Covid19 Data Analysis by Python

by Swarnadeep

Step 1 : Importing the modules

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
print('Modules are imported.')
```

Modules are imported.

Step 2 : Cleaning Covid19 dataset

2.1: importing covid19 dataset

importing "Covid19_Confirmed_dataset.csv" from "./Dataset" folder.

In [2]:
```
corona_dataset_csv = pd.read_csv('Datasets/covid19_Confirmed_dataset.csv')
corona_dataset_csv.head(10)
```

Out[2]:

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	...	4/21/20	4/22/20	4/23/20
0	NaN	Afghanistan	33.0000	65.0000	0	0	0	0	0	0	...	1092	1176	1279
1	NaN	Albania	41.1533	20.1683	0	0	0	0	0	0	...	609	634	663
2	NaN	Algeria	28.0339	1.6596	0	0	0	0	0	0	...	2811	2910	3007
3	NaN	Andorra	42.5063	1.5218	0	0	0	0	0	0	...	717	723	723
4	NaN	Angola	-11.2027	17.8739	0	0	0	0	0	0	...	24	25	25
5	NaN	Antigua and Barbuda	17.0608	-61.7964	0	0	0	0	0	0	...	23	24	24
6	NaN	Argentina	-38.4161	-63.6167	0	0	0	0	0	0	...	3031	3144	3435
7	NaN	Armenia	40.0691	45.0382	0	0	0	0	0	0	...	1401	1473	1523
8	Australian Capital Territory	Australia	-35.4735	149.0124	0	0	0	0	0	0	...	104	104	104
9	New South Wales	Australia	-33.8688	151.2093	0	0	0	0	3	4	...	2969	2971	2976

10 rows × 104 columns

Let's check the shape of the dataframe

In [3]:
```
corona_dataset_csv.shape
```

Out[3]: (266, 104)

2.2: Deleting the useless columns

In [4]:
```
corona_dataset_csv.drop(['Lat','Long'],axis = 1,inplace = True)
```

In [5]:
```
corona_dataset_csv.head(5)
```

Out[5]:

	Province/State	Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	...	4/21/20	4/22/20	4/23/20
0	NaN	Afghanistan	0	0	0	0	0	0	0	0	...	1092	1176	1279
1	NaN	Albania	0	0	0	0	0	0	0	0	...	609	634	663
2	NaN	Algeria	0	0	0	0	0	0	0	0	...	2811	2910	3007
3	NaN	Andorra	0	0	0	0	0	0	0	0	...	717	723	723
4	NaN	Angola	0	0	0	0	0	0	0	0	...	24	25	25

5 rows × 102 columns

2.3: Aggregating the rows by the country

In [6]:
```
corona_dataset_aggregated = corona_dataset_csv.groupby('Country/Region').sum()
```

In [7]:
```
corona_dataset_aggregated.head()
```

Out[7]:

Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	4/21/20	4/22/20	4/23/20	4/24/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1092	1176	1279	1351
Albania	0	0	0	0	0	0	0	0	0	0	...	609	634	663	678
Algeria	0	0	0	0	0	0	0	0	0	0	...	2811	2910	3007	3127
Andorra	0	0	0	0	0	0	0	0	0	0	...	717	723	723	731
Angola	0	0	0	0	0	0	0	0	0	0	...	24	25	25	25

5 rows × 100 columns

In [8]:
```
corona_dataset_aggregated.shape
```

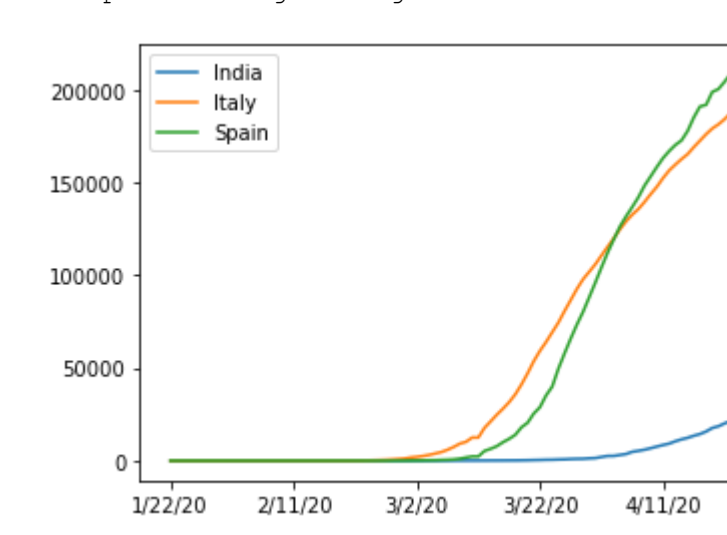
Out[8]: (187, 100)

2.4: Visualizing data related to a country for example India, Italy and Spain

visualization always helps for better understanding of our data.

In [9]:
```
corona_dataset_aggregated.loc['India'].plot()
corona_dataset_aggregated.loc['Italy'].plot()
corona_dataset_aggregated.loc['Spain'].plot()
plt.legend()
```

Out[9]: <matplotlib.legend.Legend at 0x114bd310>

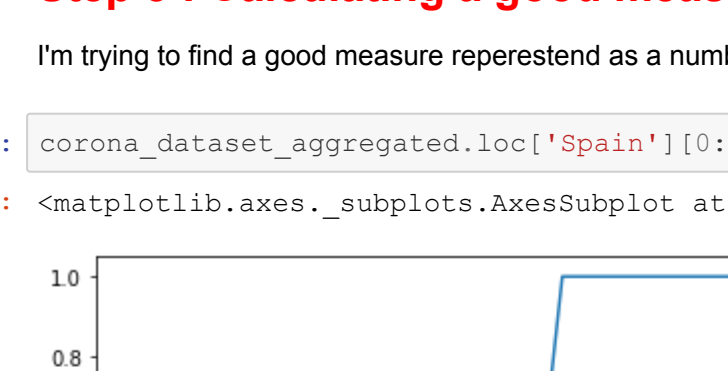


Step 3 : Calculating a good measure

I'm trying to find a good measure reperestend as a number, describing the spread of the virus in a country.

In [10]:
```
corona_dataset_aggregated.loc['Spain'][0:15].plot()
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x12593838>

