BIG DATA COLOMBIA

11 al 15 de febrero 2019
Medellín - Colombia

RADA

Python for Data Science

Web Scraping

Anna Scaife

University of Manchester

Science & Technology
Facilities Council

Training Data

All data science is only as good as the data we put in.

i.e. You can have the world's most perfect classifier, but if you have poor training data then it's worthless.

The data is worth more than the code.

Data is the New Oil

https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-d

Libraries

Import `requests` library for handling weblinks:

```
import requests
```

Import `beautifulsoup` library for searching html:

```
from bs4 import BeautifulSoup as bs
```

Import `pytube` library for extracting data from youtube:

```
from pytube import YouTube
```

requests

```
# specify a web link:
url = "https://www.somewebsite.com"

# use the requests library to get the page:
r = requests.get(url)
```

```
# extract the text from the web link:
page = r.text
```

The text that is returned is the `html` of the webpage.

HTML


```
<!doctype html><html  invert style="font-size: 10px;font-family: Roboto, Arial, sans-serif; background-color: #fafafa;"><head><meta http-equiv="origin-trial" data-feature="Media Capabilities" data-expires="2018-04-12" content="AjLq5uNF7MpG/eM34tWcJD3h8yZY1Q72ckfwdKbUNKGtUNaZkrw55eq2tI60vG0IlsCNw33W9WmuV113EAsdHAwAAABpeyJvcmlnaW4iOiJodHRwczovL3lvdXR1YmUuY29tOjQ0MyIsImZlYXR1cmUiOiJNZWRpYUNhcGFiaWxpdGllcyIsImV4cGlyeSI6MTUyMzQ5MTIwMCwiaXNTdWJkb21haW4iOnRydWV9"><meta http-equiv="origin-trial" data-feature="Long Task Observer" data-expires="2017-04-17" content="AgXf9faUpH8YmYNhInb5nw8BxXZaT8pZlj3At6FUrcvdBzs0I8VxKDkfinT4bbXfPZX8lXKfjotQZrhFVnpzFwYAAABZeyJvcmlnaW4iOiJodHRwczovL3d3dy55b3V0dWJlLmNvbTo0NDMiLCJmZWF0dXJlIjoiTG9uZ1Rhc2tPYnNlcnZlciIsImV4cGlyeSI6MTQ5MjQ3MzYwMH0="><script >var ytcfg = {d: function() {return (window.yt && yt.config_) || ytcfg.data_ || (ytcfg.data_ = {});},get: function(k, o) {return (k in ytcfg.d()) ? ytcfg.d()[k] : o;},set: function() {var a = arguments;if (a.length > 1) {ytcfg.d()[a[0]] = a[1];} else {for (var k in a[0]) {ytcfg.d()[k] = a[0][k];}}}};window.ytcfg.set('EMERGENCY_BASE_URL', "\/error_204?client.name=1\u0026client.version=2.20180905\u0026level=ERROR\u0026t=jserror");</script><link rel="shortcut icon" href="https://s.ytimg.com/yts/img/favicon-vfl8qSV2F.ico" type="image/x-icon" ><link rel="icon" href="https://s.ytimg.com/yts/img/favicon_32-vflOogEID.png" sizes="32x32" ><link rel="icon" href="https://s.ytimg.com/yts/img/favicon_48-vflVjB_Qk.png" sizes="48x48" ><link rel="icon" href="https://s.ytimg.com/yts/img/favicon_96-vflW9Ec0w.png" sizes="96x96" ><link rel="icon" href="https://s.ytimg.com/yts/img/favicon_144-vfliLAfaB.png" sizes="144x144" ><title>YouTube</title><script >var ytcsi = {gt: function(n) {n = (n || '') + 'data_';return ytcsi[n] || (ytcsi[n] = {tick: {},info: {}});},now: window.performance && window.performance.timing &&window.performance.now ? function() {return window.performance.timing.navigationStart + window.performance.now();} : function() {return (new Date()).getTime();},tick: function(l, t, n) {ticks = ytcsi.gt(n).tick;var v = t || ytcsi.now();if (ticks[l]) {ticks['_' + l] = (ticks['_' + l] || [ticks[l]]);ticks['_' + l].push(v);}ticks[l] = v;},info: function(k, v, n) {ytcsi.gt(n).info[k] = v;},setStart: function(s, t, n) {ytcsi.info('yt_sts', s, n);ytcsi.tick('_start', t, n);}};(function(w, d) {ytcsi.setStart('dhs', w.performance ? w.performance.timing.responseStart : null);var isPrerender = (d.visibilityState || d.webkitVisibilityState) == 'prerender';var vName = (!d.visibilityState && d.webkitVisibilityState)? 'webkitvisibilitychange' : 'visibilitychange';if (isPrerender) {ytcsi.info('prerender', 1);var startTick = function() {ytcsi.setStart('dhs');d.removeEventListener(vName, startTick);};d.addEventListener(vName, startTick, false);}if (d.addEventListener) {d.addEventListener(vName, function() {ytcsi.tick('vc');}, false);}var slt = function(el, t) {setTimeout(function() {var n = ytcsi.now();el.loadTime = n;if (el.slt) {el.slt();}}, t);};w.__ytRIL = function(el) {if (!el.getAttribute('data-thumb')) {if (w.requestAnimationFrame) {w.requestAnimationFrame(function() {slt(el, 0);});} else {slt(el, 16);}}};})(window, document);</script>
<script  src="https://s.ytimg.com/yts/jsbin/web-animations-next-lite.min-vflqEtsI7/web-animations-next-lite.min.js" type="text/javascript" name="web-animations-next-lite.min/web-animations-next-lite.min" ></script>
        <script >if (window.ytcsi) {window.ytcsi.tick("rsbe_dph", null, '');}</script>

<link  rel="import" href="https://s.ytimg.com/yts/htmlbin/desktop_polymer-vflsQgFuS.html" name="desktop_polymer" >        <script >if (window.ytcsi) {window.ytcsi.tick("rsae_dph", null, '');}</script>
```

