Q 1)

You are provided with the following
URL: http://eecs.qmul.ac.uk/~emmanouilb/income_table.html. This webpage includes a table on individuals' income and shopping habits - the same that was used in the Week 3 lab.

A. Inspect the HTML code of the above URL, and provide a short report on the various tags present in the code. What is the function of each unique tag present in the HTML code? [0.5 marks out of 5]
B. Using Beautiful Soup, scrape the table and convert it into a pandas dataframe. Perform data cleaning when necessary to remove extra characters (no need to handle missing values). In the report include the code that was used to scrape and convert the table and provide evidence that the table has been successfully scraped and converted (e.g. by displaying the contents of the dataframe).

Answer:- A) url = "http://eecs.qmul.ac.uk/~emmanouilb/income_table.html"

html = urlopen(url)

In this url there are different types of tags present which are as follows:

<h1> this are the heading tag in the website

<P> this are the paragraph tag in which you write paragraph in the webpage

< table > for tables,

< tr > for table rows,

< th > for table headers and

< td > for table cells.

B)

```
import requests

import lxml.html as lh

import pandas as pd

url='http://eecs.qmul.ac.uk/~emmanouilb/income_table.html'

#Create a handle, page, to handle the contents of the website

page = requests.get(url)

#Store the contents of the website under doc

doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML

tr_elements = doc.xpath('//tr')


#Check the length of the first 12 rows
```

```
[len(T) for T in tr_elements[:12]]


tr_elements = doc.xpath('//tr')
#Create empty list
col=[]
i=0
#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i+=1
    name=t.text_content()
    print (i,name)
    col.append((name,[]))

#Since out first row is the header, data is stored on the second row onwards
for j in range(1,len(tr_elements)):
    #T is our j'th row
    T=tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T)!=4:
        break

    #i is the index of our column
    i=0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data=t.text_content()
        #Check if row is empty
        if i>0:
        #Convert any numerical value to integers
```

```
        try:
            data=int(data)
        except:
            pass
    #Append the data to the empty list of the i'th column
    col[i][1].append(data)
    #Increment i for the next column
    i+=1
```

```
[len(C) for (title,C) in col]
```

```
Dict={title:column for (title,column) in col}
df=pd.DataFrame(Dict)
```

```
df.head()
```

Out[161]:

	Region	Age	Income	Online Shopper
0	India	49	86400	No
1	Brazil	32	57600	Yes
2	USA	35	64800	No
3	Brazil	43	73200	No
4	USA	45		Yes

Q 2) The list of the various MSc programmes offered by the School of EECS is provided at the following URL: http://eecs.qmul.ac.uk/postgraduate/programmes/. Perform web scraping on the table present in the above URL and convert it into a pandas dataframe that would include one row for each programme of study as shown in the webpage. The dataframe should include the following 5 columns: name of postgraduate degree programme (e.g. Advanced Electronic and Electrical Engineering), programme code for part-time study (e.g. H60C), programme code for full-time study (e.g. H60A), URL for part-time study programme details, URL for full-time study programme details. Perform data cleaning to remove unecessary characters when needed. In the report include the code that was used to scrape, convert and clean the table and provide evidence that the table has been successfully scraped (e.g. by displaying the contents of the dataframe). [1 mark out of 5]

Answer:-

```
import requests
```

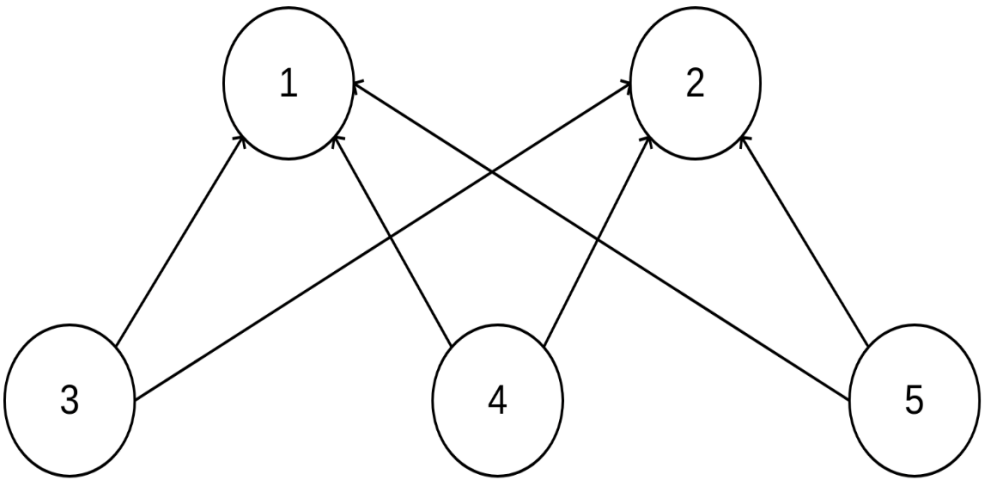
```
import lxml.html as lh
import pandas as pd
url='http://eecs.qmul.ac.uk/postgraduate/programmes/’
#Create a handle, page, to handle the contents of the website
page = requests.get(url)
#Store the contents of the website under doc
doc = lh.fromstring(page.content)
#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')
#Check the length of the first 12 rows
[len(T) for T in tr_elements[:12]]
tr_elements = doc.xpath('//tr')
#Create empty list
col=[]
i=0
#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i+=1
    name=t.text_content()
    print (i,name)
    col.append((name,[]))
#Since out first row is the header, data is stored on the second row onwards
for j in range(1,len(tr_elements)):
    #T is our j'th row
    T=tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T)!=4:
        break

    #i is the index of our column
```

```
    i=0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data=t.text_content()
        #Check if row is empty
        if i>0:
        #Convert any numerical value to integers
            try:
                data=int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i+=1
[len(C) for (title,C) in col]
Dict={title:column for (title,column) in col}
df=pd.DataFrame(Dict)
parttime_link=[]
anchor=soup.find_all('a',attrs={'title':'Use alt + click to follow the link'})
all_links=set()

for link in anchor:
    if(link.get('href') != '#'):
        parttime_link="http://eecs.qmul.ac.uk/postgraduate/programmes/" +link.get('href')
        all_links.add(parttime_link)
        print(parttime_link)

fulltime_link=[]
anchor=soup.find_all('a',attrs={'title':'Use alt + click to follow the link'})
```

```
all_links=set()


for link in anchor:
   if(link.get('href') != '#'):
      fulltime_link="http://eecs.qmul.ac.uk/postgraduate/programmes/" +link.get('href')
      all_links.add(fulltime_link)
      print(fulltime_link)
frame=[df,parttime_link,fulltime_link]
result = pd.concat(frame,axis=0)
```

