Chitramoy Mukherjee

DSC 540-T302

Week-5 and Week-6

Date : 09/29/2023

Open the page in a separate chrome/firefox tab and use something like inspect element tool to see the source HTML and understand the structure Read the page using bs4 Find the table structure you will need to deal with (how many tables are there) 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 Seperate 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.

In [9]:
```
from bs4 import BeautifulSoup
import pandas as pd
import requests
import urllib.request
import time
import numpy as np
import matplotlib.pyplot as plt
from urllib.request import urlopen
```

In [20]:
```
#Read the page using bs4
fd = open("List of countries by GDP (nominal) - Wikipedia.htm", "r",encodi
soup = BeautifulSoup(fd)
fd.close()
```

In [21]:
```
# Identify the table count
all_tables = soup.find_all("table")
print("Total number of tables are {} ".format(len(all_tables)))
```

```
Total number of tables are 9
```

In [22]:
```
data_table = soup.find("table", {"class": '"wikitable"|}'})
print(type(data_table))
```

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

In [23]:
```
sources = data_table.tbody.findAll('tr', recursive=False)[0]
sources_list = [td for td in sources.findAll('td')]
print(len(sources_list))
```

```
3
```

In [24]:
```
data = data_table.tbody.findAll('tr', recursive=False)[1].findAll('td', re
```

In [25]:
```
data_tables = []
for td in data:
    data_tables.append(td.findAll('table'))
```

In [26]:
```
len(data_tables)
```

Out[26]:
```
3
```

In [27]:
```
source_names = [source.findAll('a')[0].getText() for source in sources_lis
print(source_names)
```

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

In [28]:
```
header1 = [th.getText().strip() for th in data_tables[0][0].findAll('theac
header1
```

Out[28]:
```
['Rank', 'Country', 'GDP(US$MM)']
```

In [29]:
```
rows1 = data_tables[0][0].findAll('tbody')[0].findAll('tr')[1:]
```

In [30]:
```
data_rows1 = [[td.get_text().strip() for td in tr.findAll('td')] for tr ir
```

In [31]:
```
df1 = pd.DataFrame(data_rows1, columns=header1)
```

In [32]:
```
df1.head()
```

Out[32]:

	Rank	Country	GDP(US$MM)
0	1	United States	19,390,600
1	2	China[n 1]	12,014,610
2	3	Japan	4,872,135
3	4	Germany	3,684,816
4	5	United Kingdom	2,624,529

In [33]:
```
header2 = [th.getText().strip() for th in data_tables[1][0].findAll('theac
header2
```

Out[33]:
```
['Rank', 'Country', 'GDP(US$MM)']
```

In [34]:
```
rows2 = data_tables[1][0].findAll('tbody')[0].findAll('tr')[1:]
```

In [35]:
```
def find_right_text(i, td):
    if i == 0:
        return td.getText().strip()
    elif i == 1:
        return td.getText().strip()
    else:
        index = td.text.find("♠")
        return td.text[index+1:].strip()
```

In [37]:
```
data_rows2 = [[find_right_text(i, td) for i, td in enumerate(tr.findAll('t
```

In [38]:
```
df2 = pd.DataFrame(data_rows2, columns=header2)
```

In [39]:
```
df2.head()
```

Out[39]:

	Rank	Country	GDP(US$MM)
0	1	United States	19,390,604
1		European Union[23]	17,277,698
2	2	China[n 4]	12,237,700
3	3	Japan	4,872,137
4	4	Germany	3,677,439

In [17]:
```
# Read the visit_data.csv and display dataset with head command

df_visit = pd.read_csv("C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-5
df_visit.head()
```

Out[17]:

	id	first_name	last_name	email	gender	ip_address	visit
0	1	Sonny	Dahl	sdahl0@mysql.com	Male	135.36.96.183	1225.0
1	2	NaN	NaN	dhoovart1@hud.gov	NaN	237.165.194.143	919.0
2	3	Gar	Armal	garmal2@technorati.com	NaN	166.43.137.224	271.0
3	4	Chiarra	Nulty	cnulty3@newyorker.com	NaN	139.98.137.108	1002.0
4	5	NaN	NaN	sleaver4@elegantthemes.com	NaN	46.117.117.27	2434.0

