DATA MINING ASSIGNMENT 4.

Friday, 3 December 2021 15:37

Q1 The html page with income table data has the following html tags:-

1) html 2) body 3) h1 4) p 5) table 6) thead and th 7) tr 8) tbody 9) td 10) head

1) The <html> tag is used at the beginning of every webpage and is closed at the end of the webpage. It represents the start and the end of the webpage.

2) <body> All the data that is present in a webpage except title is written in the body tag. The body tag is closed just before the html tag at the end.

3) The <h1> is used to format a text as a heading. h1 denotes the biggest heading followed by h2, h3, h4 h5 and h6. Here the heading "ECS766P Data mining - Week 10" is written inside the h1 tag.

4) <p> is the paragraph tag in html and is used to format the text in a paragraph. The text above the table is written in a <p> tag.

5) The <table> is used to create a table in html. All the table rows, headers and body are defined inside the <table> tag.

6) <thead> tag in html is used to create table headers. In our webpage we have used <thead> before the <tbody> to define the column headings of our table. i.e Region, Age, Income and Online Shopper. A <th> tag is used inside a tr tag to intent headers for different columns. Eg:-

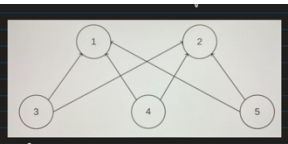
```
<thead>
  <tr>
    <th> Region </th>
  </tr>
</thead>
```

7) <tr> is used to define rows in a <table> tag.

8) <tbody> has all the data of a table except the main table header. All the rows of our data i.e <tr> tags are created inside the <tbody> tag.

9) Once all the rows are defined, we need to add data into them, therefore the <td> tag is used to add data into our table. It is usually defined under a <tr> tag. The number of <td> tags in a <tr> tag will be equal to the number of headers that is the <th> tags in the header row of the table. This is how a table is structured in html.

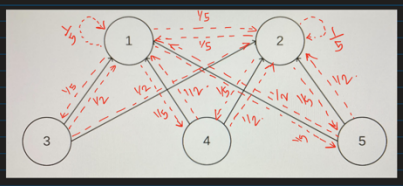
Q3)



1) In the above group of 5 web pages:-

- '1' and '2' are considered as 'authorities' because they have many inlinks. (3 inlinks each on '1' and '2' from '3', '4' and '5' (hubs)
- '3', '4' and '5' are considered as 'hubs' because they have many out links to authorities 1 and 2.

2) In the above group of webpages the initial probability for every page will be $\frac{1}{\text{no. of pages}}$, i.e $\frac{1}{5}$ for every page.



$P_{ji} = \frac{1}{\text{out } j}$ → outgoing links out j

From looking at the above webpage, we can say that the dead ends '1' and '2' have probabilities '$\frac{1}{5}$' as they are connected with '5' other nodes including a self node.
(We join a dead end with every other node, hence we have joined '1' and '2' with each and every node making its probability '$\frac{1}{5}$')

The hubs 3, 4 and 5 are each connected with '1' and '2', hence the number of outlinks are '2'. ∴ The probability for nodes '3', '4', '5' is '$\frac{1}{2}$'.

$P_{13}, P_{14}, P_{15}, P_{12}, P_{1self} = \frac{1}{5}$

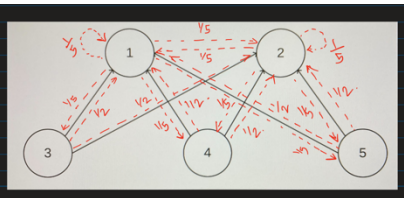
$P_{23}, P_{24}, P_{25}, P_{21}, P_{2self} = \frac{1}{5}$

$P_{31}, P_{32} = \frac{1}{2}$

$P_{41}, P_{42} = \frac{1}{2}$

$P_{51}, P_{52} = \frac{1}{2}$

3)



Page Rank for all the pages 1, 2, 3, 4, 5.
Assuming the teleportation probability = 'α'.

$\pi(i) = \frac{\alpha}{n} + (1-\alpha) \cdot \sum_{\text{In}(i)} \pi(j) \cdot p_{ji}$ → Page rank formula.

$\pi(1) = \frac{\alpha}{5} + (1-\alpha)\left[\pi(1) \cdot \frac{1}{5} + \pi(2)\frac{1}{5} + \pi(3)\frac{1}{2} + \pi(4)\frac{1}{2} + \pi(5)\frac{1}{2}\right]$

$\pi(2) = \frac{\alpha}{5} + (1-\alpha)\left[\pi(1) \cdot \frac{1}{5} + \pi(2)\frac{1}{5} + \pi(3)\frac{1}{2} + \pi(4)\frac{1}{2} + \pi(5)\frac{1}{2}\right]$

$\pi(3) = \frac{\alpha}{5} + (1-\alpha)\left[\pi(1) \cdot \frac{1}{5} + \pi(2)\frac{1}{5}\right]$

$\pi(4) = \frac{\alpha}{5} + (1-\alpha)\left[\pi(1)\frac{1}{5} + \pi(2)\frac{1}{5}\right]$

$\pi(5) = \frac{\alpha}{5} + (1-\alpha)\left[\pi(1)\frac{1}{5} + \pi(2)\frac{1}{5}\right]$

