RENKON

DATA COLLECTION - PROJECT 1

SOURCE CODE:

(Created by Ritwik Chandra Pandey - November 2021)

--

LIBRARIES USED:

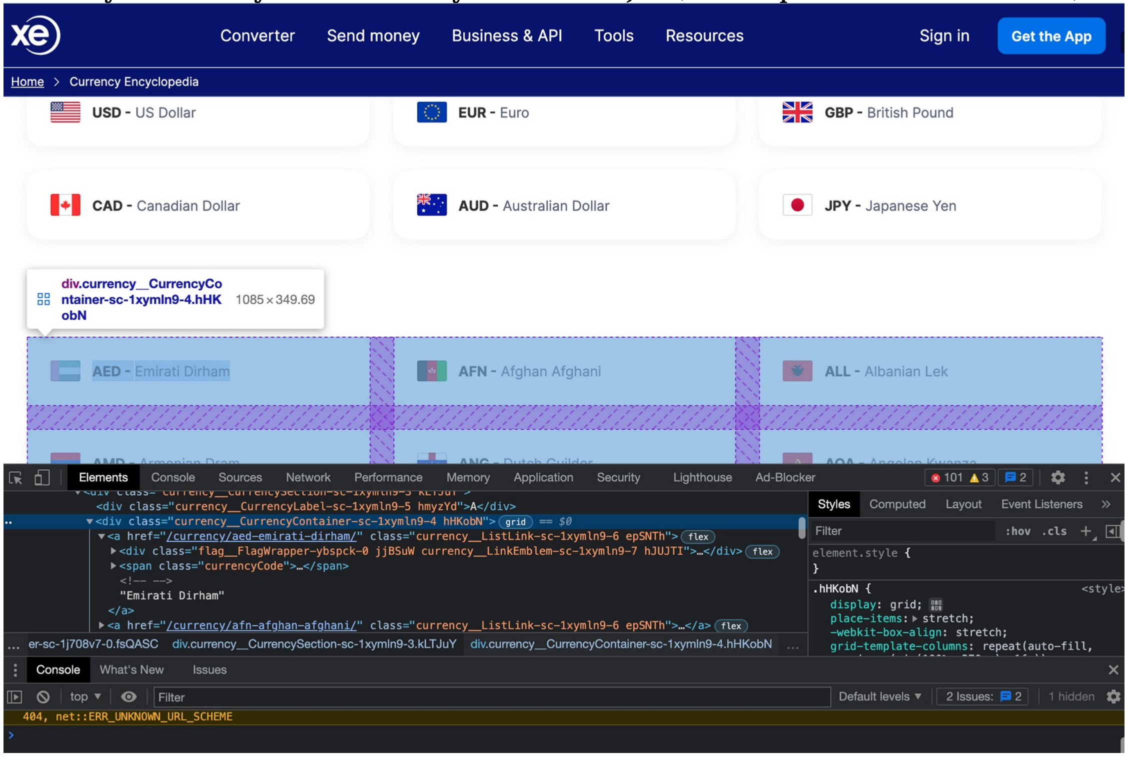
re, tabulate, requests, pandas, bs4, prettytable.

```
import re
from tabulate import tabulate
import requests
import pandas as pd
from bs4 import BeautifulSoup
from prettytable import PrettyTable
```

Function 1 : currency_info(doc)

The role of this function is to extract the name, code and link to each and every currency available in XE Currency Encyclopedia. It takes *doc* as argument which is actually a BeautifulSoup object representing the parsed document as a whole. Its basically an HTML document converted into a complex tree of Python objects. The *find_all()* function which is invoked using *doc* returns all occurences of 'div' tag having class

-'currency__CurrencySection-sc-1xymln9-3 kLTJuY', in the parsed document *doc*, as a list.



```
        <div class="currency__CurrencyLabel-sc-1xymln9-5 hmyzYd">A</div>
      ▼<div class="currency__CurrencyContainer-sc-1xymln9-4 hHKobN"> grid == $0
        ▼<a href="/currency/aed-emirati-dirham/" class="currency__ListLink-sc-1xymln9-6 epSNTh"> flex
          ▶<div class="flag__FlagWrapper-ybspck-0 jjBSuW currency__LinkEmblem-sc-1xymln9-7 hJUJTI">…</div> flex
          ▶<span class="currencyCode">…</span>
            <!-- -->
            "Emirati Dirham"
          </a>
        ▶<a href="/currency/afn-afghan-afghani/" class="currency__ListLink-sc-1xymln9-6 epSNTh">…</a> flex
… er-sc-1j708v7-0.fsQASC   div.currency__CurrencySection-sc-1xymln9-3.kLTJuY   div.currency__CurrencyContainer-sc-1xymln9-4.hHKobN
```

```
element.style {
}
.hHKobN {
  display: grid;
  place-items: ▶ stretch;
  -webkit-box-align: stretch;
  grid-template-columns: repeat(auto-fill,
```

```
404, net::ERR_UNKNOWN_URL_SCHEME
```

It goes on to define a *cur_dict* dictionary with keys 'Name', 'Code' and 'Link' as shown below. Now, we do not want the *'popular'* table on the top, thats why we start with 1 in the following for loop. The loop goes on till the length of *cur* and finds all 'div' tags and puts them in *cur_in*, it also puts all 'a' tags in *cur_in_a*.

Take a look at where 'a' tags occur in the above picture.

Now, the links, currency codes and names are extracted similarly as shown in the code below and appeneded to *cur_dict* appropriately. *cur_dict* is then converted to dataframe using *pd.DataFrame()* which is ultimately returned.

```
def currency_info(doc):
    cur = doc.find_all('div',{'class':'currency__CurrencySection-sc-
1xymln9-3 kLTJuY'})
    cur_dict = {'Name':[],'Code':[], 'Links':[]}
    for i in range(1,len(cur)):
        cur_in = cur[i].find_all('div',
{'class':'currency__CurrencyContainer-sc-1xymln9-4 hHKobN'})
        cur_in_a = cur_in[0].find_all('a',
{'class':'currency__ListLink-sc-1xymln9-6 epSNTh'})
        for j in range(len(cur_in_a)):
                    cur_in_a_link = cur_in_a[j]['href']
                    cur_in_a_cc = (cur_in_a[j].find_all('span',
{'class':'currencyCode'}))[0].text.split()[0]
                    cur_in_a_name = ' '.join(cur_in_a[j].text.split()
[2:7])
```

```
                        cur_dict['Name'].append(cur_in_a_name)
                        cur_dict['Code'].append(cur_in_a_cc)

cur_dict['Links'].append('https://www.xe.com'+cur_in_a_link)

    df = pd.DataFrame(cur_dict)
    return df
```

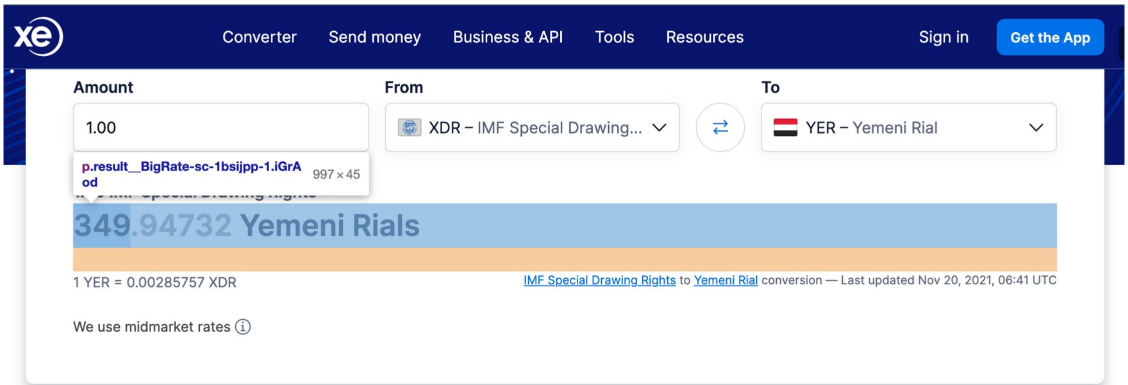
Function 2 : conversion(c1, c2, amount)

The role of this function is to return the converted amount as well as the date and time of the last updation that took place of the exchange rate, all as a single string.

