In [1]:

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

20/20

This is assignment 1 for DATA201 in 2020. It is worth 5% of your course mark. To submit, uses the ECS assignment system (https://apps.ecs.vuw.ac.nz/submit/DATA201 (https://apps.ecs.vuw.ac.nz/submit/DATA201))

The due date is Saturday 21st March by midnight. No extensions.

Dowload LifeExpectancy.csv from the course webpage, and read it into Python, skipping the necessary rows and reading the header. Make the country name be an index. Print the first few rows to ensure that you have it correct. [2 marks]

2/2

In [2]:

```
df = pd.read_csv("LifeExpectancy.csv", skiprows=4, index_col='Country Name')
df.head()
```

Out[2]:

	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	19
Country Name									
Aruba	ABW	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	65.662	66.074	66.444	66.787	67.113	67.4
Afghanistan	AFG	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	32.446	32.962	33.471	33.971	34.463	34.9
Angola	AGO	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	37.524	37.811	38.113	38.430	38.760	39.1
Albania	ALB	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	62.283	63.301	64.190	64.914	65.463	65.8
Andorra	AND	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	NaN	NaN	NaN	NaN	NaN	N

5 rows × 63 columns

Drop rows that consist of NaN values. Be careful how you do this, the naive way of just using dropna() without looking at the data a bit might not do what you expect. You might need to know that to delete a column you can use `life.drop([list of column names],axis=1` . [2 marks]

2/2

In [3]:

```
df = df.drop(['2018', '2019'], axis=1)
df = df.dropna()
df
```

Out[3]:

	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
Country Name							
Aruba	ABW	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	65.662000	66.074000	66.444000	66.787000
Afghanistan	AFG	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	32.446000	32.962000	33.471000	33.971000
Angola	AGO	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	37.524000	37.811000	38.113000	38.430000
Albania	ALB	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	62.283000	63.301000	64.190000	64.914000
Arab World	ARB	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	46.546909	47.141621	47.731783	48.320432
...	...	...	...	...	...	...	...
Samoa	WSM	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	56.902000	57.188000	57.472000	57.756000
Yemen, Rep.	YEM	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	29.919000	30.163000	30.500000	30.943000
South Africa	ZAF	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	48.406000	48.777000	49.142000	49.509000
Zambia	ZMB	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	46.687000	47.084000	47.446000	47.772000
Zimbabwe	ZWE	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	53.019000	53.483000	53.946000	54.403000

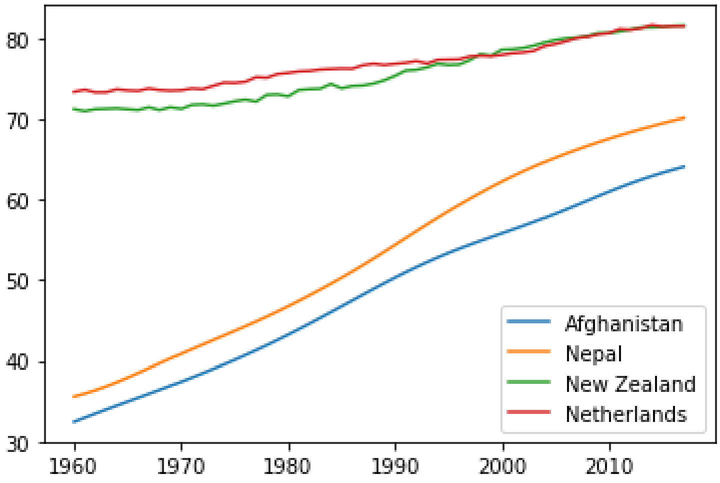
235 rows × 61 columns

Now plot the curves of life expectancy against time on 1 plot for the following countries: Afghanistan, Nepal, New Zealand, Netherlands Include a legend, and make the labels on the x-axis readable. [2 marks]

2/2 But remember chart and axis titles.

