AbhijitMandal_DSC540_Week5-6Ex

April 23, 2021

0.0.1 DSC 540 Week 5-6

Abhijit Mandal

0.0.2 Activity 7: Reading Tabular Data from a Web Page and Creating DataFrames

In this activity, you have been given a Wikipedia page where you have the GDP of all countries listed. You have been asked to create three DataFrames from the three sources mentioned in the page (https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)):

- Open the page in a separate Chrome/Firefox tab and use something like an Inspect Element tool to view the source HTML and understand its structure Read the page using bs4
- Read the page using bs4
- Find the table structure you will need to deal with (how many tables there are?)
- Find the right table using bs4
- Separate the source names and their corresponding data
- Get the source names from the list of sources you have created
- Separate the header and data from the data that you separated before for the first source only, and then create a DataFrame using that
- Repeat the last task for the other two data sources

0.0.3 Load the necessary libraries.

[132]:
```
# We are importing BeautifulSoup for web data scrapping
# urllib to open and read website html

from bs4 import BeautifulSoup
import urllib.request
import pandas as pd
```

0.0.4 Reading the wikipedia website HTML using beautifulSoup library

[133]:
```
website = "https://en.wikipedia.org/wiki/List_of_countries_by_GDP_(nominal)"
websiteData = urllib.request.urlopen(website)
bfData = BeautifulSoup(websiteData)
websiteData.close()
```

0.0.5 Reading Html table structure to find how many tables are there in the HTML

[134]:
```
no_of_tables = bfData.find_all("table")
print("There are " + str(len(no_of_tables)) + " tables in the website HTML")
```

```
There are 10 tables in the website HTML
```

0.0.6 Find the right table using bs4

[135]:
```
# finding table with class wikitable inside HTML, this table holds the data we␣
↪are looking for
tabledata = bfData.find("table", {"class": "wikitable"})
print(type(tabledata))
```

```
<class 'bs4.element.Tag'>
```

0.0.7 Separate the source names and their corresponding data

[136]:
```
# finding all rows inside table body non recusively as we are interested in␣
↪getting only the top level rows
scRow = tabledata.tbody.findAll('tr', recursive=False)[0]

# finding all table cells inside the table rows non recursively to get only the␣
↪heder cells (each cell represents a source)
scCell = [td for td in scRow.findAll('td')]

# iterating through the anchor tags in each cell to get the names
datasources = [item.findAll('a')[0].getText() for item in scCell]

print(datasources)
```

```
['International Monetary Fund', 'World Bank', 'United Nations']
```

0.0.8 Separate the header and data from the data that you separated before for the first source only, and then create a DataFrame using that

[137]:
```
# Method to read all tables and print the data with header iteratively
def printtable(data_tables):
    count = len(data_tables)
    for i in range(count):
        #getting header for the table
        tableheader = [th.getText().strip() for th in data_tables[i][0].
↪findAll('th')]

        #Geting data for the table
        rows = data_tables[i][0].findAll('tbody')[0].findAll('tr')[1:]
        data_rows =[[td.get_text().strip() for td in tr.findAll('td')] for tr␣
↪in rows]
```

```
        tableDF = pd.DataFrame(data_rows, columns=tableheader)
        display(tableDF)
```

[138]:
```
# get all rows and cells from table body
data = tabledata.tbody.findAll('tr', recursive=False)[1].findAll('td',␣
 ↪recursive=False)
data_tables = []

#iterate through the cells and find tables inside them, these are the table we␣
 ↪are looking for
for td in data:
    data_tables.append(td.findAll('table'))

#print(data_tables[0][0])

# Getting headers of first table
table1Header = [th.getText().strip() for th in data_tables[0][0].findAll('th')]
print(table1Header)
```

```
['Rank', 'Country/Territory', 'GDP(US$million)']
```

[139]:
```
#Geting data for the first table
rows1 = data_tables[0][0].findAll('tbody')[0].findAll('tr')[1:]
data_rows1 = [[td.get_text().strip() for td in tr.findAll('td')] for tr in␣
 ↪rows1]
df1 = pd.DataFrame(data_rows1, columns=table1Header)
df1.head()
```

[139]:
```
  Rank Country/Territory GDP(US$million)
0              World[21]      93,889,577
1    1     United States      22,675,271
2    2         China[n 2]      16,642,318
3    3             Japan       5,378,136
4    4           Germany       4,319,286
```