Here, we have argumemets as *c1*, *c2* and *amount*. *c1* and *c2* refer to country codes and *amount* refers to the amount that needs to be converted. The *code_url* is customized as per the link of the webpage, using *amount* that is obtained from the argument. *response* is a request object and response.text is the HTML code of the webpage.

response.text and 'html.parser' are fed to BeautifulSoup to obtain parsed document called *doc*. Then, a check takes place where the status code of the response object is checked. Anything other than 200 is an error and appropriate exception is kept raedy to deal with the error.

find_all() returns 'p' tags having the class as shown below to extact the conversion amount and similarly 'div' tags are obtained to extract the time and date of last update.



Amount: 1.00
From: XDR – IMF Special Drawing...
To: YER – Yemeni Rial

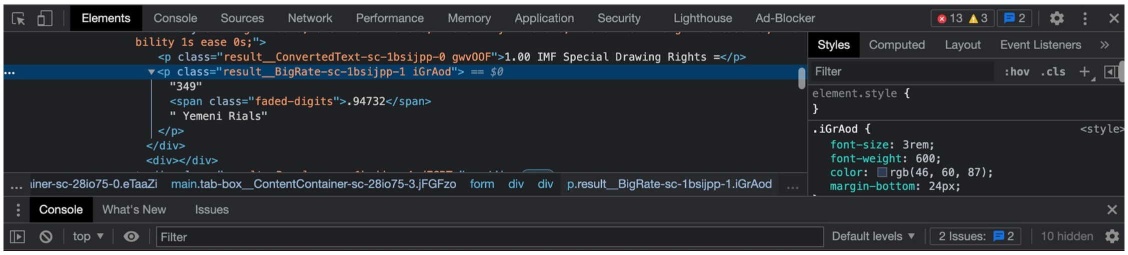
p.result__BigRate-sc-1bsijpp-1.iGrAod 997 × 45

349.94732 Yemeni Rials

1 YER = 0.00285757 XDR

IMF Special Drawing Rights to Yemeni Rial conversion — Last updated Nov 20, 2021, 06:41 UTC

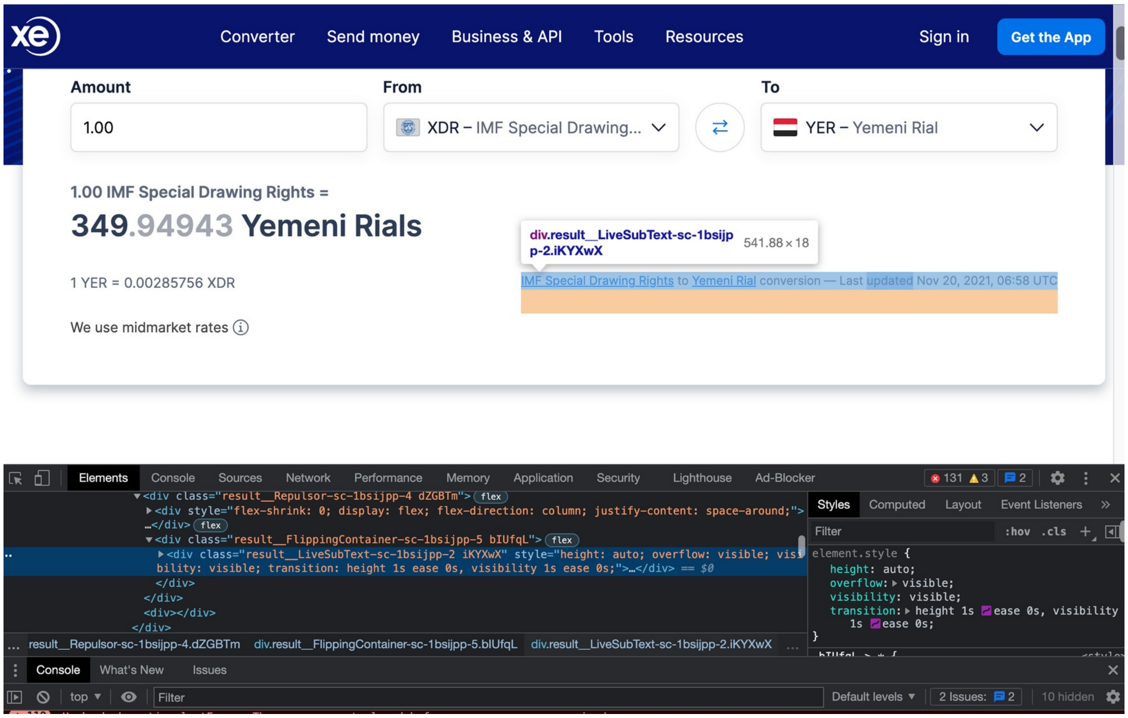
We use midmarket rates



```
bility 1s ease 0s;">
  <p class="result__ConvertedText-sc-1bsijpp-0 gwvOOF">1.00 IMF Special Drawing Rights =</p>
 ▾<p class="result__BigRate-sc-1bsijpp-1 iGrAod"> == $0
    "349"
    <span class="faded-digits">.94732</span>
    " Yemeni Rials"
  </p>
</div>
<div></div>
```

```
element.style {
}
.iGrAod {                                    <style>
    font-size: 3rem;
    font-weight: 600;
    color: rgb(46, 60, 87);
    margin-bottom: 24px;
```

… iner-sc-28io75-0.eTaaZi main.tab-box__ContentContainer-sc-28io75-3.jFGFzo form div div p.result__BigRate-sc-1bsijpp-1.iGrAod …



Amount
1.00
From
XDR – IMF Special Drawing...
To
YER – Yemeni Rial

1.00 IMF Special Drawing Rights =
349.94943 Yemeni Rials

1 YER = 0.00285756 XDR

We use midmarket rates

div.result__LiveSubText-sc-1bsijpp-2.iKYXwX 541.88 × 18

IMF Special Drawing Rights to Yemeni Rial conversion — Last updated Nov 20, 2021, 06:58 UTC

```
▼<div class="result__Repulsor-sc-1bsijpp-4 dZGBTm"> flex
  ▶<div style="flex-shrink: 0; display: flex; flex-direction: column; justify-content: space-around;">
  …</div> flex
  ▼<div class="result__FlippingContainer-sc-1bsijpp-5 bIUfqL"> flex
    ▶<div class="result__LiveSubText-sc-1bsijpp-2 iKYXwX" style="height: auto; overflow: visible; visi
    bility: visible; transition: height 1s ease 0s, visibility 1s ease 0s;">…</div> == $0
    </div>
  </div>
  <div></div>
</div>
```

... result__Repulsor-sc-1bsijpp-4.dZGBTm div.result__FlippingContainer-sc-1bsijpp-5.bIUfqL div.result__LiveSubText-sc-1bsijpp-2.iKYXwX ...

```
element.style {
  height: auto;
  overflow: ▶ visible;
  visibility: visible;
  transition: ▶ height 1s ease 0s, visibility
    1s ease 0s;
}
```

```
def conversion(c1,c2,amount):
    code_url = 'https://www.xe.com/currencyconverter/convert/?
Amount='+ str(amount) + '&From=' + c1 + '&To=' + c2
    response = requests.get(code_url)
    doc = BeautifulSoup(response.text, 'html.parser')
    if response.status_code != 200:
      raise Exception('Failed to load page {}'.format(code_url))
    conv = doc.find_all('p',{'class':'result__BigRate-sc-1bsijpp-1
iGrAod'})
    time = doc.find_all('div',{'class':'result__LiveSubText-sc-
1bsijpp-2 iKYXwX'})
    return conv[0].text + " [" + time[0].text + " ]"
```