Q 3)

1. Consider the graph in the figure below as displaying the links for a group of 5 webpages.
   A. Which of the 5 nodes would you consider hubs and which would you consider authorities? Explain why. [0.5 marks out of 5]
   B. Assume that this graph is to be used as input to the PageRank algorithm. Calculate the transition probabilities p_{ij}pij for all 5 nodes in the below graph (where ii and jj take values between 1 to 5). Add transitions with a uniform probability distribution in the case of dead-end nodes (do not consider cases of dead-end components). [1 mark out of 5].
   C. Derive the PageRank $\pi(i)$π(i) for all nodes, where $i=\{1,...,5\}$i={1,...,5} corresponds to the node index. Assume that the teleportation probability is set to αα. [1 mark out of 5]



Answer:-

1) The nodes 3, 4 and 5 should be considered HUBS while nodes 1 and 2 should be considered authorities.

   This is because according to HITS algorithm by Jon Kleinberg, if a page points to many web pages with high authority weights, it is a HUB while if a page is pointed to by many web pages with higher HUB weights, it is an AUTHORITY.

   In layman's terms, a HUB is a page which can direct users to relevant sites i.e. the root while the authority is the page the user is looking for. For example, if a person was looking for online shopping websites, and he googles it, then google which shows him a list of all the online shopping sites is the HUB as it points to many relevant pages and the search results like amazon, flipkart, myntra etc are the authorities i.e. the site the user was actually looking for.

2) Assume a small universe of five web pages: **1**, **2**, **3**, **4** and **5**. Links from a page to itself are ignored. Multiple outbound links from one page to another page are treated as a single link. PageRank is initialized to the same value for all pages. In the original form of PageRank, the sum of PageRank over all pages was the total number of pages on the web at that time, so each page in this example would have an initial value of 1. However, later versions of PageRank, and the remainder of this section, assume a probability distribution between 0 and 1. Hence the initial value for each page in this example is 0.25.

   The PageRank transferred from a given page to the targets of its outbound links upon the next iteration is divided equally among all outbound links.

   If the only links in the system were from pages **2**, **3**, and **4** to **5**, each link would transfer

0.25 PageRank to **1** upon the next iteration, for a total of 0.75.

Suppose instead that page **2** had a link to pages **4** and **1**, page **3**had a link to page **2**, and page **3** had links to all three pages. Thus, upon the first iteration, page **2** would transfer half of its existing value, or 0.125, to page **1** and the other half, or 0.125, to page **3**. Page **3** would transfer all of its existing value, 0.25, to the only page it links to, **1**. Since **4** had three outbound links, it would transfer one third of its existing value, or approximately 0.083, to **1**. At the completion of this iteration, page **1** will have a PageRank of approximately 0.458.In other words, the PageRank conferred by an outbound link is equal to the document's own PageRank score divided by the number of outbound links **L( )**.

3) At node with no output-links: teleport operation

At node with output-links: teleport operation with probability $0<\alpha<1$ and the standard random walk $1-\alpha$. α is a fixed parameter chosen in advance.

$C_B(2)= \sigma_{13}(2)/\sigma_{13} + \sigma_{31}(2)/ \sigma_{31} = 1/1 + 0 = 1$

$C_B'(2) = C_B(2)/(3-1)(3-2) = 0.5$

0	1	0
1	0	1
0	1	0
0	1	0
½	0	½

0	1	0
1/3	1/3	1/3
1/3	1/3	1/3
1/3	1/3	1/3
1/6	2/3	1/6
5/12	1/6	5/12
1/6	2/3	1/6