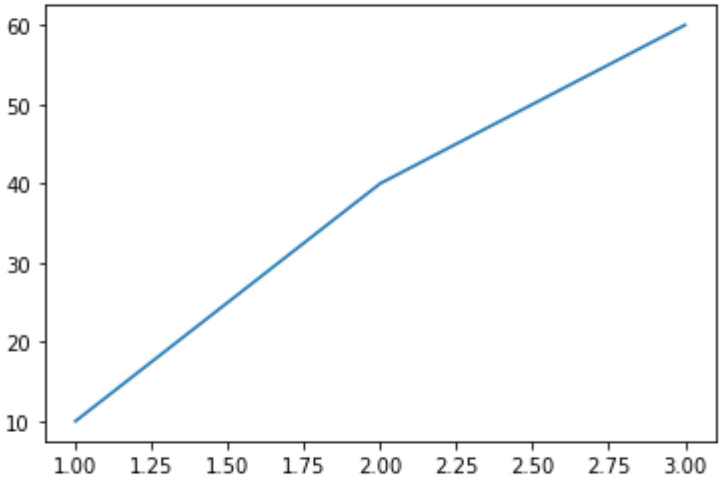
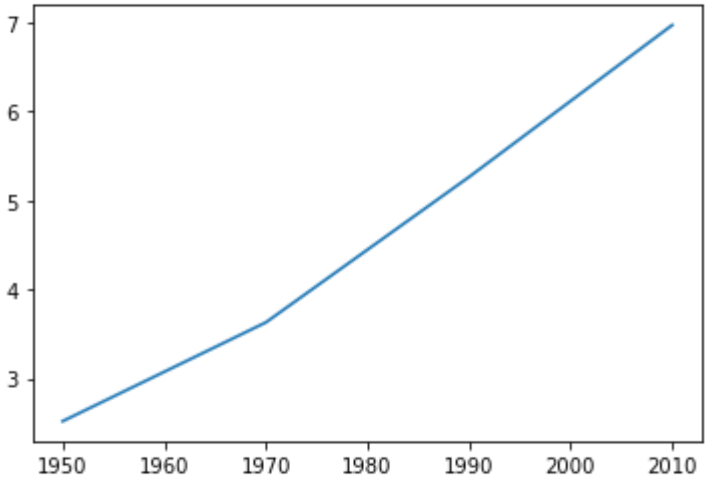
In [9]:

```
import matplotlib.pyplot as plt
%matplotlib inline

x = [1,2,3]
y = [10,40,60]
plt.plot(x,y)
plt.show()
```



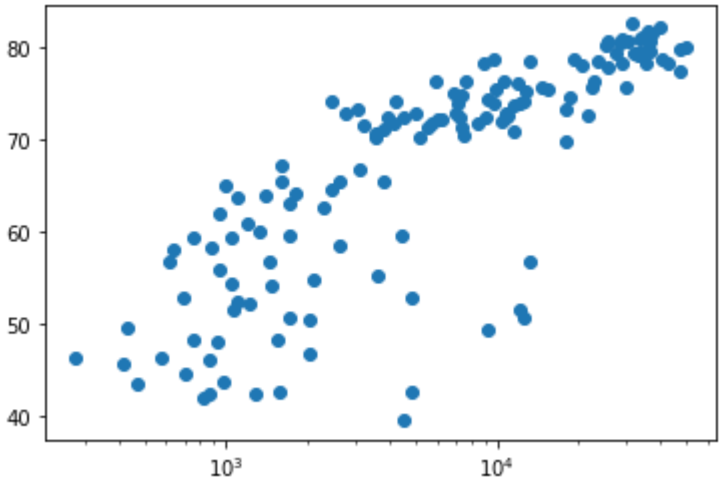
In [13]:

```
import matplotlib.pyplot as plt
%matplotlib inline
year = [1950,1970,1990,2010]
pop = [2.519, 3.629, 5.263, 6.97]
plt.plot(year, pop)
plt.show()
```



In [21]:

```
gdp_cap = [974.5803384, 5937.029525999999, 6223.367465, 4797.231267, 12779

life_exp = [43.828, 76.423, 72.301, 42.731, 75.32, 81.235, 79.829, 75.635,

pop = [31.889923, 3.600523, 33.333216, 12.420476, 40.301927, 20.434176, 8.

col = ['red', 'green', 'blue', 'blue', 'yellow', 'black', 'green', 'red',


import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(gdp_cap, life_exp)
plt.xscale('log')
plt.show()
```