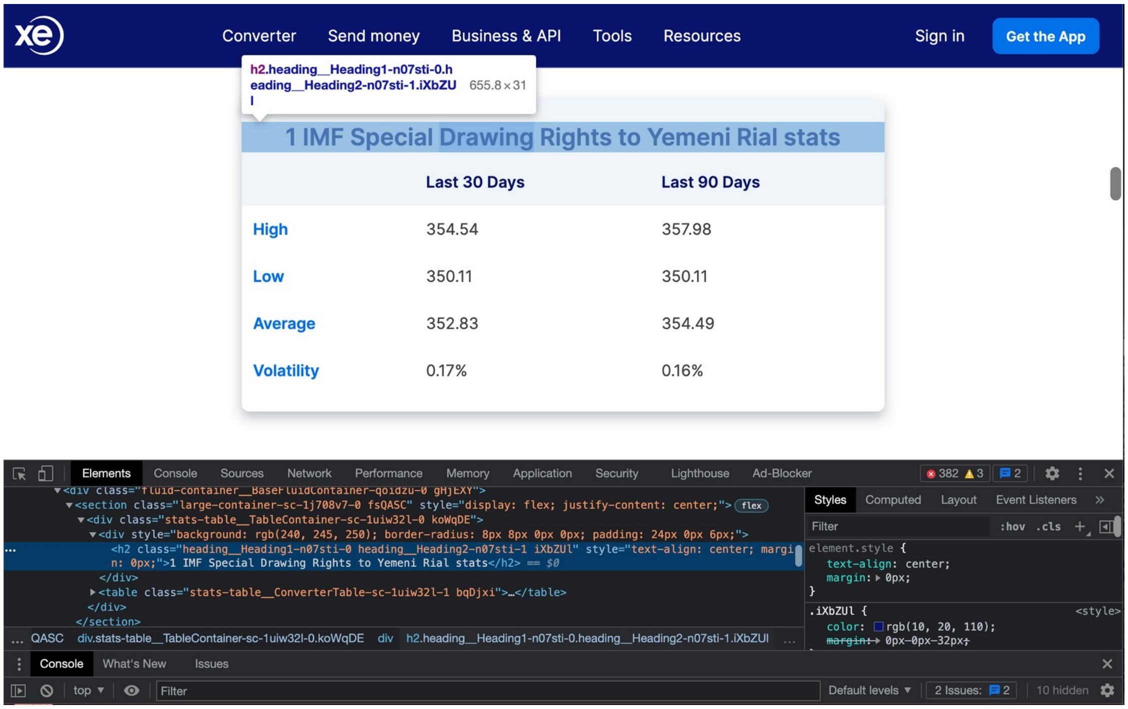
Function 3 : stats_conv(c1, c2, amount)

This function prints the stats table shown in the picture below. The stats table is unique and depends on the input that the user gives.

Here, we have argumemets as *c1*, *c2* and *amount*. *c1* and *c2* refer to country codes and *amount* refers to the amount that needs to be converted. The *code_url* is customized as per the link of the webpage, using *amount* that is obtained from the argument. *response* is a request object and *response.text* is the HTML code of the webpage.

response.text and 'html.parser' is fed to BeautifulSoup to obtained parsed document called *doc*. Then, a check takes place where the status code of the response object is checked. Anything other than 200 is an error and appropriate exception is kept raedy to deal with the error.

The *find_all()* function is then used to exatract the heading as shown. The code below and the html code in the image can be compared to understand why code has been written like this.



1 IMF Special Drawing Rights to Yemeni Rial stats

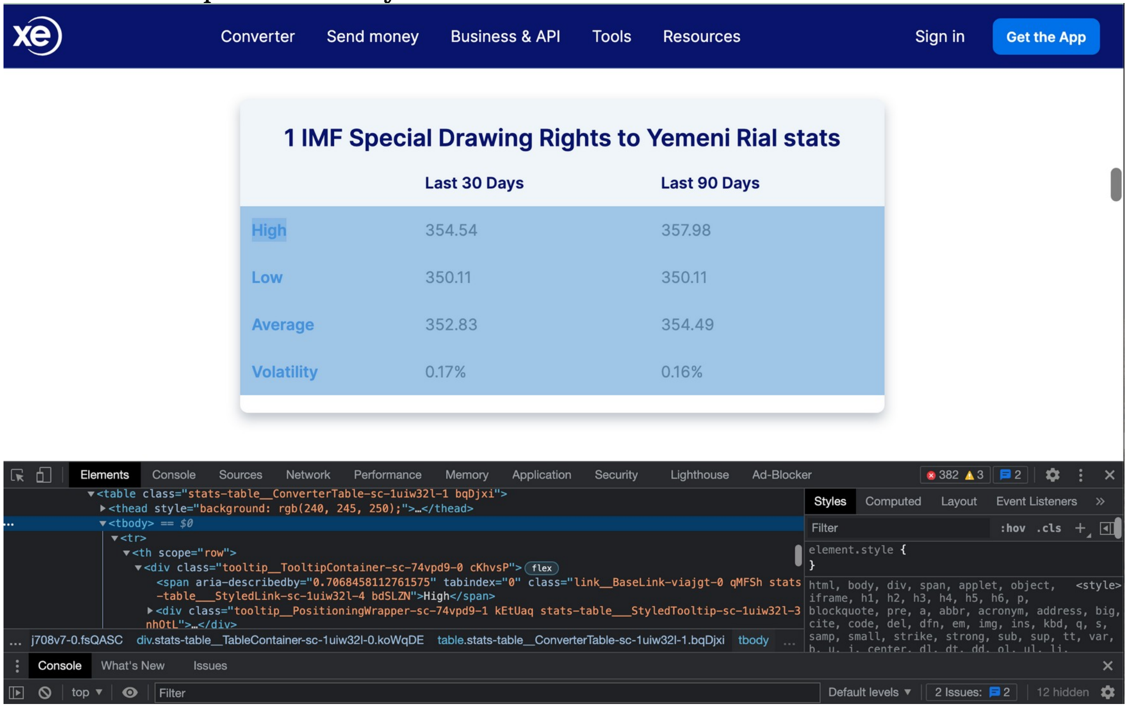
	Last 30 Days	Last 90 Days
High	354.54	357.98
Low	350.11	350.11
Average	352.83	354.49
Volatility	0.17%	0.16%

```
▼<div class="fluid-container__BaseFluidContainer-qoidzu-0 gHjEXY">
 ▼<section class="large-container-sc-1j708v7-0 fsQASC" style="display: flex; justify-content: center;"> flex
  ▼<div class="stats-table__TableContainer-sc-1uiw32l-0 koWqDE">
   ▼<div style="background: rgb(240, 245, 250); border-radius: 8px 8px 0px 0px; padding: 24px 0px 6px;">
      <h2 class="heading__Heading1-n07sti-0 heading__Heading2-n07sti-1 iXbZUl" style="text-align: center; margi
      n: 0px;">1 IMF Special Drawing Rights to Yemeni Rial stats</h2> == $0
     </div>
   ▶<table class="stats-table__ConverterTable-sc-1uiw32l-1 bqDjxi">…</table>
   </div>
  </section>
```

```
element.style {
  text-align: center;
  margin: ▶ 0px;
}
.iXbZUl {                                    <style>
  color: rgb(10, 20, 110);
  margin: ▶ 0px-0px-32px;
```

```
… QASC  div.stats-table__TableContainer-sc-1uiw32l-0.koWqDE  div  h2.heading__Heading1-n07sti-0.heading__Heading2-n07sti-1.iXbZUl …
```

To extract the table shown in the picture, the following code is written (lines starting with *sub* and *row*) and then a table is created using *PrettyTable()*. The rows are added to the table one by one by extarcting information bit by bit from the webpage, keeping in mind all the functions required to carry out the same.