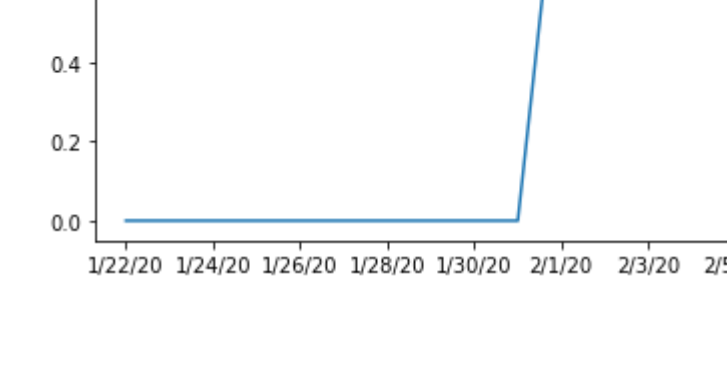


3.1: Calculating the first derivative of the curve

Maximum infection rate

In [11]:
```
corona_dataset_aggregated.loc['Spain'].diff().plot()
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x125f5808>



task 3.2: find maxmimum infection rate for Spain

In [12]:
```
corona_dataset_aggregated.loc['Spain'].diff().max()
```

Out[12]: 9630.0

In [13]:
```
corona_dataset_aggregated.loc['India'].diff().max()
```

Out[13]: 1893.0

In [14]:
```
corona_dataset_aggregated.loc['Italy'].diff().max()
```

Out[14]: 6557.0

Task 3.3: find maximum infection rate for all of the countries.

In [15]:
```
countries = list(corona_dataset_aggregated.index)
max_infection_rates = []
for c in countries :
        max_infection_rates.append(corona_dataset_aggregated.loc[c].diff().max())
corona_dataset_aggregated['max_infection_rates'] = max_infection_rates
```

In [16]:
```
corona_dataset_aggregated.head()
```

Out[16]:

Country/Region	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	1/31/20	...	4/22/20	4/23/20	4/24/20	4/25/20
Afghanistan	0	0	0	0	0	0	0	0	0	0	...	1176	1279	1351	1463
Albania	0	0	0	0	0	0	0	0	0	0	...	634	663	678	712
Algeria	0	0	0	0	0	0	0	0	0	0	...	2910	3007	3127	3256
Andorra	0	0	0	0	0	0	0	0	0	0	...	723	723	731	738
Angola	0	0	0	0	0	0	0	0	0	0	...	25	25	25	25

5 rows × 101 columns

Task 3.4: create a new dataframe with only needed column

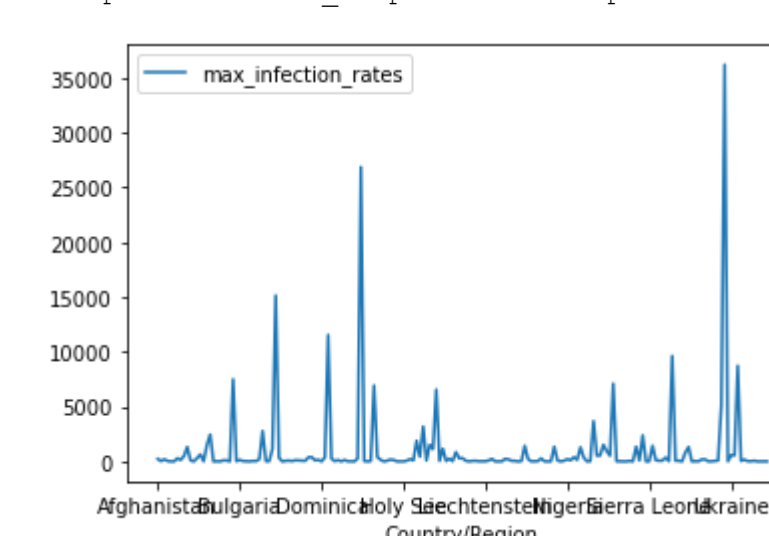
In [17]:
```
corona_data = pd.DataFrame(corona_dataset_aggregated['max_infection_rates'])
corona_data.head()
```

Out[17]:

Country/Region	max_infection_rates
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

In [18]:
```
corona_data.plot()
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1262f628>



Step 4 :

- Importing the WorldHappinessReport.csv dataset
- selecting needed columns for the analysis
- join the datasets
- calculate the correlations as the result of our analysis

4.1 : importing the dataset

In [19]:
```
happiness_report_csv = pd.read_csv('Datasets/worldwide_happiness_report.csv')
happiness_report_csv.head()