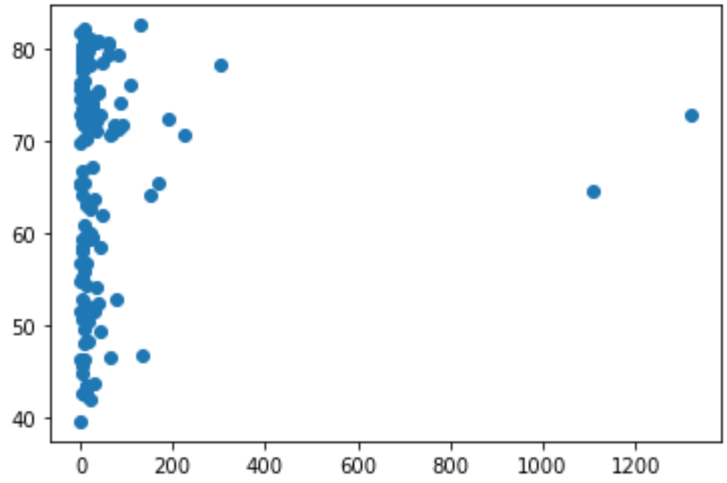


Ex1

ScatterPlot

In [28]:

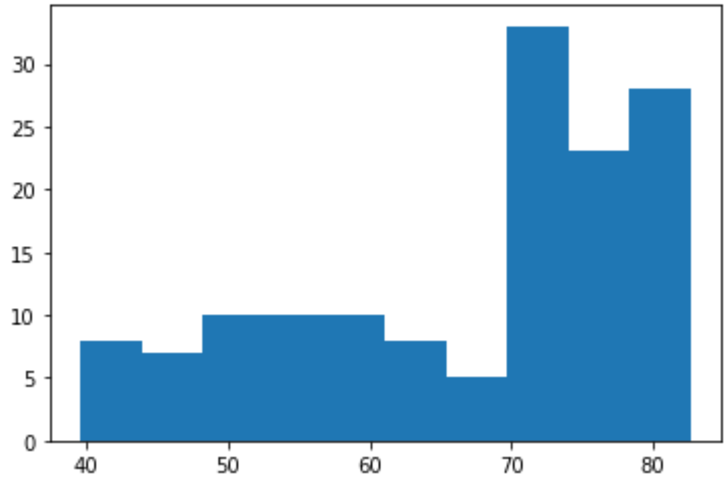
```
# gdp_cap = [974.5803384, 5937.029525999999, 6223.367465, 4797.231267, 127

life_exp = [43.828, 76.423, 72.301, 42.731, 75.32, 81.235, 79.829, 75.635,

pop = [31.889923, 3.600523, 33.333216, 12.420476, 40.301927, 20.434176, 8.



import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(pop, life_exp)
#plt.xscale('log')
plt.show()
```



