urllib

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.

XML

The problem with the string you fetch above is that it is very long, very complicated and difficult to read as text. It is so long that I cannot put it in this document, it would be ridiculous. However, to see it you can type in the html query above and look at it in a brower. Try that now!!!

That is because it is designed to be a structured text document that is to be read by other programs (not necessarily people). However, once you get past how ugly it is, we can find a few things that help us understand it.

As a structure document, this string has **tags** that mark the beginning and end of information. This is not unlike HTML, which is a simpler kind of structured text document. In our returned query string, we are particularly interested in the payload that is contained between the `<entry extra stuff> ... </entry>` tags. This is the information for each video. If we asked for 5 videos in the query, there should be 5 payloads starting and ending as indicated. We need to extract information from those payloads for our project. Each entry beginning could have some extra stuff in the tag itself, we don't care about that.

How do we find these payloads? There are many ways, and you can imagine some ways to do it. The process of looking at the contents of these payloads is called **parsing**, and you have to find a way to **parse** this XML file to extract the needed information.

For example, we **could** (emphasize could, you can do it as you like) do the following with our `results_str` from above:

```
payload_list = result_str.split('<entry')

len(payload_list) → returns 6
```

If we split the string at the beginning of each payload (the `<entry ...>` tag), then each element of the resulting list should be a payload, and should end with the `</entry>` tag. But, you say, there are 6 entries in the list and we asked for the top 5. What happened?

There's "stuff" at the front of the string we don't care about, and that is the first entry (`payload_list[0]`). We can ignore that. Elements 1-5 should be our payloads.

Even one payload is too long to place in this document. However, in this payload we are only looking for a few things. The string `find` method 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.

I urge you to try this out a bit in the python interpreter and get a feel for what you have to do.

For information about the author, you can send a different query to extract that information as indicated in the original document. The string that comes back from that query has only one entry, and we are interested in two elements: `subscriberCount, totalUploadViews`. You can search for them in the string as well.