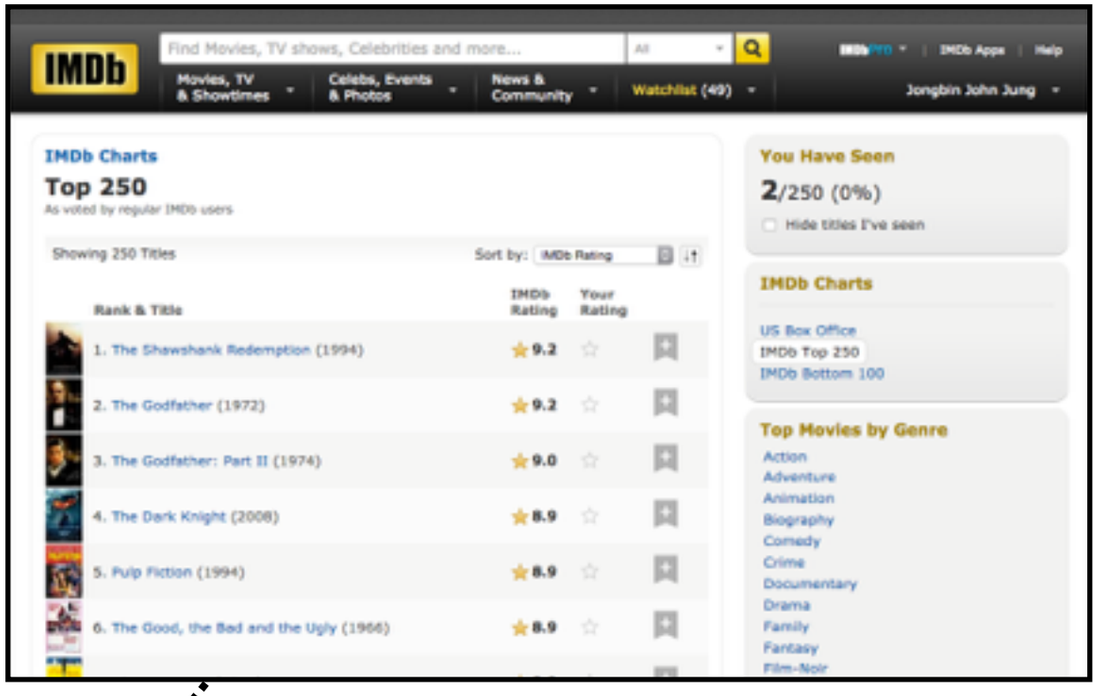
# Web Scraping

a **Data Science Drop-in** Tutorial

by **Jongbin Jung**
(jongbin@stanford.edu)

# Web Scraping?

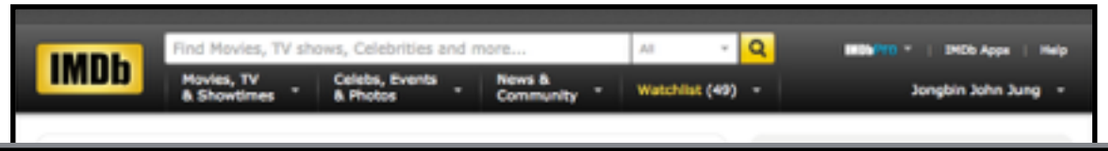



| | movie_title | actor_name | character |
|---|---|---|---|
| 1 | The Shawshank Redemption | Tim Robbins | Andy Dufresne |
| 2 | The Shawshank Redemption | Morgan Freeman | Ellis Boyd 'Red' Redding |
| 3 | The Shawshank Redemption | Bob Gunton | Warden Norton |
| 4 | The Shawshank Redemption | William Sadler | Heywood |
| 5 | The Shawshank Redemption | Clancy Brown | Captain Hadley |
| 6 | The Shawshank Redemption | Gil Bellows | Tommy |
| 7 | The Shawshank Redemption | Mark Rolston | Bogs Diamond |
| 8 | The Shawshank Redemption | James Whitmore | Brooks Hatlen |
| 9 | The Shawshank Redemption | Jeffrey DeMunn | 1946 D.A. |
| 10 | The Shawshank Redemption | Larry Brandenburg | Skeet |
| 11 | The Shawshank Redemption | Neil Giuntoli | Jigger |
| 12 | The Shawshank Redemption | Brian Libby | Floyd |
| 13 | The Shawshank Redemption | David Proval | Snooze |
| 14 | The Shawshank Redemption | Joseph Ragno | Ernie |
| 15 | The Shawshank Redemption | Jude Ciccolella | Guard Mert |
| 16 | The Godfather | Marlon Brando | Don Vito Corleone |
| 17 | The Godfather | Al Pacino | Michael Corleone |
| 18 | The Godfather | James Caan | Sonny Corleone |
| 19 | The Godfather | Richard S. Castellano | Clemenza (as Richard Castellano) |
| 20 | The Godfather | Robert Duvall | Tom Hagen |
| 21 | The Godfather | Sterling Hayden | Capt. McCluskey |
| 22 | The Godfather | John Marley | Jack Woltz |
| 23 | The Godfather | Richard Conte | Barzini |

Get from here…

… to **here**

# Web Scraping?




| | movie_title | actor_name | character |
|---|---|---|---|
| 1 | The Shawshank Redemption | Tim Robbins | Andy Dufresne |
| 2 | The Shawshank Redemption | Morgan Freeman | Ellis Boyd 'Red' Redding |
| 3 | The Shawshank Redemption | Bob Gunton | Warden Norton |
| 4 | The Shawshank Redemption | William Sadler | Heywood |