Code Question 1:-

Importing the necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

+ Code + Text

```
[2] from urllib.request import urlopen
    from bs4 import BeautifulSoup
```

Setting the url to scrape our data

```
[3] url = " http://eecs.qmul.ac.uk/~emmanouilb/income_table.html"
    html = urlopen(url)

[4] soup = BeautifulSoup(html, 'lxml')
    print(type(soup))

    <class 'bs4.BeautifulSoup'>

[5] # Get the title
    title = soup.title
    print(title)

    None
```

```
[6] text = soup.get_text()

[7] soup.find_all('a')

    []

all_links = soup.find_all("a")
for link in all_links:
    print(link.get("href"))
```

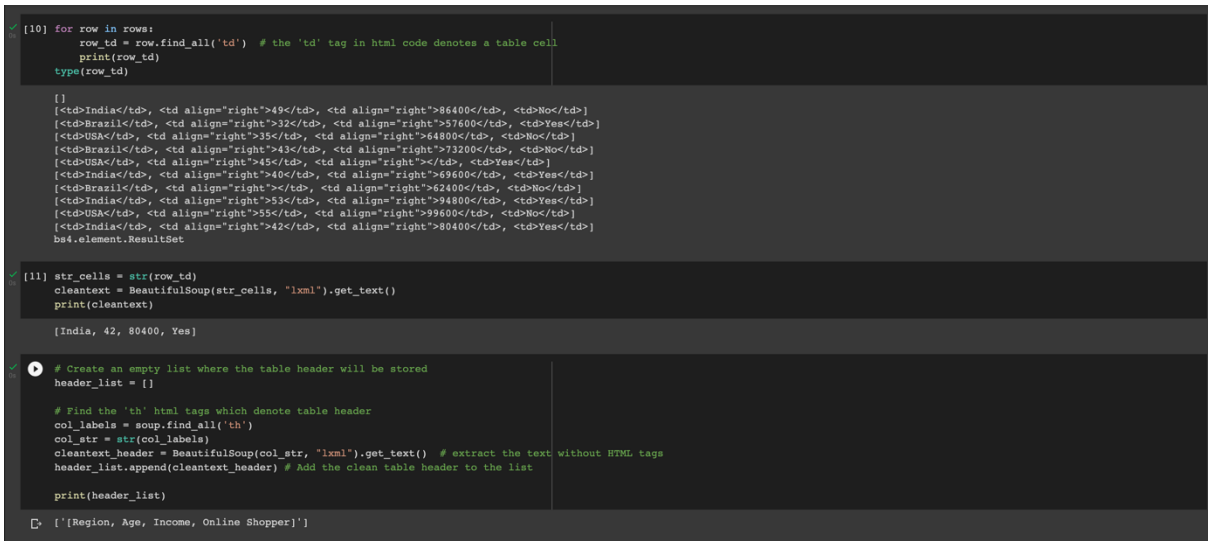
+ Code + Text

```
[9] # Print the first 10 table rows
    rows = soup.find_all('tr')  # the 'tr' tag in html denotes a table row
    print(rows[:10])
```

Finding all table rows

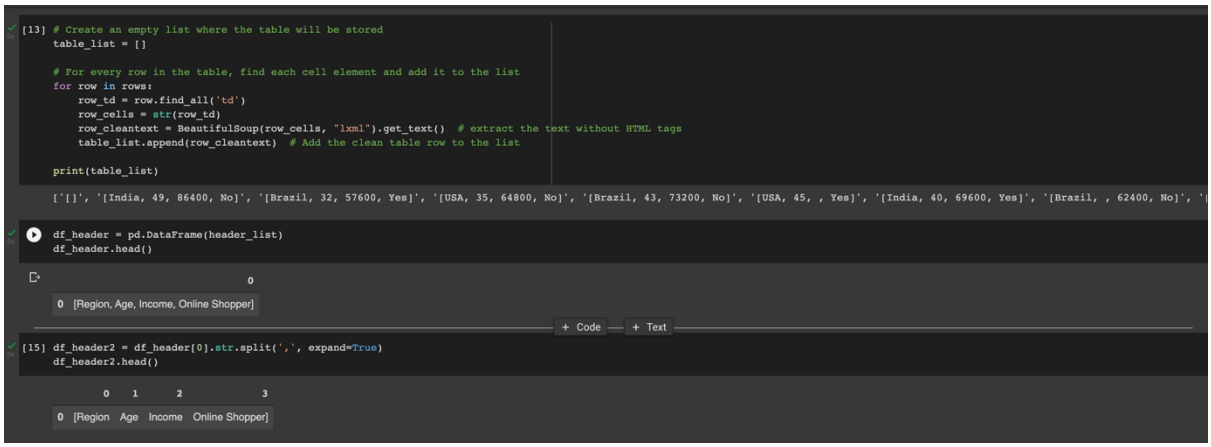
```
# Print the first 10 table rows
rows = soup.find_all('tr')  # the 'tr' tag in html denotes a table row
print(rows[:10])

[<tr><th title="Field #1">Region</th>
<th title="Field #2">Age</th>
<th title="Field #3">Income</th>
<th title="Field #4">Online Shopper</th>
</tr>, <tr>
<td>India</td>
<td align="right">49</td>
<td align="right">86400</td>
<td>No</td>
</tr>, <tr>
<td>Brazil</td>
<td align="right">32</td>
<td align="right">57600</td>
<td>Yes</td>
</tr>, <tr>
<td>USA</td>
<td align="right">35</td>
<td align="right">64800</td>
<td>No</td>
</tr>, <tr>
<td>Brazil</td>
<td align="right">43</td>
<td align="right">73200</td>
<td>No</td>
</tr>, <tr>
<td>USA</td>
<td align="right">45</td>
<td align="right"></td>
<td>Yes</td>
</tr>, <tr>
<td>India</td>
<td align="right">40</td>
<td align="right">69600</td>
<td>Yes</td>
</tr>, <tr>
<td>Brazil</td>
<td align="right"></td>
<td align="right">62400</td>
<td>No</td>
</tr>, <tr>
<td>India</td>
<td align="right">53</td>
<td align="right">94800</td>
<td>Yes</td>
</tr>, <tr>
<td>USA</td>
<td align="right">55</td>
<td align="right">99600</td>
<td>No</td>
</tr>]
```

