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

```
Requirement already satisfied: beautifulsoup4 in c:\users\hp\anaconda3\lib\site
-packages (4.10.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\hp\anaconda3\lib\site-
packages (from beautifulsoup4) (2.2.1)
Requirement already satisfied: requests in c:\users\hp\anaconda3\lib\site-packa
ges (2.26.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\anaconda3\lib
\site-packages (from requests) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\anaconda3\lib\site-p
ackages (from requests) (3.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\hp\anaconda3\l
ib\site-packages (from requests) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\hp\anacond
a3\lib\site-packages (from requests) (2.0.4)
```

In [2]:
```
import sys

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

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
```

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Fal
```

In [5]:
```
# using requests.get() method with the provided static_url
# assigning the response to a object
response = requests.get(static_url).text
```

In [6]:
```
# using BeautifulSoup() to create a BeautifulSoup object from a response text cor
soup = BeautifulSoup(response, 'html.parser')
```

In [7]:
```
# using soup.title attribute
print(soup.title)
```

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

In [8]:
```
# Using the find_all function in the BeautifulSoup object, with element type `tab
# Assigning the result to a list called `html_tables`
html_tables = soup.find_all("table")
print(html_tables)
```

```
[<table class="multicol" role="presentation" style="border-collapse: collaps
e; padding: 0; border: 0; background:transparent; width:100%;">
<tbody><tr>
<td style="text-align: left; vertical-align: top;">
<h3><span class="mw-headline" id="Rocket_configurations">Rocket configuration
s</span></h3>
<div class="chart noresize" style="margin-top:1em;max-width:420px;">
<div style="position:relative;min-height:320px;min-width:420px;max-width:420p
x;">
<div style="float:right;position:relative;min-height:240px;min-width:320px;ma
x-width:320px;border-left:1px black solid;border-bottom:1px black solid;">
<div style="position:absolute;left:3px;top:224px;height:15px;min-width:18px;m
ax-width:18px;background-color:LightSteelBlue;-webkit-print-color-adjust:exac
t;border:1px solid LightSteelBlue;border-bottom:none;overflow:hidden;" title
="[[Falcon 9 v1.0]]: 2"></div>
<div style="position:absolute;left:55px;top:224px;height:15px;min-width:18px;
max-width:18px;background-color:LightSteelBlue;-webkit-print-color-adjust:exa
ct;border:1px solid LightSteelBlue;border-bottom:none;overflow:hidden;" title
="[[Falcon 9 v1.0]]: 2"></div>
```

In [9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><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 hr
ef="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
```

In [10]:
```
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`) into
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
```

In [11]:
```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass',
'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Laun
ch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Fl
ight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbi
t', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch si
te', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'N/A',
'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'O
rbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch
site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Fligh
t 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', 'Customer', 'Launch outc
ome', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit',
'Customer', 'Launch outcome', 'Date and time ( )', 'Launch site', 'Payload', 'O
rbit', 'Customer', 'Date and time ( )', 'Launch site', 'Payload', 'Orbit', 'Cus
tomer', 'Date and time ( )', 'Launch site', 'Payload', 'Orbit', 'Customer', 'Da
te and time ( )', 'Launch site', 'Payload', 'Orbit', 'Customer', 'Demo flight
s', 'logistics', 'Crewed missions', 'Commercial satellites', 'Scientific satell
ites', 'Military satellites', 'Rideshares', 'Current', 'In development', 'Retir
ed', 'Cancelled', 'Spacecraft', 'Cargo', 'Crewed', 'Test vehicles', 'Current',
'Retired', 'Unflown', 'Orbital', 'Atmospheric', 'Landing sites', 'Other facilit
ies', 'Support', 'Contracts', 'R&D programs', 'Key people', 'Related', 'Genera
l', 'General', 'People', 'Vehicles', 'Launches by rocket type', 'Launches by sp
aceport', 'Agencies, companies and facilities', 'Other mission lists and timeli
nes']
```

In [12]:
```
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']=[]
```

In [13]:
```
xtracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowhead
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch
        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

            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.strings
            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)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_8528/2117755612.py in <module>
     15         #if it is number save cells in a dictonary
     16         if flag:
---> 17             extracted_row += 1
     18             # Flight Number value
     19

NameError: name 'extracted_row' is not defined
```

In [14]:
```
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[14]:

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

In [15]:
```
df.to_csv('spacex_web_scraped.csv', index=False)
```