# Today's **GOAL**

Collect the **cast overview** (actor and character played) for each of the **Top 10 movies** of **IMDb Charts' Top 250** (http://www.imdb.com/chart/top?ref_=nv_ch_250_4)

# First Things First

- Get on the Stanford `corn` servers
  (see, `https://farmshare.stanford.edu/`)

  - Mac/Linux: Open a terminal and type:

```
> ssh [SUNet ID]@corn.stanford.edu
```
[Terminal]

    replacing `[SUNet ID]` with your ID

  - Windows: Use PowerShell / PuTTY / Cygwin …

# First Things First

- Let's get some `python` packages!

```
> pip install --user requests beautifulsoup4 selenium
```
[Terminal]

- Clone (i.e., download) example scripts

```
> git clone https://github.com/5harad/datascience.git
> cd datascience/webscraping
```
[Terminal]

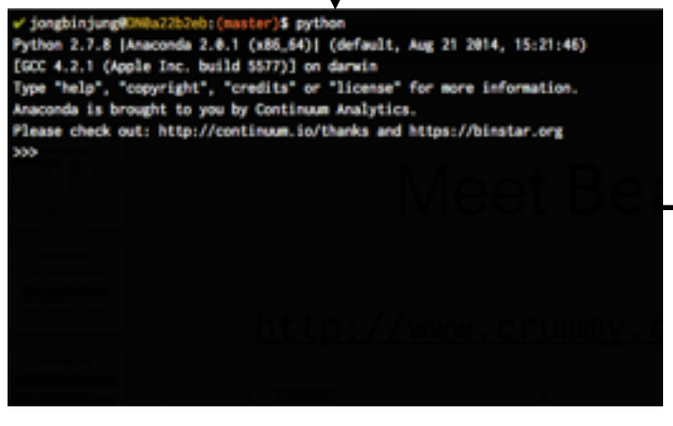
- Start `python`

```
> python
```
[Terminal]

# Meet BeautifulSoup

http://www.crummy.com/software/BeautifulSoup/

from source (`html`)
to `python`

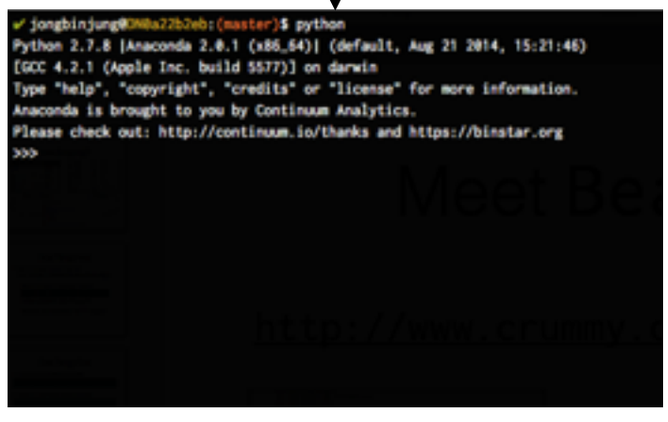



`BeautifulSoup`
goes **here**

| | movie_title | actor_name | character |
|---|---|---|---|
| 1 | The Shawshank Redemption | Tim Robbins | Andy Dufresne |
| 2 | The Shawshank Redemption | Morgan Freeman | Ellis Boyd 'Red' Redding |
| 3 | The Shawshank Redemption | Bob Gunton | Warden Norton |
| 4 | The Shawshank Redemption | William Sadler | Heywood |
| 5 | The Shawshank Redemption | Clancy Brown | Captain Hadley |
| 6 | The Shawshank Redemption | Gil Bellows | Tommy |
| 7 | The Shawshank Redemption | Mark Rolston | Bogs Diamond |
| 8 | The Shawshank Redemption | James Whitmore | Brooks Hatlen |
| 9 | The Shawshank Redemption | Jeffrey DeMunn | 1946 D.A. |
| 10 | The Shawshank Redemption | Larry Brandenburg | Skeet |
| 11 | The Shawshank Redemption | Neil Giuntoli | Jigger |
| 12 | The Shawshank Redemption | Brian Libby | Floyd |
| 13 | The Shawshank Redemption | David Proval | Snooze |
| 14 | The Shawshank Redemption | Joseph Ragno | Ernie |
| 15 | The Shawshank Redemption | Jude Ciccolella | Guard Mert |
| 16 | The Godfather | Marlon Brando | Don Vito Corleone |
| 17 | The Godfather | Al Pacino | Michael Corleone |
| 18 | The Godfather | James Caan | Sonny Corleone |
| 19 | The Godfather | Richard S. Castellano | Clemenza (as Richard Castellano) |
| 20 | The Godfather | Robert Duvall | Tom Hagen |
| 21 | The Godfather | Sterling Hayden | Capt. McCluskey |
| 22 | The Godfather | John Marley | Jack Woltz |
| 23 | The Godfather | Richard Conte | Barzini |

This can be done many ways.
But we use `requests` (for now)

from source (`html`)
to `python`




```python
import requests


web_page =

requests.get('address')

web_page.content
```
[python]

