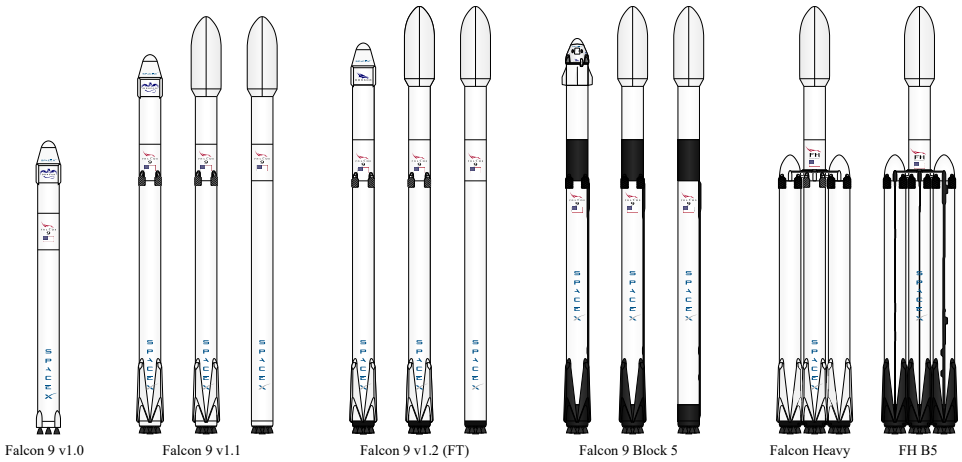
Space X Falcon 9 First Stage Landing Data Collection

Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia

Estimated time needed: **40** minutes

In this lab, you will be performing web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches`

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches



Falcon 9 first stage will land successfully

Several examples of an unsuccessful landing are shown here:

More specifically, the launch records are stored in a HTML table shown below:

2020 [edit]

In late 2019, Gwynne Shotwell stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,[490] in addition to 14 or 15 non-Starlink launches. At 26 launches, 13 of which for Starlink satellites, Falcon 9 had its most prolific year, and Falcon rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.[491]

[hide] Flight No.	Date and time (UTC)	Version, Booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[492]	F9 B5 △ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[493]								
79	19 January 2020, 15:30[494]	F9 B5 △ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[495] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[496]	NASA (CTS)[497]	Success	No attempt
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule,[498] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[419] The abort test used the capsule originally intended for the first crewed flight.[499] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[500] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.								
80	29 January 2020, 14:07[501]	F9 B5 △ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean.[502]								
81	17 February 2020, 15:05[503]	F9 B5 △ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed into a 212 km × 386 km (132 mi × 240 mi) elliptical orbit instead of launching into a circular orbit and firing the second stage engine twice. The first stage booster failed to land on the drone ship[504] due to incorrect wind data.[505] This was the first time a flight proven booster failed to land.								
82	7 March 2020, 04:50[506]	F9 B5 △ B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 △)	1,977 kg (4,359 lb)[507]	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
	Last launch of phase 1 of the CRS contract. Carries *Bartolomeo*, an ESA platform for hosting external payloads onto ISS.[508] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage engine failure. SpaceX decided to swap out the second stage instead of replacing the faulty part.[509] It was SpaceX's 50th successful landing of a first stage booster, the third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft.								
83	18 March 2020, 12:16[510]	F9 B5 △ B1048.5	KSC, LC-39A	Starlink 5 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fifth operational launch of Starlink satellites. It was the first time a first stage booster flew for a fifth time and the second time the fairings were reused (Starlink flight in May 2019).[511] Towards the end of the first stage burn, the booster suffered premature shut down of an engine, the first of a Merlin 1D variant and first since the CRS-1 mission in October 2012. However, the payload still reached the targeted orbit.[512] This was the second Starlink launch booster landing failure in a row, later revealed to be caused by residual cleaning fluid trapped inside a sensor.[513]								
84	22 April 2020, 19:30[514]	F9 B5 △ B1051.4	KSC, LC-39A	Starlink 6 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)

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

First let's import required packages for this lab

In [1]:
```
!pip3 install beautifulsoup4
!pip3 install requests
```

```
Requirement already satisfied: beautifulsoup4 in /home/jupyterlab/conda/envs/pyth
on/lib/python3.7/site-packages (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /home/jupyterlab/conda/envs/pytho
n/lib/python3.7/site-packages (from beautifulsoup4) (2.3.2.post1)
Requirement already satisfied: requests in /home/jupyterlab/conda/envs/python/li
b/python3.7/site-packages (2.29.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/jupyterlab/cond
a/envs/python/lib/python3.7/site-packages (from requests) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/envs/pytho
n/lib/python3.7/site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/conda/en
vs/python/lib/python3.7/site-packages (from requests) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/conda/envs/
python/lib/python3.7/site-packages (from requests) (2023.5.7)
```

In [2]:
```
import sys

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

and we will provide some helper functions for you to process web scraped HTML table

In [3]:
```
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_cells
    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

    #  if header:
    #     # Check if there are any <br> tags or <a> tags and remove them
    #     for tag in header.find_all(['br', 'a']):
    #         tag.extract()
    #     # Return the cleaned text of the header
    #     return header.get_text(strip=True)
    # return None
```

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Fa
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [10]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
```

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

In [11]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text con
soup = BeautifulSoup(response.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [12]:
```
# Use soup.title attribute
soup.title
```

Out[12]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [13]:
```
# Use the find_all function in the BeautifulSoup object, with element type `t
html_tables = soup.find_all('table')
# Assign the result to a list called `html_tables`
```

Starting from the third table is our target table contains the actual launch records.