```

Out[19]:

	Overall rank	Country or region	Score	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices	Generosity	Perceptions of corruption
0	1	Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
1	2	Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
2	3	Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
3	4	Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
4	5	Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298

In [ ]:

4.2: Dropping the useless columns

In [20]:
```
useless_cols = ['Overall rank','Score','Generosity','Perceptions of corruption']
```

In [21]:
```
happiness_report_csv.drop(useless_cols,axis = 1, inplace = True)
happiness_report_csv.head()
```

Out[21]:

	Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
0	Finland	1.340	1.587	0.986	0.596
1	Denmark	1.383	1.573	0.996	0.592
2	Norway	1.488	1.582	1.028	0.603
3	Iceland	1.380	1.624	1.026	0.591
4	Netherlands	1.396	1.522	0.999	0.557

4.3 : changing the indices of the dataframe

In [22]:
```
happiness_report_csv.set_index('Country or region',inplace = True)
happiness_report_csv.head()
```

Out[22]:

Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Finland	1.340	1.587	0.986	0.596
Denmark	1.383	1.573	0.996	0.592
Norway	1.488	1.582	1.028	0.603
Iceland	1.380	1.624	1.026	0.591
Netherlands	1.396	1.522	0.999	0.557

In [ ]:

4.4: now let's join two dataset we have prepared

Corona Dataset :

In [23]:
```
corona_data.head()
```

Out[23]:

Country/Region	max_infection_rates
Afghanistan	232.0
Albania	34.0
Algeria	199.0
Andorra	43.0
Angola	5.0

In [24]:
```
corona_data.shape
```

Out[24]: (187, 1)

World happiness report Dataset :

In [25]:
```
happiness_report_csv.head()
```

Out[25]:

Country or region	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Finland	1.340	1.587	0.986	0.596
Denmark	1.383	1.573	0.996	0.592
Norway	1.488	1.582	1.028	0.603
Iceland	1.380	1.624	1.026	0.591
Netherlands	1.396	1.522	0.999	0.557

In [26]:
```
happiness_report_csv.shape
```

Out[26]: (156, 4)

In [27]:
```
data = corona_data.join(happiness_report_csv,how = 'inner')
data.head()
```

