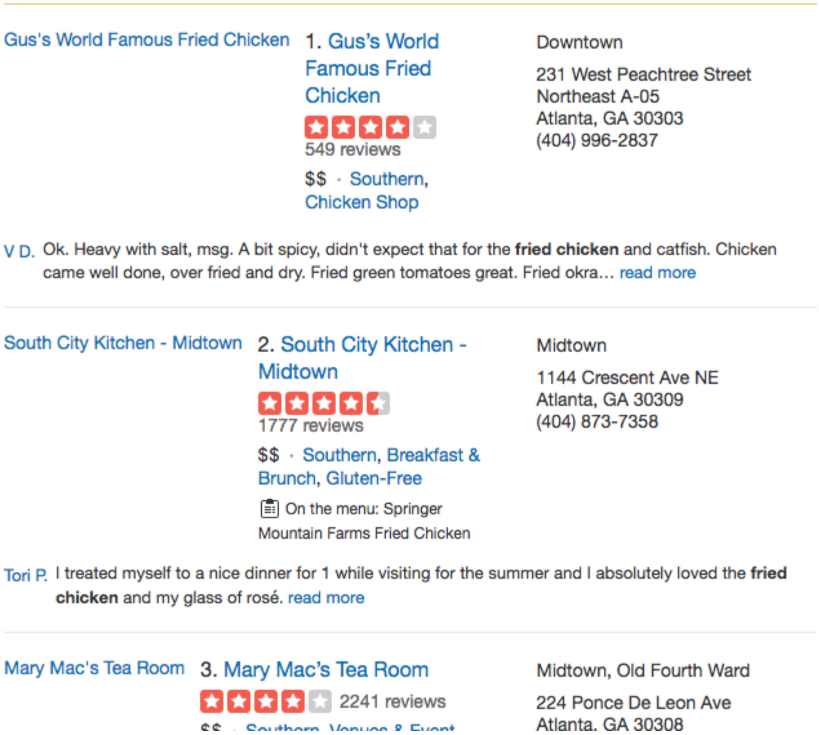
Part 1 of 2: Processing an HTML file

One of the richest sources of information is the Web! In this notebook, we ask you to use string processing and regular expressions to mine a web page, which is stored in HTML format.

The data: Yelp! reviews. The data you will work with is a snapshot of a recent search on the Yelp! site for the best fried chicken restaurants in Atlanta. That snapshot is hosted here: https://cse6040.gatech.edu/datasets/yelp-example

If you go ahead and open that site, you'll see that it contains a ranked list of places:



Gus's World Famous Fried Chicken 1. Gus's World Famous Fried Chicken
549 reviews
$$ · Southern, Chicken Shop

Downtown
231 West Peachtree Street
Northeast A-05
Atlanta, GA 30303
(404) 996-2837

V D. Ok. Heavy with salt, msg. A bit spicy, didn't expect that for the **fried chicken** and catfish. Chicken came well done, over fried and dry. Fried green tomatoes great. Fried okra… read more

South City Kitchen - Midtown 2. South City Kitchen - Midtown
1777 reviews
$$ · Southern, Breakfast & Brunch, Gluten-Free
On the menu: Springer Mountain Farms Fried Chicken

Midtown
1144 Crescent Ave NE
Atlanta, GA 30309
(404) 873-7358

Tori P. I treated myself to a nice dinner for 1 while visiting for the summer and I absolutely loved the **fried chicken** and my glass of rosé. read more

Mary Mac's Tea Room 3. Mary Mac's Tea Room
2241 reviews
$$ · Southern, Venues & Event

Midtown, Old Fourth Ward
224 Ponce De Leon Ave
Atlanta, GA 30308

Your task. In this part of this assignment, we'd like you to write some code to extract this list.

Getting the data

First things first: you need an HTML file. The following Python code will download a particular web page that we've prepared for this exercise and store it locally in a file.

> If the file exists, this command will not overwrite it. By not doing so, we can reduce accesses to the server that hosts the file. Also, if an error occurs during the download, this cell may report that the downloaded file is corrupt; in that case, you should try re-running the cell.

In [1]:
```
import requests
import os
import hashlib

if os.path.exists('.voc'):
    data_url = 'https://cse6040.gatech.edu/datasets/yelp-example/yelp.htm'
else:
    data_url = 'https://github.com/cse6040/labs-fa17/raw/master/datasets/yelp.htm'

if not os.path.exists('yelp.htm'):
    print("Downloading: {} ...".format(data_url))
    r = requests.get(data_url)
    with open('yelp.htm', 'w', encoding=r.encoding) as f:
        f.write(r.text)

with open('yelp.htm', 'r', encoding='utf-8') as f:
    yelp_html = f.read().encode(encoding='utf-8')
    checksum = hashlib.md5(yelp_html).hexdigest()
    assert checksum == "4a74a0ee9cefee773e76a22a52d45a8e", "Downloaded file has incorrect checksum!"
```

```
'yelp.htm' is ready!
```

Viewing the raw HTML in your web browser. The file you just downloaded is the raw HTML version of the data described previously. Before moving on, you should go back to that site and use your web browser to view the HTML source for the web page. Do that now to get an idea of what is in that file.

> If you don't know how to view the page source in your browser, try the instructions on this site.

Reading the HTML file into a Python string. Let's also open the file in Python and read its contents into a string named, `yelp_html`.

