From: Piruz Alemi
Subject: Web scraping & MongoDB Postings
Report on Nasa News Data Base
Date: Feb 6, 2020

Objectives met:

- Created and connected to local MongoDB databases.
- Created, read, updated, and deleted MongoDB documents using the Mongo Shell.
- Created a simple Python application that connected to and modified MongoDB databases using the PyMongo library.
- Used Beautiful Soup to scrape my own data from the web.
- Saved the results of web scraping into MongoDB.
- Became comfortable rendering templates with Flask using data retrieved from a Mongo database.
- Used Beautiful Soup to scrape data.
- Used PyMongo to save data to a Mongo database.
- Used Flask to render templates.

Helpful Links used were:

- Mongo in 30 minutes
- Python Requests
- Webscraping with BeautifulSoup
- Python Splinter

JPL Mars Space Images - Featured Image

- Visited the url for JPL Featured Space Image here.
  - https://www.jpl.nasa.gov/spaceimages/?search=&category=Mars
- Use splinter to navigate the site and find the image url for the current Featured Mars Image and assign the url string to a variable called `featured_image_url`.
- Make sure to find the image url to the full size `.jpg` image.
- Make sure to save a complete url string for this image.

```
# Example:
featured_image_url =
'https://www.jpl.nasa.gov/spaceimages/images/largesize/PIA16225_hires.jpg
'
```

Completed:

- Appended the dictionary with the image url string and the hemisphere title to a list. This list contained one dictionary for each hemisphere.

```
# Example:
hemisphere_image_urls = [
    {"title": "Valles Marineris Hemisphere", "img_url": "..."},
    {"title": "Cerberus Hemisphere", "img_url": "..."},
    {"title": "Schiaparelli Hemisphere", "img_url": "..."},
    {"title": "Syrtis Major Hemisphere", "img_url": "..."},
```

Steps:

I was able to scrape the Nasa web site and post to MongoDB:

*I was able to access **children** and **siblings** of Nasa web site HTML Tags for **different pages**…*

Navigations of these web pages were achieved by using Splinter Library for Clicks()

Snaps shots of my program follows:

```
In [43]: # go deeper into the anchor href and see what is inside?
         r = requests.get("https://mars.nasa.gov/news/8558/global-storms-on-mars-launch-dust-towers-into-the-sky/", params=dict(
             query="<p>",
             page=1
         ))

         print(r.text)
```

```
<!DOCTYPE html>
<html lang='en' xml:lang='en' xmlns='http://www.w3.org/1999/xhtml'>
<head>
<meta content='text/html; charset=utf-8' http-equiv='Content-Type'>
<!-- Always force latest IE rendering engine or request Chrome Frame -->
<meta content='IE=edge,chrome=1' http-equiv='X-UA-Compatible'>
```

I scrape various <Titles>, Divisions, <Spans>, <Li>:

```
In [47]: # Examine the results, then determine element that contains sought info
         # results are returned as an iterable list
         divClass = soup.find_all('div', class_='brand2')
         print(divClass)
```

```
[]
```

```
In [48]: # S P A N is used for in-line  vs. Div.
         span = soup.find_all('span')
         span
```

```
Out[48]: [<span class="top-nav__account-text"></span>,
          <span class="top-nav__primary__menu__item__link__text--account"></span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--scores">
                                                                  Scores
                                                                  </span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--news">
                                                                  News
                                                                  </span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--video">
                                                                  Video
                                                                  </span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--standings">
                                                                  Standings
                                                                  </span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--stats">
                                                                  Stats
                                                                  </span>,
          <span class="top-nav__primary__menu__item__link__text top-nav__primary__menu__item__link__text--schedule">
                                                                  Schedule
                                                                  </span>,
```

```
In [49]: # The HTML <li> tag represents a list item in ordered and unordered lists. The <li> tag is placed inside either
         # a <ol> tag or a <ul>  to represent each individual item within that list. It can also be used with the <menu> tag
         li = soup.find('li')
         print(li)
```

```
<li class="top-nav__primary__menu__item top-nav__primary__menu__item--account top-nav__primary__menu__item--default">
<div class="top-nav__primary__menu__item-wrapper">
<a class="top-nav__primary__menu__item__link top-nav__primary__menu__item__link--account top-nav__primary__menu__item
--account__link" href="#"><i class="top-nav__primary__menu__item__link--account-icon top-nav__account__icon top-nav__
icon top-nav__icon--account"></i><span class="top-nav__primary__menu__item__link__text--account"></span></a>
</div>
</li>
```