Out[27]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

4.5: correlation matrix

In [28]:
```
data.corr()
```

Out[28]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
max_infection_rates	1.000000	0.250118	0.191958	0.289263	0.078196
GDP per capita	0.250118	1.000000	0.759468	0.863062	0.394603
Social support	0.191958	0.759468	1.000000	0.765286	0.456246
Healthy life expectancy	0.289263	0.863062	0.765286	1.000000	0.427892
Freedom to make life choices	0.078196	0.394603	0.456246	0.427892	1.000000

Step 5 : Visualization of the results

In [29]:
```
data.head()
```

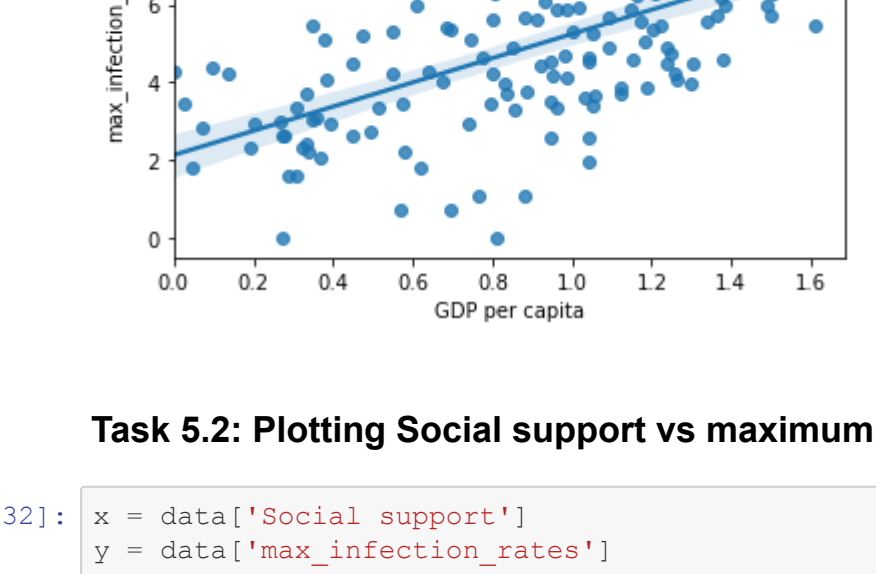
Out[29]:

	max_infection_rates	GDP per capita	Social support	Healthy life expectancy	Freedom to make life choices
Afghanistan	232.0	0.350	0.517	0.361	0.000
Albania	34.0	0.947	0.848	0.874	0.383
Algeria	199.0	1.002	1.160	0.785	0.086
Argentina	291.0	1.092	1.432	0.881	0.471
Armenia	134.0	0.850	1.055	0.815	0.283