Extracting the td tags which contain the table data



```
[10] for row in rows:
         row_td = row.find_all('td')  # the 'td' tag in html code denotes a table cell
         print(row_td)
     type(row_td)

     []
     [<td>India</td>, <td align="right">49</td>, <td align="right">86400</td>, <td>No</td>]
     [<td>Brazil</td>, <td align="right">32</td>, <td align="right">57600</td>, <td>Yes</td>]
     [<td>USA</td>, <td align="right">35</td>, <td align="right">64800</td>, <td>No</td>]
     [<td>Brazil</td>, <td align="right">43</td>, <td align="right">73200</td>, <td>No</td>]
     [<td>USA</td>, <td align="right">45</td>, <td align="right"></td>, <td>Yes</td>]
     [<td>India</td>, <td align="right">40</td>, <td align="right">69600</td>, <td>Yes</td>]
     [<td>Brazil</td>, <td align="right"></td>, <td align="right">62400</td>, <td>No</td>]
     [<td>India</td>, <td align="right">53</td>, <td align="right">94800</td>, <td>Yes</td>]
     [<td>USA</td>, <td align="right">55</td>, <td align="right">99600</td>, <td>No</td>]
     [<td>India</td>, <td align="right">42</td>, <td align="right">80400</td>, <td>Yes</td>]
     bs4.element.ResultSet

[11] str_cells = str(row_td)
     cleantext = BeautifulSoup(str_cells, "lxml").get_text()
     print(cleantext)

     [India, 42, 80400, Yes]

     # Create an empty list where the table header will be stored
     header_list = []

     # Find the 'th' html tags which denote table header
     col_labels = soup.find_all('th')
     col_str = str(col_labels)
     cleantext_header = BeautifulSoup(col_str, "lxml").get_text()  # extract the text without HTML tags
     header_list.append(cleantext_header) # Add the clean table header to the list

     print(header_list)

     ['[Region, Age, Income, Online Shopper]']
```

Joining all the row data into a list and then converting it into a dataframe



```
[13] # Create an empty list where the table will be stored
     table_list = []

     # For every row in the table, find each cell element and add it to the list
     for row in rows:
         row_td = row.find_all('td')
         row_cells = str(row_td)
         row_cleantext = BeautifulSoup(row_cells, "lxml").get_text()  # extract the text without HTML tags
         table_list.append(row_cleantext)  # Add the clean table row to the list

     print(table_list)

     ['[]', '[India, 49, 86400, No]', '[Brazil, 32, 57600, Yes]', '[USA, 35, 64800, No]', '[Brazil, 43, 73200, No]', '[USA, 45, , Yes]', '[India, 40, 69600, Yes]', '[Brazil, , 62400, No]', '
```

	0
0	[Region, Age, Income, Online Shopper]

```
[15] df_header2 = df_header[0].str.split(',', expand=True)
     df_header2.head()
```

	0	1	2	3
0	[Region	Age	Income	Online Shopper]

This is how our table looks but we need to perform some cleaning in order to remove unnecessary characters.

```
[16] df_table = pd.DataFrame(table_list)
     df_table2 = df_table[0].str.split(',', expand=True)
     df_table2.head(10)
```

	0	1	2	3
0	[]	None	None	None
1	[India	49	86400	No]
2	[Brazil	32	57600	Yes]
3	[USA	35	64800	No]
4	[Brazil	43	73200	No]
5	[USA	45		Yes]
6	[India	40	69600	Yes]
7	[Brazil		62400	No]
8	[India	53	94800	Yes]
9	[USA	55	99600	No]

```
# Remove uneccesary characters
df_table2[0] = df_table2[0].str.strip('[')
df_table2[3] = df_table2[3].str.strip(']')


# Remove all rows with any missing values
df_table3 = df_table2.dropna(axis=0, how='any')

df_table3.head(10)
```

	0	1	2	3
1	India	49	86400	No
2	Brazil	32	57600	Yes
3	USA	35	64800	No
4	Brazil	43	73200	No
5	USA	45		Yes
6	India	40	69600	Yes
7	Brazil		62400	No
8	India	53	94800	Yes
9	USA	55	99600	No
10	India	42	80400	Yes