I was able to loop through a list of items, and post the dictionary of items:

```
In [57]: # Examine the results, then determine element that contains sought info
         # results are returned as an iterable list
         results = soup.find_all('li')

         # Loop through returned results
         for result in results:
             # Error handling
             try:
                 # Identify and return title of listing
                 title = result.find('a').text
                 # Identify and return paragraph of listing
                 # pargraph = result.p.text
                 # Identify and return link to listing
                 link = result.a['href']

                 # Run only if title, price, and link are available
                 if (title or paragraph or link):
                     # Print results
                     print('-------------------------- Title is: ---------------------------------')
                     print(title)
                     print('********************** paragraph is: *******************************')
                     print(paragraph)
                     print('_________________________ Link is: _____________________________________')
                     print(link)
                     print(" <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< . >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>")
                     # ------------------------------------------------------------------
                     # Dictionary to be inserted as a MongoDB document
                     # ------------------------------------------------------------------
                     post = {
                         'title': title,
                         'paragraph': paragraph,
                         'url': link
                     }

                     collection.insert_one(post)

             except Exception as e:
                 print(e)
```

```
-------------------------- Title is: ---------------------------------

********************** paragraph is: *******************************
<p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science Mission Directorate</
p>
_________________________ Link is: _____________________________________
#
 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< . >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
cannot encode object: <p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science
Mission Directorate</p>, of type: <class 'bs4.element.Tag'>
```

```
 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< . >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
cannot encode object: <p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science
Mission Directorate</p>, of type: <class 'bs4.element.Tag'>
-------------------------- Title is: ---------------------------------
Corporate Marketing Partners
********************** paragraph is: *******************************
<p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science Mission Directorate</
p>
_________________________ Link is: _____________________________________
https://www.nhl.com/info/corporate-marketing-partners
 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< . >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
cannot encode object: <p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science
Mission Directorate</p>, of type: <class 'bs4.element.Tag'>
-------------------------- Title is: ---------------------------------
Sponsorship Opportunities
********************** paragraph is: *******************************
<p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science Mission Directorate</
p>
_________________________ Link is: _____________________________________
/info/sponsorship-opportunities
 <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< . >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
cannot encode object: <p>Managed by the Mars Exploration Program and the Jet Propulsion Laboratory for NASA's Science
Mission Directorate</p>, of type: <class 'bs4.element.Tag'>
```

```
In [58]: # Display items in MongoDB collection
         listings = db.items.find()

         for listing in listings:
             print(listing)
```

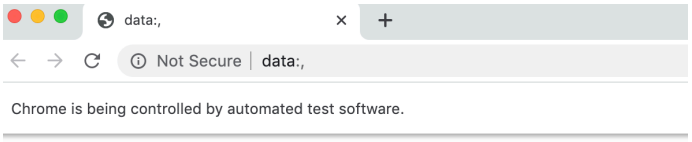
```
In [22]: # Extract title text
         title = soup.title.text
         print(title)
```

```
Global Storms on Mars Launch Dust Towers Into the Sky — NASA's Mars Exploration Program
```

```
In [23]: # Print all paragraph texts
         paragraphs = soup.find_all('p')
         for paragraph in paragraphs:
             print('------------')
             print(paragraph.text)
```

```
------------
Mars Before and After Dust Storm: Side-by-side movies shows how the 2018 global dust storm enveloped the Red Planet,
courtesy of the Mars Color Imager (MARCI) camera onboard NASA's Mars Reconnaissance Orbiter. This global dust storm c
aused NASA's Opportunity rover to lose contact with Earth. Credit: NASA/JPL-Caltech/MSSS. Full download ›
------------
Dust storms are common on Mars. But every decade or so, something unpredictable happens: A series of runaway storms b
reaks out, covering the entire planet in a dusty haze.
```

