Skills Network

(https://skills.network/?
utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1000
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2022-01-01)

Sricharan Nallan Chakravarthula

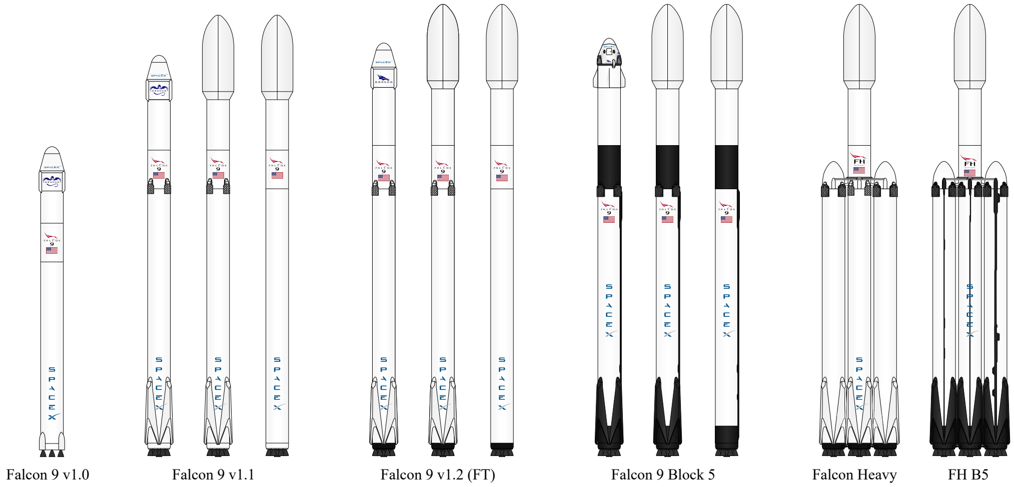
Space X Falcon 9 First Stage Landing Prediction

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`

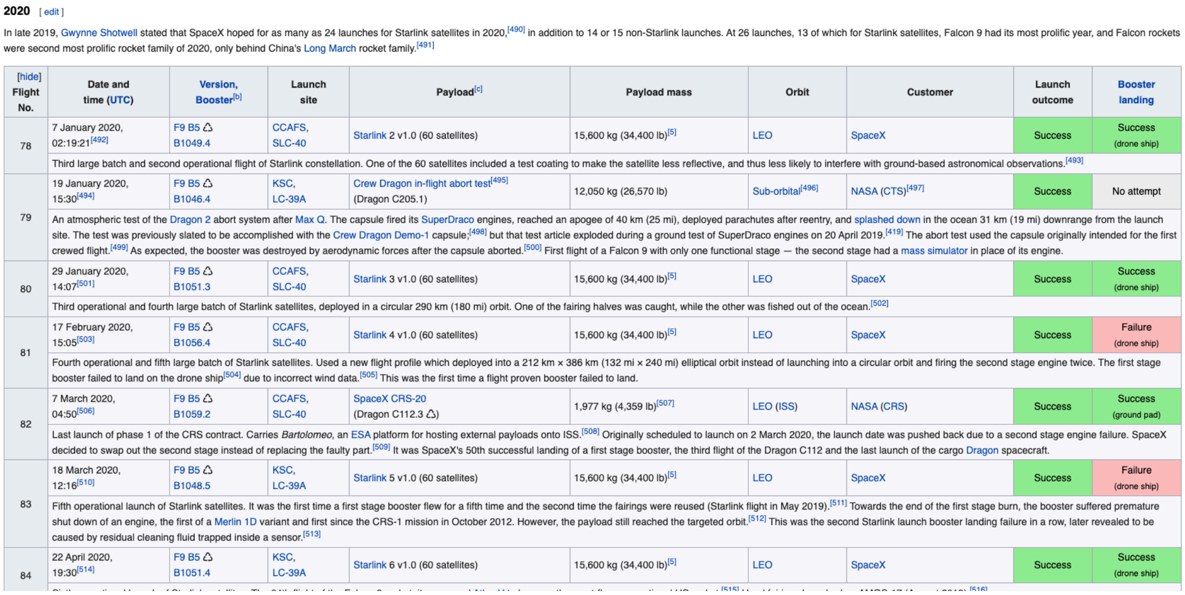
https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
(https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches?
utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1000
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2022-01-01)



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

- Use BeautifulSoup to extract/web scrape Falcon 9 launch records from an HTML table on Wikipedia
- Parse the table and convert it into a Pandas data frame

