Programming Language Learning Series
Mastery of Python Language
(YouTube Stats Scraper)

Since 2005, YouTube has allowed the public submission of videos, providing a simple platform for users to share their creations. In more recent times, viral videos and YouTube personalities have made an impact on society. With the rise in popularity, a great deal of videos are uploaded and viewed every minute generating an immense amount of statistical data. YouTube provides a means of third-party applications to "scrape" this data.

In this project, we will generate and submit queries to retrieve and view the current state of YouTube videos. Your program will allow the user to generate a YouTube query for:

- Top Rated
- Top Favourites
- Most Viewed
- Most Recent
- Most Discussed

The user may specify the number of results they wish the query to be restricted by. Information received from the query will be displayed on screen in a readable fashion. In addition to basic video information, you will also display extra information about the user who uploaded the video.

Implement the following subtasks:

1. Querying YouTube Data API : The library *urllib* is a way to create a connection to a web server and download data from the web server as if it were a file. The particular function in the library is *urlopen*, which we can use to create the connection. Like any file, you open it, read the contents, then close it. Here is an example:

```
import urllib
web_obj=
urllib.urlopen("http://gdata.youtube.com/feeds/api/standardfeeds/top_rated?max-
results=5&time=today")
results_str = web_obj.read()
web_obj.close()
```

The result is a single string that is the full content of the web query you provided. It is best to use the *read()* function (which returns the entire contents as a single string) as you will find that iteration or other functions such as *readlines()* get somewhat confused on the end of line character.

Programming Language Learning Series
Mastery of Python Language
(YouTube Stats Scraper)

A list of the base URLs to use in order to fulfill the query options are listed below and at http://code.google.com/apis/youtube/2.0/reference.html#Standard_feeds

Name	Feed Id	URL and Description
Top rated	top_rated	**URL:** http://gdata.youtube.com/feeds/api/standardfeeds/top_rated **Description:** This feed contains the most highly rated YouTube videos.
Top favorites	top_favorites	**URL:** http://gdata.youtube.com/feeds/api/standardfeeds/top_favorites **Description:** This feed contains videos most frequently flagged as favorite videos.
Most viewed	most_viewed	**URL:** http://gdata.youtube.com/feeds/api/standardfeeds/most_viewed **Description:** This feed contains the most frequently watched YouTube videos.
Most recent	most_recent	**URL:** http://gdata.youtube.com/feeds/api/standardfeeds/most_recent **Description:** This feed contains the videos most recently submitted to YouTube.
Most discussed	most_discussed	**URL:** http://gdata.youtube.com/feeds/api/standardfeeds/most_discussed **Description:** This feed contains the YouTube videos that have received the most comments.

Make sure that all the above API is working as expected by end of this task.

2. XML Parsing: Once the HTTP request is submitted, a well-defined XML file is returned containing the results. The documentation page thoroughly describes each of these elements:
http://code.google.com/apis/youtube/2.0/reference.html#API_Request_XML_Element_Definitions
A wealth of information is contained in the response, but we are only concerned with a few sections.

Response from Video Queries: Each video entry is encapsulated in a separate <entry>, </entry> tags. Contained within that section are a series of tags describing the particular video.

Example:

```
<entry>
    <media:title type='plain'>Evolution of Dance - By Judson Laipply
    </media:title>

    <author>
        <name>judsonlaipply</name>
        <uri>http://gdata.youtube.com/feeds/api/users/judsonlaipply</uri>
    </author>

    <media:description type='plain'>For more visit
                                        http://www.mightaswelldance.com
    </media:description>

    <yt:statistics favoriteCount='1042868' viewCount='172120091'/>
    .
    .
    .
</entry>
<entry>
    .
    .
</entry>
```

In order to extract the key pieces of information, you will need to parse the associated tags to retrieve the text. In this payload we are only looking for a few things. The string findmethod is most useful here. The elements we want to look for are: title, author, viewCount, favoriteCount. You can look for these in each payload and extract their values. There is still a little work to do there, but this should help quite a bit.

Response from User Queries: A user profile query produces a similar response, except there is only one <entry>.

```
<entry xmlns='http://www.w3.org/2005/Atom'… >

<yt:statistics lastWebAccess='2010-12-01T14:00:23.000-08:00'
subscriberCount='69411' videoWatchCount='0' viewCount='2842770'
totalUploadViews='174888832'/>
    .
    .
    .
</entry>
```

In order to extract the key pieces of information, you will need to parse the associated tags to retrieve the text. This response consists of only one entry, and we are interested in two elements: subscriberCount, totalUploadViews. Make sure that you parse the result and get the right information by end of this task.

