTube, and creates an HTML page (file) that displays a table containing information on the videos from the feed.

The instructor provided you with two files in Blackboard:

- The *youtube.py* file is a script that you need to download it on your computer, rename it (by adding your last names(s) as usual), edit it, and submit it in Blackboard for grading. This is your solution and is the only file that you'll submit.
- The *youtube_riverdance.html* file is a sample HTML file produced by the instructor's solution to lab5. This file shows how the HTML file created by your script should look like. Pay particular attention to the header and the data format in each column. Please do not submit any HTML file in Blackboard for grading. The instructor will run your script to generate the HTML file.

What you will learn

This lab helps you learn a number of things that are useful in many real-world applications that you may develop in Python (or other languages) in the future. More specifically, you will:

- Download and use third-party libraries, upon reading their documentation.
    - You will notice that the documentation typically does not have a step-by-step structure, because every developer uses the library in a slightly different way, and thus takes slightly different steps. The documentation lists the capabilities of the library and provides a short code snippet that illustrates their usage. It is up to you (the developer) to identify the particular facilities (e.g., classes or functions) that you want to use in your application, and then use them accordingly.
- Work with the Google Data API, which includes the YouTube Data API and several other APIs that you might find useful in your future software development endeavors. The Google Data API is one of the most powerful and popular. You can spend months or even years exploring its ever-expanding capabilities and finding new ways of using them in your own projects.
- Use Python to generate an HTML file. This is a basic and fundamental step in developing applications for the Web.
    - The HTML.py module is simple to use and a great match for our lab. However, Python has many other libraries for more advanced HTML generation, some of which are listed at the bottom of the HTML.py homepage (right above the User comments section).

The Approach

The instructor recommends that you develop your solution by taking the following steps, in order:

1. Read this entire document to get a first idea on the task.
2. Download the *youtube.py* and the *youtube_riverdamce.html* files from Blackboard to your computer. Save them in the *c:\Python27* directory (or the home directory of your Python installation). Take a look at these files to get a better idea of what you need to do. The *YouTubeAPI* class will retrieve the video feed, and the *HTMLWriter* class will use that video feed to produce the HTML file. Both classes were created by the instructor for you.
3. To retrieve the video feed, your YouTubeAPI class will call appropriate functions from the Python Client Library for the Google Data API (which includes the YouTube Data API). Before you can use that library, you need to read the developer's guide and download the library.
    a. The most useful sections of the developer's guide are **Requirements** (up to Authentication), **Understanding video feeds and entries** (esp. **Displaying a feed of videos**, **Video entry contents**), and **Retrieving and searching for videos** (esp. **Searching for videos**). Note that we will only work with public feeds in this lab, thus we do not need any form of authentication (developer key, client ID, session token, client login).
    b. The downloaded library is zipped. If you need a free Windows program to extract the files from that archive, you can try 7-zip or another tool of your choice. Make sure you download the correct version (32-bit or 64-bit) for your computer. After you unzip it, copy the **gdata-2.0.18** directory into the **c:\Python27** directory on your computer. Technically, you could copy the gdata-2.0.18 directory anywhere you want and instruct Python (via **sys.path**) to use it from there, but that will require the instructor to configure his sys.path differently for every student's submission.
4. To create an HTML file that is similar to the one provided by the instructor, your HTMLWriter class will call appropriate functions from the *HTML.py* third party module. Before you can use that module, you need to read the tutorial and download the module. The tutorial is short and easy to read. The module also comes in a zip archive. Please read the previous paragraph for instructions on extraction. After you unzip it, you need to copy the **HTML.py-0.04** directory into the **c:\Python27** directory on your computer.
5. At this point, you are ready to start editing the youtube.py file to code your solution to lab5.
    a. The first step is to configure sys.path at the beginning of the script, as instructed in the comments inserted there. If you run your script and Python doesn't complain about the three imports of third-party libraries, your sys.path is well configured. Remember that you need to append to sys.path the full paths leading to the HTML.py module and to all the other modules that you import.
    b. The next step is to go over the statements in the __main__ section of the script. These are the entry point. What are they doing? They create a YouTubeAPI object, then they create a query string, then call the getSearchFeed method on the YouTubeAPI object to get a video feed from YouTube, and then they print that video feed on display so that we can see what we got. The last three statements in our script are using the HTMLWriter class to write the entries from the feed into an HTML file. To that end, they

create an HTMLWriter object, then call the feed2html method which returns a string representation of that object, and then call the html2file method to write that string into an HTML file. Notice that the name of the file includes the query string that you sent to YouTube – please use only letters and spaces in that query, otherwise you might attempt to create an HTML file with an illegal name, and that might fail. The file will be created in the same directory as your script (which is c:\Python27\ in my case).

c. When your script is done running, you can double-click on that HTML file to open it with your default browser. If everything looks right, you are ready to submit your Python script for grading. Otherwise, keep working on it and checking on the result after each run. **Note**: you don't need to close and reopen the HTML file in your browser after each run. You can leave it open and refresh the page after each run (usually by pressing F5).

Understanding youtube.py

Class YouTubeAPI

This class handles all your communication with YouTube. The first four statements in the __main__ section of the script work with this class to that end. The __init__ constructor should not be changed in your application. The remaining three methods in this class are very similar to functions with the same names provided in the developer's guide. Copy & paste and a tiny bit of editing will do the job.

After submitting this lab, if you are interested in exploring further capabilities of the YouTube API, you can keep reading the developer's guide and add new methods to the YouTubeAPI class created by the instructor for you. If you want to use facilities that require authentication (that is, if you want to perform any YouTube action that requires authentication), you'll need to take care of that yourself.

Class HTMLWriter

This is where you will do most of the lab5 work. The __init__ constructor can remain empty because this class does not define any data attributes for its instances. It only defines methods.

The methods

The feed2html() method receives the video feed as an argument and returns a string that is ready to be written to an HTML file on disk. That string is obtained by calling the appropriate function from the HTML.py module. Since that function expects a list of lists as its argument, we first need to convert our video feed to a list of lists. To that end, six data attributes of each video from the feed should be copied into a list of strings. The order of the strings and the format of each string should be identical to those in the table from the youtube_riverdance.html file provided by the instructor.

1. The first string (displayed in the first table column) should be a HTML link (an *…* element) generated by calling the appropriate function from the HTML.py module. This link displays the video title. When the user clicks on it, it will be taken to the YouTube page that plays that video.

2. The second string (displayed in the second table column) should be the publishing date of the video, in the yyyy-mm-dd format.
3. The third string should be the video category
4. The fourth string is the duration of the video. By default, each YouTube video entry in your feed contains the duration in seconds. You need to convert the number of seconds into an hours:minutes:seconds triplet. To that end, you will call the sec2time() method in the HTMLWriter class. This receives the number of seconds as an argument and returns a string that is formatted appropriately. Note that if the duration is 45 minutes and 30 seconds, your video should display 45:30 rather than 0:45:30. This is how YouTube displays the video duration on their page as well.
5. The fifth string is the view count. You will need to insert commas at appropriate places to make this potentially big number easier to read by humans. Again, this is also how YouTube displays the view count on their pages. To format this count appropriately, you will call the formatViewCount method.
6. The sixth string is the average rating of that video. You should only display two decimals.

After we build this list of lists containing all data from our video feed that we want to display, we pass the list as an argument to the appropriate function in the HTML.py module. This function will build an HTML table that contains all the data from our list. The table should also have a header as seen in the example provided by the instructor.