In [4]:

```
plt.plot(df.loc['Afghanistan', :][3:], label='Afghanistan')
plt.plot(df.loc['Nepal', :][3:], label='Nepal')
plt.plot(df.loc['New Zealand', :][3:], label='New Zealand')
plt.plot(df.loc['Netherlands', :][3:], label='Netherlands')
ax = plt.gca()
ticks = df.columns.values[3:][::10]
ax.set_xticks(ticks)
plt.legend()
plt.show()
```



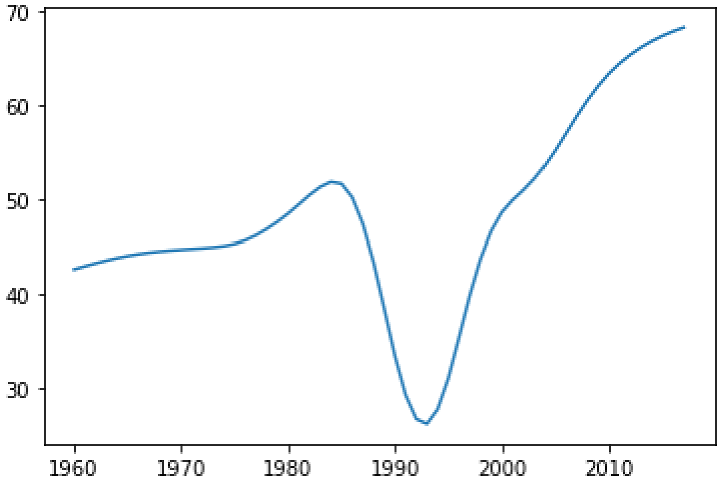
Plot Rwanda separately, and explain briefly why it has that shape (hint: use wikipedia) [2 marks]

2/2

In [5]:

```
plt.plot(df.loc['Rwanda', :][3:], label='Rwanda')
ax = plt.gca()
ticks = df.columns.values[3:][::10]
ax.set_xticks(ticks)
plt.show()

# due to the Rwandan genocide
# the mortality rate for Rwanda decreased dramatically
# Academy Award winning film Hotel Rwanda is really good
```



Can you detect any other countries where the life expectancy drops significantly?

To compute this write some loops over each country and each year. If the next value is below 95% of the current value, print the name of the country. [4 marks]

4/4

In [6]:

```
def get_drop(prev, current):
    return current/prev
```

In [7]:

```python
# convert to a numpy array, because iterating over a dataframe is non-pattern
# and computationlally expensive due to the constant creation of Series objects
df_array = df.reset_index().to_numpy()
start_year = 1960
for row in df_array:
    prev = 0
    for i, item in enumerate(row[4:]):
        if prev != 0:
            if get_drop(prev, item) < 0.95:
                year = start_year+i
                print(''.join(['In ', str(year), ' there is a significant (<95%) drop from ', str(year-1), ' in ', row[0],
                               '\'s life expectancy, going from ', str(prev), ' to ', str(item), '\n']))
        prev = item
```

```
In 1972 there is a significant (<95%) drop from 1971 in Cambodia's life expectancy, going from 39.699 to 36.676

In 1973 there is a significant (<95%) drop from 1972 in Cambodia's life expectancy, going from 36.676 to 32.667

In 1974 there is a significant (<95%) drop from 1973 in Cambodia's life expectancy, going from 32.667 to 28.04

In 1975 there is a significant (<95%) drop from 1974 in Cambodia's life expectancy, going from 28.04 to 23.595

In 1976 there is a significant (<95%) drop from 1975 in Cambodia's life expectancy, going from 23.595 to 20.317

In 1977 there is a significant (<95%) drop from 1976 in Cambodia's life expectancy, going from 20.317 to 18.907

In 1987 there is a significant (<95%) drop from 1986 in Rwanda's life expectancy, going from 50.233000000000004 to 47.409

In 1988 there is a significant (<95%) drop from 1987 in Rwanda's life expectancy, going from 47.409 to 43.361000000000004

In 1989 there is a significant (<95%) drop from 1988 in Rwanda's life expectancy, going from 43.361000000000004 to 38.439

In 1990 there is a significant (<95%) drop from 1989 in Rwanda's life expectancy, going from 38.439 to 33.413000000000004

In 1991 there is a significant (<95%) drop from 1990 in Rwanda's life expectancy, going from 33.413000000000004 to 29.248

In 1992 there is a significant (<95%) drop from 1991 in Rwanda's life expectancy, going from 29.248 to 26.691
```