In [17]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
# print(first_launch_table)
```

You should able to see the columns names embedded in the table header elements `<th>` as follows:

```
    <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 of 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="#cite_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>
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

In [19]:
```python
# Assuming you have the 'first_launch_table' variable containing the specific

# Initialize an empty list to store the column names
column_names = []

# Use find_all to locate all the <th> elements in the 'first_launch_table'
th_elements = first_launch_table.find_all('th')

# Iterate through each <th> element and extract the column name
for th in th_elements:
    column_name = extract_column_from_header(th)

    # Check if the extracted column name is not empty before appending it to t
    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)

# 'column_names' will now contain the extracted column names from the table he
```

Check the extracted column names

In [20]:
```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass',
'Orbit', 'Customer', 'Launch outcome']
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [ ]:
```
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']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

In [25]:
```
extracted_row = 0
launch_dict = []  # Initialize an empty list to store launch records
```

```
# Extract each table
for table_number, table in enumerate(soup.find_all('table', "wikitable plain
    # Get table rows
    for rows in table.find_all("tr"):
        # Check to see if the first table heading is a number corresponding
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
        else:
            flag = False
        # Get table elements
        row = rows.find_all('td')
        # If it is a number, save cells in a dictionary
        if flag:
            extracted_row += 1
            # Initialize a dictionary to store the launch record
            launch_record = {}

            # Flight Number value
            launch_record['Flight No.'] = flight_number

            datatimelist = date_time(row[0])
            # Date value
            launch_record['Date'] = datatimelist[0].strip(',')

            # Time value
            launch_record['Time'] = datatimelist[1]

            # Booster version
            bv = booster_version(row[1])
            if not (bv):
                bv = row[1].a.string
            launch_record['Version Booster'] = bv

            # Launch Site
            launch_record['Launch Site'] = row[2].a.string

            # Payload
            launch_record['Payload'] = row[3].a.string

            # Payload Mass
            launch_record['Payload mass'] = get_mass(row[4])

            # Orbit
            launch_record['Orbit'] = row[5].a.string

            # Customer
            # launch_record['Customer'] = row[6].a.string
            customer_tag = row[6].a  # Get the <a> tag if it exists
            if customer_tag:
                launch_record['Customer'] = customer_tag.string
            else:
                launch_record['Customer'] = row[6].text.strip()  # Use the c

            # Launch outcome
```

```
            launch_record['Launch outcome'] = list(row[7].strings)[0]

            # Booster landing
            launch_record['Booster landing'] = landing_status(row[8])

            # Append the launch record to the launch_dict list
            launch_dict.append(launch_record)

# Now, 'launch_dict' contains the extracted launch records from the table ro
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

In [30]:
```
df = pd.DataFrame(launch_dict)
df
```

Out[30]:

	Flight No.	Date	Time	Version Booster	Launch Site	Payload	Payload mass	Orbit	Cu
0	1	4 June 2010	18:45	F9 v1.0B0003.1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	
1	2	8 December 2010	15:43	F9 v1.0B0004.1	CCAFS	Dragon	0	LEO	
2	3	22 May 2012	07:44	F9 v1.0B0005.1	CCAFS	Dragon	525 kg	LEO	
3	4	8 October 2012	00:35	F9 v1.0B0006.1	CCAFS	SpaceX CRS-1	4,700 kg	LEO	
4	5	1 March 2013	15:10	F9 v1.0B0007.1	CCAFS	SpaceX CRS-2	4,877 kg	LEO	
...	...	...	...	...	...	...	...	...	
116	117	9 May 2021	06:42	F9 B5B1051.10	CCSFS	Starlink	15,600 kg	LEO	
117	118	15 May 2021	22:56	F9 B5B1058.8	KSC	Starlink	~14,000 kg	LEO	
118	119	26 May 2021	18:59	F9 B5B1063.2	CCSFS	Starlink	15,600 kg	LEO	
119	120	3 June 2021	17:29	F9 B5B1067.1	KSC	SpaceX CRS-22	3,328 kg	LEO	
120	121	6 June 2021	04:26	F9 B5	CCSFS	SXM-8	7,000 kg	GTO	S

121 rows × 11 columns

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

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

Author

Dev Agnihotri