Final Output:-

```
# We remove uneccessary characters from the header
df_header2[0] = df_header2[0].str.strip('[')
df_header2[3] = df_header2[3].str.strip(']')

# We concatenate the two dataframes
frames = [df_header2, df_table3]
df = pd.concat(frames)

df2 = df.rename(columns=df.iloc[0]) # We assign the first row to be the dataframe header
df3 = df2.drop(df2.index[0]) # We drop the replicated header from the first row of the dataframe

df3.head(10)
```

	Region	Age	Income	Online Shopper
1	India	49	86400	No
2	Brazil	32	57600	Yes
3	USA	35	64800	No
4	Brazil	43	73200	No
5	USA	45		Yes
6	India	40	69600	Yes
7	Brazil		62400	No
8	India	53	94800	Yes
9	USA	55	99600	No
10	India	42	80400	Yes

Code Question 2:-

Setting the url to scrape our data

```
[19] url_qmul = "http://eecs.qmul.ac.uk/postgraduate/programmes/"
     html2 = urlopen(url_qmul)
```

Calling the beaitiful soup library

```
[20] soup2 = BeautifulSoup(html2, 'lxml')
```

```
[21] rows2 = soup2.find_all('tr') # extracting all rows
     header = []
     col_labels2 = soup2.find_all('th') #extracting all the table headers (th tags)
     col_str2 = str(col_labels2)
     cleantext2 = BeautifulSoup(col_str2, "lxml").get_text()
     cleantext2 = cleantext2 + ', Part Time URL' + ', Full Time URL'
     header.append(cleantext2) # appending them into the header list
     print(header)

     ['[Postgraduate degree programmes, Part-time(2 year), Full-time(1 year)], Part Time URL, Full Time URL']
```

Adding all the extracted rows into a list

```
[22] table = []

     for row in rows2:
         row_td2 = row.find_all('td')
         row_cells2 = str(row_td2)
         row_cleantext2 = BeautifulSoup(row_cells2, "lxml").get_text()
         table.append(row_cleantext2)

     table
```

```
['[]',
 '[Advanced Electronic and Electrical Engineering, H60C, H60A]',
 '[Artificial Intelligence, I4U2\xa0, I4U1\xa0]',
 '[Big Data Science, H6J6, H6J7]',
 '[Computer Games, \xa0, I4U4]',
 '[Computer Science, G4U2, G4U1]',
 '[Computer Science by Research, G4Q2, G4Q1]',
 '[Computing and Information Systems, G5U6, G5U5]',
 '[Data Science and Artificial Intelligence by Conversion, \xa0, I4U5\xa0]',
 '[Electronic Engineering by Research, H6T6, H6T5]',
 '[Internet of Things (Data), I1T2, I1T0]',
 '[Machine Learning for Visual Data Analytics, H6JZ, H6JE]',
 '[Sound and Music Computing\xa0, H6T4, H6T8]',
 '[Telecommunication and Wireless Systems, H6JD, H6JA]',
 '[Digital and Technology Solutions (Apprenticeship), I4DA, \xa0]']
```

```
df_header = pd.DataFrame(header)
df_header.head()
```

	0
0	[Postgraduate degree programmes, Part-time(2 y...

Separating the header values using a comma to get individual headers for columns

```
df_header = df_header[0].str.split(', ', expand=True)
df_header.head()
```

	0	1	2	3	4
0	[Postgraduate degree programmes	Part-time(2 year)	Full-time(1 year)]	Part Time URL	Full Time URL

Separating the row values using a comma to get individual rows for their corresponding headers

```
df_table2 = pd.DataFrame(table)
df_table2 = df_table2[0].str.split(', ', expand=True)
df_table2
```

	0	1	2
0	[]	None	None
1	[Advanced Electronic and Electrical Engineering	H60C	H60A]
2	[Artificial Intelligence	I4U2	I4U1]
3	[Big Data Science	H6J6	H6J7]
4	[Computer Games		I4U4]
5	[Computer Science	G4U2	G4U1]
6	[Computer Science by Research	G4Q2	G4Q1]
7	[Computing and Information Systems	G5U6	G5U5]
8	[Data Science and Artificial Intelligence by C...		I4U5]
9	[Electronic Engineering by Research	H6T6	H6T5]
10	[Internet of Things (Data)	I1T2	I1T0]
11	[Machine Learning for Visual Data Analytics	H6JZ	H6JE]
12	[Sound and Music Computing	H6T4	H6T8]
13	[Telecommunication and Wireless Systems	H6JD	H6JA]
14	[Digital and Technology Solutions (Apprentices...	I4DA	]

Cleaning specific rows for items like '[' and ']'

```
df_table2[0] = df_table2[0].str.strip('[')
df_table2[2] = df_table2[2].str.strip(']')
df_table2
```

	0	1	2
0	]	None	None
1	Advanced Electronic and Electrical Engineering	H60C	H60A
2	Artificial Intelligence	I4U2	I4U1
3	Big Data Science	H6J6	H6J7
4	Computer Games		I4U4
5	Computer Science	G4U2	G4U1
6	Computer Science by Research	G4Q2	G4Q1
7	Computing and Information Systems	G5U6	G5U5
8	Data Science and Artificial Intelligence by Co...		I4U5
9	Electronic Engineering by Research	H6T6	H6T5
10	Internet of Things (Data)	I1T2	I1T0
11	Machine Learning for Visual Data Analytics	H6JZ	H6JE
12	Sound and Music Computing	H6T4	H6T8
13	Telecommunication and Wireless Systems	H6JD	H6JA
14	Digital and Technology Solutions (Apprenticeship)	I4DA	