Let's try this with the
`BeautifulSoup` web page

http://www.crummy.com/software/BeautifulSoup/

# Meet `BeautifulSoup`

Make the source (`html`) into a `BeautifulSoup`

```python
from bs4 import BeautifulSoup

soup = BeautifulSoup(web_page.content)
```

[python]


| | movie_title | actor_name | character |
|---|---|---|---|
| 1 | The Shawshank Redemption | Tim Robbins | Andy Dufresne |
| 2 | The Shawshank Redemption | Morgan Freeman | Ellis Boyd 'Red' Redding |
| 3 | The Shawshank Redemption | Bob Gunton | Warden Norton |
| 4 | The Shawshank Redemption | William Sadler | Heywood |
| 5 | The Shawshank Redemption | Clancy Brown | Captain Hadley |
| 6 | The Shawshank Redemption | Gil Bellows | Tommy |
| 7 | The Shawshank Redemption | Mark Rolston | Bogs Diamond |
| 8 | The Shawshank Redemption | James Whitmore | Brooks Hatlen |
| 9 | The Shawshank Redemption | Jeffrey DeMunn | 1946 D.A. |
| 10 | The Shawshank Redemption | Larry Brandenburg | Skeet |
| 11 | The Shawshank Redemption | Neil Giuntoli | Jigger |
| 12 | The Shawshank Redemption | Brian Libby | Floyd |
| 13 | The Shawshank Redemption | David Proval | Snooze |
| 14 | The Shawshank Redemption | Joseph Ragno | Ernie |
| 15 | The Shawshank Redemption | Jude Ciccolella | Guard Mert |
| 16 | The Godfather | Marlon Brando | Don Vito Corleone |
| 17 | The Godfather | Al Pacino | Michael Corleone |
| 18 | The Godfather | James Caan | Sonny Corleone |
| 19 | The Godfather | Richard S. Castellano | Clemenza (as Richard Castellano) |
| 20 | The Godfather | Robert Duvall | Tom Hagen |
| 21 | The Godfather | Sterling Hayden | Capt. McCluskey |
| 22 | The Godfather | John Marley | Jack Woltz |
| 23 | The Godfather | Richard Conte | Barzini |

# Use `BeautifulSoup`

```python
soup.h1
```
[python]

```output
<h1>Beautiful Soup</h1>
```
[output]


Did we just get this?

# What's Happening?

```
21 <div align="center">
22
23 <a href="bs4/download/"><h1>Beautiful Soup</h1></a>
24
25 <p>"A tremendous boon." -- <a
26 href="http://www.awaretek.com/python/index.html">Pyt
27 Podcast</a></p>
28
29 <p>[ <a href="#Download">Download</a> | <a
30 href="bs4/doc/">Documentation</a> | <a href="#HallOf
   href="https://code.launchpad.net/beautifulsoup">Sour
   href="https://groups.google.com/forum/?fromgroups#!f
   group</a> ]</p>
31
32 <small>If Beautiful Soup has saved you a lot of time
33 best way to pay me back is to check out <a
34 href="http://www.candlemarkandgleam.com/shop/constel
   <i>Constellation
35 Games</i>, my sci-fi novel about alien video games</
36 <a
```

- `soup.h1` gives us the content surrounded by `<h1>` and `</h1>`

# What's Happening?

```
21 <div align="center">
22
23 <a href="bs4/download/"><h1>Beautiful Soup</h1></a>
24
25 <p>"A tremendous boon." -- <a
26 href="http://www.awaretek.com/python/index.html">Pyt
27 Podcast</a></p>
28
29 <p>[ <a href="#Download">Download</a> | <a
30 href="bs4/doc/">Documentation</a> | <a href="#HallOf
   href="https://code.launchpad.net/beautifulsoup">Sour
   href="https://groups.google.com/forum/?fromgroups#!f
   group</a> ]</p>
31
32 <small>If Beautiful Soup has saved you a lot of time
33 best way to pay me back is to check out <a
34 href="http://www.candlemarkandgleam.com/shop/constel
   <i>Constellation
35 Games</i>, my sci-fi novel about alien video games</
36 <a
```

- The `<a>` and `</a>` make the text between them into a *link*
- The `href="bs4/download/"` indicates where the link should link to

# Get a link and its address

```python
soup.a
```
[python]

```output
<a href="bs4/download/"><h1>Beautiful Soup</h1></a>
```
[output]

Now let's `get` the address (the value assigned to `href`)

```python
soup.a.get('href')
```
[python]

```output
'bs4/download/'
```
[output]

You didn't write that awful page. You're just trying to get some data out of it. Beautiful Soup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

# Beautiful Soup

"A tremendous boon." -- Python411 Podcast

[ Download | Documentation | Hall of Fame | Source | Discussion group ]

If Beautiful Soup has saved you a lot of time and money, the best way to pay me back is to check out *Constellation Games*, my sci-fi novel about alien video games.
You can read the first two chapters for free, and the full novel starts at 5 USD. Thanks!

*If you have questions, send them to the discussion group. If you find a bug, file it.*

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. Beautiful Soup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and Beautiful Soup can't detect one. Then you just have to specify the original encoding.
3. Beautiful Soup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

Beautiful Soup parses anything you give it, and does the tree traversal stuff for you. You can tell it "Find all the links", or "Find all the links of class `externalLink`", or "Find all the links whose urls match "foo.com", or "Find the table heading that's got bold text, then give me that text."

Valuable data that was once locked up in poorly-designed websites is now within your reach. Projects that would have taken hours take only minutes with Beautiful Soup.