1 IMF Special Drawing Rights to Yemeni Rial stats

	Last 30 Days	Last 90 Days
High	354.54	357.98
Low	350.11	350.11
Average	352.83	354.49
Volatility	0.17%	0.16%

```
▼<table class="stats-table__ConverterTable-sc-1uiw32l-1 bqDjxi">
 ▶<thead style="background: rgb(240, 245, 250);">…</thead>
 ▼<tbody> == $0
  ▼<tr>
   ▼<th scope="row">
    ▼<div class="tooltip__TooltipContainer-sc-74vpd9-0 cKhvsP"> flex
      <span aria-describedby="0.7068458112761575" tabindex="0" class="link__BaseLink-viajgt-0 qMFSh stats
      -table___StyledLink-sc-1uiw32l-4 bdSLZN">High</span>
     ▶<div class="tooltip__PositioningWrapper-sc-74vpd9-1 kEtUaq stats-table___StyledTooltip-sc-1uiw32l-3
     nhOtL">…</div>
```

```
element.style {
}
html, body, div, span, applet, object,       <style>
iframe, h1, h2, h3, h4, h5, h6, p,
blockquote, pre, a, abbr, acronym, address, big,
cite, code, del, dfn, em, img, ins, kbd, q, s,
samp, small, strike, strong, sub, sup, tt, var,
```

```
… j708v7-0.fsQASC  div.stats-table__TableContainer-sc-1uiw32l-0.koWqDE  table.stats-table__ConverterTable-sc-1uiw32l-1.bqDjxi  tbody …
```

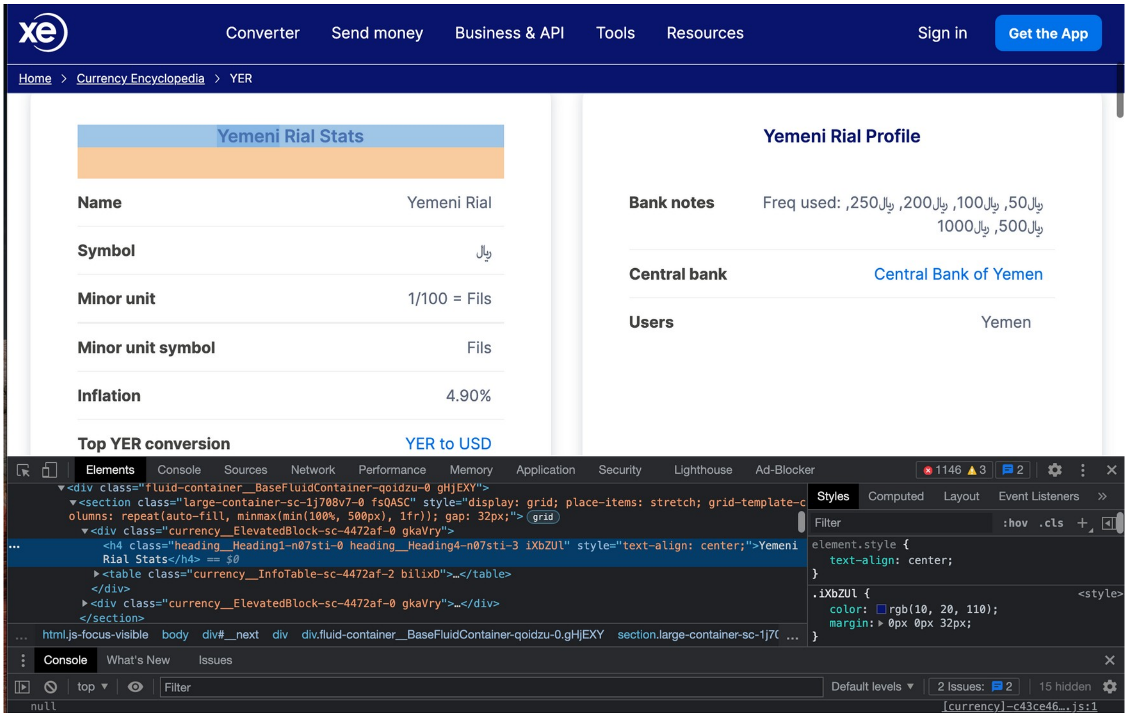
The details about the functions in prettytable library can be found in its documentation.

```
 def stats_conv(c1,c2,amount):
        code_url = 'https://www.xe.com/currencyconverter/convert/?
Amount='+ str(amount) + '&From=' + c1 + '&To=' + c2
        response = requests.get(code_url)
        doc = BeautifulSoup(response.text, 'html.parser')
        if response.status_code != 200:
            raise Exception('Failed to load page {}'.format(code_url))
        stats= doc.find_all('h2',{'class':'heading__Heading1-n07sti-0
heading__Heading2-n07sti-1 iXbZUl'})
        sub = doc.find_all('th',{'class':'stats-table__ColumnHeading-
sc-1uiw32l-2 kqQaYk'})
        row = doc.find_all('div',{'class':'tooltip__TooltipContainer-
sc-74vpd9-0 cKhvsP'})
        row_values = doc.find_all('td')
        stats_table = PrettyTable()
        stats_table.field_names = [" ",sub[0].text,sub[1].text]
        stats_table.title  = stats[0].text

stats_table.add_row([row[7].text[0:4],row_values[44].text,row_values[4
5].text])

stats_table.add_row([row[8].text[0:3],row_values[46].text,row_values[4
7].text])

stats_table.add_row([row[9].text[0:7],row_values[48].text,row_values[4
9].text])

stats_table.add_row([row[10].text[0:10],row_values[50].text,row_values
[50].text])

        print(stats_table)
```

Function 4 : stat1(doc)

This function is used to display stats of the currency given as input.

Here, the parsed document *doc* is the argument once again. It is used to call *find_all()* to extract the heading of the Stats table shown here.


xe
Converter
Send money
Business & API
Tools
Resources
Sign in
Get the App
Home > Currency Encyclopedia > YER
Elements Console Sources Network Performance Memory Application Security Lighthouse Ad-Blocker
1146 3 2
Styles Computed Layout Event Listeners
Filter :hov .cls
Console What's New Issues
top
Filter
Default levels
2 Issues: 2
15 hidden

Yemeni Rial Stats

Name	Yemeni Rial
Symbol	﷼
Minor unit	1/100 = Fils
Minor unit symbol	Fils
Inflation	4.90%
Top YER conversion	YER to USD

Yemeni Rial Profile

Bank notes	Freq used: ﷼50, ﷼100, ﷼200, ﷼250, ﷼500, ﷼1000
Central bank	Central Bank of Yemen
Users	Yemen

```
▼<div class="fluid-container__BaseFluidContainer-qoidzu-0 gHjEXY">
  ▼<section class="large-container-sc-1j708v7-0 fsQASC" style="display: grid; place-items: stretch; grid-template-columns: repeat(auto-fill, minmax(min(100%, 500px), 1fr)); gap: 32px;"> grid
    ▼<div class="currency__ElevatedBlock-sc-4472af-0 gkaVry">
      <h4 class="heading__Heading1-n07sti-0 heading__Heading4-n07sti-3 iXbZUl" style="text-align: center;">Yemeni
      Rial Stats</h4> == $0
     ▶<table class="currency__InfoTable-sc-4472af-2 bilixD">…</table>
     </div>
   ▶<div class="currency__ElevatedBlock-sc-4472af-0 gkaVry">…</div>
  </section>
```