Cleaning specific headers for itens like '[' and ']'

```
df_header[0] = df_header[0].str.strip('[')
df_header[2] = df_header[2].str.strip(']')
framesq2= [df_header, df_table2]
dataframe2 = pd.concat(framesq2)

dataframe2 = dataframe2.rename(columns=dataframe2.iloc[0]) # DF header
dataframe2 = dataframe2.drop(dataframe2.index[0])


display(dataframe2)
```

	Postgraduate degree programmes	Part-time(2 year)	Full-time(1 year)	Part Time URL	Full Time URL
1	Advanced Electronic and Electrical Engineering	H60C	H60A	NaN	NaN
2	Artificial Intelligence	I4U2	I4U1	NaN	NaN
3	Big Data Science	H6J6	H6J7	NaN	NaN
4	Computer Games		I4U4	NaN	NaN
5	Computer Science	G4U2	G4U1	NaN	NaN
6	Computer Science by Research	G4Q2	G4Q1	NaN	NaN
7	Computing and Information Systems	G5U6	G5U5	NaN	NaN
8	Data Science and Artificial Intelligence by Co...		I4U5	NaN	NaN
9	Electronic Engineering by Research	H6T6	H6T5	NaN	NaN
10	Internet of Things (Data)	I1T2	I1T0	NaN	NaN
11	Machine Learning for Visual Data Analytics	H6JZ	H6JE	NaN	NaN
12	Sound and Music Computing	H6T4	H6T8	NaN	NaN
13	Telecommunication and Wireless Systems	H6JD	H6JA	NaN	NaN
14	Digital and Technology Solutions (Apprenticeship)	I4DA		NaN	NaN

Finding the td tags and extracing all the links (a tags) from our webpage

```
[34] table_listnew = []
     for row in rows2:
       try:
         row_tdnew = row.find_all('td')
         row_cellsnew = str(row_tdnew)
         row_cleantextnew = BeautifulSoup(row_cellsnew, "lxml").find('a').get('href')
         table_listnew.append(row_cleantextnew)
       except:
         continue

     table_listnew
```

```
['https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/advanced-electronic-and-electrical-engineering-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/artificial-intelligence-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/big-data-science-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-games-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-by-research-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computing-and-information-systems-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/data-science-and-artificial-intelligence-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/electronic-engineering-by-research-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/internet-of-things-data-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/machine-learning-for-visual-data-analytics-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/sound-and-music-computing-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/telecommunication-and-wireless-systems-msc/',
 'https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/digital-and-technology-solutions-specialist-msc/']
```

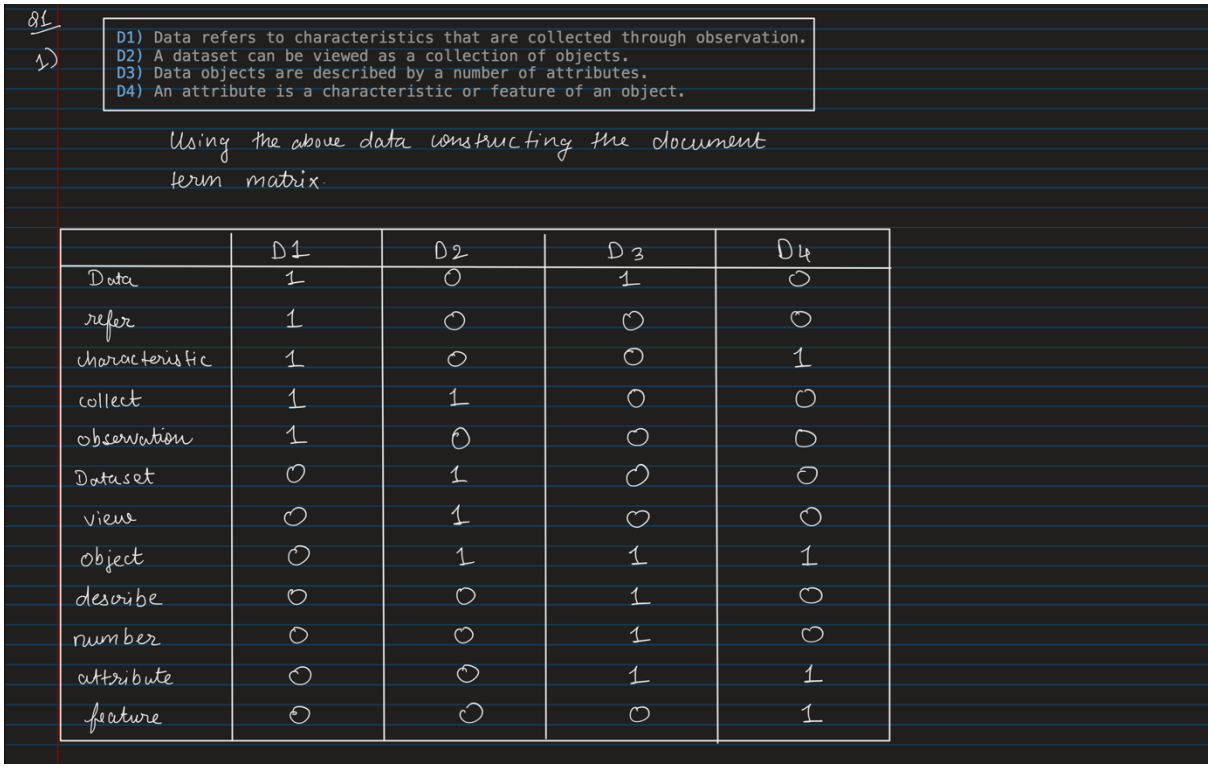
Filling our table with the extracted links for both Part Time and Full Time and displaying the table using IPython.display in order to make the links clickable

```
for i in range(len(dataframe2)):
  if dataframe2.loc[i+1, 'Part-time(2 year)'] != ' ' and dataframe2.at[i+1, 'Full-time(1 year)'] != ' ':
    dataframe2.loc[i+1, 'Part Time URL'] = table_listnew[i]
    dataframe2.loc[i+1, 'Full Time URL'] = table_listnew[i]
  elif dataframe2.loc[i+1, 'Part-time(2 year)'] == ' ' and dataframe2.at[i+1, 'Full-time(1 year)'] != ' ':
    dataframe2.loc[i+1, 'Full Time URL'] = table_listnew[i]
    dataframe2.loc[i+1, 'Part Time URL'] = ' '
  elif dataframe2.loc[i+1, 'Part-time(2 year)'] != ' ' and dataframe2.at[i+1, 'Full-time(1 year)'] == ' ':
    dataframe2.loc[i+1, 'Part Time URL'] = table_listnew[i]
    dataframe2.loc[i+1, 'Full Time URL'] = ' '

from IPython.display import HTML
HTML(dataframe2.to_html(render_links=True,escape=False))
```