Interested? Read more.

What about all these other links?

# Get more links and addresses

Lets `find_all` the links (`a`)

```python
soup.find_all('a')
```

[python]

```
[<a href="bs4/download/"><h1>Beautiful Soup</h1></a>, <a href="http://
www.awaretek.com/python/index.html">Python411 Podcast</a>, ...
...
... , <a href="http://www.crummy.com/">http://www.crummy.com/</a>, <a
href="http://www.crummy.com/software/">software/</a>, <a href="http://
www.crummy.com/software/BeautifulSoup/">BeautifulSoup/</a>]
```

[output]

- Notice `[ ..., ..., ...]` is a list in `python`
- We can `iterate` through a `list` with a `for` loop

# Get more links and addresses

Lets `get` all the addresses with a `for` loop

```python
for link in soup.find_all('a'):
    link.get('href')
```

[python]

```output
'bs4/download/'
'http://www.awaretek.com/python/index.html'
...
'http://www.crummy.com/software/'
'http://www.crummy.com/software/BeautifulSoup/'
```

[output]

Goto IMDb.com

# Today's **GOAL**

Collect the **cast overview** (actor and character played) for each of the **Top 10 movies** of **IMDb Charts' Top 250** (http://www.imdb.com/chart/top?ref_=nv_ch_250_4)

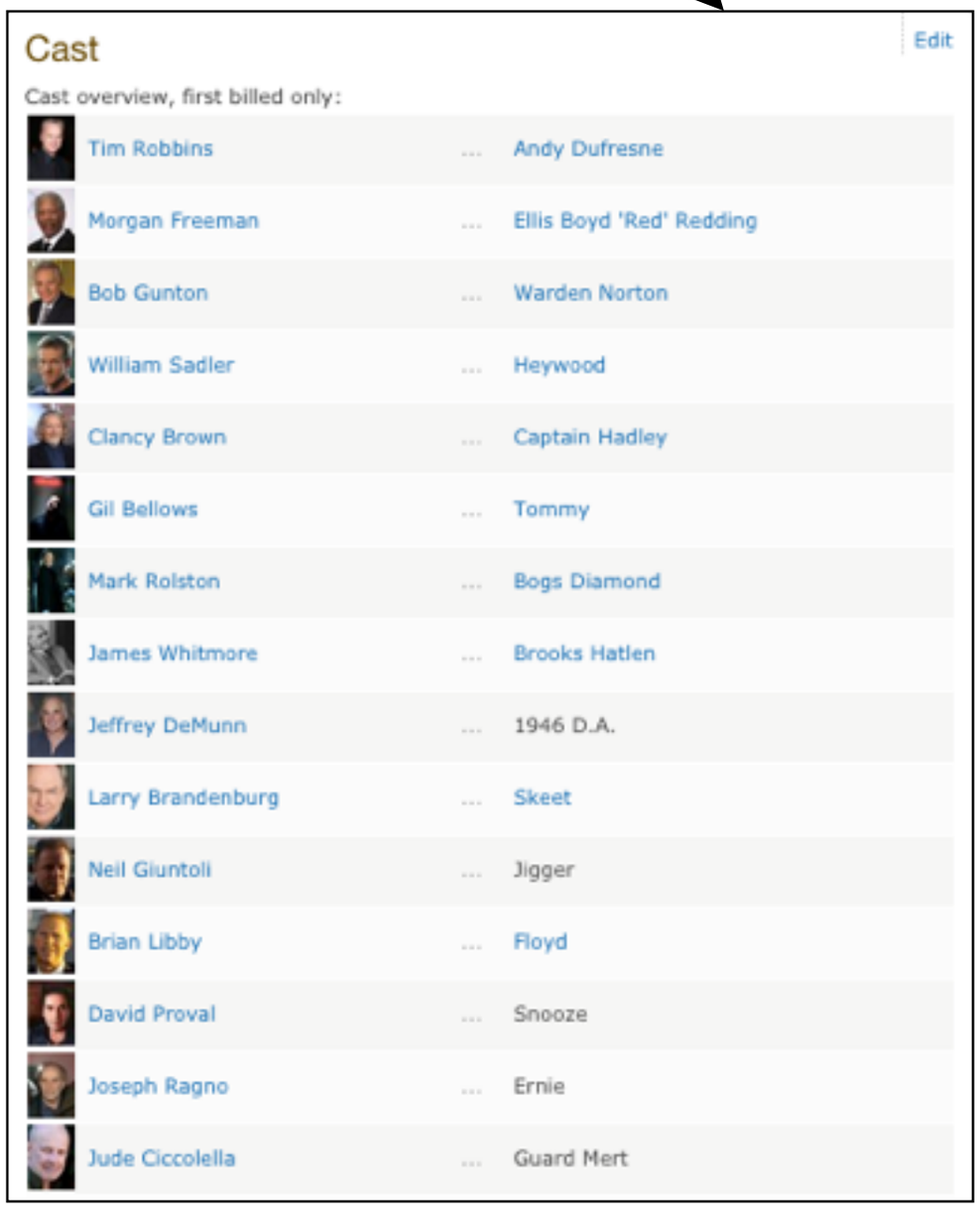
Let's start with

"the **cast overview** (actor and character played)"

for just *one* movie.

