CSE 231 Fall 2011

Programming Project 10

This assignment is worth 60 points (6.0% of the course grade) and must be **completed and turned in before 11:59 on Wed, November 30, 2011.**

Assignment Overview
This assignment will give you more experience on the use of:
1. classes
2. class methods

The goal of this project is to become more comfortable with creating and using various types of classes together. Knowing how to create and manage multiple instances of a class is important to leverage the power of object-oriented programming. This project will give you the opportunity to construct a series of classes that manage one another.

Create a text-based application that submits YouTube queries, allowing the user to view various statistical information about videos.

Background

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.

Project Description / Specification

Your program will allow the user to generate a YouTube query for:
- Top Rated
- Top Favorites
- 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.

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.

YouTube Queries
To perform the queries, we will use the Google Data API (see *http://code.google.com/apis/youtube/getting_started.html#data_api*). The API is used by sending well-defined HTTP requests, which returns an XML stream containing the results. The Data API has been thoroughly developed and contains a lot of features for doing just anything with the YouTube website; however, we will only deal with a small subset of these features.

In order to communicate with API, we can use the `urllib` module provided in python to send HTTP requests and to read the responses. The request is sent using the `urlopen()` function that takes a string containing the URL. A file-like object is returned and used to read the response.

The extra documentation file discusses the use of the urllib and what XML is (and what you have to do with it).

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.

For example, the URL for obtaining the Top Rated YouTube videos is:
http://gdata.youtube.com/feeds/api/standardfeeds/top_rated

Extra restrictions on the request are added to the URL. The additional parameters are separated by the '&' symbol following an initial '?'. For this project, we will only deal with the maximum number of results parameter (`max-results`) and time parameter (`time`). Time can take one of the following parameters:

`today`	Last 24 hours
`this_week`	Last 7 days
`this_month`	Last month
`all_time`	Since youtube started

To get five (5) of the currently top rated videos today, the query URL would look like:
http://gdata.youtube.com/feeds/api/standardfeeds/top_rated?max-results=5&time=today

To get the top rated video of all time:
http://gdata.youtube.com/feeds/api/standardfeeds/top_rated?max-results=1&time=all_time

Obtaining information about a YouTube user is performed through the following URL, where USERNAME is replaced with the YouTube username.
http://gdata.youtube.com/feeds/api/users/USERNAME

YouTube Results
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 (see 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. 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>
```

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.

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`)

Required Functions

`__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.

`__str__(self)`: Prints out information on each video and YouTube user, including the aforementioned statistics data.

User Class
The User class will perform another query to obtain detailed information about the uploader of a video. This class will store:

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

Required Functions

`__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.

`__str__(self)`: Include this function to display the stored data about the user.

Video Class
The Video class tracks information about a single YouTube video. This class will store:

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

Required Functions

`__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.

`__str__(self)`: Include this function to display the stored Video data, as well as data for the associated uploader.

Main function
The main() function in your program should prompt the user with a choice to select a query on one of the previously mentioned feeds (top rated, most discussed, …). They should also be prompted for the maximum number of results. Results of the query are displayed, and the user is reprompted for another query until they select to quit. Error checking is to be performed on all input, with reprompts for invalid values.

Deliverables

Turn in proj10.py using the handin program.
Save a copy to your H drive.

Notes and Hints:

1. Parsing the XML results requires you to locate certain XML tags. The `find()` function allows you to search for specific text within a string. It returns the index of the first appearance of the search text, or -1 if not found. For example `'abc'.find('bc')` returns `1`.

2. Simplify your code by making a function that takes the name of a tag and a the XML string, then returns the encapsulated text.

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
-----------------------------------------------------------------------------

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:
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
-----------------------------------------------------------------------------

1) today
2) this week
3) this month
4) since youtube started

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