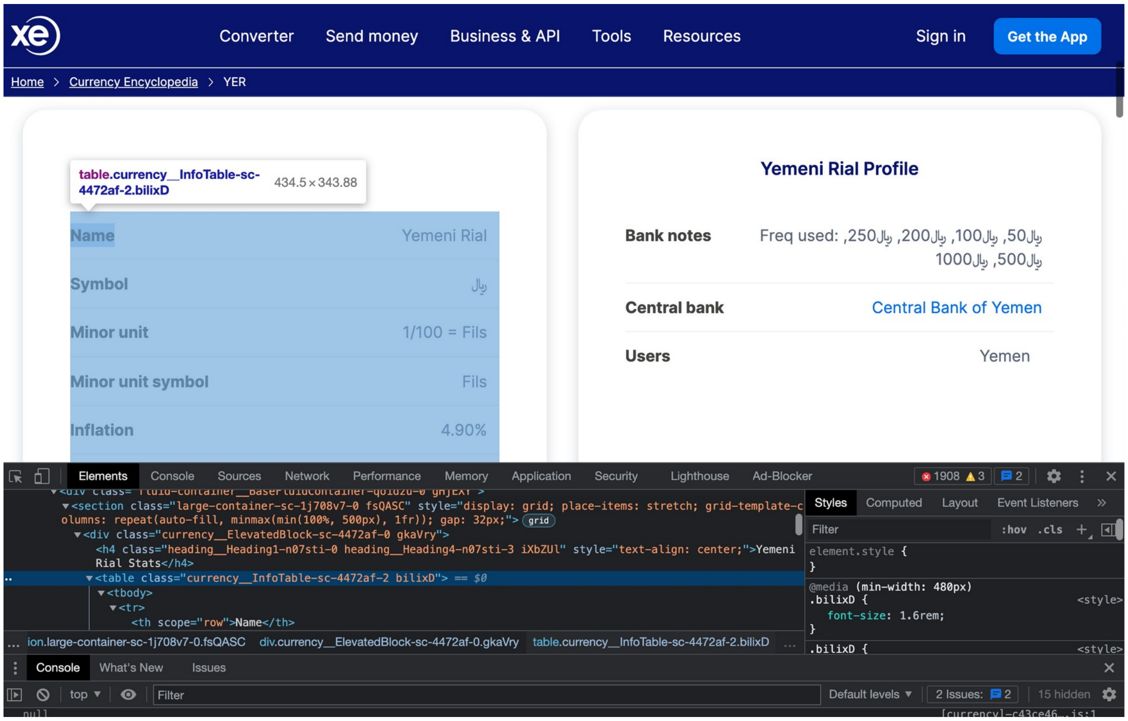
```
… html.js-focus-visible  body  div#__next  div  div.fluid-container__BaseFluidContainer-qoidzu-0.gHjEXY  section.large-container-sc-1j7( …
```

```
element.style {
  text-align: center;
}
.iXbZUl {                                  <style>
  color: rgb(10, 20, 110);
  margin: ▶ 0px 0px 32px;
}
```

```
null                                       [currency]-c43ce46….js:1
```

A table is created once again using *PrettyTable()* and field names (column names) are assigned to it. Now, *Heading* has two headings from the two tables shown in the picture above. The first heading is selected by specifying *Heading[0]* and later on (as we shall see), the second heading is used by specifying *Heading[1]*.

Again, 'table' tags are extracted to get the table.


xe
Converter
Send money
Business & API
Tools
Resources
Sign in
Get the App
Home > Currency Encyclopedia > YER
table.currency__InfoTable-sc-4472af-2.bilixD 434.5 × 343.88
Elements Console Sources Network Performance Memory Application Security Lighthouse Ad-Blocker
1908 3 2
Styles Computed Layout Event Listeners
Filter :hov .cls
Console What's New Issues
top
Filter
Default levels
2 Issues: 2
15 hidden

Name	Yemeni Rial
Symbol	﷼
Minor unit	1/100 = Fils
Minor unit symbol	Fils
Inflation	4.90%

Yemeni Rial Profile

Bank notes	Freq used: ﷼50, ﷼100, ﷼200, ﷼250, ﷼500, ﷼1000
Central bank	Central Bank of Yemen
Users	Yemen

```
▼<div class="fluid-container__BaseFluidContainer-qoidzu-0 gHjEXY">
  ▼<section class="large-container-sc-1j708v7-0 fsQASC" style="display: grid; place-items: stretch; grid-template-columns: repeat(auto-fill, minmax(min(100%, 500px), 1fr)); gap: 32px;"> grid
    ▼<div class="currency__ElevatedBlock-sc-4472af-0 gkaVry">
      <h4 class="heading__Heading1-n07sti-0 heading__Heading4-n07sti-3 iXbZUl" style="text-align: center;">Yemeni
      Rial Stats</h4>
     ▼<table class="currency__InfoTable-sc-4472af-2 bilixD"> == $0
       ▼<tbody>
         ▼<tr>
            <th scope="row">Name</th>
```

```
… ion.large-container-sc-1j708v7-0.fsQASC  div.currency__ElevatedBlock-sc-4472af-0.gkaVry  table.currency__InfoTable-sc-4472af-2.bilixD …
```

```
element.style {
}
@media (min-width: 480px)
.bilixD {                                  <style>
  font-size: 1.6rem;
}
.bilixD {                                  <style>
```

```
null                                       [currency]-c43ce46….js:1
```

The rows are added one by one to *stats_curr_table* by using *add_row* function. There is a case that needs to be handled here - sometimes 'Minor unit symbol' is empty. To handle this, the if else cases have been written. Sometimes, 'Minor unit' might also be empty, but that is automatically taken care without any if else cases.

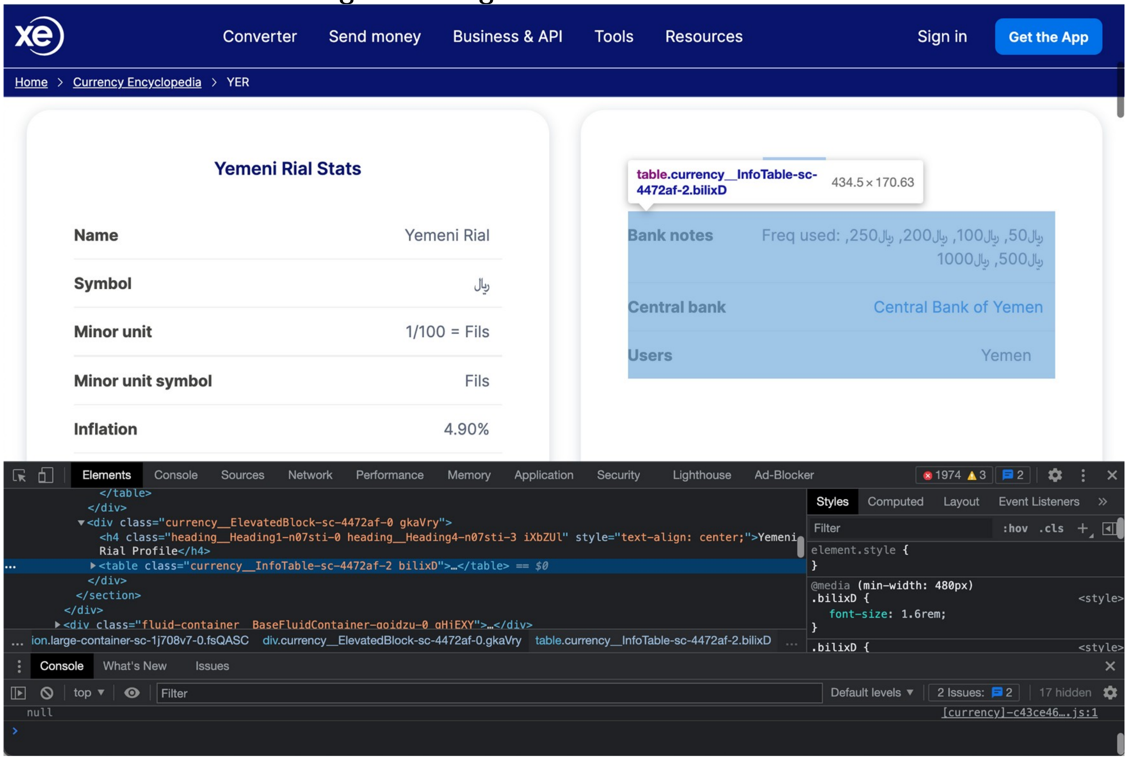
In the end, the *stats_curr_table* is returned.

```
def stat1(doc):
    Heading = doc.find_all('h4',{'class','heading__Heading1-
n07sti-0 heading__Heading4-n07sti-3 iXbZUl'})
    stats_curr_table = PrettyTable()
    stats_curr_table.field_names = [" "," Details"]
    stats_curr_table.title = Heading[0].text
    Table = doc.find_all('table',{'class','currency__InfoTable-sc-
4472af-2 bilixD'})
    stats_curr_table.add_row((Table[0].find_all('tr')[0].text[0:4]
+ " " + Table[0].find_all('tr')[0].text[4:]).split(" ",1))
    stats_curr_table.add_row((Table[0].find_all('tr')[1].text[0:6]
+ " " + Table[0].find_all('tr')[1].text[6:]).split(" ",1))
    L1 = (Table[0].find_all('tr')[2].text[0:10] +" " +
Table[0].find_all('tr')[2].text[10:]).split(" ",2)
    stats_curr_table.add_row((" ".join(L1[0:2])).split("-") +
L1[2:]  )
    L2 = (Table[0].find_all('tr')[3].text[0:17] + " " +
Table[0].find_all('tr')[3].text[17:]).split()
    if(len((" ".join(L2[0:3])).split("-") + L2[3:]) == 1):
      stats_curr_table.add_row((" ".join(L2[0:3])).split("-") +
L2[3:] + [' '])
    else:
      stats_curr_table.add_row((" ".join(L2[0:3])).split("-") +
L2[3:])


    return stats_curr_table
```

Function 5 : stat2(doc)

This function is used to display the profile of the currency given as input.