In [20]:
```
# Check for duplicates

print("First name is duplictaed - {}".format(any(df_visit.first_name.dupli
print("Last name is duplictaed - {}".format(any(df_visit.last_name.duplica
print("Email is duplictaed - {}".format(any(df_visit.email.duplicated())))
print("ip_address is duplictaed - {}".format(any(df_visit.ip_address.dupli
```

```
First name is duplictaed - True
Last name is duplictaed - True
Email is duplictaed - False
ip_address is duplictaed - False
```

In [21]:
```
# Check if any essential column contains NaN.

# Notice that we have different ways to format boolean values for the % op
print("The column Email contains NaN - %r " % df_visit.email.isnull().valu
print("The column IP Address contains NaN - %s " % df_visit.ip_address.isn
print("The column Visit contains NaN - %s " % df_visit.visit.isnull().valu
print("The column ip_address contains NaN - {}".format(any(df_visit.ip_add


```

```
The column Email contains NaN - False
The column IP Address contains NaN - False
The column Visit contains NaN - True
The column ip_address contains NaN - False
```

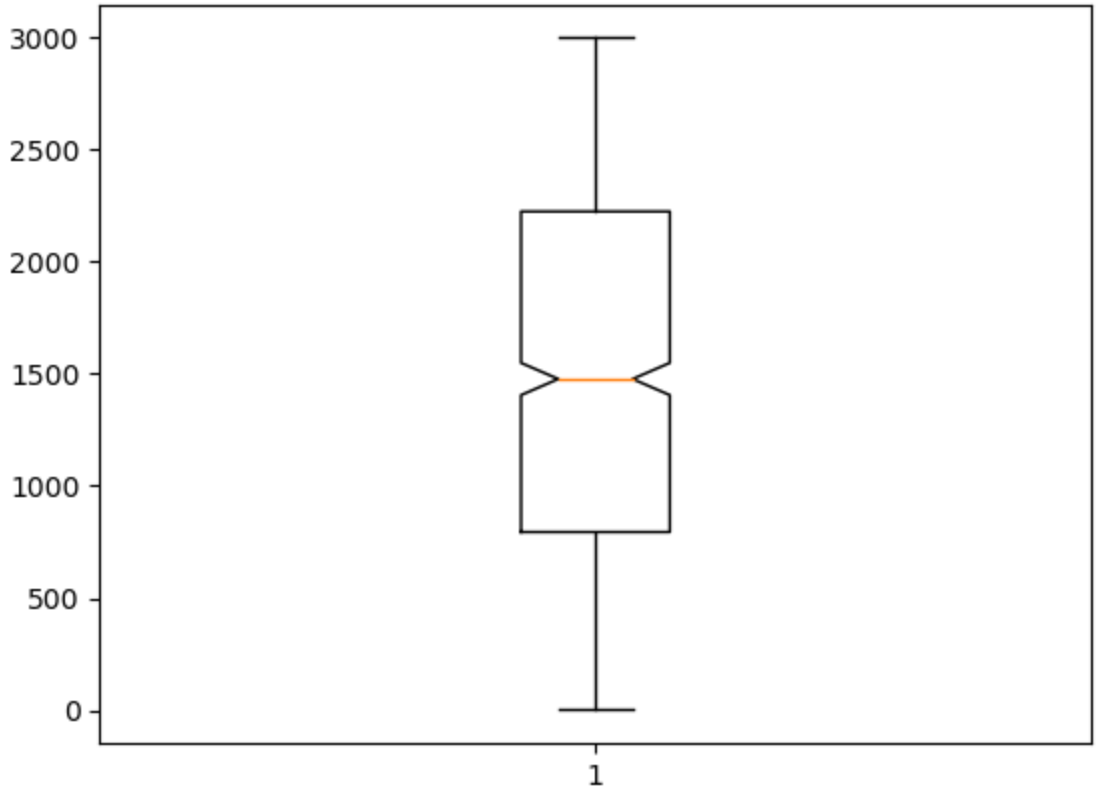
In [23]:
```
# Get rid of the outliers

size_prev = df_visit.shape
df = df_visit[np.isfinite(df_visit['visit'])]
size_after = df.shape
```