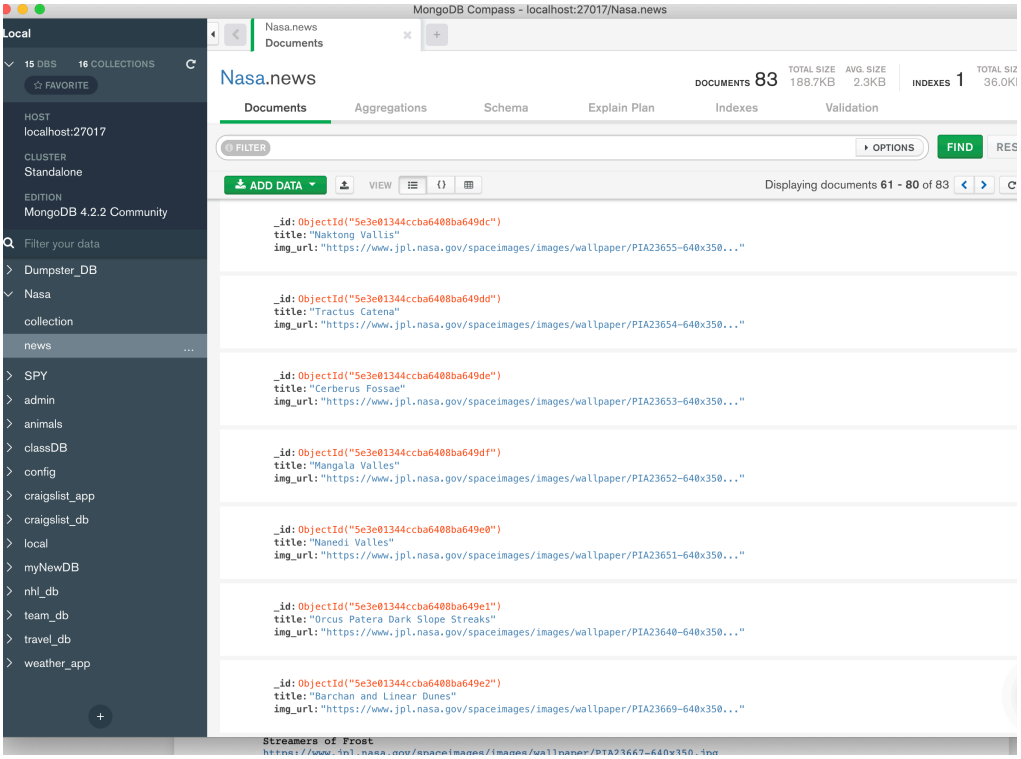
Via Chromdrive:



On the relation between “a” and “href” :

```
In [ ]: #-------------------------------------------------------------------------------------
        #### Scraped the NASA Mars News Site and collected the latest News Title and Paragraph Text.
        #### Saved these parsed variables for future reference
        ####
        #-------------------------------------------------------------------------------------

In [12]: anchor_href = soup.find("a")["href"]
         anchor_href

Out[12]: 'http://www.nasa.gov'

In [72]: # finding all the links
         link_list = [a['href'] for a in soup.find_all('a', href=True)]
         link_list

Out[72]: ['#content-wrap',
          '/',
          '#',
          'https://www.nhl.com/scores',
          '/news',
          '/news/t-277548856',
          '/news/t-278542340',
          '/news/t-277729150',
          '/news/t-278386388',
          '/news/t-278387726',
          '/news/t-292275938',
          '/news/t-277729400',
          '/news/t-277764372',
          '/news/t-281011650',
          '/info/nhl-declaration-of-principles/',
          '/community/hockey-is-for-everyone',
          '/news/t-277729160',
          '/news/t-277549076',
          '/hockeyfightscancer',
          '/info/nhl-green',
```

Insertions in MongoDb from a ollection of Nasa News Image links and its corresponding titles follows:

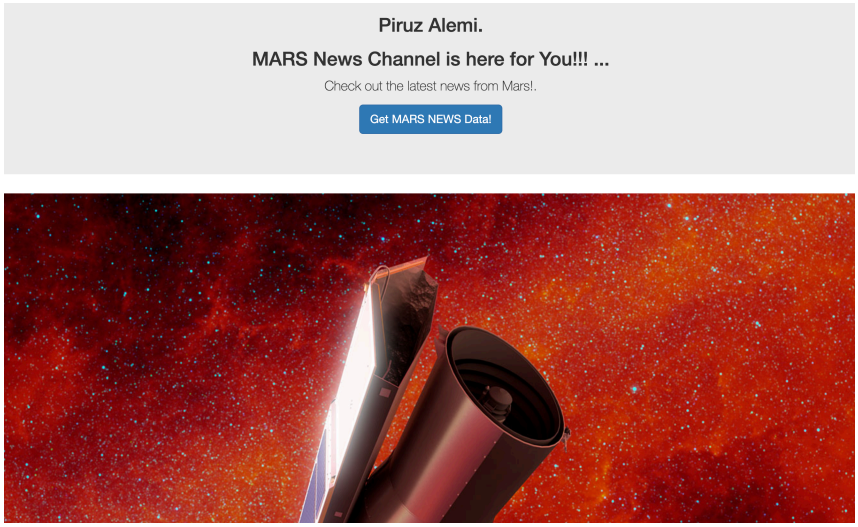


But we need an array of dictionaries to insert all the data! Which we did!!
As noted in: https://www.youtube.com/watch?v=pWbMrx5rVBE