Here, the parsed document *doc* is the argument once again. It is used to call *find_all()* to extract the heading of the Profile table shown here. A table is once again created using *PrettyTable()*. *Heading[1]* is used as the second heading is what we are looking for this time. The is extracted using 'table' tag.



Yemeni Rial Stats

Name	Yemeni Rial
Symbol	ريال
Minor unit	1/100 = Fils
Minor unit symbol	Fils
Inflation	4.90%

Bank notes	Freq used: ريال50, ريال100, ريال200, ريال250, ريال500, ريال1000
Central bank	Central Bank of Yemen
Users	Yemen

```
        </table>
      </div>
    ▼<div class="currency__ElevatedBlock-sc-4472af-0 gkaVry">
        <h4 class="heading__Heading1-n07sti-0 heading__Heading4-n07sti-3 iXbZUl" style="text-align: center;">Yemeni
        Rial Profile</h4>
...    ▶<table class="currency__InfoTable-sc-4472af-2 bilixD">…</table> == $0
      </div>
    </section>
  </div>
 ▶<div class="fluid-container  BaseFluidContainer-goidzu-0 gHiEXY">…</div>
... ion.large-container-sc-1j708v7-0.fsQASC   div.currency__ElevatedBlock-sc-4472af-0.gkaVry   table.currency__InfoTable-sc-4472af-2.bilixD ...
```

```
element.style {
}
@media (min-width: 480px)
.bilixD {                                 <style>
    font-size: 1.6rem;
}
.bilixD {                                 <style>
```

```
null                                  [currency]-c43ce46….js:1
>
```

There are maximum 5 rows possible for the table: Nicknames, Coins, Bank notes, Central bank and Users (always in the same order). Now, it is possible to have only Users row; only Central Bank and Users rows; only Bank notes, Central bank and Users rows and so on like this. All these cases need to be handled, that is why a number of if elif cases are considered as shown in the code below.

For the first if, it checks (using *find_all()*) whether the only row is Users row or not, if yes, then it extracts that row and returns the *stats_curr_table*. (A corner case is handled here when the no. of users is more than three : '...' is added after three users and this detail is added as the row of the table. This is taken care in all if elif cases that follow.)

For the elif that follows, it checks (using *find_all()*) whether the first row is Central Bank or not, if yes, then Central bank row and Users rows are added to the table and table is returned.

For the second elif, it checks (using *find_all()*) whether the first row has 'Bank' in it or not. If yes, then Bank notes, Central bank and Users rows are added to the table and the table is returned.

The same pattern is followed for the next one and the last else case is for the case when all the rows are present.

One corner case occures for Bank notes and Coins, where there is additional 'Rarely used' information. That is also taken care for all currencies using *re.search()* (imported re for this) function and searching for the word 'Rarely' in the Coins and Bank notes rows.

```
def stat2(doc):
        Heading = doc.find_all('h4',{'class','heading__Heading1-
n07sti-0 heading__Heading4-n07sti-3 iXbZUl'})
        stats_curr_table = PrettyTable()
        stats_curr_table.field_names = [" "," Details"]
        stats_curr_table.title = Heading[1].text
        Table = doc.find_all('table',{'class','currency__InfoTable-sc-
4472af-2 bilixD'})


        if(Table[1].find_all('tr')[0].text[0:5].split("-")[0] ==
'Users'):
            list1 = (Table[1].find_all('tr')[0].text[0:5].split("-"))
            list2 = (doc.find_all('span',
{'class':'currency__OneLineSpan-sc-4472af-3 bzUWsw'})
[0].text.split("-"))
            if(len(list2[0].split(','))>3):
                list2 = list2[0].split(',')[0:3] + ['...']
                list2 = (','.join(list2)).split('-')

            stats_curr_table.add_row(list1+list2)


        elif((Table[1].find_all('tr')[0].text[0:12] +" " +
Table[1].find_all('tr')[0].text[12:]).split(" ",2)[0] == 'Central
bank'):
            L2 = (Table[1].find_all('tr')[0].text[0:12] +" " +
Table[1].find_all('tr')[0].text[12:]).split(" ",2)
            stats_curr_table.add_row((" ".join(L2[0:2])).split("-") +
L2[2:]  )
            list1 = (Table[1].find_all('tr')[i].text[0:5].split("-"))
            list2 = (doc.find_all('span',
{'class':'currency__OneLineSpan-sc-4472af-3 bzUWsw'})
[0].text.split("-"))
            if(len(list2[0].split(','))>3):
                list2 = list2[0].split(',')[0:3] + ['...']
                list2 = (','.join(list2)).split('-')

            stats_curr_table.add_row(list1+list2)




        elif((Table[1].find_all('tr')[0].text[0:5] + " " +
Table[1].find_all('tr')[0].text[5:]).split(" ")[0] == 'Bank'):
            L1 = (Table[1].find_all('tr')[0].text[0:10] + " " +
Table[1].find_all('tr')[0].text[10:]).split(" ",2)
            if(re.search('Rarely',L1[1:][0])):
                x = L1[1:][0].find('Rarely')
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + (L1[2:][0][:x] + " " + L1[2:][0]
[x:]).split("-"))
            else:
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + L1[2:]  )
            L2 = (Table[1].find_all('tr')[1].text[0:12] +" " +
Table[1].find_all('tr')[1].text[12:]).split(" ",2)
            stats_curr_table.add_row((" ".join(L2[0:2])).split("-") +
L2[2:]  )
            list1 = (Table[1].find_all('tr')[i].text[0:5].split("-"))
            list2 = (doc.find_all('span',
{'class':'currency__OneLineSpan-sc-4472af-3 bzUWsw'})
[0].text.split("-"))
            if(len(list2[0].split(','))>3):
                list2 = list2[0].split(',')[0:3] + ['...']
                list2 = (','.join(list2)).split('-')


            stats_curr_table.add_row(list1+list2)

        elif ((Table[1].find_all('tr')[0].text[0:5] + " " +
Table[1].find_all('tr')[0].text[5:]).split(" ",1)[0] == 'Coins'):
            L0 = (Table[1].find_all('tr')[0].text[0:5] + " " +
Table[1].find_all('tr')[0].text[5:]).split(" ",1)
            if(re.search('Rarely',L0[1:][0])):
                x = L0[1:][0].find('Rarely')
                stats_curr_table.add_row(" ".join(L0[0:1]).split("-")
+ (L0[1:][0][:x] + " " + L0[1:][0][x:]).split('-'))
            else:
                stats_curr_table.add_row((Table[1].find_all('tr')
[0].text[0:5] + " " + Table[1].find_all('tr')[0].text[5:]).split("
",1))

            L1 = (Table[1].find_all('tr')[1].text[0:10] + " " +
Table[1].find_all('tr')[1].text[10:]).split(" ",2)

            if(re.search('Rarely',L1[2:][0])):
                x = L1[2:][0].find('Rarely')
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + (L1[2:][0][:x] + " " + L1[2:][0]
[x:]).split("-"))
            else:
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + L1[2:]  )


            L2 = (Table[1].find_all('tr')[2].text[0:12] +" " +
Table[1].find_all('tr')[2].text[12:]).split(" ",2)
            stats_curr_table.add_row((" ".join(L2[0:2])).split("-") +
L2[2:]  )
```

```
            list1 = (Table[1].find_all('tr')[3].text[0:5].split("-"))
            list2 = (doc.find_all('span',
{'class':'currency__OneLineSpan-sc-4472af-3 bzUWsw'})
[0].text.split("-"))
            if(len(list2[0].split(','))>=3):
                list2 = list2[0].split(',')[0:3] + ['...']
                list2 = (','.join(list2)).split('-')

            stats_curr_table.add_row(list1+list2)



        else:

            if(Table[1].find_all('tr')[0].text[0:9] == 'Nicknames'):
                stats_curr_table.add_row((Table[1].find_all('tr')
[0].text[0:9] + " " + Table[1].find_all('tr')[0].text[9:]).split("
",1))


            L0 = (Table[1].find_all('tr')[1].text[0:5] + " " +
Table[1].find_all('tr')[1].text[5:]).split(" ",1)
            if(re.search('Rarely',L0[1:][0])):
                x = L0[1:][0].find('Rarely')
                stats_curr_table.add_row(" ".join(L0[0:1]).split("-")
+ (L0[1:][0][:x] + " " + L0[1:][0][x:]).split('-'))
            else:
                stats_curr_table.add_row((Table[1].find_all('tr')
[1].text[0:5] + " " + Table[1].find_all('tr')[1].text[5:]).split("
",1))




            L1 = (Table[1].find_all('tr')[2].text[0:10] + " " +
Table[1].find_all('tr')[2].text[10:]).split(" ",2)

            if(re.search('Rarely',L1[2:][0])):
                x = L1[2:][0].find('Rarely')
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + (L1[2:][0][:x] + " " + L1[2:][0]
[x:]).split("-"))
            else:
                stats_curr_table.add_row(("
".join(L1[0:2])).split("-") + L1[2:]  )

            L2 = (Table[1].find_all('tr')[3].text[0:12] +" " +
Table[1].find_all('tr')[3].text[12:]).split(" ",2)
```

```
            stats_curr_table.add_row((" ".join(L2[0:2])).split("-") +
L2[2:]  )

            list1 = (Table[1].find_all('tr')[4].text[0:5].split("-"))
            list2 = (doc.find_all('span',
{'class':'currency__OneLineSpan-sc-4472af-3 bzUWsw'})
[0].text.split("-"))
            if(len(list2[0].split(','))>3):
                list2 = list2[0].split(',')[0:3] + ['...']
                list2 = (','.join(list2)).split('-')


            stats_curr_table.add_row(list1+list2)

        return(stats_curr_table)
```

