In [ ]:

```
```

In [1]:

```
import requests
import lxml.html as lh
import pandas as pd
data = pd.DataFrame()
```

In [2]:

```
url = 'https://www.worldometers.info/coronavirus/' #assign the wiki page
#WHO_url = 'https://www.worldometers.info/coronavirus/'

page = requests.get(url) # create a handle for contents of the wiki page

doc = lh.fromstring(page.content) # store content of the wiki page under doc

tr_elements = doc.xpath('//tr') # parse data stored between tr in the html

[len(T) for T in tr_elements[:12]] # check the length of the first 12 rows
```

Out[2]:

```
[22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22]
```

In [3]:
```
tr_elements = doc.xpath('//tr') # parse first row as header

col = [] # create empty list
i = 0

for t in tr_elements[0]: # for each row, store each first element (header) and an empty list
    i+=1
    name=t.text_content()
    print("%d:%s" % (i,name))
    col.append((name,[]))
```

```
1:#
2:Country,Other
3:TotalCases
4:NewCases
5:TotalDeaths
6:NewDeaths
7:TotalRecovered
8:NewRecovered
9:ActiveCases
10:Serious,Critical
11:Tot Cases/1M pop
12:Deaths/1M pop
13:TotalTests
14:Tests/
1M pop

15:Population
16:Continent
17:1 Caseevery X ppl
18:1 Deathevery X ppl
19:1 Testevery X ppl
20:New Cases/1M pop
21:New Deaths/1M pop
22:Active Cases/1M pop
```

In [4]:
```
for j in range(1,len(tr_elements)): # Because header is the first row, data would be store in the subsequent row
    T = tr_elements[j] #T is j'th row

    if len(T)!=22: #if row is not size 3, //tr data is not from the table.
        break

    i = 0 #i is the index of the first column

    for t in T.iterchildren(): #iterate through each element of the row
        data=t.text_content()

        col[i][1].append(data) #append the data to the empty list of the i'th column

        i+=1 #increment i for the next column
```

In [6]:
```
def dataframeCleaner(data):

    for columnname in data: #looping through titles of the table
        temp = []
        for column in data[columnname]:   #geting column elements for the each title
            column = str(column)
            column = column.replace(',','')# Removing unwanted data clutter
            column = column.replace('+','')#Removing unwanted '+'sign
            try:   #using try except block to convert datatype string to integer while avoiding error
                column = int(column)
            except:
                pass

            temp.append(column)
        data[columnname] = temp

    data = data.drop(data.tail(1).index) # Deleting the last row
    data = data.replace(r'^\s*$', 0, regex=True)# converting empty string to 0
    return data
```

In [7]:
```
data
```

Out[7]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	...
0		\nNorth America\n	41,869,772	+17,938	933,623	+372	34,970,438	+9,884	5,965,711	13,788	...
1		\nAsia\n	60,213,546	+159,029	865,923	+2,832	56,652,096	+129,488	2,695,527	31,353	...
2		\nSouth America\n	35,025,677	+2,416	1,074,277	+120	32,497,876	+3,548	1,453,524	26,461	...
3		\nEurope\n	50,584,099	+48,614	1,125,795	+892	46,452,986	+36,685	3,005,318	6,841	...
4		\nAfrica\n	6,455,740	+1,527	162,863	+29	5,650,803	+1,623	642,074	4,293	...
...	...	...	...	...	...	...	...	...	...	...	...
711		Total:	50,394,384	+146,000	1,123,859	+1,060	46,365,304	+46,865	2,905,221	6,706	...
712		Total:	6,423,318	+38,672	162,037	+995	5,610,144	+37,465	651,137	4,335	...
713		Total:	94,045	+1,178	1,447	+16	73,868	+227	18,730	65	...
714		Total:	721		15		706		0	0	...
715		Total:	193,440,579	+571,482	4,151,058	+8,930	175,713,125	+389,081	13,576,396	82,370	...

716 rows × 22 columns

In [8]:
```
data.dtypes
```

Out[8]:
```
#                      object
Country,Other          object
TotalCases             object
NewCases               object
TotalDeaths            object
NewDeaths              object
TotalRecovered         object
NewRecovered           object
ActiveCases            object
Serious,Critical       object
Tot Cases/1M pop       object
Deaths/1M pop          object
TotalTests             object
Tests/\n1M pop\n       object
Population             object
Continent              object
1 Caseevery X ppl      object
1 Deathevery X ppl     object
1 Testevery X ppl      object
New Cases/1M pop       object
New Deaths/1M pop      object
Active Cases/1M pop    object
dtype: object
```

In [12]:
```
# Get names of indexes for which column
indexNames = data[data['Continent'] == 'All'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Continent'] == 'Continent'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Continent'] == 0].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
```