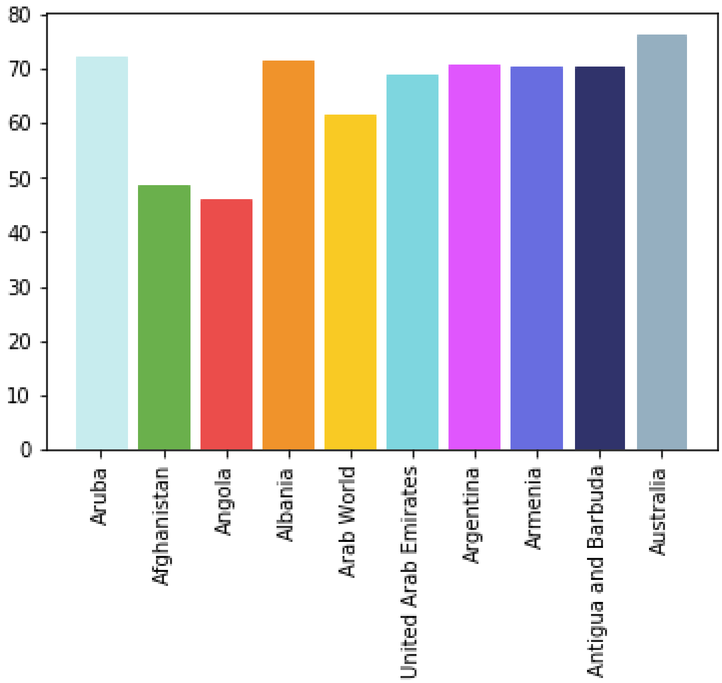
Compute the mean life expectancy for each of the countries over the whole 57 years. Plot a bar chart of this for the first 10 countries. [4 marks]

4/4

You can use colormaps to avoid having to specify each colour by code.

In [8]:

```
means = df.mean(numeric_only=True, axis=1)
bars = plt.bar(df.index[:10], means[:10])
colors = ['#c7ecee', '#6ab04c', '#eb4d4b', '#f0932b', '#f9ca24',
          '#7ed6df', '#e056fd', '#686de0', '#30336b', '#95afc0']
for i in range(10):
    bars[i].set_color(colors[i])
plt.xticks(rotation=90)
plt.show()
```

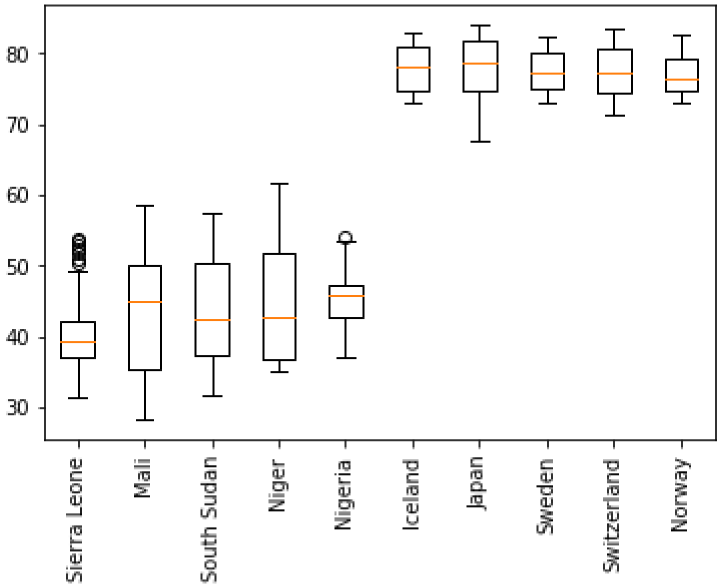


Find the 5 countries with the highest mean life expectancy and the 5 with the lowest. You might find `life.sort_values()` helpful, as well as `pd.concat` and `life.transpose` . Plot a box and whisker plot of these countries. [4 marks]

4/4

In [9]:

```
df_means = pd.concat([df, means], axis=1)
df_means = df_means.rename({0: 'mean'}, axis='columns')
means_lowest = df_means.sort_values(by=['mean'])[:5]
means_highest = df_means.sort_values(by=['mean'], ascending=False)[:5]
means_low_high = pd.concat([means_lowest, means_highest])
plt.boxplot(means_low_high.iloc[:,3:-1])
plt.xticks(np.arange(10)+1, means_low_high.index)
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

```
```