Skills
Network

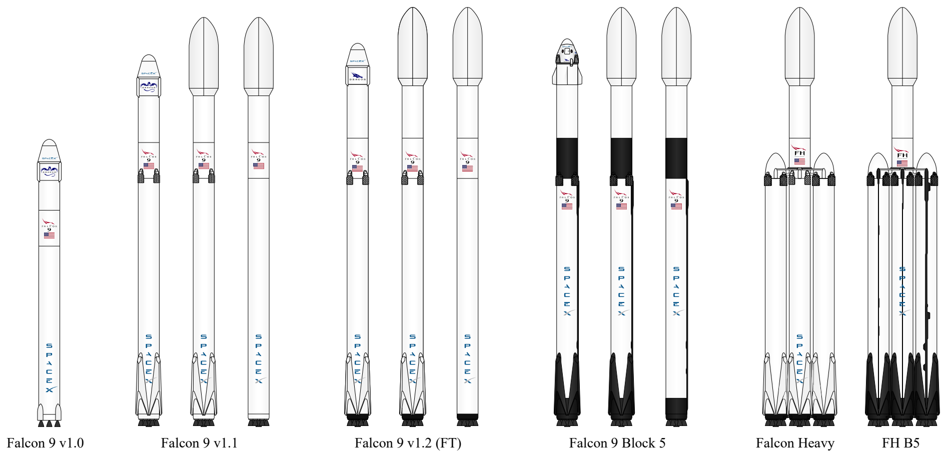
Space X Falcon 9 First Stage Landing Prediction

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`

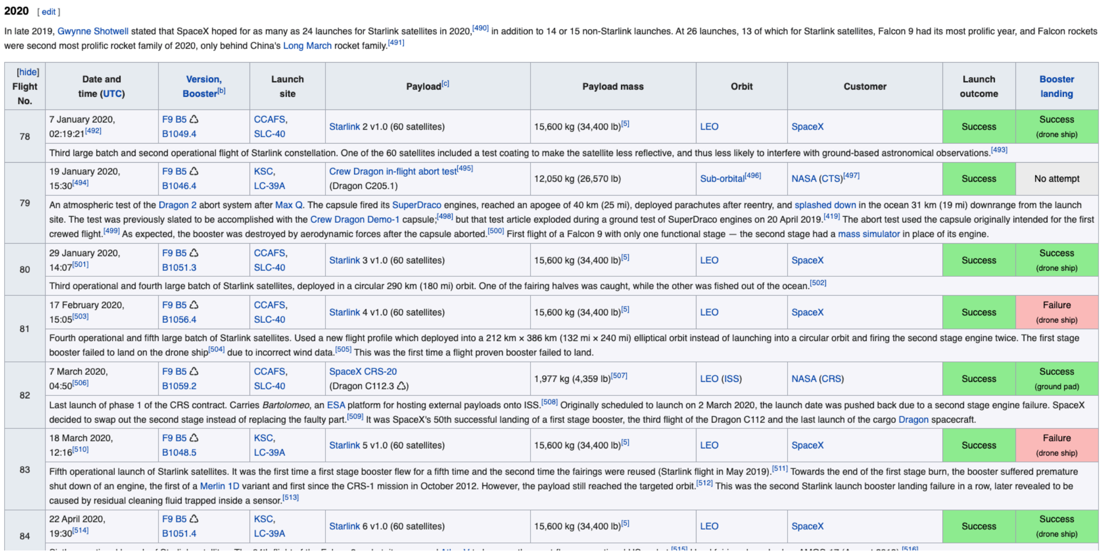
https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches



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]

<table>
<tr><th>[hide]
Flight No.</th><th>Date and time (UTC)</th><th>Version, Booster[b]</th><th>Launch site</th><th>Payload[c]</th><th>Payload mass</th><th>Orbit</th><th>Customer</th><th>Launch outcome</th><th>Booster landing</th></tr>
<tr><td rowspan="2">78</td><td>7 January 2020,
02:19:21[492]</td><td>F9 B5 ♺
B1049.4</td><td>CCAFS,
SLC-40</td><td>Starlink 2 v1.0 (60 satellites)</td><td>15,600 kg (34,400 lb)[5]</td><td>LEO</td><td>SpaceX</td><td>Success</td><td>Success
(drone ship)</td></tr>
<tr><td colspan="9">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]</td></tr>
<tr><td rowspan="2">79</td><td>19 January 2020,
15:30[494]</td><td>F9 B5 ♺
B1046.4</td><td>KSC,
LC-39A</td><td>Crew Dragon in-flight abort test[495]
(Dragon C205.1)</td><td>12,050 kg (26,570 lb)</td><td>Sub-orbital[496]</td><td>NASA (CTS)[497]</td><td>Success</td><td>No attempt</td></tr>
<tr><td colspan="9">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.</td></tr>
<tr><td rowspan="2">80</td><td>29 January 2020,
14:07[501]</td><td>F9 B5 ♺
B1051.3</td><td>CCAFS,
SLC-40</td><td>Starlink 3 v1.0 (60 satellites)</td><td>15,600 kg (34,400 lb)[5]</td><td>LEO</td><td>SpaceX</td><td>Success</td><td>Success
(drone ship)</td></tr>
<tr><td colspan="9">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]</td></tr>
<tr><td rowspan="2">81</td><td>17 February 2020,
15:05[503]</td><td>F9 B5 ♺
B1056.4</td><td>CCAFS,
SLC-40</td><td>Starlink 4 v1.0 (60 satellites)</td><td>15,600 kg (34,400 lb)[5]</td><td>LEO</td><td>SpaceX</td><td>Success</td><td>Failure
(drone ship)</td></tr>
<tr><td colspan="9">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.</td></tr>
<tr><td rowspan="2">82</td><td>7 March 2020,
04:50[506]</td><td>F9 B5 ♺
B1059.2</td><td>CCAFS,
SLC-40</td><td>SpaceX CRS-20
(Dragon C112.3 ♺)</td><td>1,977 kg (4,359 lb)[507]</td><td>LEO (ISS)</td><td>NASA (CRS)</td><td>Success</td><td>Success
(ground pad)</td></tr>
<tr><td colspan="9">Last launch of phase 1 of the CRS contract. Carries <i>Bartolomeo</i>, 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.</td></tr>
<tr><td rowspan="2">83</td><td>18 March 2020,
12:16[510]</td><td>F9 B5 ♺
B1048.5</td><td>KSC,
LC-39A</td><td>Starlink 5 v1.0 (60 satellites)</td><td>15,600 kg (34,400 lb)[5]</td><td>LEO</td><td>SpaceX</td><td>Success</td><td>Failure
(drone ship)</td></tr>
<tr><td colspan="9">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]</td></tr>
<tr><td>84</td><td>22 April 2020,
19:30[514]</td><td>F9 B5 ♺
B1051.4</td><td>KSC,
LC-39A</td><td>Starlink 6 v1.0 (60 satellites)</td><td>15,600 kg (34,400 lb)[5]</td><td>LEO</td><td>SpaceX</td><td>Success</td><td>Success
(drone ship)</td></tr>
</table>