[140]:
```
# Iteratively fetching all the tables and printing their data
printtable(data_tables)
```

```
    Rank Country/Territory GDP(US$million)
0                  World[21]      93,889,577
1       1     United States      22,675,271
2       2         China[n 2]      16,642,318
3       3             Japan       5,378,136
4       4           Germany       4,319,286
..    …                 …               …
191   189  Marshall Islands             234
192   190          Kiribati             231
```

```
193  191             Palau             229
194  192             Nauru             133
195  193            Tuvalu              57

[196 rows x 3 columns]
    Rank        Country/Territory GDP(US$million)
0                           World      87,813,420
1      1            United States      21,433,226
2      2               China[n 9]      14,342,903
3      3                    Japan       5,081,770
4      4                  Germany       3,861,124
..   …                          …               …
186  186                    Palau             268
187  187  Marshall Islands (2018)             221
188  188                 Kiribati             195
189  189                    Nauru             118
190  190                   Tuvalu              47

[191 rows x 3 columns]
    Rank            Country/Territory GDP(US$million)
0                           World[22]      87,461,674
1      1                United States      21,433,226
2      2                   China[n 9]      14,342,933
3      3                        Japan       5,082,465
4      4                      Germany       3,861,123
..   …                              …               …
208  209             Marshall Islands             237
209  210                     Kiribati             194
210  211                        Nauru             132
211  212  Montserrat (United Kingdom)              67
212  213                       Tuvalu              47

[213 rows x 3 columns]
```

0.0.9 Activity 8: Handling Outliers and Missing Data

- Read the visit_data.csv file.
- Check for duplicates.
- Check if any essential column contains NaN.
- Get rid of the outliers.
- Report the size difference.
- Create a box plot to check for outliers.
- Get rid of any outliers.

[141]:
```
#load libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

[142]:
```
#read the file
dataDF = pd.read_csv("../visit_data.csv")
dataDF.head()
```

[142]:
```
   id first_name last_name                       email gender  \
0   1      Sonny      Dahl           sdahl0@mysql.com   Male
1   2        NaN       NaN           dhoovart1@hud.gov    NaN
2   3        Gar     Armal      garmal2@technorati.com    NaN
3   4    Chiarra     Nulty       cnulty3@newyorker.com    NaN
4   5        NaN       NaN  sleaver4@elegantthemes.com    NaN

        ip_address   visit
0    135.36.96.183  1225.0
1  237.165.194.143   919.0
2   166.43.137.224   271.0
3   139.98.137.108  1002.0
4    46.117.117.27  2434.0
```

[143]:
```
# Checking for duplicates by email as email id is the unique value in the data
print("Does data has duplicate Email Ids : {}".format(any(dataDF.email.
 ↪duplicated())))
```

```
Does data has duplicate Email Ids : False
```

[144]:
```
# Checking for Nan Values in dataframe, First name last name can contain null,␣
 ↪checking for email, gender, ip address and visit
print("IP Address contains NaN - {}".format(dataDF.ip_address.isnull().values.
 ↪any()))
print("Visit contains NaN - {}".format(dataDF.visit.isnull().values.any()))
print("Email contains NaN - {}".format(dataDF.ip_address.isnull().values.any()))
print("Gender contains NaN - {}".format(dataDF.visit.isnull().values.any()))
```

```
IP Address contains NaN - False
Visit contains NaN - True
Email contains NaN - False
Gender contains NaN - True
```

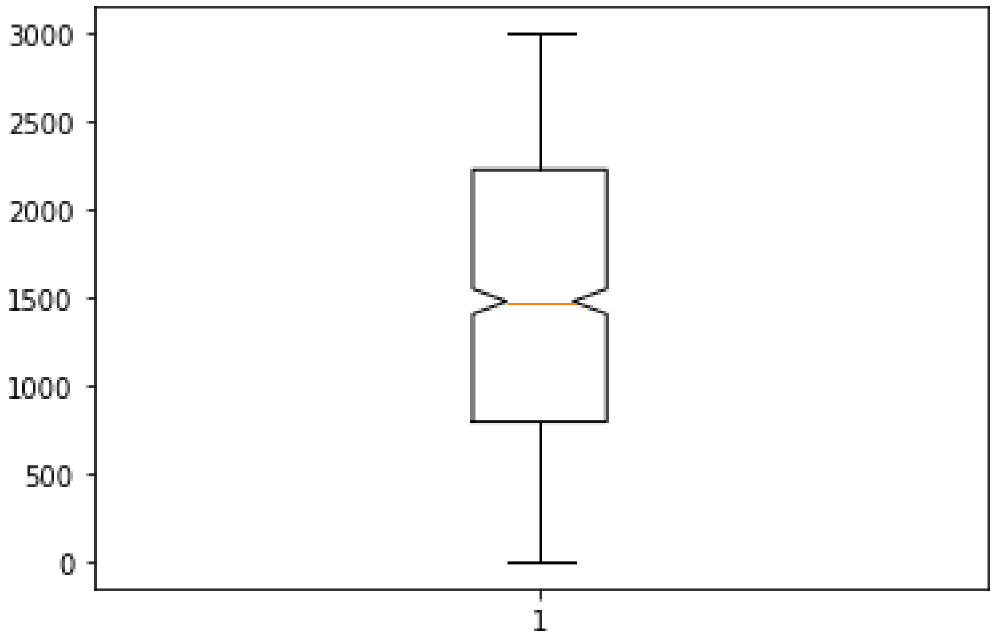
[145]:
```
# Getting rid of outliers and reporting the size difference
sizeWithOutliers = len(dataDF)
print("Size before removing outliers : " + str(sizeWithOutliers))

cleanDataDF = dataDF[dataDF.visit.notnull()]
print("Size after removing outliers : " + str(len(cleanDataDF)))
```