Final Output:-

	Postgraduate degree programmes	Part-time(2 year)	Full-time(1 year)	Part Time URL	Full Time URL
1	Advanced Electronic and Electrical Engineering	H60C	H60A	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/advanced-electronic-and-electrical-engineering-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/advanced-electronic-and-electrical-engineering-msc/
2	Artificial Intelligence	I4U2	I4U1	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/artificial-intelligence-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/artificial-intelligence-msc/
3	Big Data Science	H6J6	H6J7	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/big-data-science-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/big-data-science-msc/
4	Computer Games		I4U4		https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-games-msc/
5	Computer Science	G4U2	G4U1	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-msc/
6	Computer Science by Research	G4Q2	G4Q1	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-by-research-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computer-science-by-research-msc/
7	Computing and Information Systems	G5U6	G5U5	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computing-and-information-systems-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/computing-and-information-systems-msc/
8	Data Science and Artificial Intelligence by Conversion		I4U5		https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/data-science-and-artificial-intelligence-msc/
9	Electronic Engineering by Research	H6T6	H6T5	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/electronic-engineering-by-research-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/electronic-engineering-by-research-msc/
10	Internet of Things (Data)	I1T2	I1T0	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/internet-of-things-data-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/internet-of-things-data-msc/
11	Machine Learning for Visual Data Analytics	H6JZ	H6JE	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/machine-learning-for-visual-data-analytics-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/machine-learning-for-visual-data-analytics-msc/
12	Sound and Music Computing	H6T4	H6T8	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/sound-and-music-computing-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/sound-and-music-computing-msc/
13	Telecommunication and Wireless Systems	H6JD	H6JA	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/telecommunication-and-wireless-systems-msc/	https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/telecommunication-and-wireless-systems-msc/
14	Digital and Technology Solutions (Apprenticeship)	I4DA		https://www.qmul.ac.uk/postgraduate/taught/coursefinder/courses/digital-and-technology-solutions-specialist-msc/	

ASSIGNMENT 4 PART 2:-



Q1

1)

D1) Data refers to characteristics that are collected through observation.
D2) A dataset can be viewed as a collection of objects.
D3) Data objects are described by a number of attributes.
D4) An attribute is a characteristic or feature of an object.

Using the above data constructing the document term matrix.

	D1	D2	D3	D4
Data	1	0	1	0
refer	1	0	0	0
characteristic	1	0	0	1
collect	1	1	0	0
observation	1	0	0	0
Dataset	0	1	0	0
view	0	1	0	0
object	0	1	1	1
describe	0	0	1	0
number	0	0	1	0
attribute	0	0	1	1
feature	0	0	0	1

2) Calculating the inverse document frequency idf(w) for all the words.