In [24]:
```
# Report the size difference

print("The size of previous data was - {prev[0]} rows and the size of the
      format(prev=size_prev, after=size_after))
```

```
The size of previous data was - 1000 rows and the size of the new one is
- 974 rows
```

In [26]:
```
# Create a box plot to check for outliers

plt.boxplot(df.visit, notch=True)
```

Out[26]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x133df69c640>,
  <matplotlib.lines.Line2D at 0x133df69c910>],
 'caps': [<matplotlib.lines.Line2D at 0x133df69cbe0>,
  <matplotlib.lines.Line2D at 0x133df69ceb0>],
 'boxes': [<matplotlib.lines.Line2D at 0x133df69c370>],
 'medians': [<matplotlib.lines.Line2D at 0x133df6b41c0>],
 'fliers': [<matplotlib.lines.Line2D at 0x133df6b4490>],
 'means': []}
```



In [27]:
```
# Get rid of any outliers

df1 = df[(df['visit'] <= 2900) & (df['visit'] >= 100)]
print("After getting rid of outliers the new size of the data is - {}".for
```

```
After getting rid of outliers the new size of the data is - 923
```

In [5]:
```
# Insert data into a SQL Lite database - create a table with the following
# Name, Address, City, State, Zip, Phone Number


import sqlite3
query = """
create table cust_infrmn (Name VARCHAR(100), Address VARCHAR(250), City VA
                        Phone_nmuber VARCHAR(12));"""

con = sqlite3.connect("mydata.sqlite")
con.execute(query)
con.commit()
```

In [6]:
```
# Add at least 10 rows of data and submit your code with a query generatin

data=[("JASON","123 Maple st", "Elkhorn", "NE", 68022, "402-111-2222"),
     ("JACK","234 Pacific st", "Omaha", "NE", 68107, "402-222-3344"),
     ("SAM","123 Blondo st", "Elkhorn", "NE", 68022, "402-222-5566"),
     ("LETTY","234 WC Road st", "Omaha", "NE", 68135, "402-123-1234"),
     ("REMI","345 181st Av", "Elkhorn", "NE", 68022, "402-112-1954"),
     ("RIG","234 Fort St.", "Elkhorn", "NE", 68022, "402-744-2345"),
     ("RON","456 DEVENPORT AV", "Omaha", "NE", 68135, "402-546-1982"),
     ("SAMUEL","777 J ST", "Omaha", "NE", 68135, "402-221-1122"),
     ("NATHAN","1234 M ST", "Omaha", "NE", 68135, "402-234-1972"),
     ("NICK","567 L ST", "Omaha", "NE", 68135, "402-123-1234"),]

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

cursor = con.execute("select * from cust_infrmn")

rows = cursor.fetchall()
rows
```

Out[6]:
```
[('JASON', '123 Maple st', 'Elkhorn', 'NE', 68022, '402-111-2222'),
 ('JACK', '234 Pacific st', 'Omaha', 'NE', 68107, '402-222-3344'),
 ('SAM', '123 Blondo st', 'Elkhorn', 'NE', 68022, '402-222-5566'),
 ('LETTY', '234 WC Road st', 'Omaha', 'NE', 68135, '402-123-1234'),
 ('REMI', '345 181st Av', 'Elkhorn', 'NE', 68022, '402-112-1954'),
 ('RIG', '234 Fort St.', 'Elkhorn', 'NE', 68022, '402-744-2345'),
 ('RON', '456 DEVENPORT AV', 'Omaha', 'NE', 68135, '402-546-1982'),
 ('SAMUEL', '777 J ST', 'Omaha', 'NE', 68135, '402-221-1122'),
 ('NATHAN', '1234 M ST', 'Omaha', 'NE', 68135, '402-234-1972'),
 ('NICK', '567 L ST', 'Omaha', 'NE', 68135, '402-123-1234')]
```