We want this

...start by making a soup



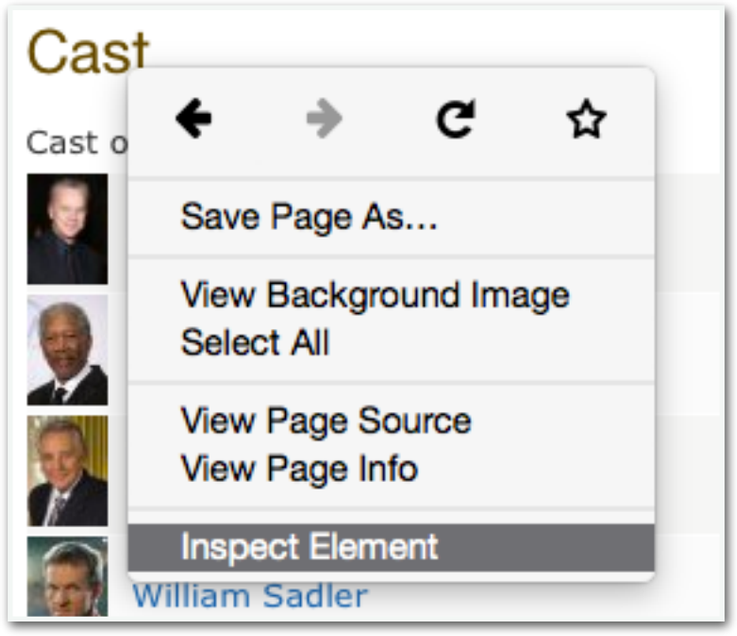

```python
from bs4 import BeautifulSoup
import requests


web_page = requests.get('http://...')
soup = BeautifulSoup(web_page.content)
```

(http://www.imdb.com/title/tt0111161/?ref_=chttp_tt_1)

# Find Stuff in a Soup of `html`

Using your browser's Developer Mode

(A demo is better than a thousand slides)




But in case you forget, it's

`[right click]`

`> [Inspect Element]`

... in most modern browsers

# finding a Specific tag

```
<table class='cast_list'>
```

```python
soup.find('table', class_='cast_list')
```

Note that we use `class_` instead of `class`.

This is because `class` means something else in `python`.

Anything other than `class`, you should use as is.

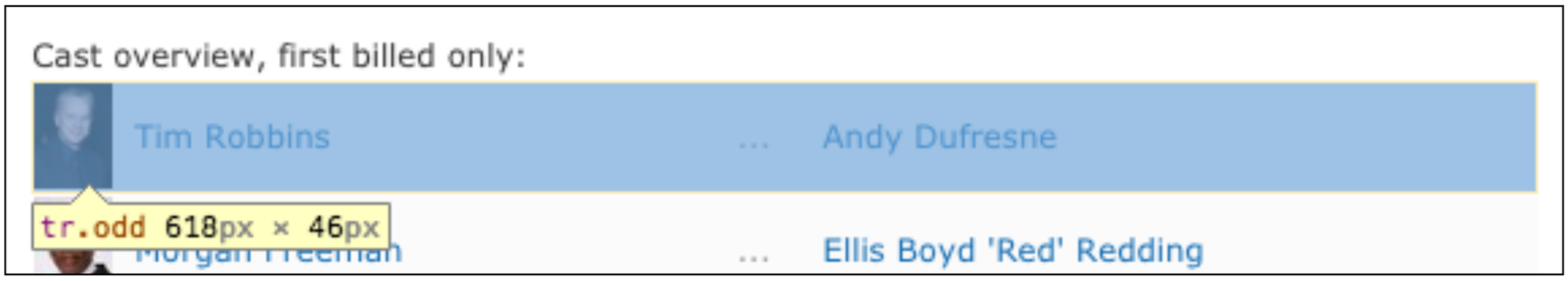
# `Strain`ing the Soup

When dealing with only a small portion of the entire page (like ourselves), it might speed things up a little to `strain` the soup with `SoupStrainer`, before `find`ing things in it.

```python
from bs4 import BeautifulSoup, SoupStrainer
import requests

cast_strainer = SoupStrainer('table', class_='cast_list')

web_page = requests.get('http://...')
soup = BeautifulSoup(web_page.content, parse_only=cast_strainer)
```

[python]

# What's in a `table`?




```
<table>
    <tr>
        <td></td>
        <td></td>
    </tr>
    <tr>
        <td></td>
    </tr>
</table>
```

within a `table` tag,

`<tr>` defines rows

while `<td></td>` defines columns within rows.

`html` tables are written row-by-row

# Iterate a `table` by its rows

We want to `iterate` each row of our strained `soup`

(`<table class="cast_list">`)

```python
for row in soup.find_all('tr'):
    # do useful things with each row
```

