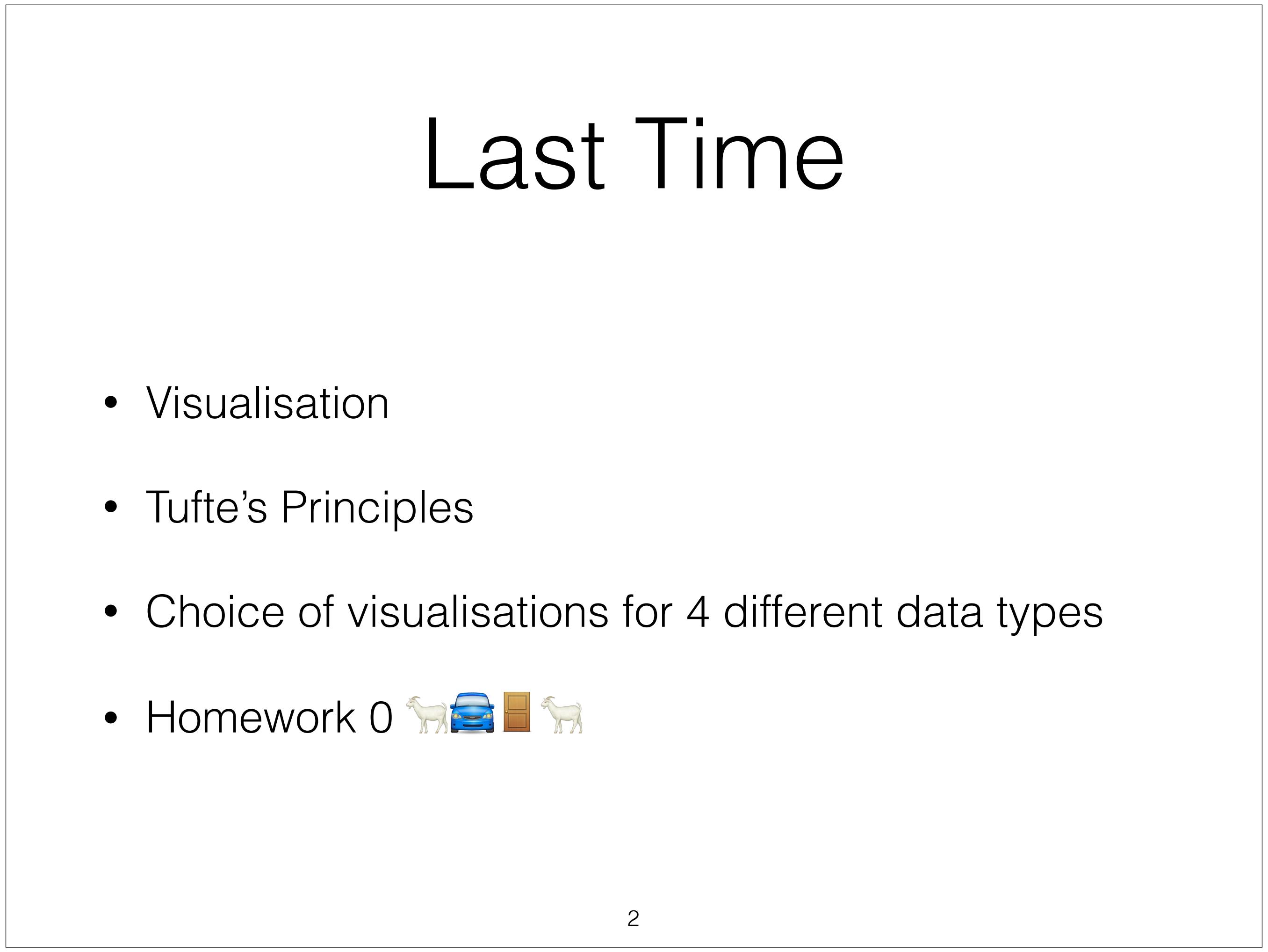
Freie Universität Berlin

Data Science Lecture 02

24.04.2015

Dr. Kashif Rasul
kashif @krasul #167

Welcome to Data Science Lecture 2.



Last Time

- Visualisation
- Tufte’s Principles
- Choice of visualisations for 4 different data types
- Homework 0 🐐🚘🚪🐐

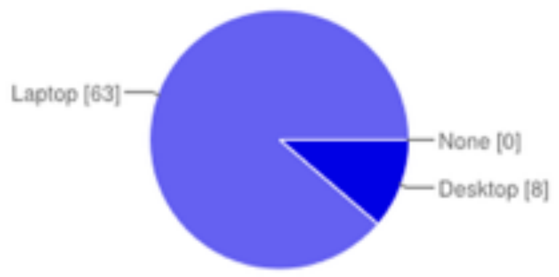
Last time we gave you an overview of information visualisation and did not really get into story telling.

MAKING COMICS
STORYTELLING SECRETS OF COMICS, MANGA
UNDERSTANDING COMICS
THE INVISIBLE ART
SCOTT McCLOUD
"A REMARKABLE NEW BAEDEKER OF THE TOONS."
REINVENTING COMICS
Imagination and Technology Revolutionizing an Art Form

And if we get time near the end of the course, we'll dive into story telling a bit more, but if you want to understand story telling at least from the point of view of comics, then all of Scott McCloud's books are worth reading. If you get one of his books then we recommend "Understanding Comics" from 1993.

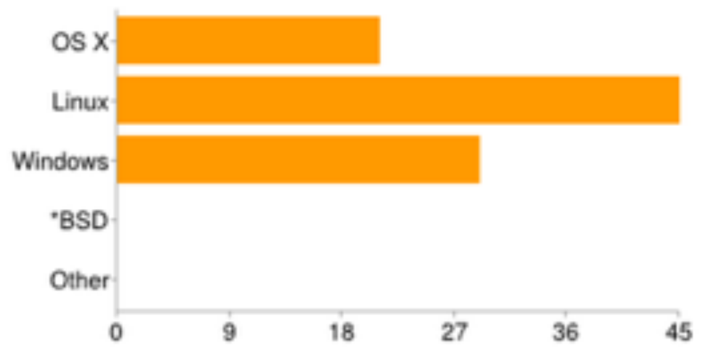
The pedagogical potential of the comic form is vast and mostly untapped. This book explores the formal aspects of comics, the historical development of the medium, its fundamental vocabulary, and various ways in which these elements have been used. It expounds theoretical ideas about comics as an art form and medium of communication and story telling.

What kind(s) of computer(s) do you own?



Desktop	**8**	11.3%
Laptop	**63**	88.7%
None	**0**	0%

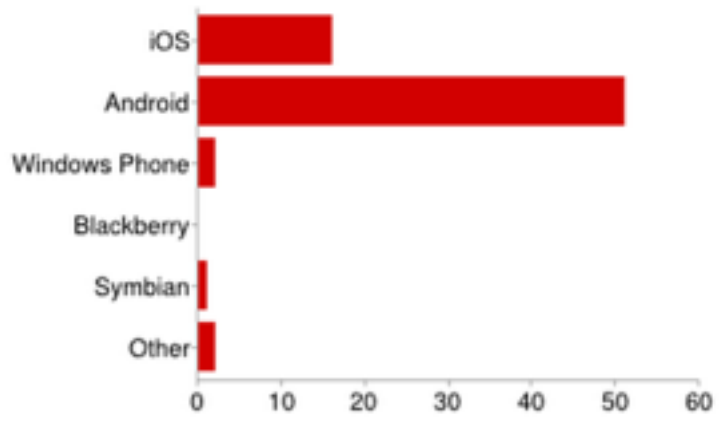
What operating system(s) do you run on your computer(s)?



OS X	**21**	29.6%
Linux	**45**	63.4%
Windows	**29**	40.8%
*BSD	**0**	0%
Other	**0**	0%

Here are the results of the initial student survey. Thank you for that. We will take into account the great feedback.

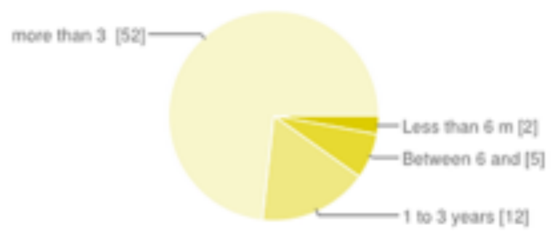
What operating system(s) do you run on your mobile(s)?



iOS	**16**	22.5%
Android	**51**	71.8%
Windows Phone	**2**	2.8%
Blackberry	**0**	0%
Symbian	**1**	1.4%
Other	**2**	2.8%

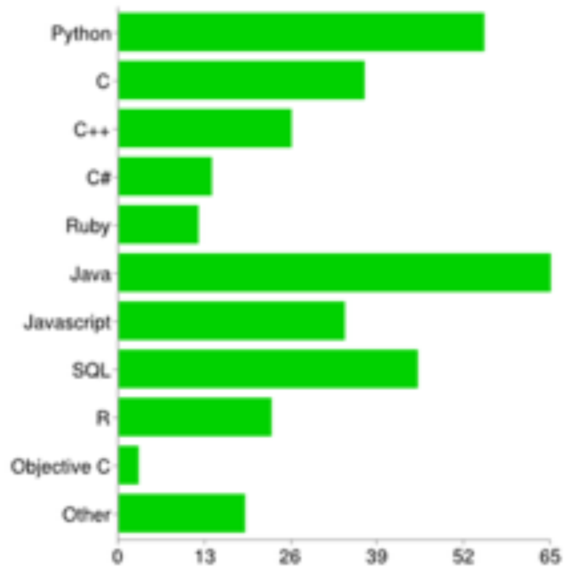
Android seems the dominant mobile OS here.

How long have you been programming?



Less than 6 months	2	2.8%
Between 6 and 12 months	5	7%
1 to 3 years	12	16.9%
more than 3 years	52	73.2%

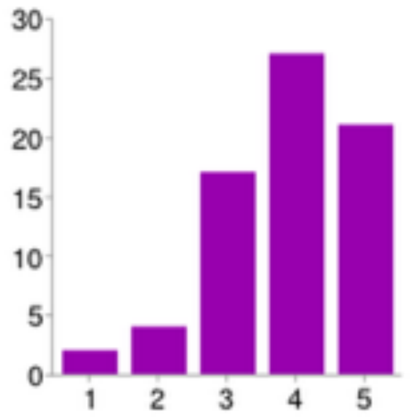
What languages do you know?



Python	55	77.5%
C	37	52.1%
C++	26	36.6%
C#	14	19.7%
Ruby	12	16.9%
Java	65	91.5%
Javascript	34	47.9%
SQL	45	63.4%
R	23	32.4%
Objective C	3	4.2%
Other	19	26.8%

Ok so Python seems like a good choice for this course. No surprise here at that Java is the most popular language here.

Overall, how comfortable are you with programming?