Function 6 : stats_curr(c1, c2, df)

This function takes currency codes *c1*, *c2* and a dataframe consisting of name of the currency, code of the currency and links to the information regarding each currency.

The dataframe is used to obtain the links to the currency data using the currency codes that are passed as *c1* and *c2*.

As two links are involved, requests is used twice and status code is also checked twice. Two parse documents are generated, *doc1* and *doc2*. They are passed to *stat1()* and *stat2()* functions as arguments to obtain necessary stats and profiles of currencies.

Note that here also if elif has been used because currency code 'XBT' which corresponds to Bitcoin, does not have a profile table, unlike any other currency.

```
 def stats_curr(c1,c2,df):

        c1_link = df[df['Code'] == c1]['Links'].values[0]

        c2_link = df[df['Code'] == c2]['Links'].values[0]

        response = requests.get(c1_link)
        doc1 = BeautifulSoup(response.text, 'html.parser')
        if response.status_code != 200:
            raise Exception('Failed to load page {}'.format(c1_link))

        response = requests.get(c2_link)
        doc2 = BeautifulSoup(response.text, 'html.parser')
        if response.status_code != 200:
            raise Exception('Failed to load page {}'.format(c2_link))

        if(c1 == 'XBT'):
            print(stat1(doc1))
            print(stat1(doc2))
            print(stat2(doc2))
        elif(c2 == 'XBT'):
            print(stat1(doc1))
            print(stat1(doc2))
            print(stat2(doc1))
        else:
            print(stat1(doc1))
            print(stat1(doc2))
            print(stat2(doc1))
            print(stat2(doc2))
        return
```

Function 7 : renkon( )

This is the main driver function of the entire program and it calls all other functions of the program, directly or indirectly.
The url of the currency encyclopedia is taken to generate the request object, which leads to the development of *doc* parsed document. *currency_info()* function is then called and *doc* is passed as the argument to this function. The first two columns of the dataframe obtained as an output of this function is converted to a table using *tabulate()* function and displayed for the user so as to act as a reference for currency codes.

The remaining of the entire function is kept indside while loop to keep *renkon()* running till the user tells it to stop. The necessary inputs : currency codes (case insensitive) and amount are taken as inputs from the user and the converted amount, stats and profiles for the two currencies entered as input, are displyed for the user.
On entering 0 when aksed, the loop stops and the execution of *renkon()* function comes to an end successfully.

```
def renkon():
    url = 'https://www.xe.com/currency/'
    response = requests.get(url)
    doc = BeautifulSoup(response.text, 'html.parser')
    if response.status_code != 200:
        raise Exception('Failed to load page {}'.format(url))
    cur_df = currency_info(doc)
    print('\t\t\t\tRENKON: CURRENCY CONVERTER\n')
    with pd.option_context('display.max_rows', None, 
'display.max_columns', None):
        print(tabulate(cur_df[['Name','Code']], headers='keys', 
tablefmt='psql',showindex = False))
    y = '1'
    while(y != '0'):
        user_code1 = input('From: Enter Currency Code (Refer table 
above for functional codes)')
        user_code2 = input('To: Enter Currency Code (Refer table above 
for functional codes)')
```

```python
        status = 1
        user_code1 = user_code1.upper()
        user_code2 = user_code2.upper()
        if cur_df[cur_df['Code'] == user_code1].empty:
            print('Please enter correct code, {} doesn\'t 
exist.'.format(user_code1.upper()))
            status = 0
        elif cur_df[cur_df['Code'] == user_code2].empty:
            print('Please enter correct code! {} doesn\'t 
exist.'.format(user_code2.upper()))
            status = 0
        if(status == 1):
            user_m1 = input('Enter the amount you have\n')
            money = conversion(user_code1,user_code2,user_m1)
            print('It\'s {}\n'.format(money))
            stats_conv(user_code1,user_code2,user_m1)
            stats_curr(user_code1,user_code2,cur_df)


        print('Do you want to try once again?\n')
        y = input('Enter 0 to stop OR any other number to continue.\
n')
    print('Thank you for using RENKON!')
    return

renkon()
```

```
                      RENKON: CURRENCY CONVERTER

+--------------------------------+--------+
| Name                           | Code   |
|--------------------------------+--------|
| Emirati Dirham                 | AED    |
| Afghan Afghani                 | AFN    |
| Albanian Lek                   | ALL    |
| Armenian Dram                  | AMD    |
| Dutch Guilder                  | ANG    |
| Angolan Kwanza                 | AOA    |
| Argentine Peso                 | ARS    |
| Australian Dollar              | AUD    |
| Aruban or Dutch Guilder        | AWG    |
| Azerbaijan Manat               | AZN    |
| Bosnian Convertible Mark       | BAM    |
| Barbadian or Bajan Dollar      | BBD    |
| Bangladeshi Taka               | BDT    |
| Bulgarian Lev                  | BGN    |
| Bahraini Dinar                 | BHD    |
| Burundian Franc                | BIF    |
| Bermudian Dollar               | BMD    |
| Bruneian Dollar                | BND    |
```