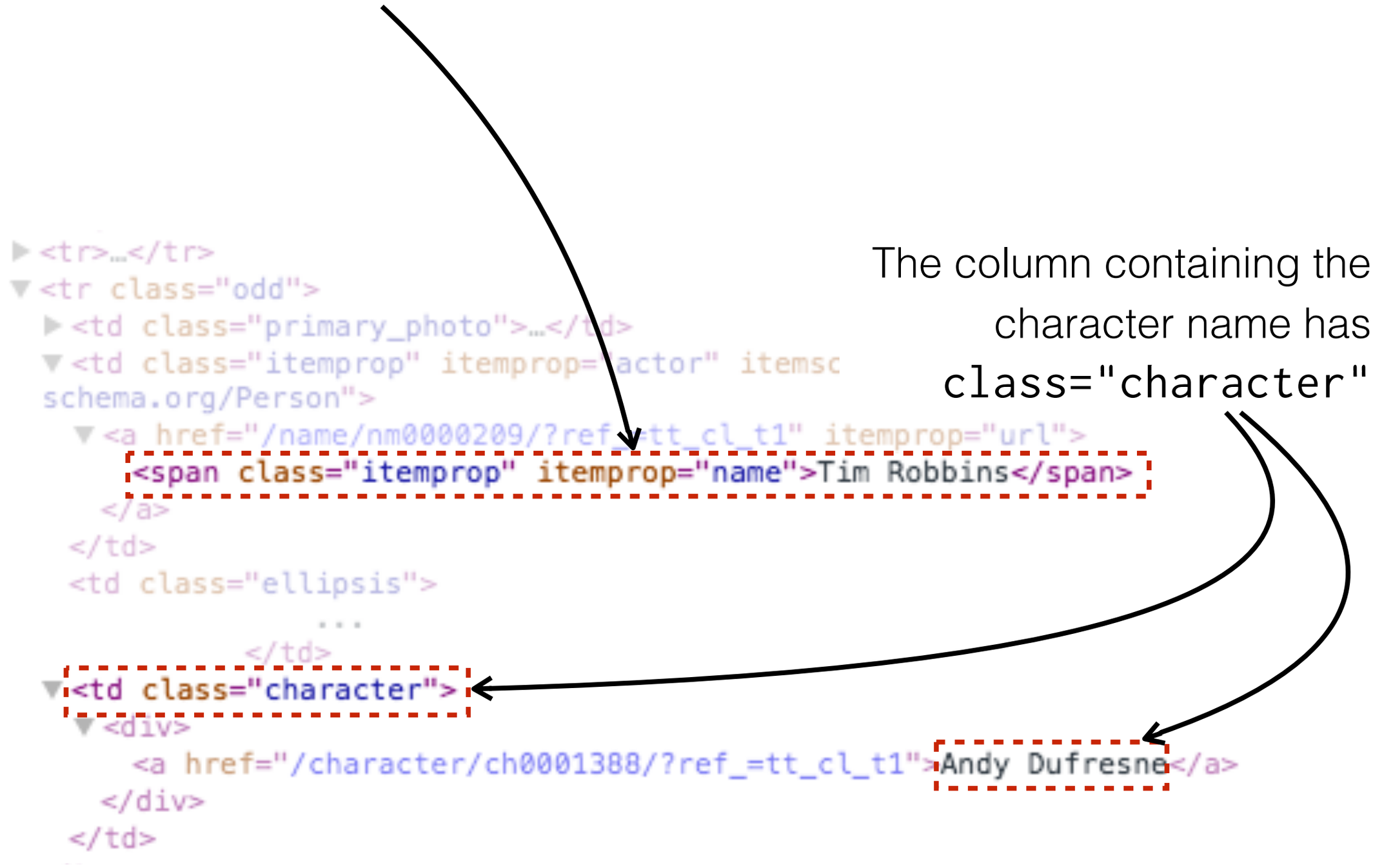
# Picking out the Cherries

How should we identify the **actor's name** and **character** played, given a single row (`<tr>`)?

```
▶ <tr>…</tr>
▼ <tr class="odd">
  ▶ <td class="primary_photo">…</td>
  ▼ <td class="itemprop" itemprop="actor" itemscope itemtype="http://
schema.org/Person">
    ▼ <a href="/name/nm0000209/?ref_=tt_cl_t1" itemprop="url">
        <span class="itemprop" itemprop="name">Tim Robbins</span>
      </a>
    </td>
    <td class="ellipsis">
                    ...
                </td>
  ▼ <td class="character">
    ▼ <div>
        <a href="/character/ch0001388/?ref_=tt_cl_t1">Andy Dufresne</a>
      </div>
    </td>
</tr>
```

An actor's name seems to be uniquely identified by the property `itemprop="name"`

The column containing the character name has `class="character"`




(*There's usually more than one way ...* )

An actor's name seems to be uniquely identified by the property `itemprop="name"`

```python
for row in soup.find_all('tr'):
    actor = row.find(itemprop='name').text
    role = row.find(class_='character').text
```

The column containing the character name has `class="character"`

`find_all('tr')` gives an extra row, which doesn't have anything that matches `itemprop='name'`

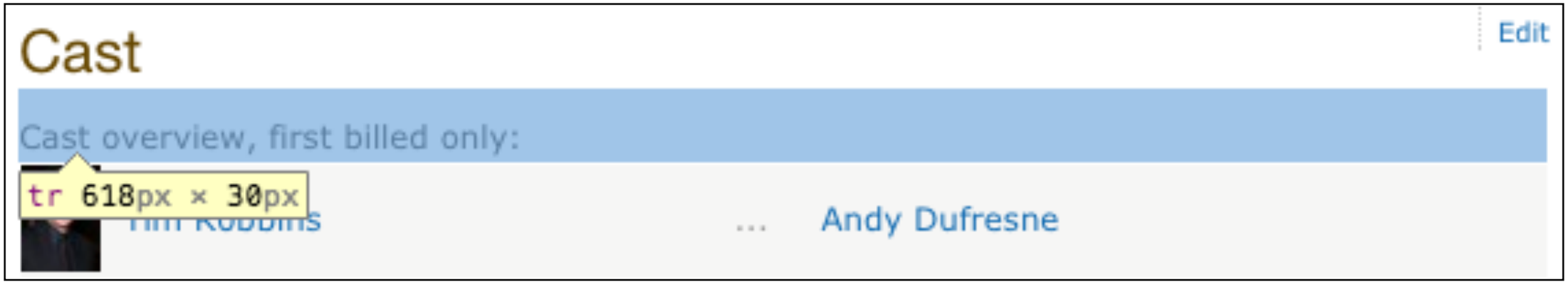
```
for row in soup.find_all('tr'):
    actor = row.find(itemprop='name').text
    role = row.find(class_='character').text
```