5.1: Plotting GDP vs maximum Infection rate

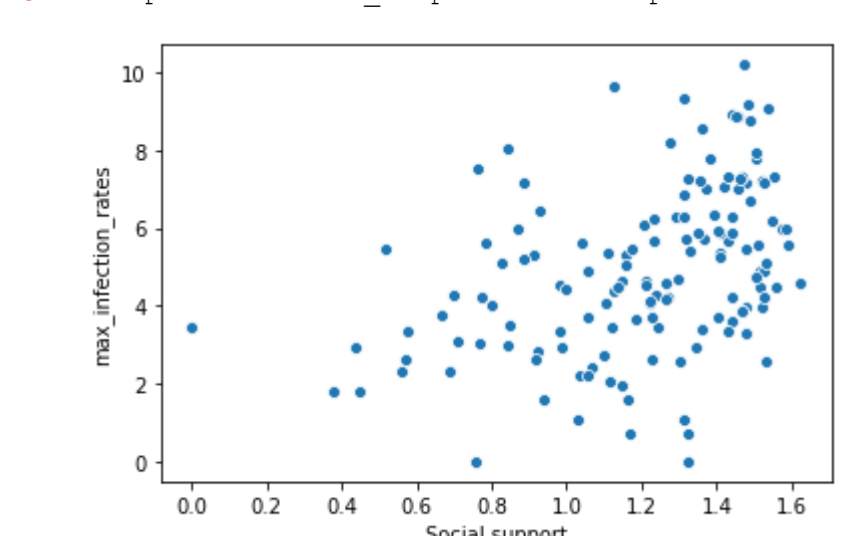
In [30]:
```
x = data['GDP per capita']
y = data['max_infection_rates']
sns.scatterplot(x,np.log(y))
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x12672970>



In [31]:
```
sns.regplot(x,np.log(y))
```

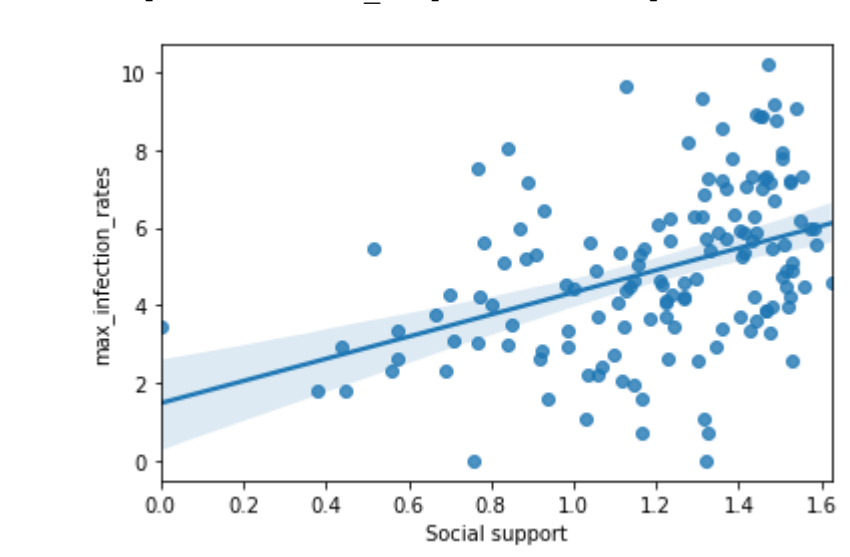
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1269fb50>



Task 5.2: Plotting Social support vs maximum Infection rate

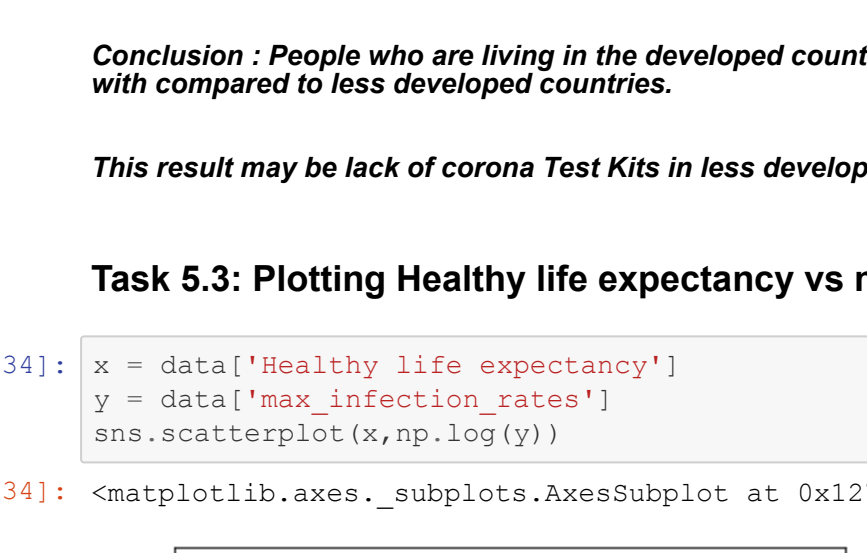
In [32]:
```
x = data['Social support']
y = data['max_infection_rates']
sns.scatterplot(x,np.log(y))
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x126e0d18>