Formula for IDF = $\log\left(\frac{n_d}{dw}\right)$

∴ idf(data) = $\log\left(\frac{4}{2}\right) = \log 2 = 0.301$.

idf(refer) = $\log\left(\frac{4}{1}\right) = \log 4 = 0.602$

idf(characteristic) = $\log\left(\frac{4}{2}\right) = \log 2 = 0.301$.

idf(collect) = $\log\left(\frac{4}{2}\right) = \log 2 = 0.301$.

idf(observation) = $\log\left(\frac{4}{1}\right) = \log 4 = 0.602$.

idf(dataset) = $\log\left(\frac{4}{1}\right) = \log 4 = 0.602$

idf(views) = $\log\left(\frac{4}{1}\right) = \log 4 = 0.602$

idf(object) = $\log\left(\frac{4}{3}\right) = \log\left(\frac{4}{3}\right) = 0.1249$.

idf(describes) = $\log 4 = 0.602$

idf(numbers) = $\log 4 = 0.602$

idf(attribute) = $\log(4/2) = 0.301$

idf(feature) = $\log(4/1) = 0.602$

Q2

Time series = {0.1, 0.15, 0.2, 0.2, 0.3, 0.4, 0.25, 0.6, 0.5}

To perform time series binning using k=3 values per bin.

k = 3

$$y'_{i+1} = \sum_{r=1}^{k} \frac{y_{i+r}}{k}$$

$y'_1 = \frac{0.1 + 0.15 + 0.2}{3} = 0.15$

$y'_2 = \frac{0.2 + 0.3 + 0.4}{3} = 0.3$

$y'_3 = \frac{0.25 + 0.6 + 0.5}{3} = 0.45$

$y' = \{0.15, 0.3, 0.45\}$

→ Resulting Time series after k=3 binning.

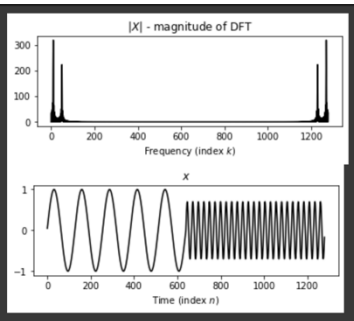
Question 3

Importing necessary libraries and loading the time series data. Later converting the data into a flattened(1-D) numpy array.

```
[ ] import numpy as np
    import pandas as pd
    time_data = pd.read_csv('timeseries.csv')
    time_np = np.array(time_data)
    time_np = time_np.flatten()
```

Computing the DFT of the time series data and comparing it with the plot of the original data

```
import matplotlib.pyplot as plt

# numpy implementation of FFT
Xfft = np.fft.fft(time_np)
# Plot
plt.subplot(2,1,1)
plt.title('$|X|$ - magnitude of DFT')
plt.plot(np.abs(Xfft), 'k')
plt.xlabel('Frequency (index $k$)')
plt.show()

plt.subplot(2,1,2)
plt.title('$x$')
plt.plot(time_np, 'k')
plt.xlabel('Time (index $n$)')
plt.show()
```



Conclusion

It is evident that the time series has 2 predominant frequency components.

The reason being, we can see two different kinds of wave frequencies here (1 from 0-600 and another from 600-1200 , therefore 2 frequency components)

Question 4

```
birth_data = pd.read_csv('births.csv')
df = pd.DataFrame(birth_data)
df
```

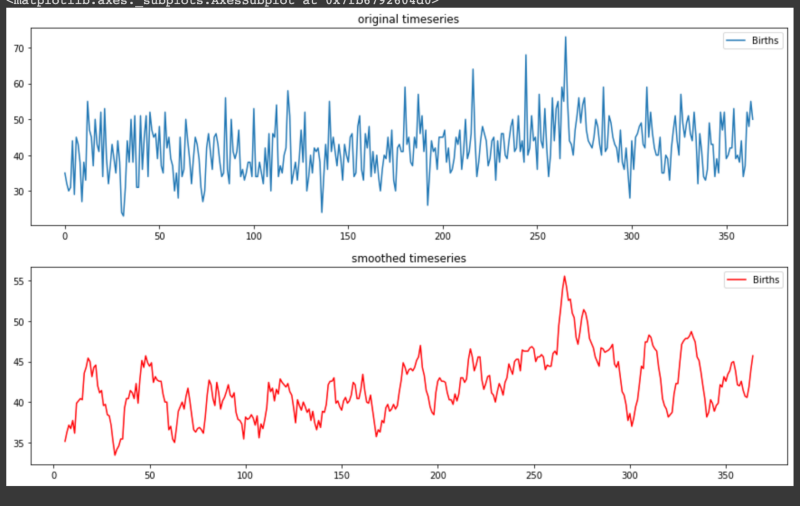
	Date	Births
0	1959-01-01	35
1	1959-01-02	32
2	1959-01-03	30
3	1959-01-04	31
4	1959-01-05	44
...	...	...
360	1959-12-27	37
361	1959-12-28	52
362	1959-12-29	48
363	1959-12-30	55
364	1959-12-31	50

365 rows × 2 columns

Smoothing using window size 7

```
# Perform trailing moving average smoothing
rolling = birth_data.rolling(window=7)
rolling_mean = rolling.mean()
print(rolling_mean.head(10))

# plot original and transformed dataset
time_data.plot(figsize=(15,4),title='original timeseries')
rolling_mean.plot(color='red', figsize=(15,4),title='smoothed timeseries')
```

```
      Births
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
5        NaN
6  35.142857
7  36.285714
8  37.142857
9  36.714286
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6792604d0>
```



Replacing the Nan values with 0 using the fillna function in pandas.

```
rolling_mean['Births'] = rolling_mean['Births'].fillna(0)
rolling_mean['Births']
```

```
0       0.000000
1       0.000000
2       0.000000
3       0.000000
4       0.000000
          ...
360    40.714286
361    40.571429
362    41.857143
363    44.000000
364    45.714286
Name: Births, Length: 365, dtype: float64
```

```
rolling_mean['Births'] = rolling_mean['Births'].fillna(0)
ar_data = np.array(rolling_mean['Births'])

df['Births'] = rolling_mean['Births']
df
```