[python]

we want `python` to ignore these errors

```
...
AttributeError: 'NoneType' object has no
attribute 'text'
```

[output]



python will `try` this

```python
for row in soup.find_all('tr'):
    try:
        actor = row.find(itemprop='name').text
        role = row.find(class_='character').text

    except AttributeError:
        pass
```

if an `AttributeError` happens, it will do this (in this case, ignore the error and `pass`)

**WARNING!**

Practice caution with `try-except`. Don't `pass` an error, unless you're certain you it's an error you *want to* ignore.

```python
for row in soup.find_all('tr'):
    try:
        actor = row.find(itemprop='name').text
        role = row.find(class_='character').text

    except AttributeError:
        pass
```
[python]

```
u'Tim Robbins'
u'\n\nAndy Dufresne\n\n'
...
```
[output]

What's with all the
\n\n\n\n\n\n ... ?

Web designers sometimes use hidden white spaces for layout purposes.

A good way to deal with these in `python` is to surround your string with `' '.join(string.split())`

```python
for row in soup.find_all('tr'):
    try:
        actor = ' '.join(row.find(itemprop='name').text.split())
        role = ' '.join(row.find(class_='character').text.split())

    except AttributeError:
        pass
```

[python]

# Write the Data to a File

Usually, `print` is sufficient in `python`.

But `print` doesn't play well with weird characters ...

But the web is full of weird characters!

*One* workaround is to use `codecs`

```python
import codecs

...

with codecs.open('file_name.tsv', 'a', encoding='utf-8') as fid:
    print>> fid, '\t'.join([actor, role])
```

**PRO:** Works with most languages on the web (Chinese, Korean, ...)

**CON:** Trickier to pipe/redirect output using command line

# So far ...

```python
from bs4 import BeautifulSoup, SoupStrainer
import requests
import codecs

cast_strainer = SoupStrainer('table', class_='cast_list')

web_page = requests.get('http://...')
soup = BeautifulSoup(web_page.content, parse_only=cast_strainer)

for row in soup.find_all('tr'):
    try:
        actor = ' '.join(row.find(itemprop='name').text.split())
        role = ' '.join(row.find(class_='character').text.split())

        with codecs.open('file_name.tsv', 'a', encoding='utf-8') as fid:
            print>> fid, '\t'.join([actor, role])

    except AttributeError:
        pass
```

[python]

# Finally!

## Today's **GOAL**

Collect the **cast overview** (actor and character played) for each of the **Top 10 movies** of **IMDb Charts' Top 250** (http://www.imdb.com/chart/top?ref_=nv_ch_250_4)

