YouTube webscraping script documentation

This is the documentation of a script which is written to collect information about videos from the main page of YouTube. It collects the title of the video, the channel name, the number of views and the amount of time passed since the video has been uploaded.

The programming language of this script is Python. I used multiple libraries of Python for this task. I used BeautifulSoup to collect data, selenium to simulate a certain behaviour of a user, in this case to scroll down to the bottom of the page. Finally, I used pandas to export the collected data to an xlsx Excel file.

So, let's see the solution:

At first, we have to import the needed libraries:

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time
import pandas as pd
```

Then we have to define a browser, which opens http://youtube.com and we have to accept the cookies to do any further interactions with the site, so we simulate a click on accepting cookies. We find this by the xpath selector of this element. By xpath selector, we can select the nodes of this item.

```
browser = webdriver.Chrome()

browser.get('http://www.youtube.com')

#click to accept cookies
browser.find_element_by_xpath('/html/body/ytd-app/ytd-consent-bump-v2-
lightbox/tp-yt-paper-dialog/div[2]/div[2]/div[5]/div[2]/ytd-button-
renderer[2]/a/tp-yt-paper-button').click()
```

The following part of the code simulates the scrolling to the bottom of the page. At first, we make the program to „sleep" for 2 seconds. It decreases the chance of that the site will detect us as a robot. Some sites denies access to robots or denies access to users if it detects you as a robot (based on earlier experiences of mine on other projects).

So, what this code does, is at first it gets the actual height of the site. At first it is the height of the last loaded pixel. For example if 2500 pixel is loaded of the site it is 2500. We scroll until we reach the bottom of the page. We always store and overwrite the previous height. If the previous height and the current height is the same, it means that we reached the bottom of this page.

```
time.sleep(2)
```

```
prev_height = browser.execute_script("return 
document.documentElement.scrollHeight")

#scrolling to the bottom of the page
while True:
    browser.execute_script('window.scrollTo(0, 
document.documentElement.scrollHeight);')
    time.sleep(2)

    new_height = browser.execute_script("return 
document.documentElement.scrollHeight")

    if new_height == prev_height:
        print("We have reached the bottom of the page successfully!")
        break
    prev_height = new_height
```

We can parse the site, we define it by a BeautifulSoup constructor and define some empty lists in which we will collect data.

```
bs = BeautifulSoup(browser.page_source, "lxml")

channel = []
title = []
views = []
upload = []
```

At first we collect all ’video elements’. We can make it by the ’ytd-rich-item-renderer’ in this case, which has the ’style-scope ytd-rich-grid-renderer’ class selector. After that we search for video titles, by ’yt-formatted-string’ and ’video-title’ selector in each element of the previously selected ’video-elements’ (each of the whole rectangle, which contains, the index image and details of the video).

So, we go through each of these ’video-elements’ to find all video titles. We have to do this, because I recognized, that there are not only videos on the page, but video playlists also, and it doesn’t have the exact same structure as a ’simple’ video. So I ignored the playlists, because if we consider them, it makes our work much more difficult, because there are video titles and information about the videos of the playlist, but they are not visible if you look at the site, and if the length of the collected datas are not the same, we would not be able to export the data, because we would get error messages.

That is why I only considered the ’video elements’ which has only one title, because it means that it is a video and surely not a playlist, which we want to ignore in this case.

```
divTag = bs.find_all('ytd-rich-item-renderer', {'class': 'style-scope ytd-
rich-grid-renderer'})
```

```python
for tag in divTag:
    titles = tag.findAll('yt-formatted-string', {'id': 'video-title'})

    if len(titles)==1:
        title.append(titles[0].text)
```

I did the same with the channel names:

```python
for tag in divTag:
    channel_raw = tag.findAll('a', {'class': 'yt-simple-endpoint style-scope
yt-formatted-string'})
    if len(channel_raw)==1:
        channel.append(channel_raw[0].text)
```

I did something similar with the views and the time has passed since the upload. They had the similar tags and class selectors, so I collected them together. There were some special cases, which had to be handled. For example a live stream did not have a detail about the upload time/date, only about the active viewers, so if that was the case, I wrote 'élő' (live) into the list.

Another special case was, if there were a video before premier but there was not any waiting viewer. I handled this that I wrote to the list 'premier előtt' (before premier). In any other cases I just attached the text which can be seen on the site.

```python
for tag in divTag:
    v = tag.findAll('span', {'class': 'style-scope ytd-video-meta-block'})

    for i in range(len(v)):
        if "aktív néző" in v[i].text:
            views.append(v[i].text)
            views.append('élő')

        elif "Premier" in v[i].text and "várakozik" not in v[i-1].text:
            views.append('premier előtt')
            views.append(v[i])

        else:
            views.append(v[i].text)
```

Later I separated the viewer numbers and upload data. First a I created another empty list in which I collected the viewer data. I attached every second element starting from the first element. I did the same with the upload data, but started from the second element.

```python
view_final = []

for i in range(0,len(views),2):
    view_final.append(views[i])
```

```python
for i in range(1,len(views),2):
    upload.append(views[i])
```

Finally I created a new empty pandas dataframe. I created columns in which I put the right list and then exported it to an xlsx Excel file.

```python
final = pd.DataFrame()


final['channel'] = channel
final['title'] = title
final['views'] = view_final
final['since_upload'] = upload
final.to_excel('results.xlsx', index=False)
```