1	**2**	2.8%
2	**4**	5.6%
3	**17**	23.9%
4	**27**	38%
5	**21**	29.6%

Today

- Data Scrapping, Munging and Cleaning
- Live coding
- Exercise 1
- Homework 1

Getting WWW data

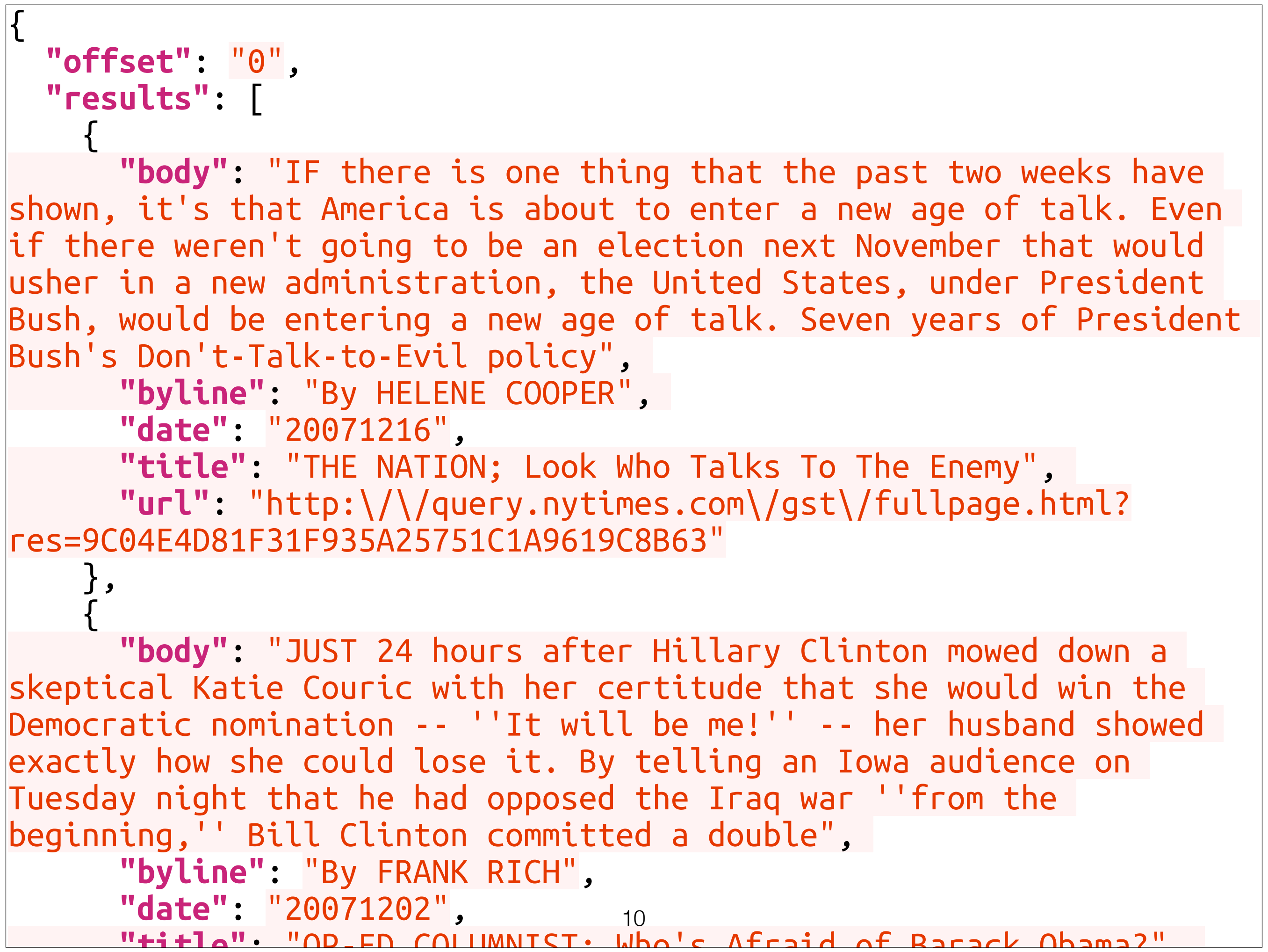
- Typical workflow: Direct Download (check wiki)
- Make an API call (twitter, facebook etc.)
- Web scrapping: Homework 1

Let us turn our attention now to the problem of getting data from the WWW and cover some basics so we can be all on the same page.

When a website offers data then it is relatively easy to get the data in some standard format. On the github wiki there is a Data sources page with links to sources of data.

More and more often a service provides an API to get data.

Lastly, if you can't get the data in any other way, you might be forced to scrape the data.

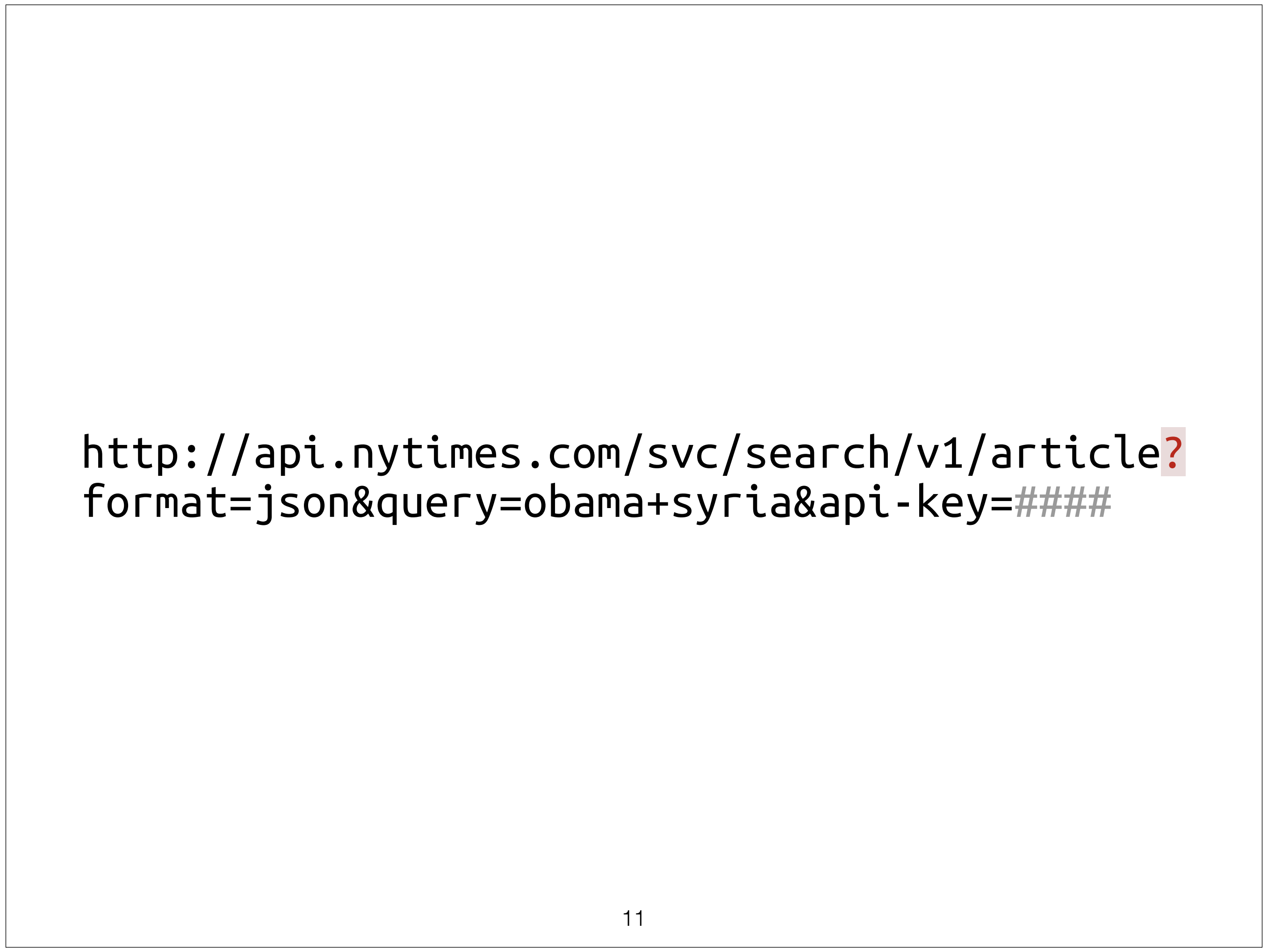


```
{
  "offset": "0",
  "results": [
    {
      "body": "IF there is one thing that the past two weeks have
shown, it's that America is about to enter a new age of talk. Even
if there weren't going to be an election next November that would
usher in a new administration, the United States, under President
Bush, would be entering a new age of talk. Seven years of President
Bush's Don't-Talk-to-Evil policy",
      "byline": "By HELENE COOPER",
      "date": "20071216",
      "title": "THE NATION; Look Who Talks To The Enemy",
      "url": "http:\/\/query.nytimes.com\/gst\/fullpage.html?
res=9C04E4D81F31F935A25751C1A9619C8B63"
    },
    {
      "body": "JUST 24 hours after Hillary Clinton mowed down a
skeptical Katie Couric with her certitude that she would win the
Democratic nomination -- ''It will be me!'' -- her husband showed
exactly how she could lose it. By telling an Iowa audience on
Tuesday night that he had opposed the Iraq war ''from the
beginning,'' Bill Clinton committed a double",
      "byline": "By FRANK RICH",
      "date": "20071202",
      "title": "OP-ED COLUMNIST; Who's Afraid of Barack Obama?"
```

Typically an API call will return data in a format called JSON (JavaScript Object Notation). It corresponds to a mix between Dictionaries (key, values) and Lists.

It’s popularity is growing and is the interface we will try to use most often.

Here is and example of the result of an API request to the New York Times.



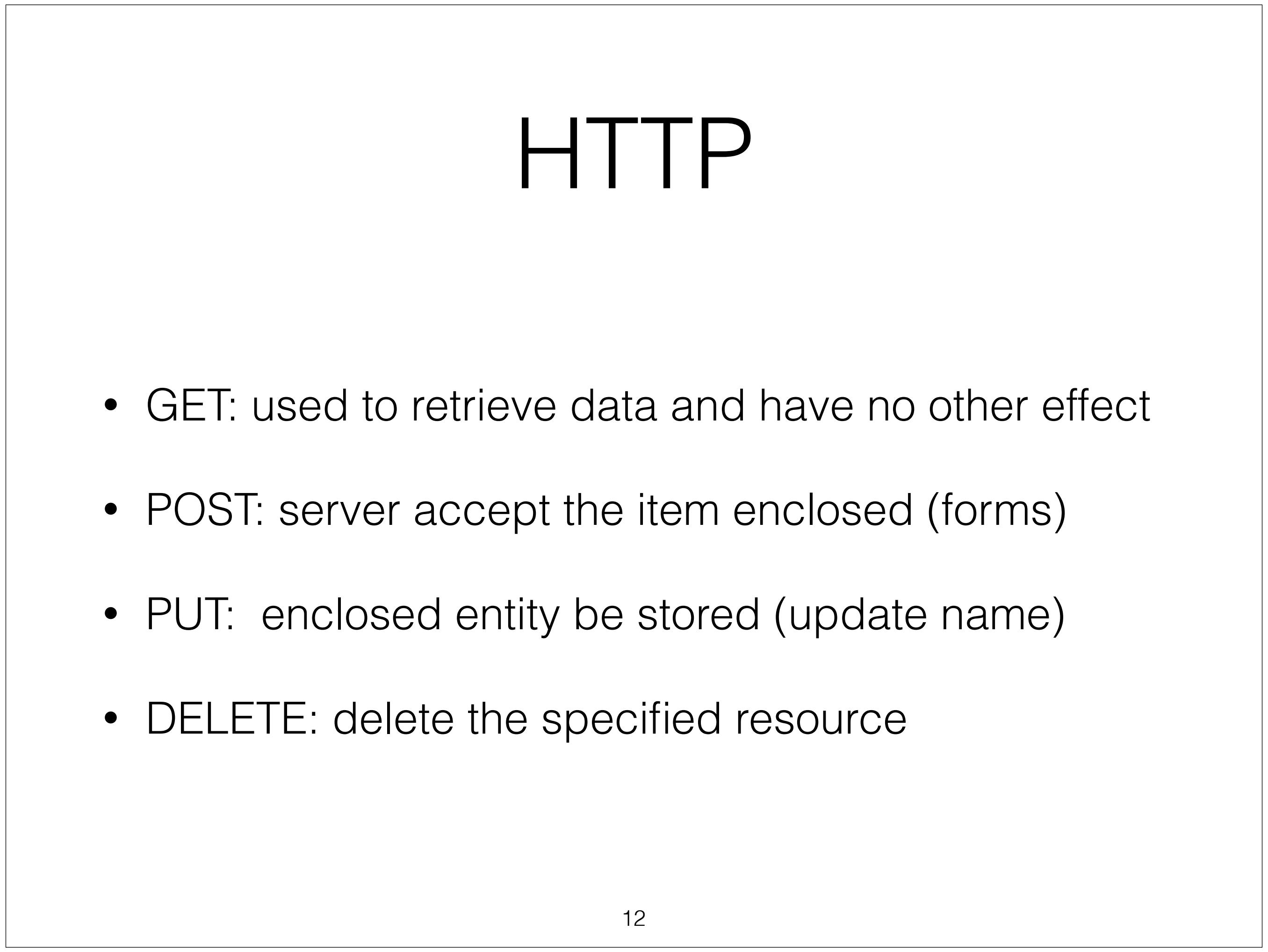
```
http://api.nytimes.com/svc/search/v1/article?
format=json&query=obama+syria&api-key=####
```

Now this particular JSON was received after we made a call to this Request URL.