# Workflow

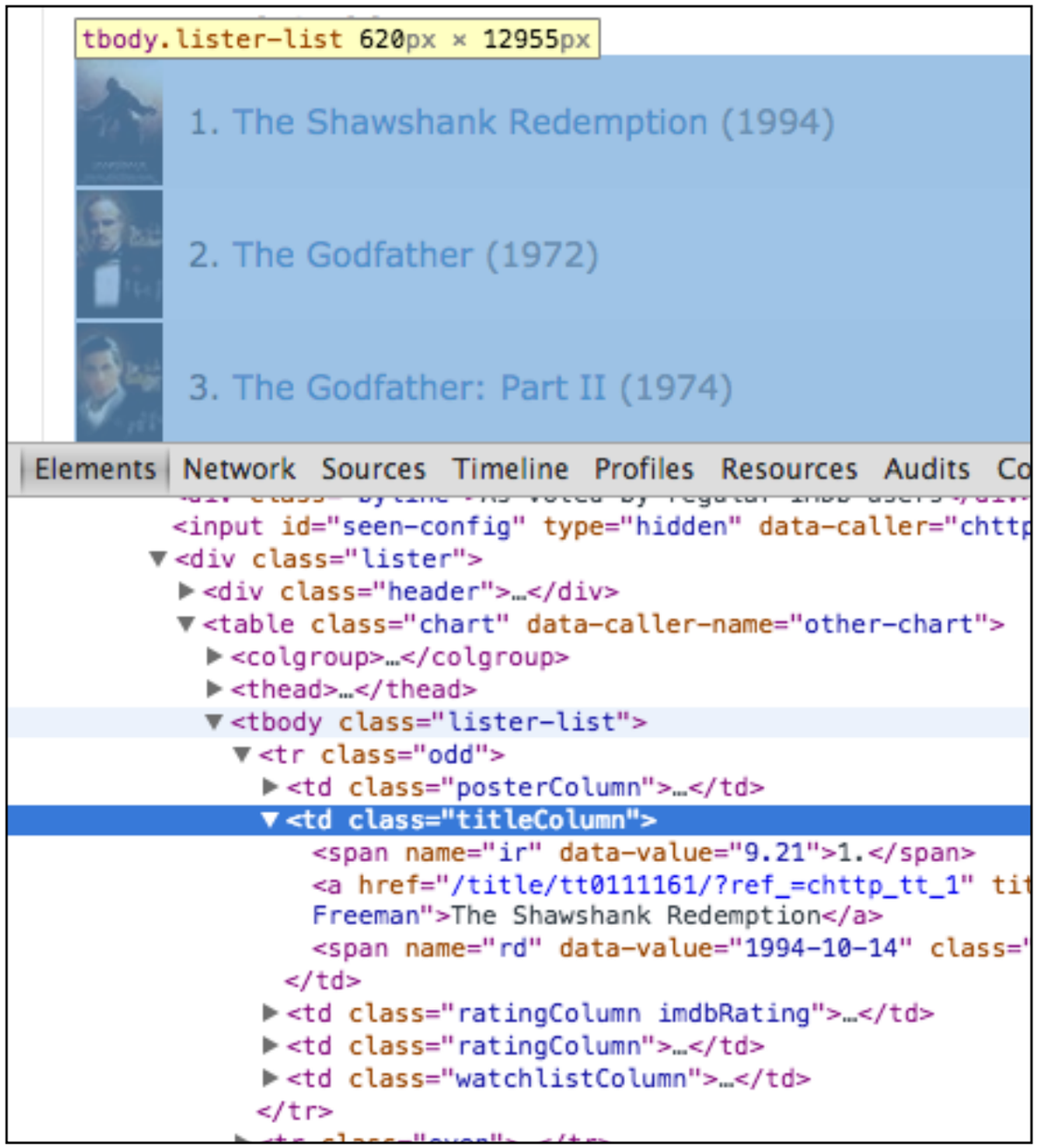

http://www.imdb.com/chart/top?ref_=nv_ch_250_4

`for` these links do ***this***

```python
from bs4 import BeautifulSoup, SoupStrainer
import requests
import codecs

cast_strainer = SoupStrainer('table', class_='cast_list')

web_page = requests.get('http://...')
soup = BeautifulSoup(web_page.content, parse_only=cast_strainer)

for row in soup.find_all('tr'):
    try:
        actor = ' '.join(row.find(itemprop='name').text.split())
        role = ' '.join(row.find(class_='character').text.split())

        with codecs.open('file_name.tsv', 'a', encoding='utf-8') as fid:
            print>> fid, '\t'.join([actor, role])

    except AttributeError:
        pass
```

[python]

# The Links




What do we need, and how should we get it?

remember:

```python
soup.a.get('href')
```

# Workflow

`requests` ▶ `BeatifulSoup` ▶ `.get('href')` ▶ `requests` ...

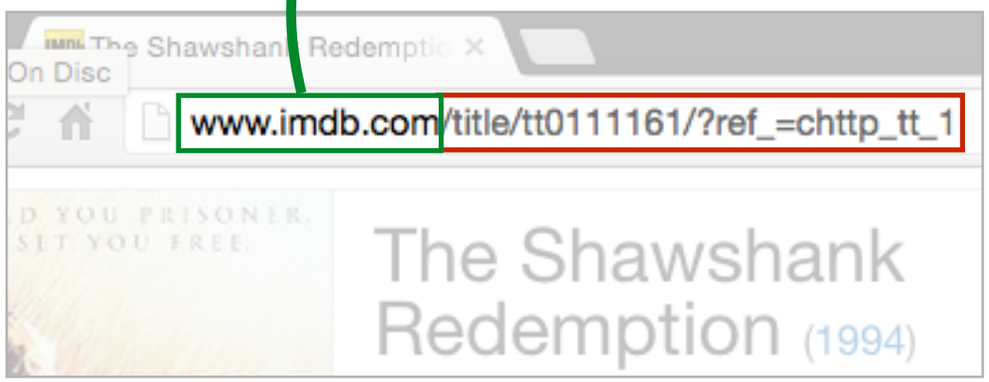
```python
web_page = requests.get('http://...')


list_strainer = SoupStrainer('tbody', class_='lister-list')
soup = BeautifulSoup(web_page.content, parse_only=list_strainer)


movie_list = soup.find_all('td', class_='titleColumn')

for movie in movie_list:
    movie_title = movie.a.text

    link = 'http://imdb.com' + movie.a.get('href')
    soup = requests.get(link)
    ...
```

[python]

# Workflow

**Build** a link from the `href` value

```python
link = 'http://imdb.com' + movie.a.get('href')
```




```
<a href="/title/tt0111161/?ref_=chttp_tt_1" title="Frank Darabont
(dir.), Tim Robbins, Morgan Freeman">The Shawshank Redemption</a>
```

"**Top 10 movies** of **IMDb Charts' Top 250**"

```python
soup.find_all('td', class_='titleColumn', limit=10)
```

The `limit` option will return only the first `10` results of `find_all`

# Be Nice

Don’t harass the servers

Rule-of-thumb(?): three requests per second

Space your requests using `sleep`

```python
from time import sleep

...
for link in movie_list:
    # request the link and do your thing
    ...
    sleep(0.3)
```

Unit is seconds.

So, `sleep(0.3)` = (approx.) three requests per second

# Thank you.

**Data Science Drop-in**
https://5harad.com/drop-in/

**Mondays @ 4–6 pm in Y2E2 253**