3. Class Design: In order to easily track and store information obtained from the query, you will use 3 classes:

- Query
- Video
- User

The Query class will store the results, i.e., videos, produced for a single query request from the user. It is also responsible for calculating general statistics regarding the results. The Video class stores basic information about a single video. Similarly, the User class will store various information about the YouTube user who submitted the related video.

a. Design User Class: The User class will performs query to obtain detailed information about the uploader of a video. This class must allow us to store the following details of user:

- Username (name)
- Number of Subscribers (subscriberCount)
- Total Views of Uploaded Videos (totalUploadViews)

Implementation Detail:

1. **__init__(self, author_str):** The author_str will contain the text encapsulated in the <author> tag of an <entry>. The string must be parsed to extract the YouTube
   username of the Video's uploader. Another HTTP request must be constructed and submitted to obtain the extra information about the uploader.
2. **__repr__(self):** Include this function to display the stored data about the user.

b. Design Video Class: The Video class tracks information about a single YouTube video. This class must allow us to store the following details of video:

- Title (media:title)

- User instance
- Description (media:description)
- Favorite Count (favoriteCount)
- Total Views (viewCount)

Implementation Detail:

1. **__init__(self, entry_str):** The entry_str will contain the text encapsulated in a single <entry> section. The string must be parsed to extract the various pieces of information and the text to create the User instance.
2. **__repr__(self):** Include this function to display the stored Video data, as well as data for the associated uploader.

c. Design Query Class: The Query class will perform the actual HTTP request and initial parsing to build the Video objects from the response. It will also calculate the following information based on the video and user results.

Video Data

- Total Video Favorite Count (favoriteCount)
- Total Video View Count (viewCount)

User Data

- Total User Subscriber Count (subscriberCount)
- Total User Upload View Count (totalUploadViews)

Implementation Detail:

1. **__init__(self, feed_id, max_results):** Takes as input the type of query (feed_id) and the maximum number of results (max_results) that the query should obtain. The correct HTTP request must be constructed and submitted. The results are converted into Video objects, which are stored within this class.
2. **__repr__(self):** Prints out information on each video and YouTube user, including the aforementioned statistics data.

4. Unit Testing: Unit test each class you have written sothat you can be more confident on the project outcome

5. Main Method: Write the code to integrate subtasks and test it on provided dataset.

Example Run

Welcome to the YouTube text-based query application.

You can select a popular feed to perform a query on and view statistical information about the related videos and users.

1) today

2) this week

3) this month

4) since youtube started

Please select a time(or 'Q' to quit):1

1) Top Rated

2) Top Favorited

3) Most Viewed

4) Most Recent

5) Most Discussed

Please select a feed (or 'Q' to quit): 1

Enter the maximum number of results to obtain: 2

****

Top Rated Videos

****

****

Title: Minecraft - "Shadow of Israphel" Part 35: Lastwatch Hold (Minecon Special!)

Views: 903,180

Favorited: 4,907

Uploader: BlueXephos

Author's Statistics:

Subscriber Count: 1,222,878

Total Upload Views: 527,462,846

****

****

Title: Overcrowded Minecraft Farm!?

Views: 113,096

Favorited: 10,819

Uploader: TheSyndicateProject

Author's Statistics:

Subscriber Count: 585,493

Total Upload Views: 182,663,315

****

Total Favorited: 15,726

Total Viewed: 1,016,276

Total Subscribed: 1,808,371

Total Views of Uploaded Videos: 710,126,161

Programming Language Learning Series
Mastery of Python Language
(YouTube Stats Scraper)

---

1) today

2) this week

3) this month

4) since youtube started

Please select a time(or 'Q' to quit):4

1) Top Rated

2) Top Favorited

3) Most Viewed

4) Most Recent

5) Most Discussed

Please select a feed (or 'Q' to quit): 2

Enter the maximum number of results to obtain: 2

****

Top Favorited Videos

****

****

Title: Evolution of Dance - By Judson Laipply

Views: 184,334,352

Favorited: 1,085,632

Uploader: judsonlaipply

Author's Statistics:

Programming Language Learning Series
Mastery of Python Language
(YouTube Stats Scraper)

Subscriber Count: 72,597

Total Upload Views: 187,215,602

****

****

Title: Charlie bit my finger - again !

Views: 389,866,571

Favorited: 1,046,159

Uploader: HDCYT

Author's Statistics:

Subscriber Count: 161,895

Total Upload Views: 501,782,556

****

Total Favorited: 2,131,791

Total Viewed: 574,200,923

Total Subscribed: 234,492

Total Views of Uploaded Videos: 688,998,158

Programming Language Learning Series
Mastery of Python Language
(YouTube Stats Scraper)

1) today

2) this week

3) this month

4) since youtube started

Please select a time(or 'Q' to quit):q

>>>