Now this Web query string is made up of the URL and a search part which is separated by a ‘?’ character. The search part is made up of a series of key value fields separated by the ‘&’ character.

One final thing to note is that most API’s require you to register with an API key. By changing the search part we can request terms we want on the fly.

A good API will also come with some documentation on the query terms and the type of responses one should accept. For example if I want to search for some term in the title of articles, I can look up the NewYork Times’ API documentation.



HTTP

- GET: used to retrieve data and have no other effect
- POST: server accept the item enclosed (forms)
- PUT: enclosed entity be stored (update name)
- DELETE: delete the specified resource

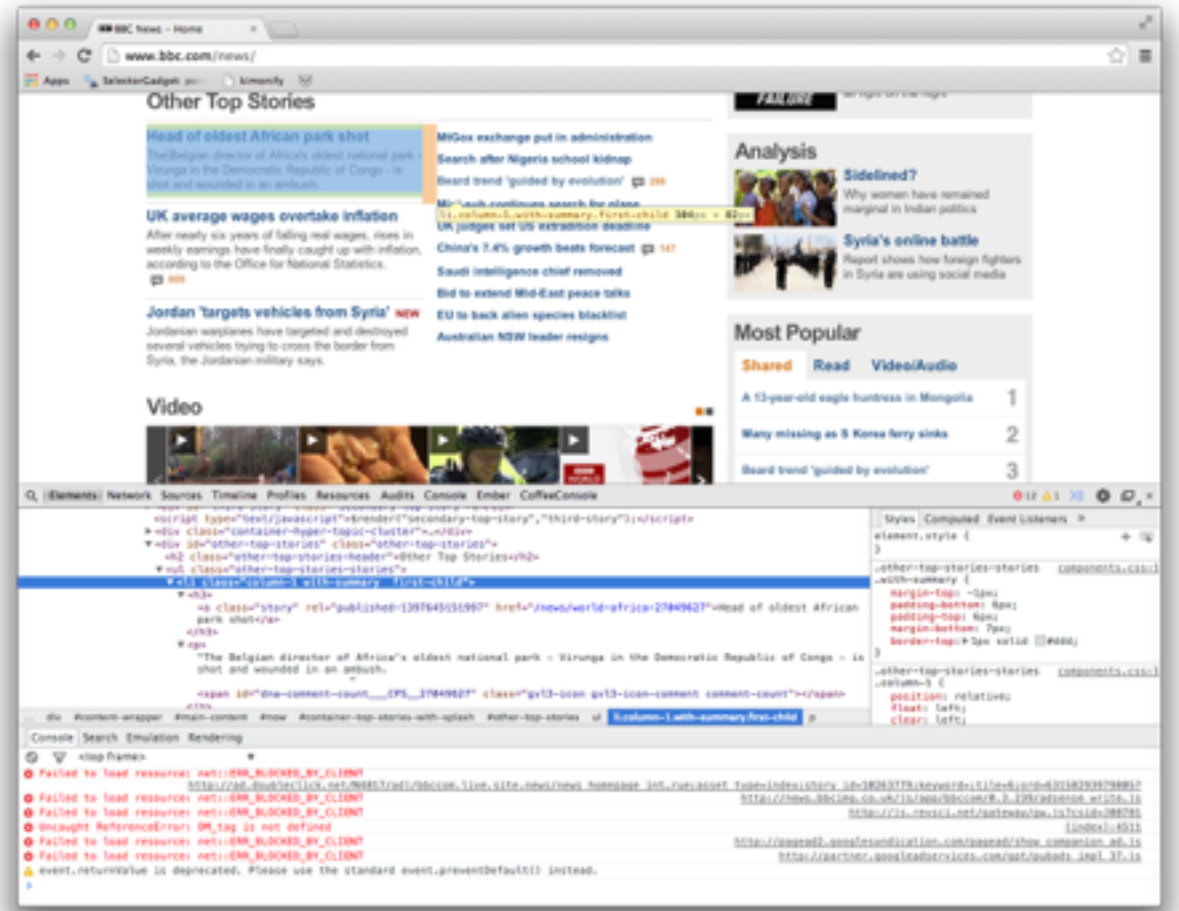
We mentioned a GET request, but to step back a bit, we need to talk about HTTP. HTTP is a application protocol for distributed, collaborative, hypermedia information systems. Web servers and web browsers communicate with each other via this protocol.

HTTP defines methods (sometimes referred to as verbs) to indicate the desired action to be performed on the identified resource. The four most common verbs are:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<result_set>
    <status>
        OK
    </status>
    <copyright>
        Copyright (c) 2014 The New York Times Company.
    </copyright>
    <num_results>
        25
    </num_results>
    <last_modified>
        2014-04-11T13:07:16-04:00
    </last_modified>
    <results>
        <book>
            <list_name>
                Hardcover Fiction
            </list_name>
            <display_name>
                Hardcover Fiction
            </display_name>
            <updated>
                WEEKLY
            </updated>
```

Sometimes the response format is in XML and is characterised by opening and closing tags in a hierarchical or nested manner.

This is different from Json, but XML is extremely powerful in its own sense and more verbose and a bit more cumbersome to use for humans.



```
www.bbc.com/news/

Other Top Stories

Head of oldest African park shot
The Belgian director of Africa's oldest national park - Virunga in the Democratic Republic of Congo - is shot and wounded in an ambush.

UK average wages overtake inflation
After nearly six years of falling real wages, rises in weekly earnings have finally caught up with inflation, according to the Office for National Statistics.

Jordan 'targets vehicles from Syria' NEW
Jordanian warplanes have targeted and destroyed several vehicles trying to cross the border from Syria, the Jordanian military says.

MtGox exchange put in administration
Search after Nigeria school kidnap
Beard trend 'guided by evolution'
UK judges set US extradition deadline
China's 7.4% growth beats forecast
Saudi intelligence chief removed
Bid to extend Mid-East peace talks
EU to back alien species blacklist
Australian NSW leader resigns

Analysis
Sidelined?
Why women have remained marginal in Indian politics
Syria's online battle
Report shows how foreign fighters in Syria are using social media

Most Popular
Shared  Read  Video/Audio
A 13-year-old eagle huntress in Mongolia  1
Many missing as S Korea ferry sinks  2
Beard trend 'guided by evolution'  3

Video
```

```
<script type="text/javascript">$render("secondary-top-story","third-story");</script>
▶<div class="container-hyper-topic-cluster">…</div>
▼<div id="other-top-stories" class="other-top-stories">
   <h2 class="other-top-stories-header">Other Top Stories</h2>
  ▼<ul class="other-top-stories-stories">
   ▼<li class="column-1 with-summary  first-child">
     ▼<h3>
        <a class="story" rel="published-1397645151997" href="/news/world-africa-27049627">Head of oldest African park shot</a>
      </h3>
     ▼<p>
        "The Belgian director of Africa's oldest national park - Virunga in the Democratic Republic of Congo - is shot and wounded in an ambush.
        <span id="dna-comment-count___CPS__27049627" class="gvl3-icon gvl3-icon-comment comment-count"></span>
```

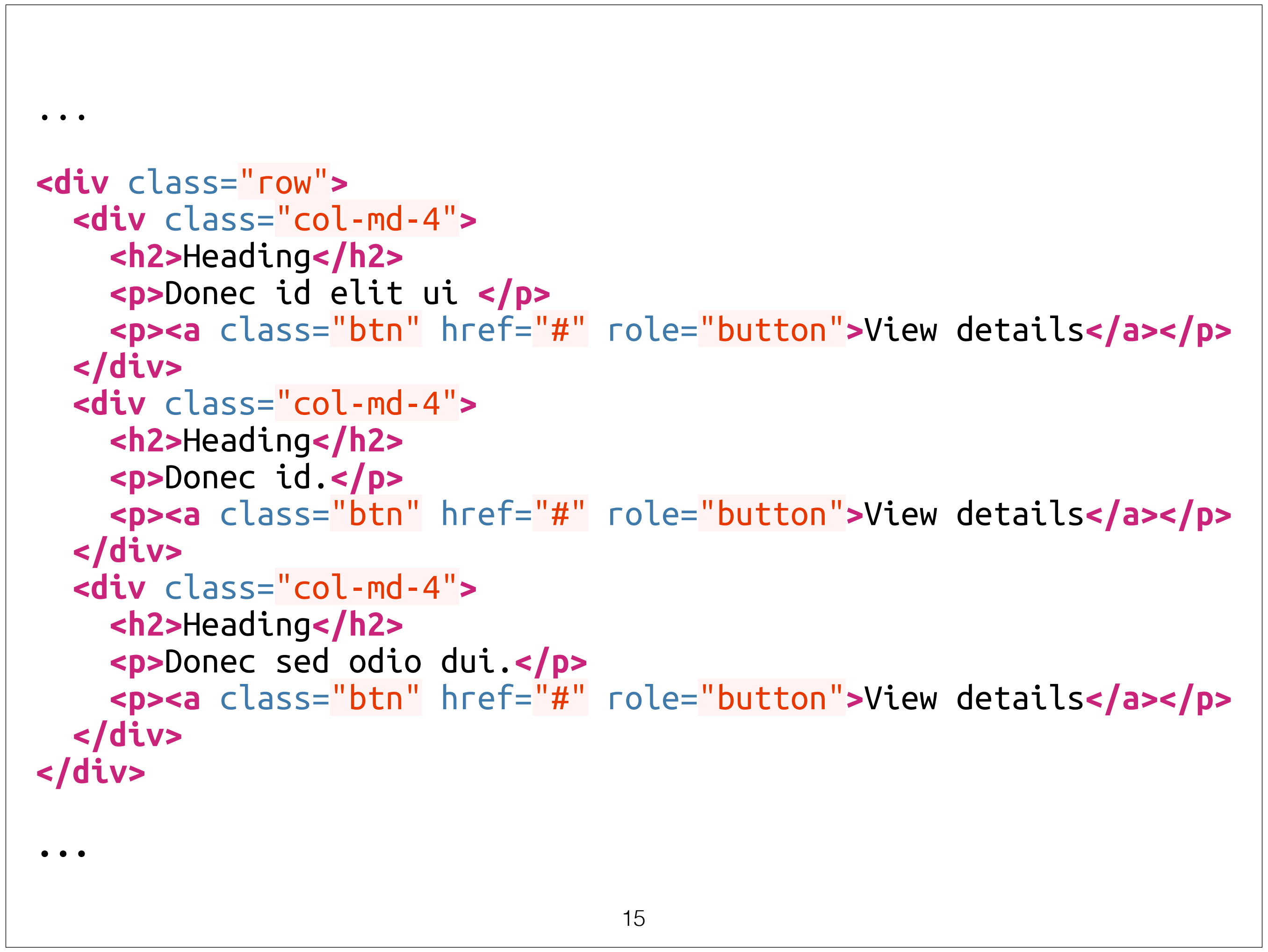
```
element.style {
}
.other-top-stories-stories .with-summary {
  margin-top: -1px;
  padding-bottom: 4px;
  padding-top: 4px;
  margin-bottom: 7px;
  border-top: 1px solid #ddd;
}
.other-top-stories-stories .column-1 {
  position: relative;
  float: left;
  clear: left;
```