In [14]:
```
# Get names of indexes for which column
indexNames = data[data['Country,Other'] == 'Total:'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'Africa'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'Asia'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'Europe'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'North America'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'South America'].index
# Delete these row indexes from dataFrame
data.drop(indexNames , inplace= True)
indexNames = data[data['Country,Other'] == 'Australia/Oceania'].index
# Delete these roindexNames = indexes from dataFrame
data.drop(indexNames , inplace= True)
```

In [15]:
```python
# Function to Clean the DataSet
def dataframeCleaner(data):

    for columnname in data: #looping through titles of the table
        temp = []
        for column in data[columnname]:   #geting column elements for the each title
            column = str(column)
            column = column.replace(',','')# Removing unwanted data clutter
            column = column.replace('\n','')# Removing unwanted \n
            column = column.replace('N/A','')# Removing unwanted N/A
            column = column.replace('+','')#Removing unwanted '+'sign
            try:   #using try except block to convert datatype string to integer while avoiding error
                column = int(column)
            except:
                pass

            temp.append(column)
        data[columnname] = temp

    data = data.drop(data.tail(1).index) # Deleting the last row
    data = data.replace(r'^\s*$', 0, regex=True)# converting empty string to 0
    return data
```

In [16]:
```
data
```

Out[16]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	.
0		\nNorth America\n	41869772	17938	933623	372	34970438	9884	5965711	13788	.
1		\nAsia\n	60213546	159029	865923	2832	56652096	129488	2695527	31353	.
2		\nSouth America\n	35025677	2416	1074277	120	32497876	3548	1453524	26461	.
3		\nEurope\n	50584099	48614	1125795	892	46452986	36685	3005318	6841	.
4		\nAfrica\n	6455740	1527	162863	29	5650803	1623	642074	4293	.
...	...	...	...	...	...	...	...	...	...	...	.
703	218	Vanuatu	4		1		3		0		.
704	219	Marshall Islands	4				4		0		.
705	220	Samoa	3				3		0		.
706	221	Saint Helena	2				2		0		.
707	222	Micronesia	1				1		0		.

687 rows × 22 columns

In [19]:
```
# Cleaning the dataset using user defined function
data = dataframeCleaner(data)
data
```

Out[19]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical
0	0	North America	41869772	17938	933623	372	34970438	9884	5965711	13788
1	0	Asia	60213546	159029	865923	2832	56652096	129488	2695527	31353
2	0	South America	35025677	2416	1074277	120	32497876	3548	1453524	26461
3	0	Europe	50584099	48614	1125795	892	46452986	36685	3005318	6841
4	0	Africa	6455740	1527	162863	29	5650803	1623	642074	4293
...	...	...	...	...	...	...	...	...	...	...
701	216	Western Sahara	10	0	1	0	8	0	1	0
702	217	MS Zaandam	9	0	2	0	7	0	0	0
703	218	Vanuatu	4	0	1	0	3	0	0	0
704	219	Marshall Islands	4	0	0	0	4	0	0	0
705	220	Samoa	3	0	0	0	3	0	0	0

685 rows × 22 columns

In [20]:
```
data.dtypes
```

Out[20]:
```
#                        int64
Country,Other           object
TotalCases               int64
NewCases                 int64
TotalDeaths              int64
NewDeaths                int64
TotalRecovered           int64
NewRecovered             int64
ActiveCases              int64
Serious,Critical         int64
Tot Cases/1M pop         int64
Deaths/1M pop           object
TotalTests               int64
Tests/\n1M pop\n         int64
Population               int64
Continent               object
1 Caseevery X ppl        int64
1 Deathevery X ppl       int64
1 Testevery X ppl        int64
New Cases/1M pop        object
New Deaths/1M pop       object
Active Cases/1M pop     object
dtype: object
```

In [21]:
```
data = data.infer_objects()
```

In [22]:
```
data
```

Out[22]:

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	.
0	0	North America	41869772	17938	933623	372	34970438	9884	5965711	13788	.
1	0	Asia	60213546	159029	865923	2832	56652096	129488	2695527	31353	.
2	0	South America	35025677	2416	1074277	120	32497876	3548	1453524	26461	.
3	0	Europe	50584099	48614	1125795	892	46452986	36685	3005318	6841	.
4	0	Africa	6455740	1527	162863	29	5650803	1623	642074	4293	.
...	...	...	...	...	...	...	...	...	...	...	.
701	216	Western Sahara	10	0	1	0	8	0	1	0	.
702	217	MS Zaandam	9	0	2	0	7	0	0	0	.
703	218	Vanuatu	4	0	1	0	3	0	0	0	.
704	219	Marshall Islands	4	0	0	0	4	0	0	0	.
705	220	Samoa	3	0	0	0	3	0	0	0	.

685 rows × 22 columns

In [ ]:
```

```