Import Libraries

In [1]:

```
!pip3 install beautifulsoup4
!pip3 install requests
print("Installation complete.")
```

```
Requirement already satisfied: beautifulsoup4 in /home/jupyterlab/conda/e
nvs/python/lib/python3.7/site-packages (4.11.1)
Requirement already satisfied: soupsieve>1.2 in /home/jupyterlab/conda/en
vs/python/lib/python3.7/site-packages (from beautifulsoup4) (2.3.2.post1)
Requirement already satisfied: requests in /home/jupyterlab/conda/envs/py
thon/lib/python3.7/site-packages (2.28.1)
Requirement already satisfied: charset-normalizer<3,>=2 in /home/jupyterl
ab/conda/envs/python/lib/python3.7/site-packages (from requests) (2.1.1)
Requirement already satisfied: certifi>=2017.4.17 in /home/jupyterlab/con
da/envs/python/lib/python3.7/site-packages (from requests) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /home/jupyterlab/
conda/envs/python/lib/python3.7/site-packages (from requests) (1.26.13)
Requirement already satisfied: idna<4,>=2.5 in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from requests) (3.4)
Installation complete.
```

In [2]:

```
import sys
import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
print("All libraries have been imported.")
```

```
All libraries have been imported.
```

Define Useful Functions

> **Note**
>
> These code bits were provided.

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)][

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_
    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
```

Start Here

Task
1

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.

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]:
```
# We use the requests.get() method with the provided static_url and assign
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_a
response = requests.get(static_url).text
```

In [5]:
```
# Confirm that the response contains HTML markup
response[0:100]
```

Out[5]:
```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>
\n<meta charset="UTF-8"/>\n<title'
```

In [6]:

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response tex
soup = BeautifulSoup(response, "html.parser")
print(soup.prettify()[0:500], '\n==============\n', soup.prettify()[5000:5
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   List of Falcon 9 and Falcon Heavy launches - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFrame
s":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","Februar
y","March","April","May","June","July","August","September","October","No
vember","December"],"wgRequ
==============
 ;skin=vector">
  </script>
  <meta content="" name="ResourceLoaderDynamicStyles"/>
  <link href="/w/load.php?lang=en&amp;modules=site.styles&amp;only=styles
&amp;skin=vector" rel="stylesheet"/>
  <meta content="MediaWiki 1.40.0-wmf.17" name="generator"/>
  <meta content="origin" name="referrer"/>
  <meta content="origin-when-crossorigin" name="referrer"/>
  <meta content="origin-when-cross-origin" name="referrer"/>
  <meta content="noindex,nofollow,max-image-preview:standard" name="robot
s"/>
  <m
```

In [7]:

```
# Find the title using .title attribute
soup.title
```

Out[7]:

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

In [8]:

```
# Find the title using .find_all()
title_soup = soup.find_all('title')
title_soup
```

Out[8]:

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

Task
2

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

In [9]:

```
# Use the find_all function in the BeautifulSoup object, with element type
html_tables = soup.find_all('table')
```

In [10]:

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(str(first_launch_table)[0:500])
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 10
0%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_T
ime" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" tit
le="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
```

In [11]:

```
# Count the number of <th> elements in 'first_launch_table' using .find_al
th_elements = first_launch_table.find_all('th')
len(th_elements)
```

Out[11]:

```
17
```

In [12]:

```
# Print the 'th' elements from the table
for count,i in enumerate(th_elements):
    print(extract_column_from_header(i))
```

```
Flight No.
Date and time ( )

Launch site
Payload
Payload mass
Orbit
Customer
Launch outcome

None
None
None
None
None
None
None
```

In [13]:

```python
# Iterate through each th element and apply the provided extract_column_fr
# Append the Non-empty column name (`if name is not None and len(name) > 0
column_names = []

for count,i in enumerate(th_elements):
    extract = extract_column_from_header(i)
    if extract is not None and len(extract) > 0:
        print(count, extract)
        column_names.append(extract)
    else:
        print(f'{count} {extract} --- --- --- [Discarded]')

column_names
```

```
0 Flight No.
1 Date and time ( )
2  --- --- --- [Discarded]
3 Launch site
4 Payload
5 Payload mass
6 Orbit
7 Customer
8 Launch outcome
9  --- --- --- [Discarded]
10 None --- --- --- [Discarded]
11 None --- --- --- [Discarded]
12 None --- --- --- [Discarded]
13 None --- --- --- [Discarded]
14 None --- --- --- [Discarded]
15 None --- --- --- [Discarded]
16 None --- --- --- [Discarded]
```

Out[13]:

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

In [14]:

```python
# Check the extracted column names
print(column_names)
```

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

Task
3

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

In [15]:

```
# Create dictionary from keys from the extracted column names.
launch_dict = dict.fromkeys(column_names)
print(f'All Keys:\n{launch_dict} \n')

# Remove an irrelvant column
del launch_dict['Date and time ( )']
print(f'Removed \'Date and time ( )\':\n{launch_dict} \n')