```
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Uncaught ReferenceError: DM_tag is not defined
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
Failed to load resource: net::ERR_BLOCKED_BY_CLIENT
event.returnValue is deprecated. Please use the standard event.preventDefault() instead.
```

Finally we can always use Chrome's developer tools and look at the HTML source of any webpage which is online. Go to View -> Developer Tools.

HTML is a simplified version of XML with predefined tags. So for web scraping it is important that you understand HTML.

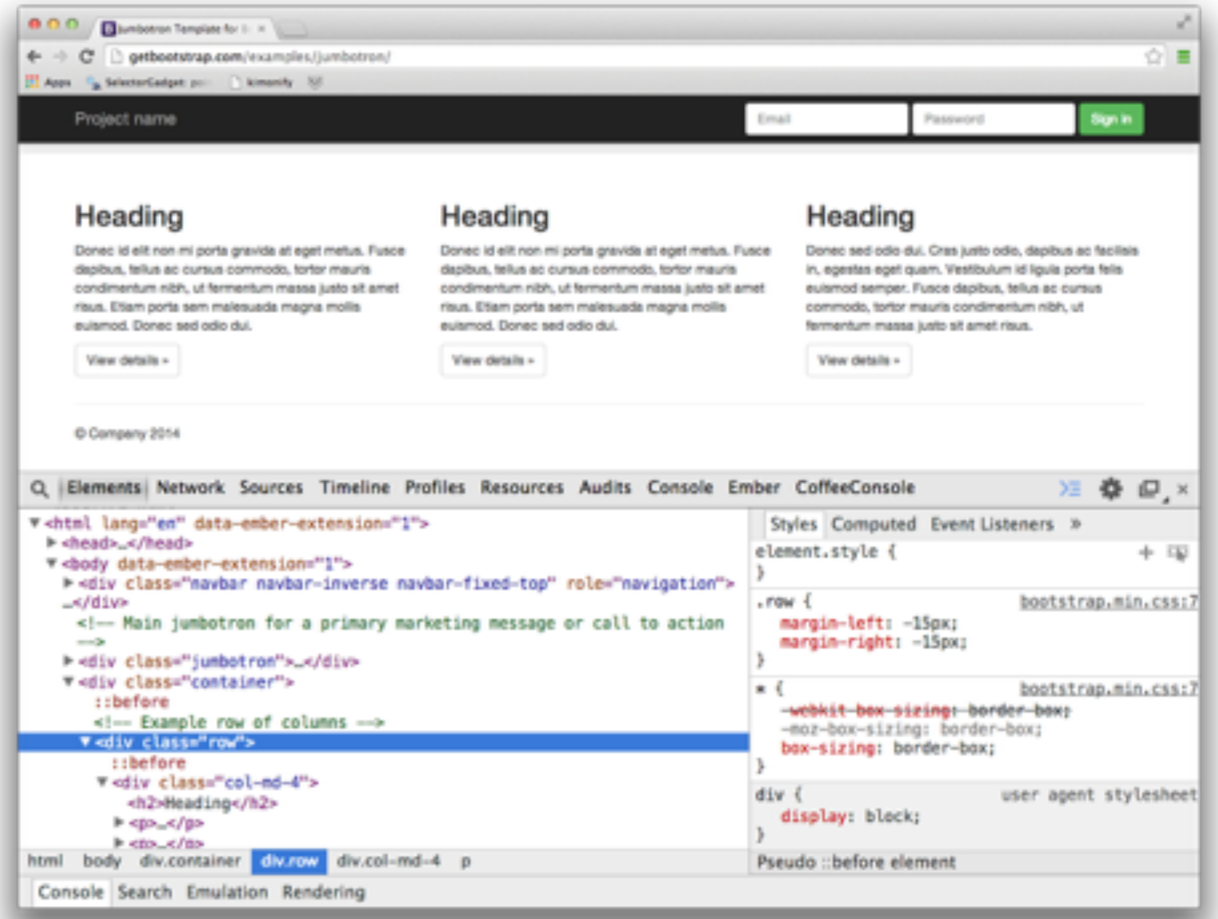
Your code will need to get web page via a GET request.



```
...

<div class="row">
  <div class="col-md-4">
    <h2>Heading</h2>
    <p>Donec id elit ui </p>
    <p><a class="btn" href="#" role="button">View details</a></p>
  </div>
  <div class="col-md-4">
    <h2>Heading</h2>
    <p>Donec id.</p>
    <p><a class="btn" href="#" role="button">View details</a></p>
  </div>
  <div class="col-md-4">
    <h2>Heading</h2>
    <p>Donec sed odio dui.</p>
    <p><a class="btn" href="#" role="button">View details</a></p>
  </div>
</div>

...
```

In looking at HTML source we have some predefined tags like ‘div’ together with class attributes like “row” etc. So the purpose of the class attribute is for styling via CSS and when a number of objects are alike for example the styling of a row or column we use the class attribute.

There is also the ‘id’ attribute which is used for styling specific elements of the page like the for example if a particular item in a list is non-editable, we might make it grey depending on it’s ‘id’.



Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

View details »

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

View details »

Heading

Donec sed odio dui. Cras justo odio, dapibus ac facilisis in, egestas eget quam. Vestibulum id ligula porta felis euismod semper. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.

View details »

© Company 2014

Elements Network Sources Timeline Profiles Resources Audits Console Ember CoffeeConsole
Styles Computed Event Listeners »
html body div.container div.row div.col-md-4 p
Pseudo ::before element
Console Search Emulation Rendering

```
▼<html lang="en" data-ember-extension="1">
  ▶<head>…</head>
  ▼<body data-ember-extension="1">
    ▶<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    …</div>
     <!-- Main jumbotron for a primary marketing message or call to action
     -->
    ▶<div class="jumbotron">…</div>
    ▼<div class="container">
       ::before
       <!-- Example row of columns -->
      ▼<div class="row">
         ::before
        ▼<div class="col-md-4">
           <h2>Heading</h2>
          ▶<p>…</p>
          ▶<p>…</p>
```

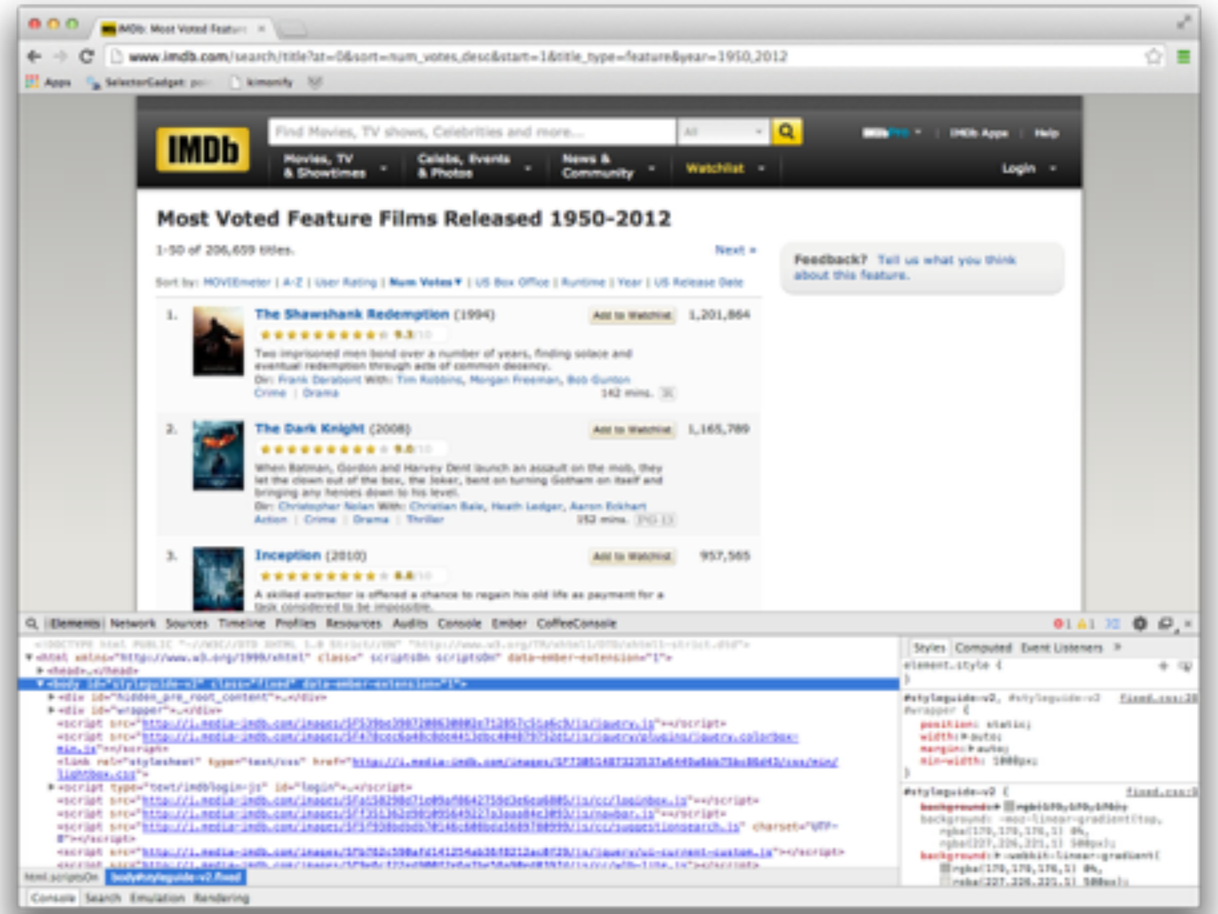
```
element.style {
}
.row {                                  bootstrap.min.css:7
   margin-left: -15px;
   margin-right: -15px;
}
* {                                     bootstrap.min.css:7
   ~~-webkit-box-sizing: border-box;~~
   -moz-box-sizing: border-box;
   box-sizing: border-box;
}
div {                               user agent stylesheet
   display: block;
}
```

And this hierarchical or tree representation of the webpage is referred to as the DOM (Document Object Model). And the chrome inspector makes it easy to explore the DOM.

For scrapping we need to be able to parse the DOM, traverse it and extract appropriate elements from it that we need from this DOM, all from python.

Ethics of Scrapping

- Scrape public websites only
- Non-sensitive information only
- If you do scrape try to anonymise the data
- Always cite your sources (url, date, time etc.)



www.imdb.com/search/title?at=0&sort=num_votes,desc&start=1&title_type=feature&year=1950,2012

IMDb

Movies, TV & Showtimes | Celebs, Events & Photos | News & Community | Watchlist | Login

Most Voted Feature Films Released 1950-2012

1-50 of 206,659 titles. Next »

Sort by: MOVIEmeter | A-Z | User Rating | Num Votes ▾ | US Box Office | Runtime | Year | US Release Date

Feedback? Tell us what you think about this feature.

1. **The Shawshank Redemption** (1994) — 9.3/10 — 1,201,864
   Two imprisoned men bond over a number of years, finding solace and eventual redemption through acts of common decency.
   Dir: Frank Darabont With: Tim Robbins, Morgan Freeman, Bob Gunton
   Crime | Drama — 142 mins.

2. **The Dark Knight** (2008) — 9.0/10 — 1,165,789
   When Batman, Gordon and Harvey Dent launch an assault on the mob, they let the clown out of the box, the Joker, bent on turning Gotham on itself and bringing any heroes down to his level.
   Dir: Christopher Nolan With: Christian Bale, Heath Ledger, Aaron Eckhart
   Action | Crime | Drama | Thriller — 152 mins. PG-13

3. **Inception** (2010) — 8.8/10 — 957,565
   A skilled extractor is offered a chance to regain his old life as payment for a task considered to be impossible.

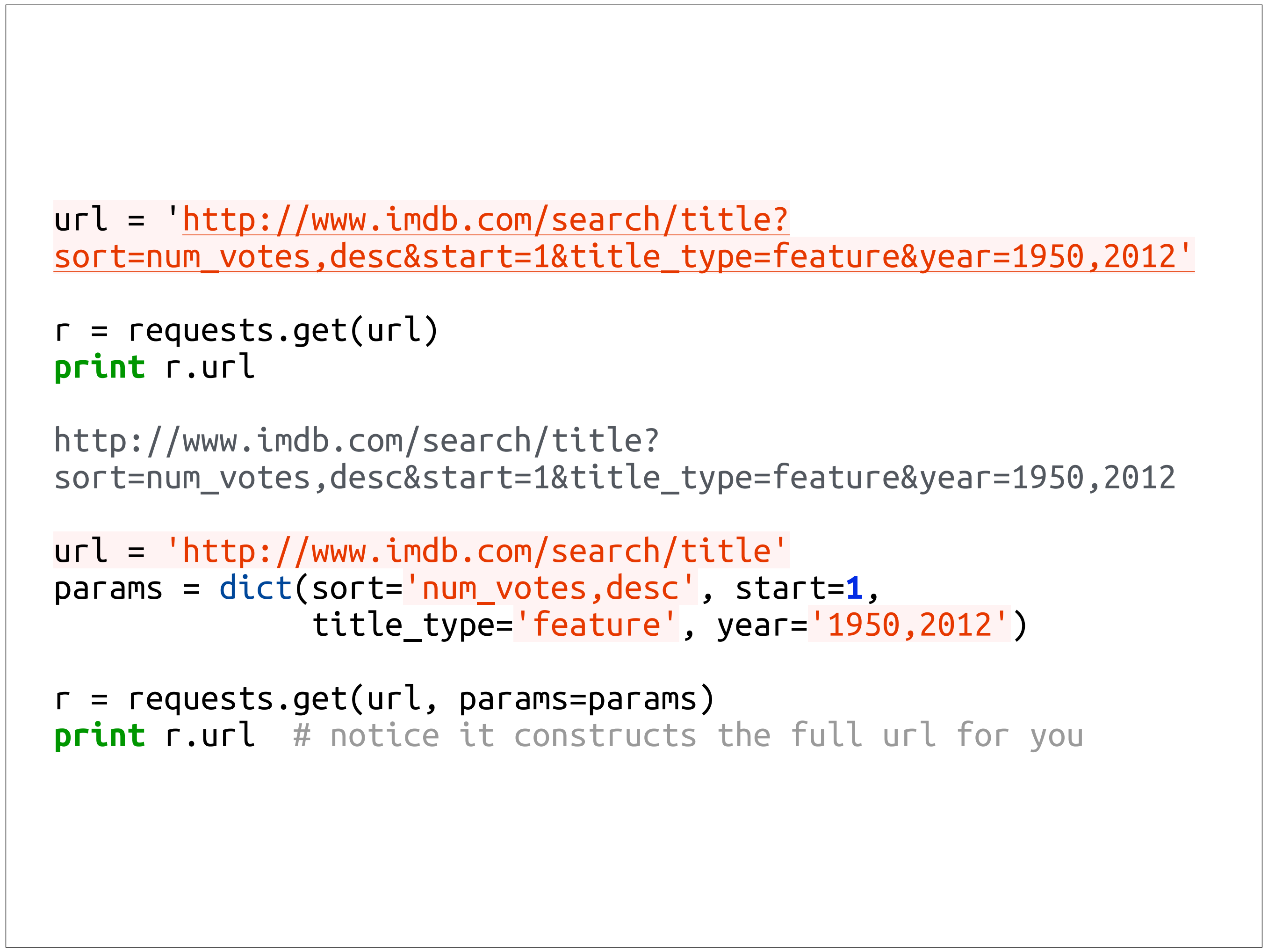
Elements | Network | Sources | Timeline | Profiles | Resources | Audits | Console | Ember | CoffeeConsole