beautifulsoup

`beautifulsoup` makes it easier for us to search through the html.

```
# use beautifulsoup to parse the html:
soup=bs(page,'html.parser')
```

YouTube

```
# define the base url and the search string:
base = "https://www.youtube.com/results?search_query="
qstring = "cat+videos"

# use the requests library to get the page:
r = requests.get(base+qstring)

# extract the text from the web link:
page = r.text

# use beautifulsoup to parse the html:
soup=bs(page,'html.parser')
```

YouTube

Find all the hyperlink elements with the html `class attribute` "yt-uix-tile-link":

```
vids = soup.findAll('a',attrs={'class':'yt-uix-tile-link'})
```

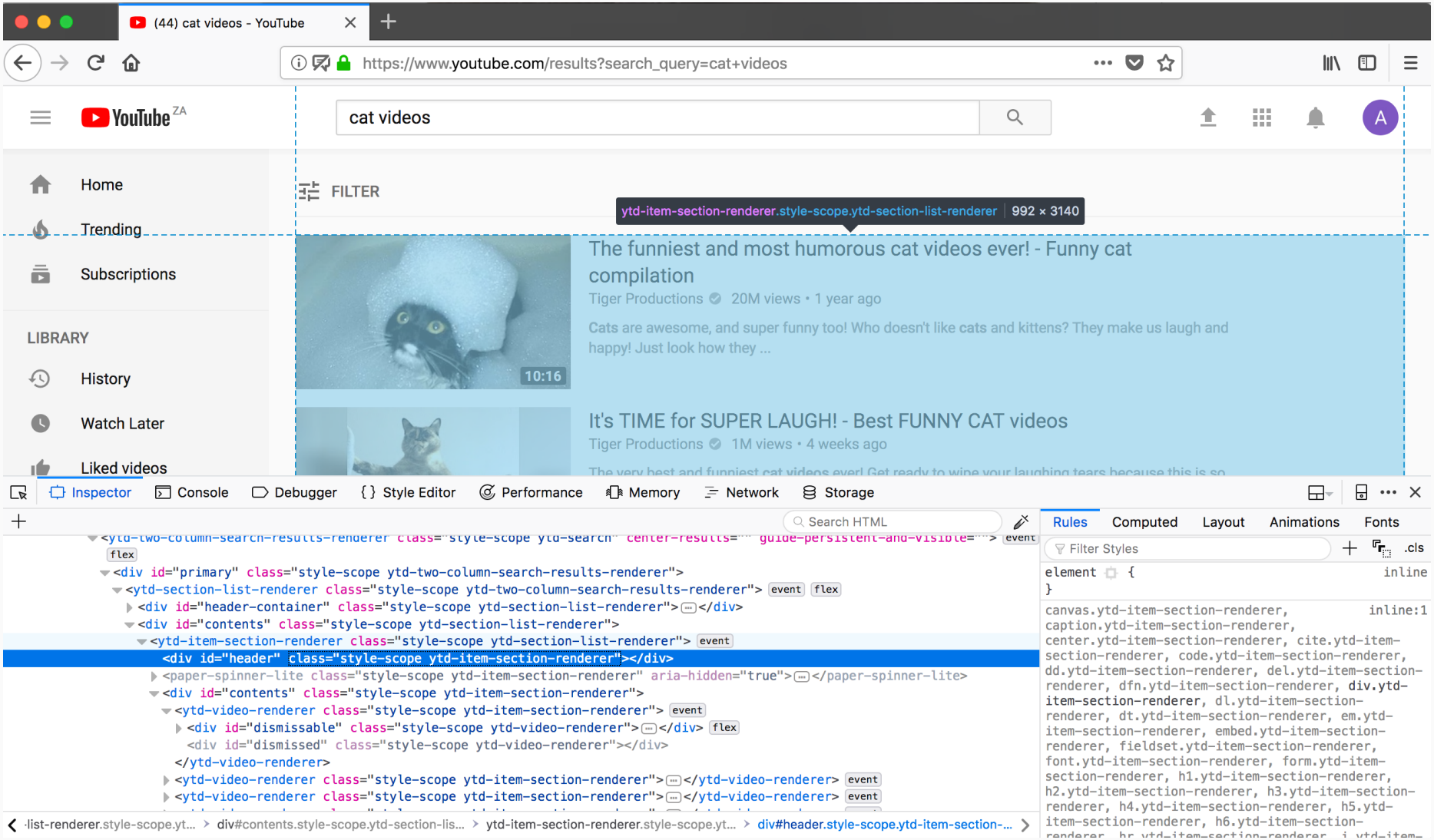
```
In [17]: print vids[0]

<a aria-describedby="description-id-172285" class="yt-uix-tile-link yt-ui-ellipsis yt-ui-ellipsis-2 yt-uix-sessionlin
k spf-link " data-sessionlink="itct=CFoQ3DAYACITCL_sxavhrd0CFU-o1QodmzYIFij0JFIKY2F0IHZpZGVvcw" dir="ltr" href="/watc
h?v=pOmu0LtcI6Y" rel="spf-prefetch" title="It's TIME for SUPER LAUGH! - Best FUNNY CAT videos">It's TIME for SUPER LA
UGH! - Best FUNNY CAT videos</a>
```