# Initialize launch_dict with empty lists
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 new columns
launch_dict['Version Booster'] = []
launch_dict['Booster landing'] = []
launch_dict['Date'] = []
launch_dict['Time'] = []

print(f'Full Launch Dictionary:\n{launch_dict}')
```

```
All Keys:
{'Flight No.': None, 'Date and time ( )': None, 'Launch site': None, 'Pay
load': None, 'Payload mass': None, 'Orbit': None, 'Customer': None, 'Laun
ch outcome': None}

Removed 'Date and time ( )':
{'Flight No.': None, 'Launch site': None, 'Payload': None, 'Payload mas
s': None, 'Orbit': None, 'Customer': None, 'Launch outcome': None}

Full Launch Dictionary:
{'Flight No.': [], 'Launch site': [], 'Payload': [], 'Payload mass': [],
'Orbit': [], 'Customer': [], 'Launch outcome': [], 'Version Booster': [],
'Booster landing': [], 'Date': [], 'Time': []}
```

In [16]:

```python
# Extract the tables
extracted_row = 0

# Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plain

    # Get table row
    for rows in table.find_all("tr"):

        # (1) ============================================================
        # Check to see if first table heading is as number corresponding t
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
        else:
            flag = False

        # (2) ============================================================
        # Get table element
        row = rows.find_all('td')

        # (3) ============================================================
        # If it is number save cells in a dictonary
        if flag:
            extracted_row += 1

            # (1) Flight Number value |=========| Append the flight_number
            datatimelist = date_time(row[0])
            launch_dict['Flight No.']+=[flight_number]

            # (10) Date value |=========| Append the date into launch_dict
            date = datatimelist[0].strip(',')
            launch_dict['Date']+=[date]

            # (11) Time value |=========| Append the time into launch_dict
            time = datatimelist[1]
            launch_dict['Time']+=[time]

            # (8) Booster version |=========| Append the bv into launch_di
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster']+=[bv]

            # (2) Launch Site |=========| Append the bv into launch_dict w
            launch_site = row[2].a.string
            launch_dict['Launch site']+=[launch_site]

            # (3) Payload |=========| Append the payload into launch_dict i
            payload = row[3].a.string
            launch_dict['Payload']+=[payload]

            # (4) Payload Mass |=========| Append the payload_mass into La
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass']+=[payload_mass]
            # (5) Orbit |=========| Append the orbit into launch_dict with
            orbit = row[5].a.string
            launch_dict['Orbit']+=[orbit]

            # (6) *** Customer |=========| Append the customer into launch_
            if str(row[6])[0:7] == '<td>Var':
                print('HERE !!', row[6])
                customer = 'Various'
                print(f'Customer Name: {customer}')
            elif str(row[6])[0:26] == '<td><a href="/wiki/Turkmen':
                print('HERE !!', row[6])
                customer = 'Turkmenistan National Space Agency'
                print(f'Customer Name: {customer}')
            else:
                print('>>>Test', row[6].a.string)
                customer = row[6].a.string
            launch_dict['Customer']+=[customer]

            # (7) *** Launch outcome |=========| Append the launch_outcome
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome']+=[launch_outcome]

            # (9) *** Booster landing |=========| Append the launch_outcom
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing']+=[booster_landing]
```

```
>>>Test Sirius XM
>>>Test NRO
>>>Test Türksat
>>>Test SpaceX
HERE !! <td>Various
</td>
Customer Name: Various
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test NASA
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test SpaceX
>>>Test NASA
```

In [17]:

```python
# Count the number of data points in each column as a check for any errors
for count,i in enumerate(launch_dict):
    column_name = list(launch_dict.keys())[count]
    length_column = str(len(launch_dict[column_name]))
    print(f'{column_name}: {length_column}')
```

```
Flight No.: 121
Launch site: 121
Payload: 121
Payload mass: 121
Orbit: 121
Customer: 121
Launch outcome: 121
Version Booster: 121
Booster landing: 121
Date: 121
Time: 121
```

In [18]:

```python
# Get the keys of the dictionary in a list
list(launch_dict.keys())
```

Out[18]:

```
['Flight No.',
 'Launch site',
 'Payload',
 'Payload mass',
 'Orbit',
 'Customer',
 'Launch outcome',
 'Version Booster',
 'Booster landing',
 'Date',
 'Time']
```

In [19]:

```
# Convert dictionary to DataFrame
df = pd.DataFrame(launch_dict)
df
```

Out[19]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booste landin
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failur
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failur
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	N attempt\
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	N attemp
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	N attempt\
...	...	...	...	...	...	...	...	...	.
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Succes
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Succes
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Succes
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Succes
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Succes

121 rows × 11 columns

Export DataFrame to .CSV

Note

spacex_web_scraped.csv

In [20]:

```
# Export DataFrame as .csv
df.to_csv('spacex_web_scraped.csv', index=False)
```

End Here