Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia

Nick Belgau

Objectives

Web scrap Falcon 9 launch records with `BeautifulSoup` :

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame

Launch records are stored online in a HTML table shown below:

Import Libraries

In [3]:
```
!pip3 install beautifulsoup4
!pip3 install requests
```

```
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/d
hcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecated,
use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/u
til.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated, use
int.from_bytes instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: beautifulsoup4 in /opt/conda/envs/Python-3.7-O
penCE/lib/python3.7/site-packages (4.9.1)
Requirement already satisfied: soupsieve>1.2 in /opt/conda/envs/Python-3.7-Op
enCE/lib/python3.7/site-packages (from beautifulsoup4) (2.0.1)
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/d
hcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecated,
use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/u
til.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated, use
int.from_bytes instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: requests in /opt/conda/envs/Python-3.7-OpenCE/
lib/python3.7/site-packages (2.22.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-
3.7-OpenCE/lib/python3.7/site-packages (from requests) (2021.5.30)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /opt/conda/envs/Pytho
n-3.7-OpenCE/lib/python3.7/site-packages (from requests) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /op
t/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages (from requests)
(1.25.9)
Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/envs/Python-3.7-O
penCE/lib/python3.7/site-packages (from requests) (2.8)
```

In [4]:
```
import sys

import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

Define Functions

In [5]:
```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cel
ls.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

Web Scraping

Setup BeautifulSoup Object

```
In [6]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_
        Falcon_Heavy_launches&oldid=1027686922"
```

Request the HTML page from the URL using HTTP GET method and get a `response` object

```
In [7]: # use requests.get() method with the provided static_url
        # Save the text of the response

        html_text = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
In [8]: soup=BeautifulSoup(html_text, 'html5lib')
```

```
In [9]: # Print page title
        print(soup.title)

        <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Extract all column/variable names from the HTML table header

Collect all relevant column names from the HTML table header

Find all tables on the wiki page first

```
In [9]: # Use the find_all function in the BeautifulSoup object, with element type `ta
        ble`
        # Assign the result to a list called `html_tables`
        html_tables = soup.find_all('table')
```

The third table contains the actual launch records.

```
In [28]: # Show the third table and check its content
         first_launch_table = html_tables[2]
```

```
<tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title
="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List o
f Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" id
="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#c
ite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
<th scope="col">Launch<br/>outcome
</th>
<th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9
 first-stage landing tests">Booster<br/>landing</a>
</th></tr>
```

Iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

In [29]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header()
 to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) i
nto a list called column_names

temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

Check the extracted column names

```
In [30]: print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass',
'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'La
unch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome',
'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass',
'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'La
unch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome',
'N/A', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload
mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time (
)', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch ou
tcome', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload
mass', 'Orbit', 'Customer', 'Launch outcome', 'FH 2', 'FH 3', 'Flight No.',
'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Cust
omer', 'Launch outcome', 'Date and time ( )', 'Launch site', 'Payload', 'Payl
oad mass', 'Orbit', 'Customer', 'Launch outcome', 'Date and time ( )', 'Launc
h site', 'Payload', 'Orbit', 'Customer', 'Date and time ( )', 'Launch site',
'Payload', 'Orbit', 'Customer', 'Date and time ( )', 'Launch site', 'Payloa
d', 'Orbit', 'Customer', 'Date and time ( )', 'Launch site', 'Payload', 'Orbi
t', 'Customer', 'Demo flights', 'logistics', 'Crewed missions', 'Commercial s
atellites', 'Scientific satellites', 'Military satellites', 'Rideshares', 'Cu
rrent', 'In development', 'Retired', 'Canceled', 'Spacecraft', 'Cargo', 'Crew
ed', 'Test vehicles', 'Current', 'Retired', 'Unflown', 'Orbital', 'Atmospheri
c', 'Landing sites', 'Other facilities', 'Support', 'Contracts', 'R&D program
s', 'Key people', 'General', 'General', 'Vehicles', 'Launches by rocket typ
e', 'Launches by spaceport', 'Agencies, companies and facilities', 'Other mis
sion lists and timelines']
```

Create a data frame by parsing the launch HTML tables

Create empty dictionary which will be pd df later.

Keys: extracted column names

In [31]:
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Fill up the `launch_dict` with launch records extracted from table rows.

Clean up unexpected annotations and other types of noise.

This includes reference links `B0004.1[8]` , missing values `N/A [e]` , inconsistent formatting, etc.

In [36]:
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowh
eaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to lau
nch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False

        row=rows.find_all('td') #get table element

        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1

            # Flight Number value
            launch_dict["Flight No."].append(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)

            # Time value
            time = datatimelist[1]
            launch_dict["Time"].append(time)

            # Booster version
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict["Version Booster"].append(bv)

            # Launch Site
            launch_site = row[2].a.string
            launch_dict["Launch site"].append(launch_site)

            # Payload
            payload = row[3].a.string
            launch_dict["Payload"].append(payload)

            # Payload Mass
            payload_mass = get_mass(row[4])
            launch_dict["Payload mass"].append(payload_mass)

            # Orbit
            orbit = row[5].a.string
            launch_dict["Orbit"].append(orbit)

            # Customer
            customer = row[6].a.string
            launch_dict["Customer"].append(customer)

            # Launch outcome
            launch_outcome = list(row[7].strings)[0]
            launch_dict["Launch outcome"].append(launch_outcome)

            # Booster landing
            booster_landing = landing_status(row[8])
            launch_dict["Booster landing"].append(booster_landing)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-36-488df74fa56f> in <module>
     58 
     59             # Customer
---> 60             customer = row[6].a.string
     61             launch_dict["Customer"].append(customer)
     62 

AttributeError: 'NoneType' object has no attribute 'string'
```

After filling parsed launch records into the dictionary, convert to pd df.

In [38]:
```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame.from_dict(launch_dict)
df.head()
```

Out[38]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	'
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	Dec
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	2
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 C
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1

Export to CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```