HTML reference:

`https://www.w3schools.com/html/default.asp`

YouTube


(44) cat videos - YouTube
https://www.youtube.com/results?search_query=cat+videos
YouTube ZA
cat videos
Home
Trending
Subscriptions
LIBRARY
History
Watch Later
Liked videos
FILTER
ytd-item-section-renderer.style-scope.ytd-section-list-renderer | 992 × 3140
The funniest and most humorous cat videos ever! - Funny cat compilation
Tiger Productions 20M views • 1 year ago
Cats are awesome, and super funny too! Who doesn't like cats and kittens? They make us laugh and happy! Just look how they ...
10:16
It's TIME for SUPER LAUGH! - Best FUNNY CAT videos
Tiger Productions 1M views • 4 weeks ago
Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Search HTML
Rules
Computed
Layout
Animations
Fonts
Filter Styles
element { inline }

<ytd-section-list-renderer class="style-scope ytd-two-column-search-results-renderer">

<ytd-item-section-renderer class="style-scope ytd-section-list-renderer">

<paper-spinner-lite class="style-scope ytd-item-section-renderer" aria-hidden="true">

<ytd-video-renderer class="style-scope ytd-item-section-renderer">

</ytd-video-renderer>
<ytd-video-renderer class="style-scope ytd-item-section-renderer">
<ytd-video-renderer class="style-scope ytd-item-section-renderer">
div#contents.style-scope.ytd-section-lis...
ytd-item-section-renderer.style-scope.yt...
div#header.style-scope.ytd-item-section-...

YouTube

`beautifulsoup findAll` returns a list. We need to extract the `hyperref` for each element, i.e. the url:

```
videolist=[]
for v in vids:
    tmp = 'https://www.youtube.com' + v['href']
    videolist.append(tmp)
```

YouTube

```
count=0
for item in videolist:

    # increment counter:
    count+=1

    # initiate the class:
    yt = YouTube(item)

    # have a look at the different formats available:
    #formats = yt.get_videos()

    # grab the video:
    video = yt.get('mp4', '360p')

    # set the output file name:
    yt.set_filename('Video_'+str(count))

    # download the video:
    video.download('./')
```

YouTube

A cautionary note: the YouTube ToS do state that:

> **5.1 H.** you agree not to use or launch any automated system (including, without limitation, any robot, spider or offline reader) that accesses the Service in a manner that sends more request messages to the YouTube servers in a given period of time than a human can reasonably produce in the same period by using a publicly available, standard (i.e. not modified) web browser;

YouTube


ALL YOUR BASE ARE BELONG TO US
Physics. Data Science. General Geekery. Probably Coffee.