```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sat Oct 29 22:00:35 2016

@author: JKM
"""

import pandas as pd
import requests
import numpy as np
import time
from bs4 import BeautifulSoup as bs4

loc_prefixes = ['doc', 'nva', 'mid']

def find_prices(results):
    prices = []
    for rw in results:
        price = rw.find('span', {'class': 'result-price'})
        if price is not None:
            price = float(price.text.strip('$'))
        else:
            price = np.nan
        prices.append(price)
    return prices

def find_times(results):
    times = []
    for rw in autos:
        if time is not None:
            date = time['datetime']
            date = pd.to_datetime(date)
        else:
            date = np.nan
        times.append(date)
    return times
```

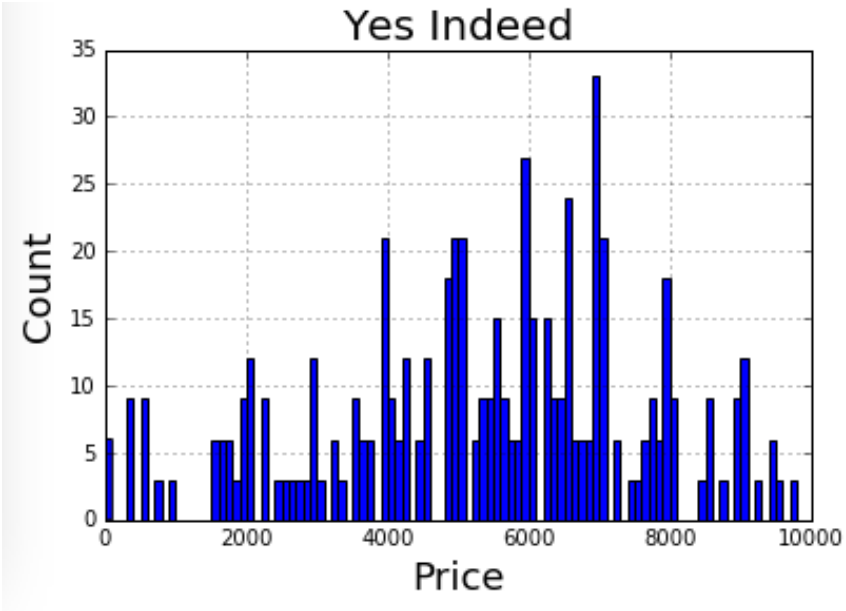
```
results = []
search_indices = np.arange(0, 300, 100)

for loc in loc_prefixes:
    print loc
    time.sleep(5)
    for i in search_indices:
        url = 'https://washingtondc.craigslist.org/search/cto'.format(loc)
        resp = requests.get(url, params={'min_auto_year': 2010,
'max_auto_miles': 100000, 's': i})
        txt = bs4(resp.text, 'html.parser')
        autos = txt.findAll(attrs={'class': "result-row"})

        title = [rw.find('a', attrs={'class': 'result-title hdrlnk'}).text for rw in autos]
        auto_id = [rw.find('a', attrs={'class': 'result-title hdrlnk'})['data-id'] for rw
in autos]
        link = [rw.find('a', attrs={'class': 'result-title hdrlnk'})['href'] for rw in
autos]
        date = [pd.to_datetime(rw.find('time')['datetime']) for rw in autos]
        price = find_prices(autos)

        data = np.array([date, auto_id, price, title, link])
        col_names = ['date', 'auto_id', 'price', 'title', 'link']
        df = pd.DataFrame(data.T, columns=col_names)
        df = df.set_index('date')
        df['loc'] = loc

        results.append(df)

results = pd.concat(results, axis=0)

results[['price']] = results[['price']].convert_objects(convert_numeric=True)

results.head()

df

ax = results.hist('price', bins=np.arange(0, 10000, 100))[0, 0]
```

```
ax.set_title('DC Metro Count Used Cars, Dist by Price', fontsize=20)
ax.set_xlabel('Price', fontsize=18)
ax.set_ylabel('Count', fontsize=18)

import string
use_chars = string.ascii_letters +\
    ''.join([str(i) for i in range(10)]) +\
    ' /\.'
results['title'] = results['title'].apply(
    lambda a: ''.join([i for i in a if i in use_chars]))


#df.to_csv('/Users/newuser/Desktop/craigslist_results.csv')
```

For Auto Supply Data, Hit Craigslist

- I was able to develop a successful python code to “scrape” and “parse” data from Craigslist into the statistical package, *pandas*
  - This code delivers a snapshot of counts in a region, restricted by age-of-car and mileage, and distributed by price
  - As a capability, this approach provides a real-time glimpse into the used-car market, and is helpful for tracking a region over time
  - The scraped data can be off-loaded into .csv or .xls
- I see much utility in this method as an approach to answering CarMax’s question of supply; however, it is one so large in scope that it is not useful to this specific assignment

DC Metro Vehicles, >2006, <150k, (top 10k entries), on Sat, Oct 29



Learning Curve

- Because of how the HTML code is structured, Make and Model information can be pulled only from the post's title, which means that we would need to write SQL scripts to further parse and scrub the data, and that some values would be missing
  - Each has a unique identifier, however, and this might be used to produce secondary queries for specific Makes and Models, but only if you REALLY want to upset the internet gods
- Because of how CL structures their website pages (which is what this code searches), it is difficult to know how to restrict a search by date or within a date range

- There is a steep learning curve that makes further development of this code inefficient for our class's purposes
  - The whole point of coding/scripting is to *save* time
- My next – albeit, "off the clock" – goal is to distribute counts in a region, restricted by vehicle age and mileage, by date
  - Also to finish editing how to export to .csv