```
[illegible]
```

So let's try to find and print the movie title, list of genres, runtime, and score of all movies on this page: "The most voted feature films released from 1950-2012" on IMDb.

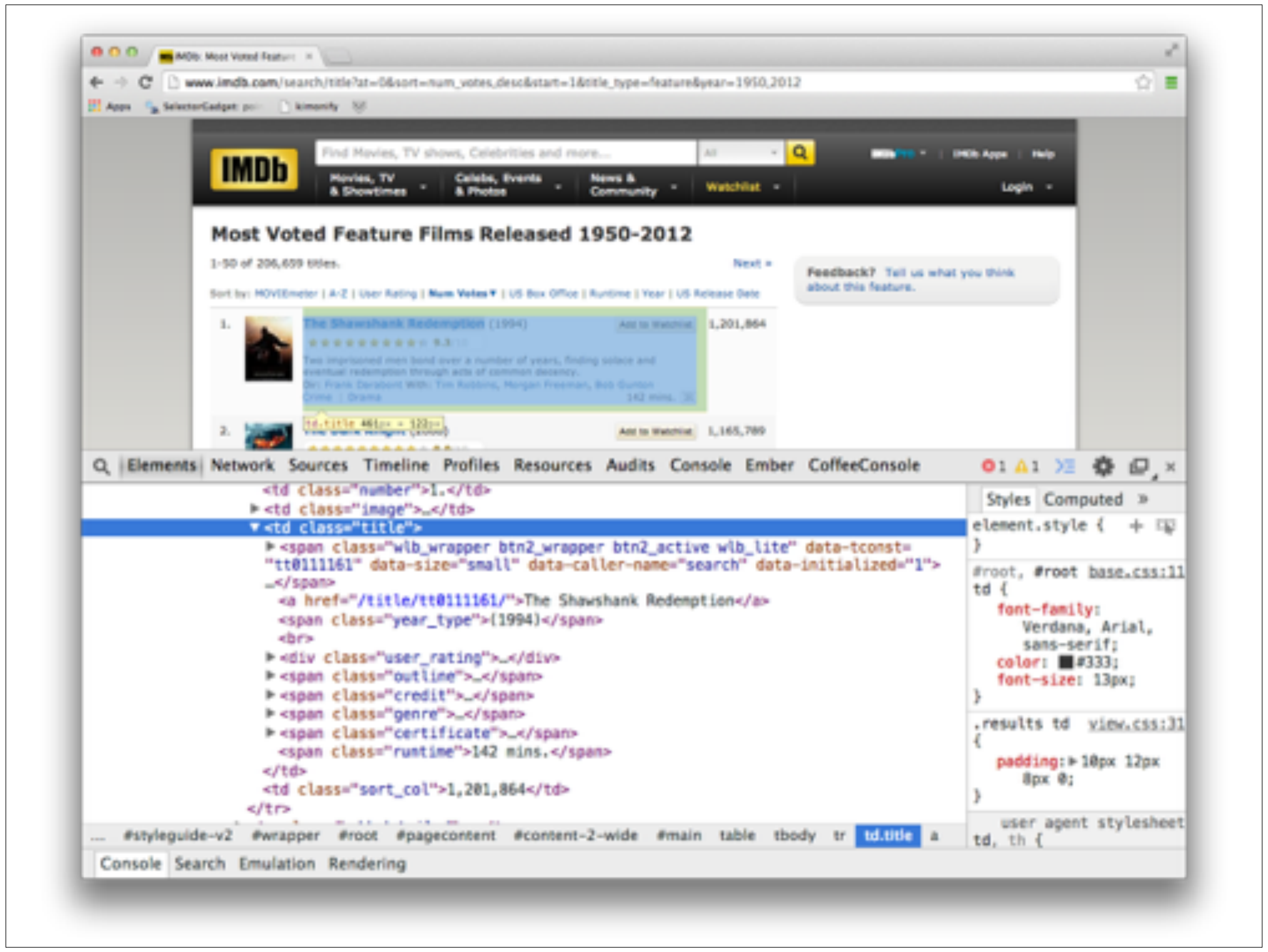
```python
import requests
from pattern import web
from bs4 import BeautifulSoup
```

In python scrapping is usually done with requests, pattern or BeautifulSoup. Let’s import them all.



```
url = 'http://www.imdb.com/search/title?
sort=num_votes,desc&start=1&title_type=feature&year=1950,2012'

r = requests.get(url)
print r.url

http://www.imdb.com/search/title?
sort=num_votes,desc&start=1&title_type=feature&year=1950,2012

url = 'http://www.imdb.com/search/title'
params = dict(sort='num_votes,desc', start=1,
              title_type='feature', year='1950,2012')

r = requests.get(url, params=params)
print r.url  # notice it constructs the full url for you
```

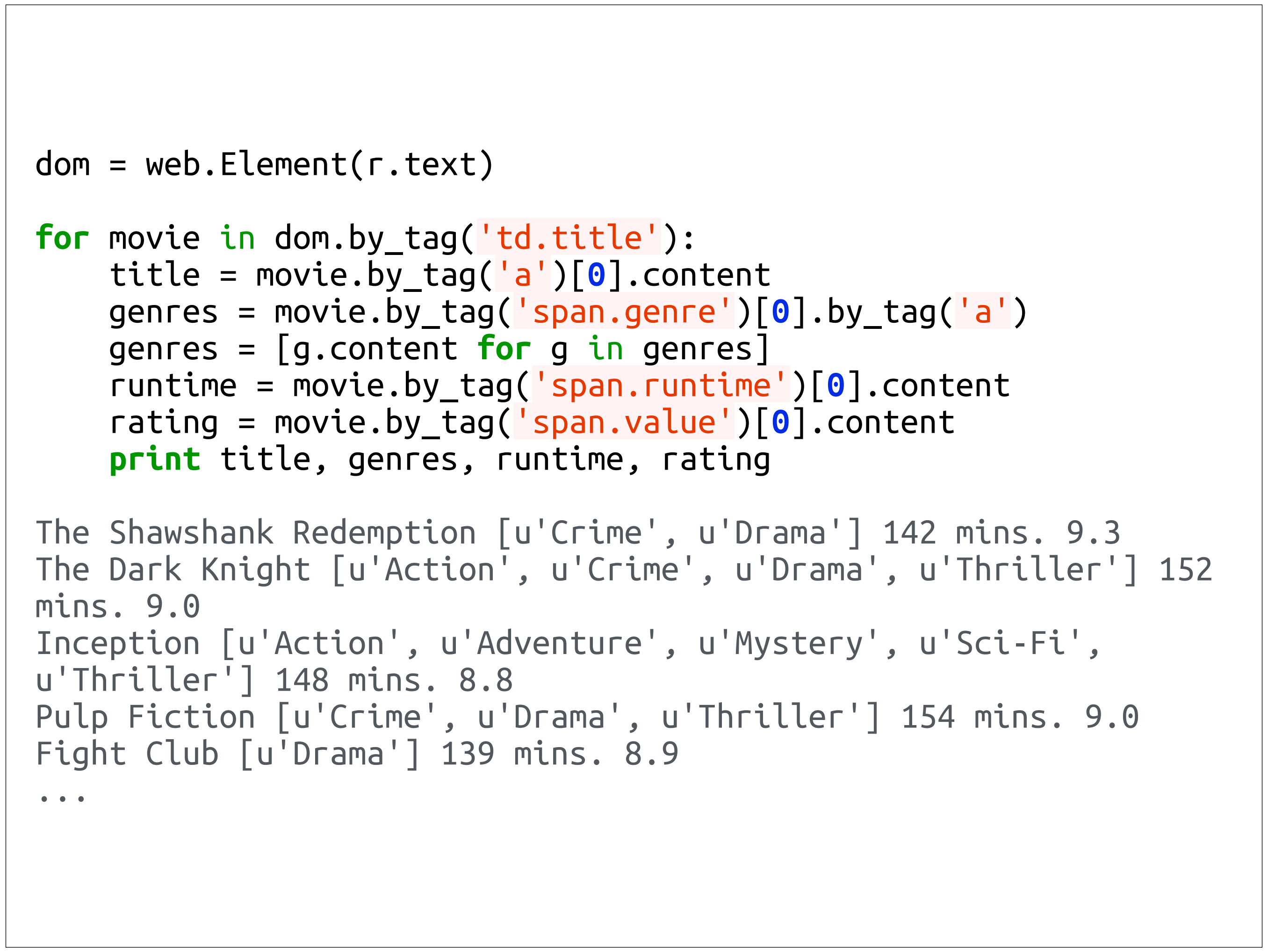
Well we can explicitly to a GET request on this url. Or we can construct the url via a base URL together with a dictionary of params.



