Space X Falcon 9 First Stage Landing Prediction

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

```
!pip3 install beautifulsoup4
!pip3 install requests

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

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.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

static_url = "https://en.wikipedia.org/w/index.php?
title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

# use requests.get() method with the provided static_url
# assign the response to a object
r = requests.get(static_url)
data = r.text

# Use BeautifulSoup() to create a BeautifulSoup object from a response
text content
soup = BeautifulSoup(data,"html.parser")

print(soup.title)
```

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

```
# Use the find_all function in the BeautifulSoup object, with element
type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')

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

column_names = []
```

```python
# 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`) into a list called column_names
table_headers = first_launch_table.find_all('th')
# print(table_headers)
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)

print(column_names)
```

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

```python
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']=[]

extracted_row = 0
for table_number,table in enumerate(soup.find_all('table',"wikitable
plainrowheaders collapsible")):
    for rows in table.find_all("tr"):  # get table row
        if rows.th:  #check to see if first table heading is a number
corresponding to launch a number
            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 flag:  #if it is number save cells in a dictonary
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key
`Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            # print(flight_number)
            datatimelist=date_time(row[0])

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

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            # 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
            launch_dict['Version Booster'].append(bv)
            # print(bv)
            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch
Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            # print(launch_site)

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

            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key
`Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            # print(payload)
            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            # print(orbit)

            # Customer
            # TODO: Append the customer into launch_dict with key
`Customer`
            try:
                customer = row[6].a.string
            except:
                customer = "None"

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

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

            # debugging: find length differences
            # print("---------------------------")
            # curr_len = [len(val) for key, val in
launch_dict.items()]
            # print(curr_len)
            # print("---------------------------")
print("number of extracted rows: ", extracted_row)
```

```
number of extracted rows:  121
```

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

```
# debugging: checking length of lists in dictionary
for key, val in launch_dict.items():
    print(f"{key}: #: {len(val)}")
```

```
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

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

df_scraped.head()

  Flight No. Launch site                               Payload Payload
mass  \
0          1       CCAFS  Dragon Spacecraft Qualification Unit
0
1          2       CCAFS                                Dragon
0
2          3       CCAFS                                Dragon
525 kg
3          4       CCAFS                          SpaceX CRS-1
4,700 kg
4          5       CCAFS                          SpaceX CRS-2
4,877 kg

  Orbit Customer Launch outcome   Version Booster Booster landing  \
0   LEO   SpaceX      Success\n  F9 v1.07B0003.18         Failure
1   LEO     NASA        Success  F9 v1.07B0004.18         Failure
2   LEO     NASA        Success  F9 v1.07B0005.18    No attempt\n
3   LEO     NASA      Success\n  F9 v1.07B0006.18      No attempt
4   LEO     NASA      Success\n  F9 v1.07B0007.18    No attempt\n

              Date   Time
0      4 June 2010  18:45
1  8 December 2010  15:43
2      22 May 2012  07:44
3   8 October 2012  00:35
4     1 March 2013  15:10

df_scraped.tail()

    Flight No. Launch site        Payload Payload mass Orbit
Customer  \
116         117       CCSFS       Starlink    15,600 kg   LEO
SpaceX
117         118         KSC       Starlink   ~14,000 kg   LEO
SpaceX
118         119       CCSFS       Starlink    15,600 kg   LEO
SpaceX
119         120         KSC  SpaceX CRS-22     3,328 kg   LEO
NASA
120         121       CCSFS          SXM-8     7,000 kg   GTO  Sirius
```

```
XM

    Launch outcome   Version Booster Booster landing         Date
Time
116       Success\n  F9 B5B1051.10657         Success   9 May 2021
06:42
117       Success\n   F9 B5B1058.8660         Success  15 May 2021
22:56
118       Success\n   F9 B5B1063.2665         Success  26 May 2021
18:59
119       Success\n   F9 B5B1067.1668         Success  3 June 2021
17:29
120       Success\n             F9 B5         Success  6 June 2021
04:26
```