	Date	Births
0	1959-01-01	0.000000
1	1959-01-02	0.000000
2	1959-01-03	0.000000
3	1959-01-04	0.000000
4	1959-01-05	0.000000
...	...	...
360	1959-12-27	40.714286
361	1959-12-28	40.571429
362	1959-12-29	41.857143
363	1959-12-30	44.000000
364	1959-12-31	45.714286

365 rows × 2 columns

Fitting the auto-regressive model and predicting first 5 values for 1960 janurary

```
[43] # Install/upgrade wikipedia and statsmodels packages for the lab
     !pip install statsmodels --upgrade

     Requirement already satisfied: statsmodels in /usr/local/lib/python3.7/dist-packages (0.13.1)
     Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.19.5)
     Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.1.5)
     Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (0.5.2)
     Requirement already satisfied: scipy>=1.3 in /usr/local/lib/python3.7/dist-packages (from statsmodels) (1.4.1)
     Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.25->statsmodels) (2018.9)
     Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.25->statsmodels) (2.8.2)
     Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.5.2->statsmodels) (1.15.0)

     # Initialise
     from statsmodels.tsa.ar_model import AutoReg

     # Fit Autoregressive model
     model = AutoReg(df['Births'], lags=2,old_names=False) # "lags" indicates the model order which is 2 in our case
     model_fit = model.fit()

     # Make prediction
     yhat = model_fit.predict(len(df['Births']), len(df['Births'])+4)
     print(yhat)

     365    45.380177
     366    44.960852
     367    44.590676
     368    44.271699
     369    43.997395
     dtype: float64
```

Hence we have predicted the daily births for the first 5 days of 1960 using the auto regressive model.

Fitting the ARIMA model and predicting first 5 values for 1960 janurary

```
     # Fit ARMA model
     model = ARIMA(df['Births'], order=(2, 0, 2)) # p=2, q=2
     model_fit = model.fit()

     # Make prediction
     yhat = model_fit.predict(len(df['Births']), len(df['Births'])+4)
     print(yhat)

     365    45.810249
     366    45.818768
     367    45.728095
     368    45.564020
     369    45.347310
     Name: predicted_mean, dtype: float64
```

Hence we have predicted the daily births for the first 5 days of 1960 using the ARMA model.

Question 5

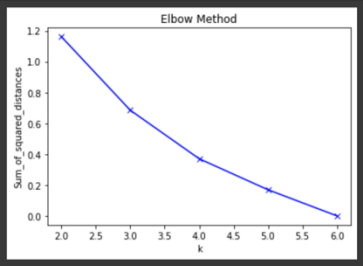
```
[ ] import wikipedia

    articles=['anomaly detection','cluster analysis','k-means clustering', 'data mining', 'data warehouse', 'association rule learning']
    wiki_lst=[]
    title=[]


    # Load wikipedia articles
    for article in articles:
       print("loading content: ",article)
       wiki_lst.append(wikipedia.page(article, auto_suggest=False).content)
       title.append(article)

    loading content:  anomaly detection
    loading content:  cluster analysis
    loading content:  k-means clustering
    loading content:  data mining
    loading content:  data warehouse
    loading content:  association rule learning

    from sklearn.feature_extraction.text import TfidfVectorizer

    vectorizer = TfidfVectorizer(stop_words={'english'})
    X = vectorizer.fit_transform(wiki_lst) # Create tf-idf feature of the wikipedia dataset

    print(X.shape) # Print dimensions of tf-idf feature

    (6, 3485)
```

Looking at the above plot of sum if squared distances vs k, using the elbow method we can say that the optimal value of k is 4 as the line graph linearly declines aftet k=4.

```
# Fit k-means model with k=4
true_k = 4
model = KMeans(n_clusters=true_k, init='k-means++', max_iter=200, n_init=10)
model.fit(X)

# Print list of documents and associated clusters
labels=model.labels_
wiki_cl=pd.DataFrame(list(zip(title,labels)),columns=['title','cluster'])
print(wiki_cl.sort_values(by=['cluster']))
```

```
                        title  cluster
0          anomaly detection        0
1           cluster analysis        1
2         k-means clustering        1
3              data mining        2
4            data warehouse        2
5  association rule learning        3
```

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

Sum_of_squared_distances = []
K = range(2,7)

for k in K:
    km = KMeans(n_clusters=k, max_iter=200, n_init=10)
    km = km.fit(X)
    Sum_of_squared_distances.append(km.inertia_)

plt.plot(K, Sum_of_squared_distances, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum_of_squared_distances')
plt.title('Elbow Method')
plt.show()
```



Another reason to choose the cluster value k = 4 apart from the linear decline, is that the resulting clusters make most sense out of all the other clusters generated from different values of k. The resulting clusters for k=4 are:-

1st Anomaly Detection
2nd Cluster Analysis and K-means Clustering (which should be in the same cluster)
3rd Data Mining and Data Warehouse (which should also be in the same cluster)
4th Association Rule mining (this cannot be grouped into any of these clusters)