```
<td class="number">1.</td>
▶<td class="image">…</td>
▼<td class="title">
  ▶<span class="wlb_wrapper btn2_wrapper btn2_active wlb_lite" data-tconst=
  "tt0111161" data-size="small" data-caller-name="search" data-initialized="1">
  …</span>
   <a href="/title/tt0111161/">The Shawshank Redemption</a>
   <span class="year_type">(1994)</span>
   <br>
  ▶<div class="user_rating">…</div>
  ▶<span class="outline">…</span>
  ▶<span class="credit">…</span>
  ▶<span class="genre">…</span>
  ▶<span class="certificate">…</span>
   <span class="runtime">142 mins.</span>
 </td>
 <td class="sort_col">1,201,864</td>
</tr>
```

Most Voted Feature Films Released 1950-2012
Elements Network Sources Timeline Profiles Resources Audits Console Ember CoffeeConsole
#styleguide-v2 #wrapper #root #pagecontent #content-2-wide #main table tbody tr td.title a
Console Search Emulation Rendering

Each movie is in the DOM element table row 'tr'->'td.title'. Inside each 'td.title' we have the title in the link tag, the genres are in the span class 'genre' etc. So we need to loop over all the 'td.title' tags and get the information we need.



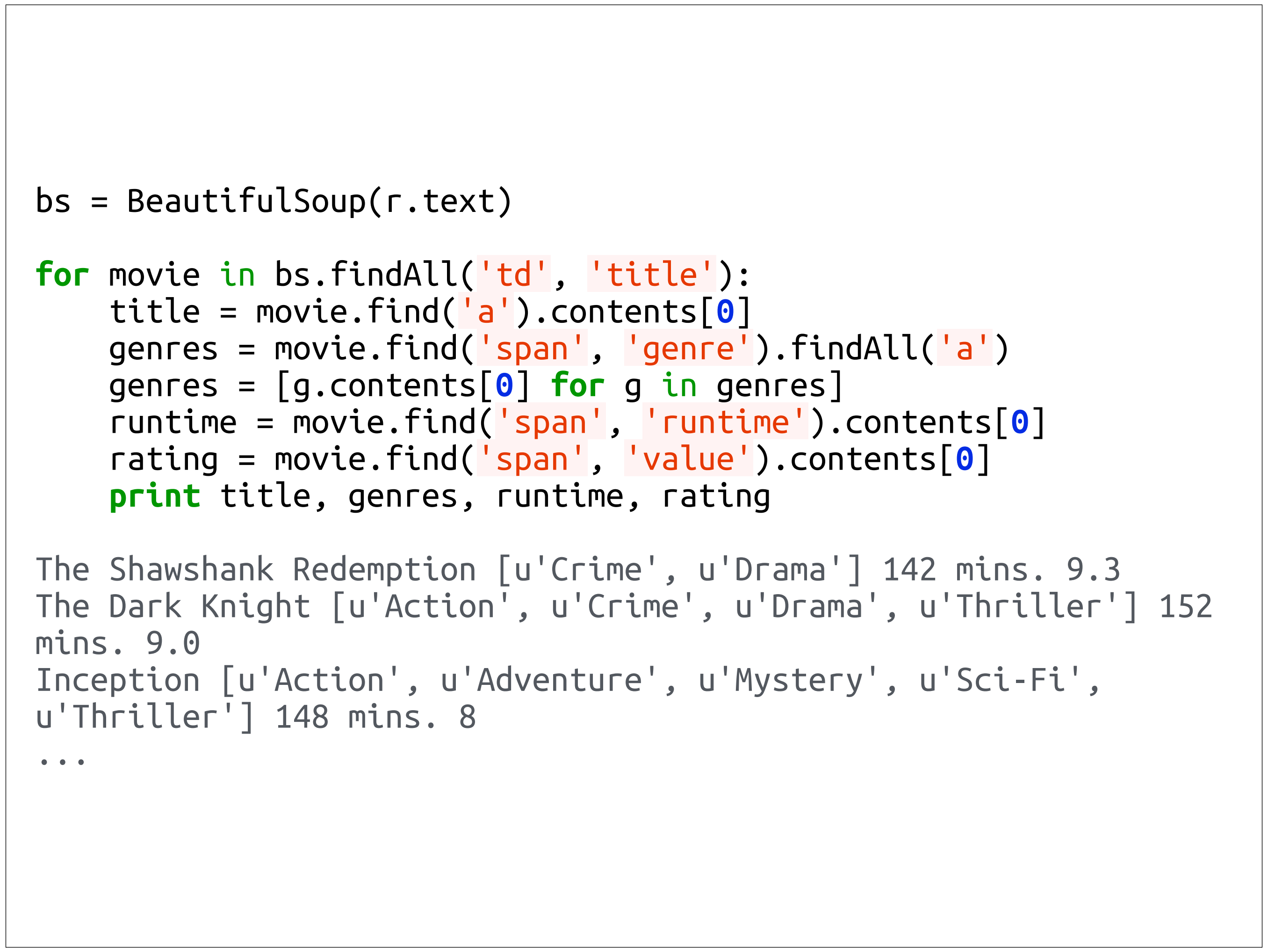
```python
dom = web.Element(r.text)

for movie in dom.by_tag('td.title'):
    title = movie.by_tag('a')[0].content
    genres = movie.by_tag('span.genre')[0].by_tag('a')
    genres = [g.content for g in genres]
    runtime = movie.by_tag('span.runtime')[0].content
    rating = movie.by_tag('span.value')[0].content
    print title, genres, runtime, rating
```

```
The Shawshank Redemption [u'Crime', u'Drama'] 142 mins. 9.3
The Dark Knight [u'Action', u'Crime', u'Drama', u'Thriller'] 152
mins. 9.0
Inception [u'Action', u'Adventure', u'Mystery', u'Sci-Fi',
u'Thriller'] 148 mins. 8.8
Pulp Fiction [u'Crime', u'Drama', u'Thriller'] 154 mins. 9.0
Fight Club [u'Drama'] 139 mins. 8.9
...
```

So with patterns we can get the particular DOM by it’s css tag. Some tags like the link has the information inside it’s content and so we use ‘.content’ to get that information. Most of the work of scrapping is iterating this process for the tag you want.

Note the list of genres are created via the python’s syntactic sugar which is equivalent to doing:

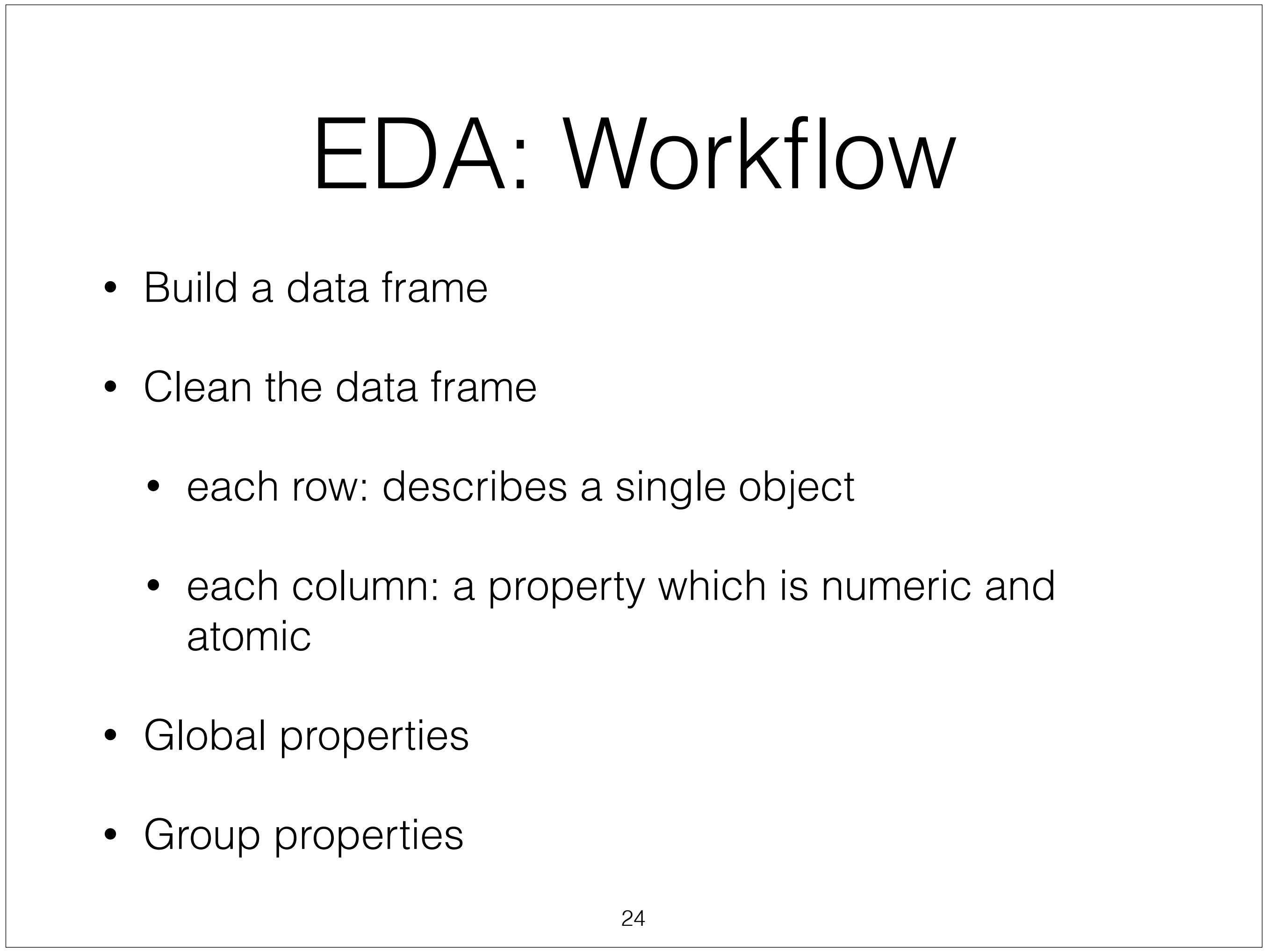
```
genres = []
for g in generes
        genres.append(g.content)
```