In [1]:

```
with open('yelp.htm', 'r', encoding='utf-8') as yelp_file:
    yelp_html = yelp_file.read()

# Print first few hundred characters of this string:
print("*** type(yelp_html) == {} ***".format(type(yelp_html)))
n = 1000
```

```
*** type(yelp_html) == <class 'str'> ***
*** Contents (first 1000 characters) ***
<!DOCTYPE html>
<!-- saved from url=(0079)https://www.yelp.com/search?find_desc=fried+chicken&find_loc=Atlanta%2C+GA&ns=1 -->
<html xmlns:fb="http://www.facebook.com/2008/fbml" class="js gr__yelp_com" lang="en"><!--<![endif]--><head data-component-bou
nd="true"><meta http-equiv="Content-Type" content="text/html; charset=UTF-8"><link type="text/css" rel="stylesheet" href="./B
est Fried chicken in Atlanta, GA - Yelp_files/css"><style type="text/css">.gm-style .gm-style-cc span,.gm-style .gm-style-cc
a,.gm-style .gm-style-mtc div{font-size:10px}
</style><style type="text/css">@media print {  .gm-style .gmnoprint, .gmnoprint {    display:none  }}@media screen {  .gm-sty
le .gmnoscreen, .gmnoscreen {    display:none  }}</style><style type="text/css">.gm-style-pbc{transition:opacity ease-in-out;
background-color:rgba(0,0,0,0.45);text-align:center}.gm-style-pbt{font-size:22px;color:white;font-family:Roboto,Arial,sans-se
rif;position:relative;margin:0;top:50%;-webkit-transform:translateY(-50%);-ms-transform:translateY(-50%);transform:translateY
(-50%)}
</style><script src="./Best Fried chicken in Atlanta, GA - Yelp_files/rules-p-M4yfUTCPeS3vn.js" style=""></script><script src
="./Best Fried chicken in Atlanta, GA - Yelp_files/segments.json" async="" type="text/javascript"></script>


    <script src="./Best Fried chicken in Atlanta, GA - Yelp_files/102029836881428" async=""></script><script async="" src="./
Best Fried chicken in Atlanta, GA - Yelp_files/fbevents.js"></script><script async="" src="./Best Fried chicken in Atlanta, G
```

Oy, what a mess! It will be great to have some code read and process the information contained within this file.

Exercise (5 points): Extracting the ranking

Write some Python code to create a variable named `rankings`, which is a list of dictionaries set up as follows:

- `rankings[i]` is a dictionary corresponding to the restaurant whose rank is `i+1`. For example, from the screenshot above, `rankings[0]` should be a dictionary with information about Gus's World Famous Fried Chicken.
- Each dictionary, `rankings[i]`, should have these keys:
  - `rankings[i]['name']`: The name of the restaurant, a string.
  - `rankings[i]['stars']`: The star rating, as a string, e.g., `'4.5'`, `'4.0'`
  - `rankings[i]['numrevs']`: The number of reviews, as an **integer.**
  - `rankings[i]['price']`: The price range, as dollar signs, e.g., `'$'`, `'$$'`, `'$$$'`, or `'$$$$'`.

Of course, since the current topic is regular expressions, you might try to apply them (possibly combined with other string manipulation methods) find the particular patterns that yield the desired information.

In [ ]:

```
#
# YOUR CODE HERE
#
```

In [ ]:

```
# Test cell: `rankings_test`

assert type(rankings) is list, "`rankings` must be a list"
assert all([type(r) is dict for r in rankings]), "All `rankings[i]` must be dictionaries"

print("=== Rankings ===")
for i, r in enumerate(rankings):
    print("{}. {} ({}): {} stars based on {} reviews".format(i+1,
                                                             r['name'],
                                                             r['price'],
                                                             r['stars'],
                                                             r['numrevs']))

assert rankings[0] == {'numrevs': 549, 'name': 'Gus’s World Famous Fried Chicken', 'stars': '4.0', 'price': '$$'}
assert rankings[1] == {'numrevs': 1777, 'name': 'South City Kitchen - Midtown', 'stars': '4.5', 'price': '$$'}
assert rankings[2] == {'numrevs': 2241, 'name': 'Mary Mac’s Tea Room', 'stars': '4.0', 'price': '$$'}
assert rankings[3] == {'numrevs': 481, 'name': 'Busy Bee Cafe', 'stars': '4.0', 'price': '$$'}
assert rankings[4] == {'numrevs': 108, 'name': 'Richards’ Southern Fried', 'stars': '4.0', 'price': '$$'}
assert rankings[5] == {'numrevs': 93, 'name': 'Greens &amp; Gravy', 'stars': '3.5', 'price': '$$'}
assert rankings[6] == {'numrevs': 350, 'name': 'Colonnade Restaurant', 'stars': '4.0', 'price': '$$'}
assert rankings[7] == {'numrevs': 248, 'name': 'South City Kitchen Buckhead', 'stars': '4.5', 'price': '$$'}
assert rankings[8] == {'numrevs': 1558, 'name': 'Poor Calvin’s', 'stars': '4.5', 'price': '$$'}
assert rankings[9] == {'numrevs': 67, 'name': 'Rock’s Chicken &amp; Fries', 'stars': '4.0', 'price': '$'}
```

Fin! This cell marks the end of Part 1. Don't forget to save, restart and rerun all cells, and submit it. When you are done, proceed to Part 2.