In [33]:
```
sns.regplot(x,np.log(y))
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x127454d8>



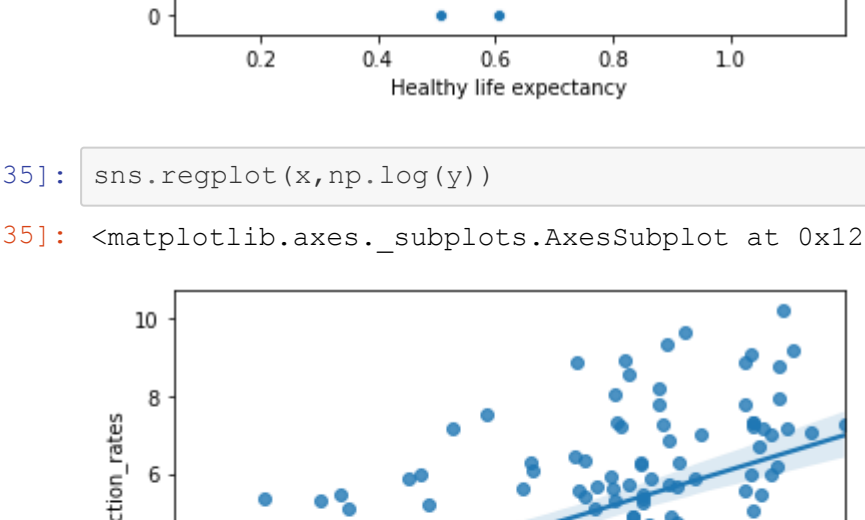
Conclusion : People who are living in the developed countries are more prone to getting the infection of Corona Virus compare of with compared to less developed countries.

This result may be lack of corona Test Kits in less developed countries

Task 5.3: Plotting Healthy life expectancy vs maximum Infection rate

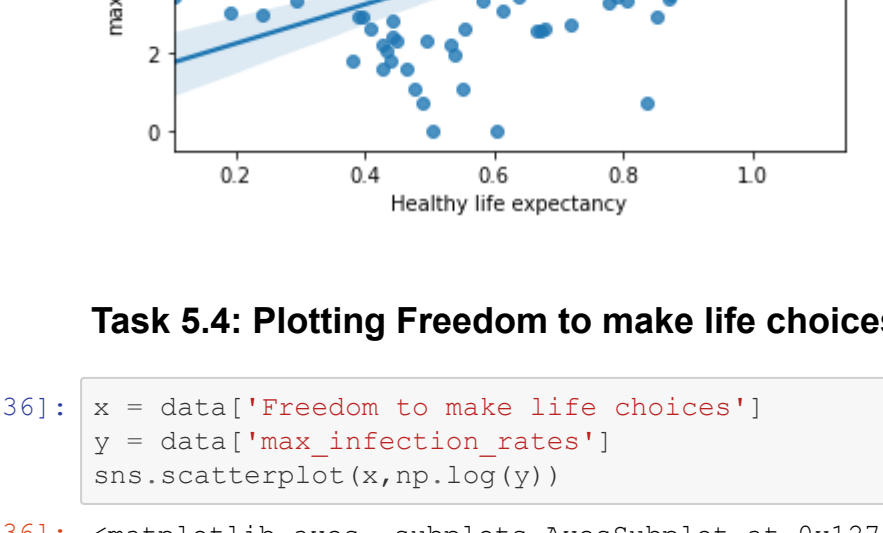
In [34]:
```
x = data['Healthy life expectancy']
y = data['max_infection_rates']
sns.scatterplot(x,np.log(y))
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x127779e8>



In [35]:
```
sns.regplot(x,np.log(y))
```

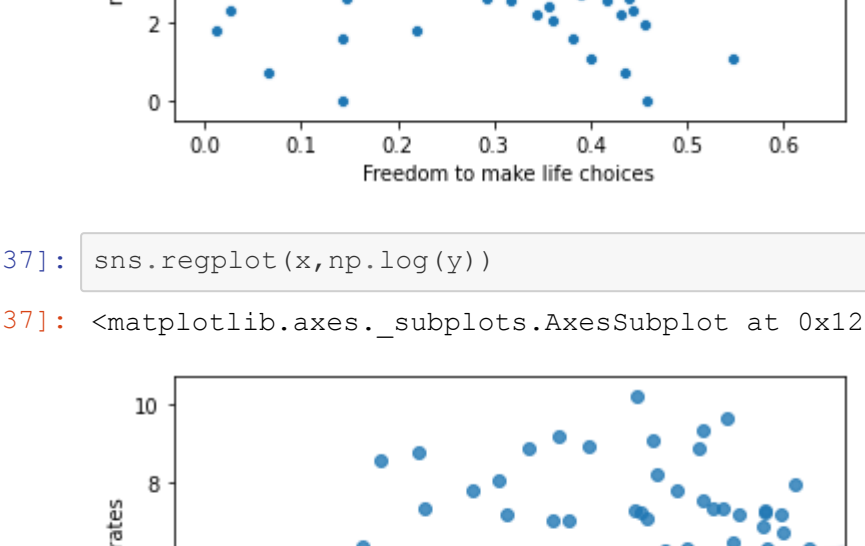
Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x127982c8>



Task 5.4: Plotting Freedom to make life choices vs maximum Infection rate

In [36]:
```
x = data['Freedom to make life choices']
y = data['max_infection_rates']
sns.scatterplot(x,np.log(y))
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x127dc6d0>



In [37]:
```
sns.regplot(x,np.log(y))
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x12768508>