```
Size before removing outliers : 1000
Size after removing outliers : 974
```

[146]:
```
# plotting box plot to check any other outliers
plt.boxplot(cleanDataDF.visit, notch=True)
```

[146]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7fee988e50a0>,
  <matplotlib.lines.Line2D at 0x7fee988e5400>],
 'caps': [<matplotlib.lines.Line2D at 0x7fee988e5760>,
  <matplotlib.lines.Line2D at 0x7fee988e5ac0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7fee988d7d00>],
 'medians': [<matplotlib.lines.Line2D at 0x7fee988e5e20>],
 'fliers': [<matplotlib.lines.Line2D at 0x7fee988ee1c0>],
 'means': []}
```



The range of data is from 0- 3000 with major concentration between 750 to 2250. We can look to get rid of any values which are far away from the major concentration

[147]:
```
cleanDataDF = cleanDataDF[(cleanDataDF['visit'] <= 2800) &␣
 ↪(cleanDataDF['visit'] >= 200)]
print("Data size after removing Nan and outliers is : " + str(len(cleanDataDF)))
```

```
Data size after removing Nan and outliers is : 864
```

0.0.10 Insert data into a SQL Lite database – create a table with the following data

a. Name, Address, City, State, Zip, Phone Number

b. Add at least 10 rows of data and submit your code with a query generating your results.

[148]:
```
# importing sqlite3 library
import sqlite3
```

[149]:
```
# making a connection to sql lite db
con = sqlite3.connect('mydata.sqlite')
#defining DDL query
query = "CREATE TABLE UserData (Name VARCHAR(50), Address VARCHAR(500),City␣
↪VARCHAR(50), State VARCHAR(50),PhoneNumber VARCHAR(50),Zip INTEGER);"

#execute the query to create the table
con.execute(query)
con.commit()
```

[150]:
```
# Add data into table
data = [('James Butt', '6649 N Blue Gum St', 'New Orleans', 'LA', 70116,␣
↪'504-621-8927'),
        ('Josephine Darakjy','4 B Blue Ridge␣
↪Blvd','Brighton','MI',48116,'810-292-9388'),
        ('Art Venere','8 W Cerritos Ave␣
↪#54','Bridgeport','NJ',8014,'856-636-8749'),
        ('Lenna Paprocki','639 Main St','Anchorage','AK',99501,'907-385-4412'),
        ('Donette Foller','34 Center St','Hamilton','OH',45011,'513-570-1893'),
        ('Simona Morasca','3 Mcauley Dr','Ashland','OH',44805,'419-503-2484'),
        ('Mitsue Tollner','7 Eads St','Chicago','IL',60632,'773-573-6914'),
        ('Leota Dilliard','7 W Jackson Blvd','San␣
↪Jose','CA',95111,'408-752-3500'),
        ('Sage Wieser','5 Boston Ave #88','Sioux␣
↪Falls','SD',57105,'605-414-2147'),
        ('Kris Marrier','228 Runamuck Pl␣
↪#2808','Baltimore','MD',21224,'410-655-8723')]

stmt = "INSERT INTO UserData VALUES(?, ?, ?, ?, ?, ?)"
con.executemany(stmt, data)
con.commit()
```

[151]:
```
# Try reading the data from Sql lite table to verify if the data got inserted
cursor = con.execute('select * from UserData')
rows = cursor.fetchall()
rows
```

[151]:
```
[('James Butt',
  '6649 N Blue Gum St',
  'New Orleans',
  'LA',
  '70116',
  '504-621-8927'),
 ('Josephine Darakjy',
  '4 B Blue Ridge Blvd',
  'Brighton',
  'MI',
  '48116',
  '810-292-9388'),
 ('Art Venere',
  '8 W Cerritos Ave #54',
  'Bridgeport',
  'NJ',
  '8014',
  '856-636-8749'),
 ('Lenna Paprocki', '639 Main St', 'Anchorage', 'AK', '99501', '907-385-4412'),
 ('Donette Foller', '34 Center St', 'Hamilton', 'OH', '45011', '513-570-1893'),
 ('Simona Morasca', '3 Mcauley Dr', 'Ashland', 'OH', '44805', '419-503-2484'),
 ('Mitsue Tollner', '7 Eads St', 'Chicago', 'IL', '60632', '773-573-6914'),
 ('Leota Dilliard',
  '7 W Jackson Blvd',
  'San Jose',
  'CA',
  '95111',
  '408-752-3500'),
 ('Sage Wieser',
  '5 Boston Ave #88',
  'Sioux Falls',
  'SD',
  '57105',
  '605-414-2147'),
 ('Kris Marrier',
  '228 Runamuck Pl #2808',
  'Baltimore',
  'MD',
  '21224',
  '410-655-8723')]
```

```
[ ]:
```