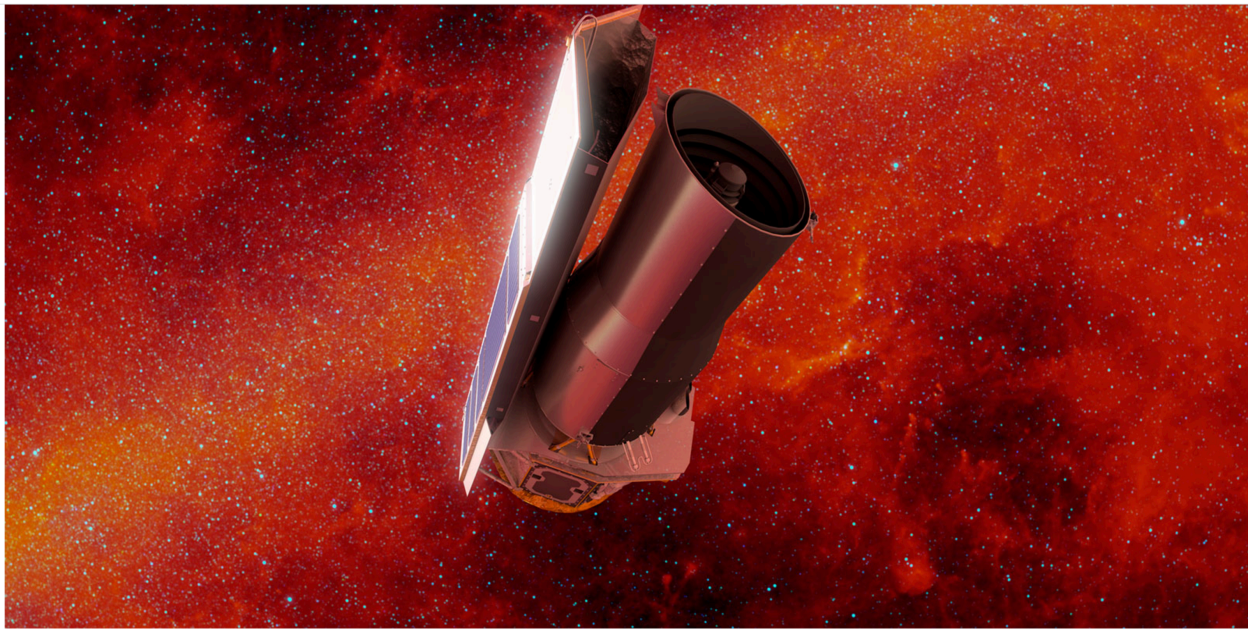
Next I ran the APP.py

```
web-scraping-challenge — python ◂ python app.py — 80×24
~/web-scraping-challenge — python ◂ python app.py
app.py
app2.py
app3.py
bin
export_dataframe.csv
scrapeAlemi.ipynb
scrape_costa.ipynb.py
scrape_craigslist.py
scrape_nasa.py
static
templates
~$ruz Alemi NASA Report on feb 6 2020 Images & Titles.docx
(base) Piruzs-MBP:web-scraping-challenge piruzalemi$ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 153-827-043
```

Running the above APP.PY with its corresponding Template & Index-HTML under Flask and its Routes / & /Scrape We get:



With its corresponding New Title:

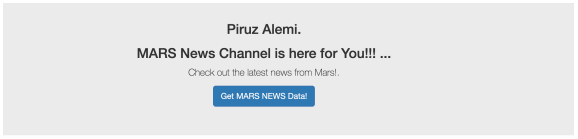


The typical Mars News is:

Nasa News Title: NASA Jet Propulsion Laboratory (JPL) - Space Mission and Science News, Videos and Images

Nasa News Links: /contact_JPL.php

And a click() on Piruz Alemi Channel News, scrapes the following:



The typical Mars News is:

Nasa News Title: NASA Jet Propulsion Laboratory (JPL) - Space Mission and Science News, Videos and Images

Nasa News Links: /contact_JPL.php