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 [ ]:

```
!pip3 install beautifulsoup4
!pip3 install requests
```

In [ ]:

```
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 [ ]:

```
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
```

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 [ ]:

```
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 [2]:

```
# use requests.get() method with the provided static_url
# assign the response to a object

import requests

# URL to fetch the Falcon 9 Launch HTML page
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Fa

# Send a GET request to the URL
response = requests.get(static_url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    print("Request successful with status code 200")

    # Print the content of the response (HTML content)
    print(response.content[:500])  # Displaying first 500 characters of the HTML
else:
    print("Failed to retrieve data, status code:", response.status_code)
```

```
Request successful with status code 200
b'<!DOCTYPE html>\n<html class="client-nojs vector-feature-language-in-header-ena
bled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-h
eader-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinne
d-clientpref-1 vector-feature-main-menu-pinned-disabled vector-feature-limited-wi
dth-clientpref-1 vector-feature-limited-width-content-enabled vector-feature-cust
om-font-size-clientpref-1 vector-feature-appearance-enabled vector-feature-appear
ance-pinned-clien'
```

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

In [3]:

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text cont
import requests
from bs4 import BeautifulSoup

# URL to fetch the Falcon 9 Launch HTML page
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Fa

# Send a GET request to the URL
response = requests.get(static_url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    print("Request successful with status code 200")

    # Create a BeautifulSoup object from the HTML content
    soup = BeautifulSoup(response.content, 'html.parser')

    # Now you can work with the BeautifulSoup object
    # For example, print the title of the webpage
    print("Title of the webpage:", soup.title)
else:
    print("Failed to retrieve data, status code:", response.status_code)
```

```
Request successful with status code 200
Title of the webpage: <title>List of Falcon 9 and Falcon Heavy launches - Wikiped
ia</title>
```

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

In [10]:

```
import requests
from bs4 import BeautifulSoup

# URL to fetch the Falcon 9 Launch HTML page
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Fa

# Send a GET request to the URL
response = requests.get(static_url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    print("Request successful with status code 200")

    # Create a BeautifulSoup object from the HTML content
    soup = BeautifulSoup(response.content, 'html.parser')

    # Print the title of the webpage
    print("Page Title:", soup.title.text)
else:
    print("Failed to retrieve data, status code:", response.status_code)
```

```
Request successful with status code 200
Page Title: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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 [11]:

```python
# Use the find_all function in the BeautifulSoup object, with element type `tabl
# Assign the result to a list called `html_tables`
import requests
from bs4 import BeautifulSoup

# URL of the webpage with HTML tables
wiki_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falc

# Send a GET request to the URL
response = requests.get(wiki_url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    print("Request successful with status code 200")

    # Create a BeautifulSoup object from the HTML content
    soup = BeautifulSoup(response.content, 'html.parser')

    # Find all tables on the webpage
    html_tables = soup.find_all('table')

    # Print the number of tables found
    print(f"Number of tables found: {len(html_tables)}")

    # Print the first few characters of the first table to verify
    if html_tables:
        print("First few characters of the first table:")
        print(html_tables[0].prettify()[:200])  # Displaying first 200 character
    else:
        print("No tables found on the webpage")
else:
    print("Failed to retrieve data, status code:", response.status_code)
```

```
Request successful with status code 200
Number of tables found: 25
First few characters of the first table:
<table class="col-begin" role="presentation">
 <tbody>
  <tr>
   <td class="col-break">
    <h3>
     <span class="mw-headline" id="Rocket_configurations">
      Rocket configurations
     </span>
```

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

In [ ]:

```python
# 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 [ ]:

```
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
# Append the Non-empty column name (`if name is not None and len(name) > 0`) int
```

Check the extracted column names

In [ ]:

```
print(column_names)
```

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 [ ]:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowhea
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launc
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight N
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            #print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            #print(payload)

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload m
            payload_mass = get_mass(row[4])
            #print(payload)

            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            #print(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].a.string
            #print(customer)

            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch
            launch_outcome = list(row[7].strings)[0]
            #print(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster
            booster_landing = landing_status(row[8])
            #print(booster_landing)
```

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

In [ ]:

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

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.

Following labs will be using a provided dataset to make each lab independent.

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

Authors

Yan Luo

Nayef Abou Tayoun

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-06-09	1.0	Yan Luo	Tasks updates
2020-11-10	1.0	Nayef	Created the initial version