```
bs = BeautifulSoup(r.text)

for movie in bs.findAll('td', 'title'):
    title = movie.find('a').contents[0]
    genres = movie.find('span', 'genre').findAll('a')
    genres = [g.contents[0] for g in genres]
    runtime = movie.find('span', 'runtime').contents[0]
    rating = movie.find('span', 'value').contents[0]
    print title, genres, runtime, rating
```

```
The Shawshank Redemption [u'Crime', u'Drama'] 142 mins. 9.3
The Dark Knight [u'Action', u'Crime', u'Drama', u'Thriller'] 152
mins. 9.0
Inception [u'Action', u'Adventure', u'Mystery', u'Sci-Fi',
u'Thriller'] 148 mins. 8
...
```

In BeautifulSoup we use the findAll() and find() functions to get the DOM elements we want.



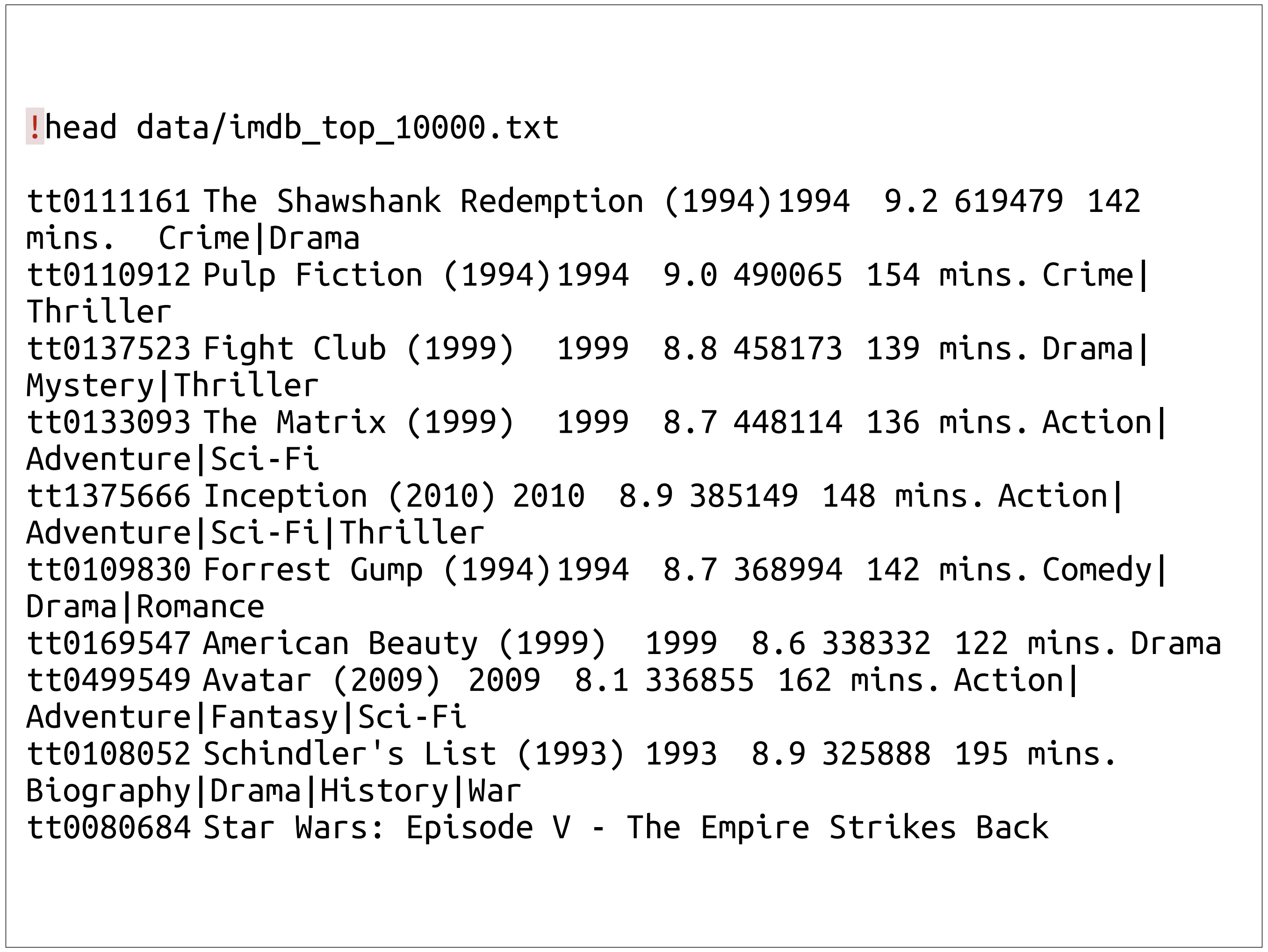
EDA: Workflow

- Build a data frame
- Clean the data frame
  - each row: describes a single object
  - each column: a property which is numeric and atomic
- Global properties
- Group properties

The basic workflow is to build a data frame, clean up the data frame, explore the global properties and the group properties of the data.

This process transforms your data into a format which is easier to work with, gives you a basic overview of the data's properties, and likely generates several questions for you to follow up in subsequent analysis.

Cool that was easy. Well this is the feeling one gets when doing Data wrangling and exploration.



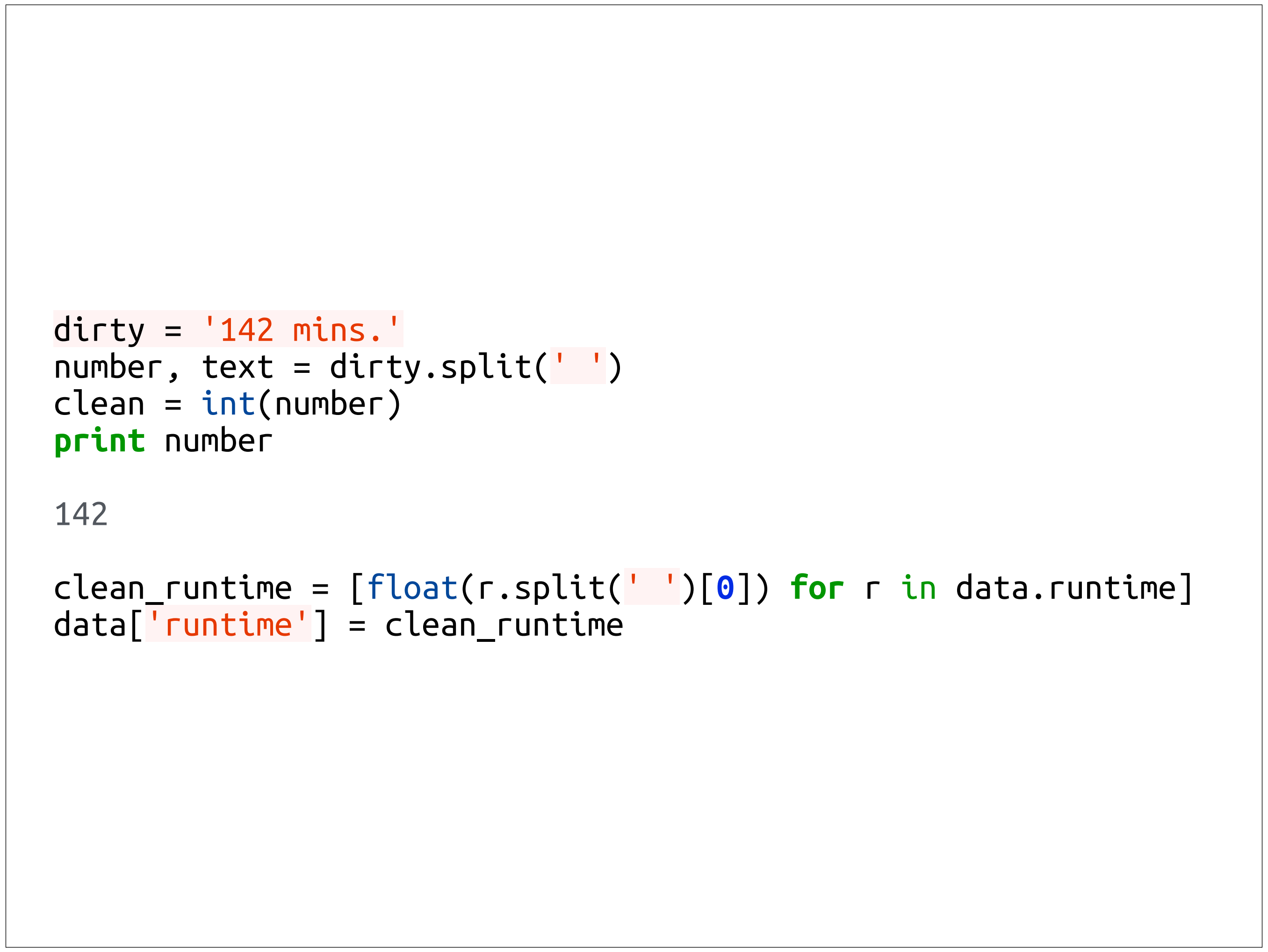
```
!head data/imdb_top_10000.txt

tt0111161 The Shawshank Redemption (1994) 1994  9.2 619479  142
mins.  Crime|Drama
tt0110912 Pulp Fiction (1994) 1994  9.0 490065  154 mins. Crime|
Thriller
tt0137523 Fight Club (1999)   1999  8.8 458173  139 mins. Drama|
Mystery|Thriller
tt0133093 The Matrix (1999)   1999  8.7 448114  136 mins. Action|
Adventure|Sci-Fi
tt1375666 Inception (2010) 2010  8.9 385149  148 mins. Action|
Adventure|Sci-Fi|Thriller
tt0109830 Forrest Gump (1994) 1994  8.7 368994  142 mins. Comedy|
Drama|Romance
tt0169547 American Beauty (1999)  1999  8.6 338332  122 mins. Drama
tt0499549 Avatar (2009)  2009  8.1 336855  162 mins. Action|
Adventure|Fantasy|Sci-Fi
tt0108052 Schindler's List (1993) 1993  8.9 325888  195 mins.
Biography|Drama|History|War
tt0080684 Star Wars: Episode V - The Empire Strikes Back
```

Here's a preview of the raw data we'll use -- it's a list of the 10,000 movies made since 1950 with the most IMDB user ratings.