Bolivian Bolíviano	BOB
Brazilian Real	BRL
Bahamian Dollar	BSD
Bhutanese Ngultrum	BTN
Botswana Pula	BWP
Belarusian Ruble	BYN
Belarusian Ruble	BYR
Belizean Dollar	BZD
Canadian Dollar	CAD
Congolese Franc	CDF
Swiss Franc	CHF
Chilean Peso	CLP
Chinese Yuan Renminbi	CNY
Colombian Peso	COP
Costa Rican Colon	CRC
Cuban Convertible Peso	CUC
Cuban Peso	CUP
Cape Verdean Escudo	CVE
Czech Koruna	CZK
Djiboutian Franc	DJF
Danish Krone	DKK
Dominican Peso	DOP
Algerian Dinar	DZD
Estonian Kroon	EEK
Egyptian Pound	EGP
Eritrean Nakfa	ERN
Ethiopian Birr	ETB
Euro	EUR
Fijian Dollar	FJD
Falkland Island Pound	FKP
British Pound	GBP
Georgian Lari	GEL
Guernsey Pound	GGP
Ghanaian Cedi	GHS
Gibraltar Pound	GIP
Gambian Dalasi	GMD
Guinean Franc	GNF
Guatemalan Quetzal	GTQ
Guyanese Dollar	GYD
Hong Kong Dollar	HKD
Honduran Lempira	HNL
Croatian Kuna	HRK
Haitian Gourde	HTG
Hungarian Forint	HUF
Indonesian Rupiah	IDR
Israeli Shekel	ILS
Isle of Man Pound	IMP
Indian Rupee	INR
Iraqi Dinar	IQD
Iranian Rial	IRR

Icelandic Krona	ISK
Jersey Pound	JEP
Jamaican Dollar	JMD
Jordanian Dinar	JOD
Japanese Yen	JPY
Kenyan Shilling	KES
Kyrgyzstani Som	KGS
Cambodian Riel	KHR
Comorian Franc	KMF
North Korean Won	KPW
South Korean Won	KRW
Kuwaiti Dinar	KWD
Caymanian Dollar	KYD
Kazakhstani Tenge	KZT
Lao Kip	LAK
Lebanese Pound	LBP
Sri Lankan Rupee	LKR
Liberian Dollar	LRD
Basotho Loti	LSL
Lithuanian Litas	LTL
Latvian Lat	LVL
Libyan Dinar	LYD
Moroccan Dirham	MAD
Moldovan Leu	MDL
Malagasy Ariary	MGA
Macedonian Denar	MKD
Burmese Kyat	MMK
Mongolian Tughrik	MNT
Macau Pataca	MOP
Mauritanian Ouguiya	MRU
Mauritian Rupee	MUR
Maldivian Rufiyaa	MVR
Malawian Kwacha	MWK
Mexican Peso	MXN
Malaysian Ringgit	MYR
Mozambican Metical	MZN
Namibian Dollar	NAD
Nigerian Naira	NGN
Nicaraguan Cordoba	NIO
Norwegian Krone	NOK
Nepalese Rupee	NPR
New Zealand Dollar	NZD
Omani Rial	OMR
Panamanian Balboa	PAB
Peruvian Sol	PEN
Papua New Guinean Kina	PGK
Philippine Peso	PHP
Pakistani Rupee	PKR
Polish Zloty	PLN
Paraguayan Guarani	PYG

Qatari Riyal	QAR
Romanian Leu	RON
Serbian Dinar	RSD
Russian Ruble	RUB
Rwandan Franc	RWF
Saudi Arabian Riyal	SAR
Solomon Islander Dollar	SBD
Seychellois Rupee	SCR
Sudanese Pound	SDG
Swedish Krona	SEK
Singapore Dollar	SGD
Saint Helenian Pound	SHP
Sierra Leonean Leone	SLL
Somali Shilling	SOS
Seborgan Luigino	SPL
Surinamese Dollar	SRD
Sao Tomean Dobra	STN
Salvadoran Colon	SVC
Syrian Pound	SYP
Swazi Lilangeni	SZL
Thai Baht	THB
Tajikistani Somoni	TJS
Turkmenistani Manat	TMT
Tunisian Dinar	TND
Tongan Pa'anga	TOP
Turkish Lira	TRY
Trinidadian Dollar	TTD
Tuvaluan Dollar	TVD
Taiwan New Dollar	TWD
Tanzanian Shilling	TZS
Ukrainian Hryvnia	UAH
Ugandan Shilling	UGX
US Dollar	USD
Uruguayan Peso	UYU
Uzbekistani Som	UZS
Venezuelan Bolívar	VEF
Venezuelan Bolívar	VES
Vietnamese Dong	VND
Ni-Vanuatu Vatu	VUV
Samoan Tala	WST
Central African CFA Franc BEAC	XAF
Silver Ounce	XAG
Gold Ounce	XAU
Bitcoin	XBT
East Caribbean Dollar	XCD
IMF Special Drawing Rights	XDR
CFA Franc	XOF
Palladium Ounce	XPD
CFP Franc	XPF
Platinum Ounce	XPT

```
| Yemeni Rial                    | YER    |
| South African Rand             | ZAR    |
| Zambian Kwacha                 | ZMK    |
| Zambian Kwacha                 | ZMW    |
| Zimbabwean Dollar              | ZWD    |
+--------------------------------+--------+
From: Enter Currency Code (Refer table above for functional codes)TRY
To: Enter Currency Code (Refer table above for functional codes)LBP
Enter the amount you have
3400.50
It's 456,330.64 Lebanese Pounds [Turkish Lira to Lebanese Pound
conversion — Last updated Nov 20, 2021, 12:21 UTC ]

+------------------------------------------+
|  1 Turkish Lira to Lebanese Pound stats  |
+------------+--------------+--------------+
|            | Last 30 Days | Last 90 Days |
+------------+--------------+--------------+
|    High    |    163.36    |    181.93    |
|    Low     |    134.13    |    134.13    |
|  Average   |    153.52    |    166.82    |
| Volatility |    1.18%     |    1.18%     |
+------------+--------------+--------------+

+-----------------------------------+
|         Turkish Lira Stats        |
+-------------------+---------------+
|                   |    Details    |
+-------------------+---------------+
|        Name       |  Turkish Lira |
|       Symbol      |       TL      |
|     Minor unit    | 1/100 = kuruş |
| Minor unit symbol |       Kr      |
+-------------------+---------------+
+-------------------------------------+
|         Lebanese Pound Stats        |
+-------------------+-----------------+
|                   |     Details     |
+-------------------+-----------------+
|        Name       |  Lebanese Pound |
|       Symbol      |       ل.ل    |
|     Minor unit    | 1/100 = Piastre |
| Minor unit symbol |     Piastre     |
+-------------------+-----------------+
+---------------------------------------------------------------------
--+
|                          Turkish Lira Profile
|
+--------------
+---------------------------------------------------------+
|              |                         Details
|
+--------------
+--------------------------------------------------------+
|  Nicknames   |                  Kağıt, Mangır, Papel
|
|    Coins     | Freq used: 5Kr, 10Kr, 25Kr, 50Kr, TL1 Rarely used:
1Kr |
|  Bank notes  |     Freq used: TL5, TL10, TL20, TL50, TL100, TL200
|
| Central bank |         Central Bank of the Republic of Turkey
|
|    Users     |                  Turkey, North Cyprus
|
+--------------
+--------------------------------------------------------+
+----------------------------------------------------------------------
---------------+
|                                Lebanese Pound Profile
|
+--------------
+----------------------------------------------------------------------
-+
|              |                                Details
|
+--------------
+----------------------------------------------------------------------
-+
|    Coins     |         Freq used: 500ل.ل ,250ل.ل Rarely used: ,50ل.ل
100ل.ل         |
|  Bank notes  | Freq used: ل.ل ,20000ل.ل ,10000ل.ل ,5000ل.ل ,1000ل.ل
100000ل.ل ,50000 |
| Central bank |                           Bangue du Liban
|
|    Users     |                               Lebanon
|
+--------------
+----------------------------------------------------------------------
-+
Do you want to try once again?

Enter 0 to stop OR any other number to continue.
0
Thank you for using RENKON!
```