```
%matplotlib inline

import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

names = ['imdbID', 'title', 'year', 'score', 'votes', ‘runtime',
         'genres']
data = pd.read_csv(‘data/imdb_top_10000.txt', delimiter='\t',
                   names=names).dropna()

print "Number of rows: %i" % data.shape[0]
data.head()  # print the first 5 rows

Number of rows: 9999
      imdbID                            title  year  score
votes    runtime   genres
0  tt0111161  The Shawshank Redemption (1994)  1994    9.2
619479  142 mins.   Crime|Drama
1  tt0110912              Pulp Fiction (1994)  1994    9.0
490065  154 mins.   Crime|Thriller
...
```

The text file is tab-separated, and doesn't have any column headers. We set the appropriate keywords in pd.read_csv() to handle this and drop any empty values.

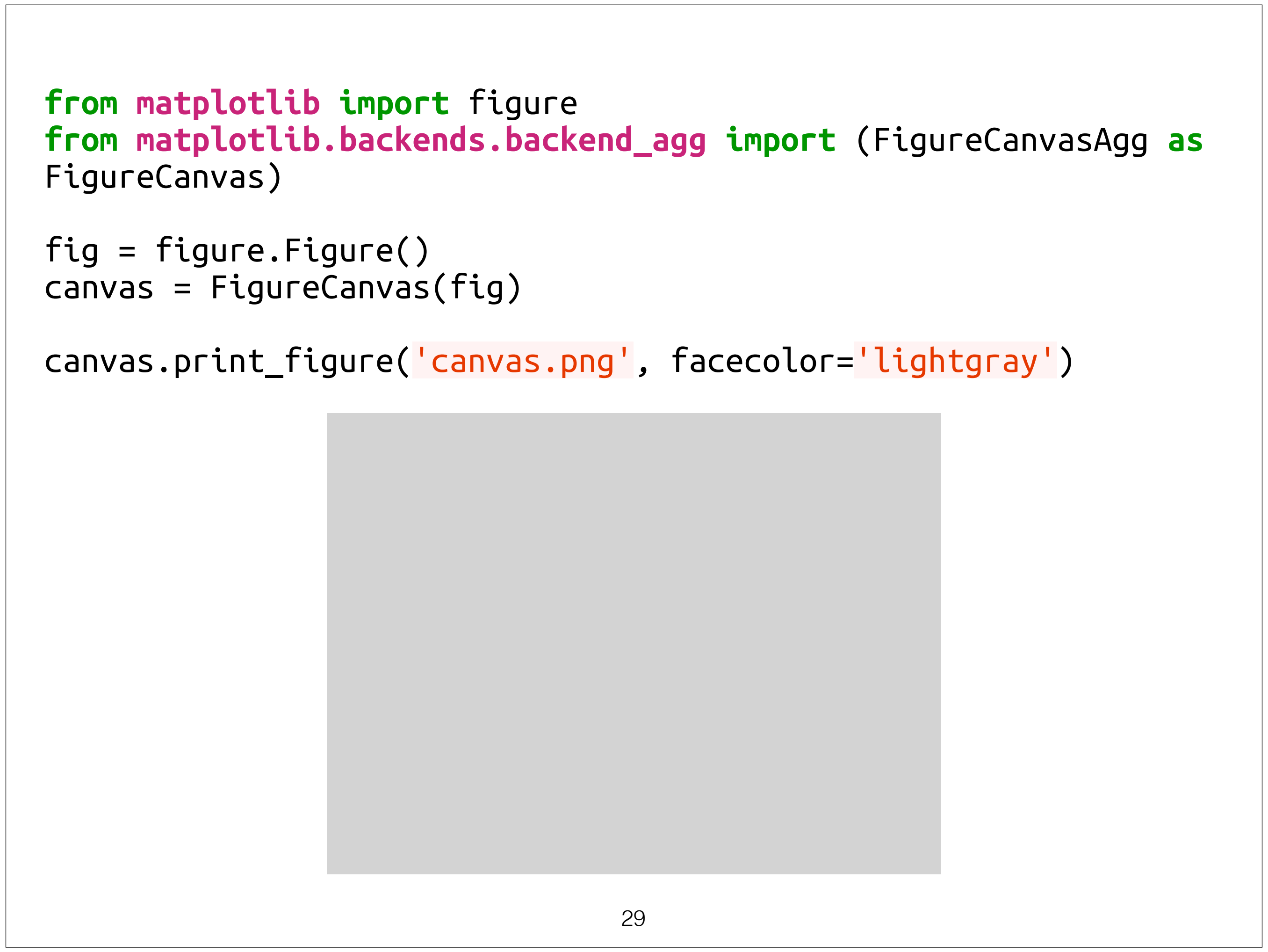
Notice the runtime is currently a string rather than an integer number. The genres column is not atomic. Finally the movie year is repeated in the year and title column.



```
dirty = '142 mins.'
number, text = dirty.split(' ')
clean = int(number)
print number

142

clean_runtime = [float(r.split(' ')[0]) for r in data.runtime]
data['runtime'] = clean_runtime
```

Let’s fix the runtime column. The following snippet converts a string like '142 mins.' to the number 142. So we use it to replace the runtime column with the cleaned runtime.

Actually, lets do the rest in iPython. But before we continue let me talk a bit about matplotlib.



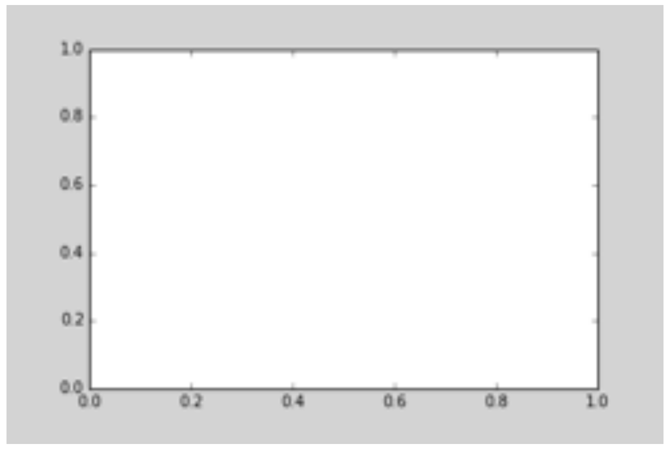
```
from matplotlib import figure
from matplotlib.backends.backend_agg import (FigureCanvasAgg as
FigureCanvas)

fig = figure.Figure()
canvas = FigureCanvas(fig)

canvas.print_figure('canvas.png', facecolor='lightgray')
```

matplotlib can be used in a procedural manner, as suggested by it’s name which was trying to copy matlab. But to get anything decent out of matplotlib you need to go object oriented.

So typically the way to use matplotlib is to define states in steps were we create a figure and add a canvas to it and then print it out. It’s like a sandwich which makes it hard to pull out pieces in the middle to inspect or reuse them.

The object oriented way is to create objects and attach properties to different aspects of this object. So as shown above we will end up with just a blank light grey canvas.

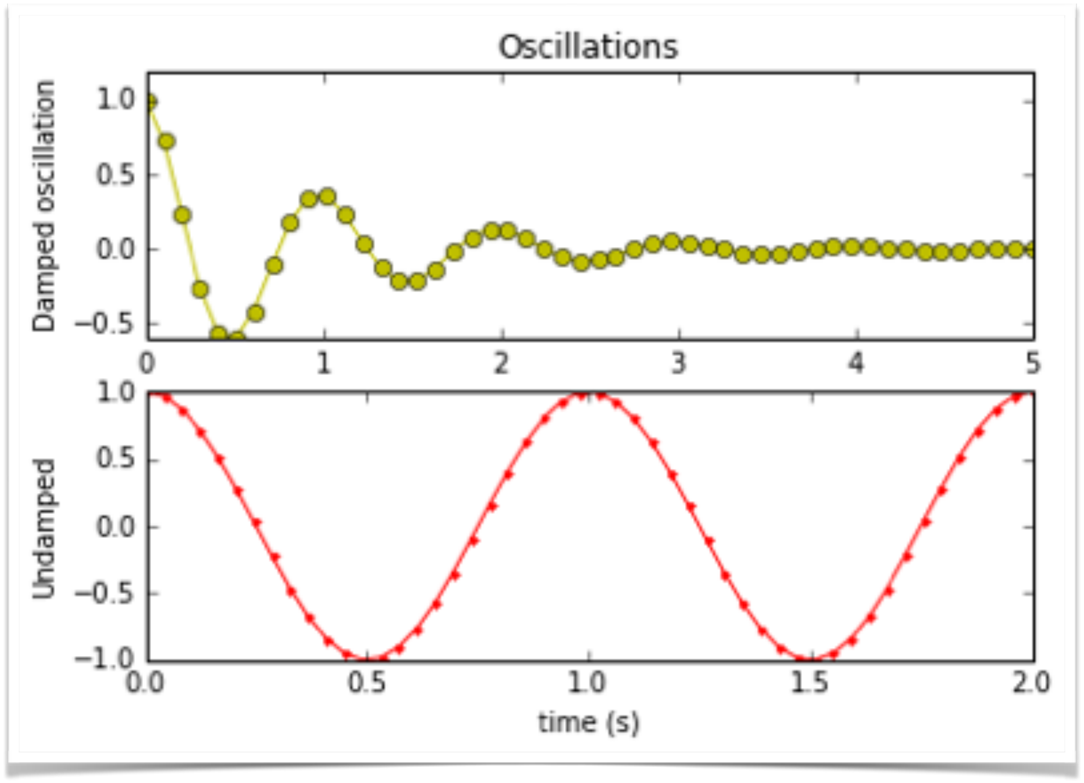
```
fig = figure.Figure()
canvas = FigureCanvas(fig)
ax = fig.add_subplot(1,1,1)

canvas.print_figure('plot.png', facecolor='lightgray')
```



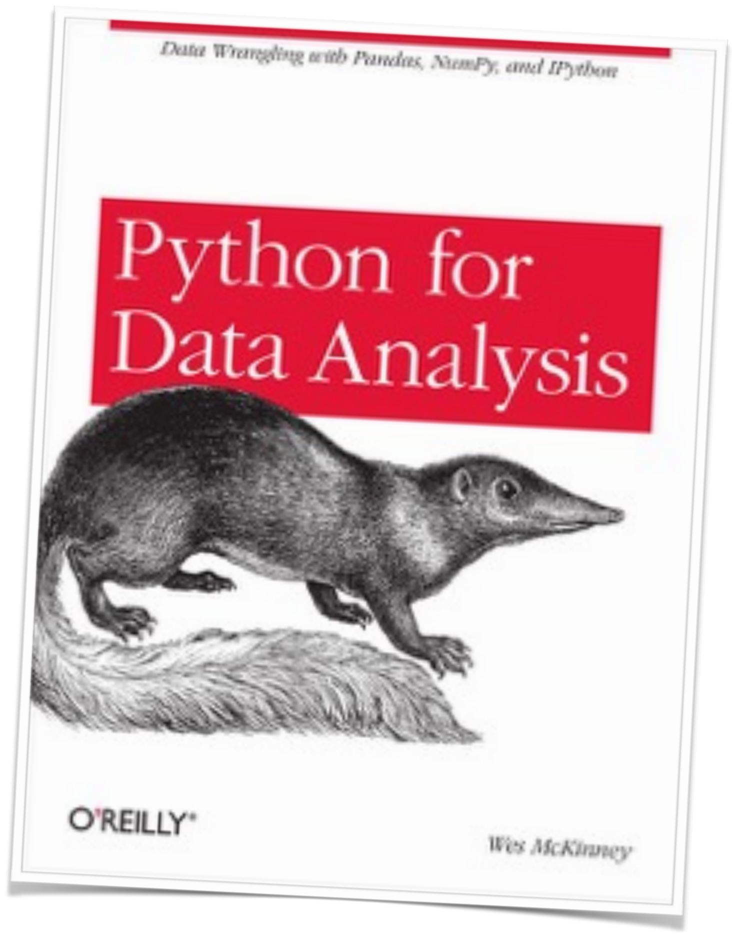
Now to add a plot we use the add_subplot() function to our figure. We always use the add_subplot() to add plots to our figure since it gives us control over where we put things on our canvas. add_subplot() takes the total number of rows, columns and the current plot's index.

```python
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)

y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)

plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'yo-')
plt.title('Oscillations')
plt.ylabel('Damped oscillation')

plt.subplot(2, 1, 2)
plt.plot(x2, y2, 'r.-')
plt.xlabel('time (s)')
plt.ylabel('Undamped')

plt.show()
```

pyplot provides a matlab like plotting environment. Here we use subplot to show two plots and configure the properties of the object like the plot label etc.



And this is what it looks like.



For further information check chapters 5 and 7 in this book.