Histogram of life_exp

In [31]:

```
plt.hist(life_exp)
plt.show()
```

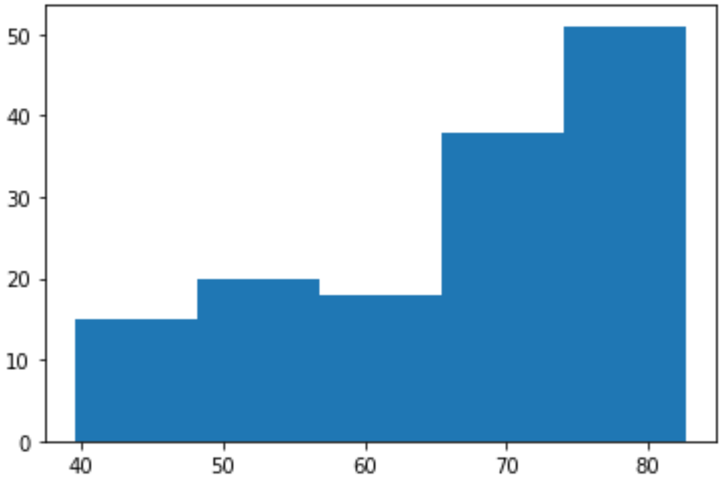


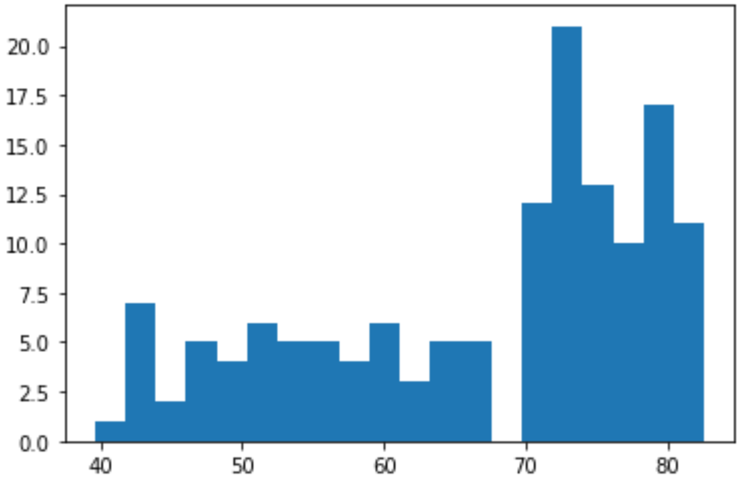
Ex2

Histogram of Life_exp

In [40]:

```
plt.hist(life_exp, bins = 5)
plt.show()


plt.hist(life_exp, bins = 20)
plt.show()
```





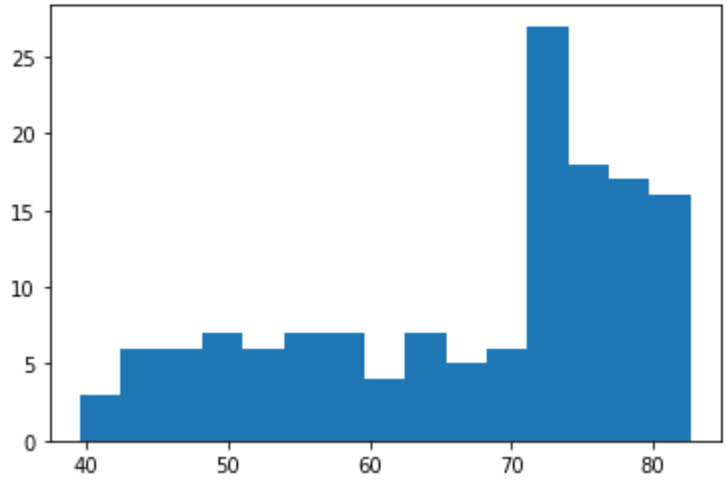
the bins 5 historgram is not easy to understand.

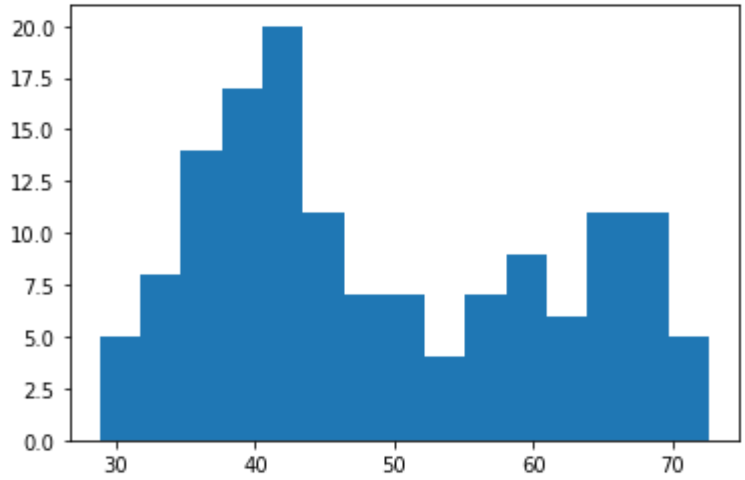
this bins 20 histogram is better because we can clearly see which timeline has more or less than the other.

Ex3

In [45]:

```
life_exp1950 = [28.8, 55.23, 43.08, 30.02, 62.48, 69.12, 66.8, 50.94, 37.4

plt.hist(life_exp, bins = 15)
plt.show()

plt.hist(life_exp1950, bins = 15)
plt.show()
```





I observe that they have differnt frequency even if the bins is the same.

life_exp 2007 has the highest frequency.

Ex4

I will use Histogram because the grades on your exam follow a particular distribution

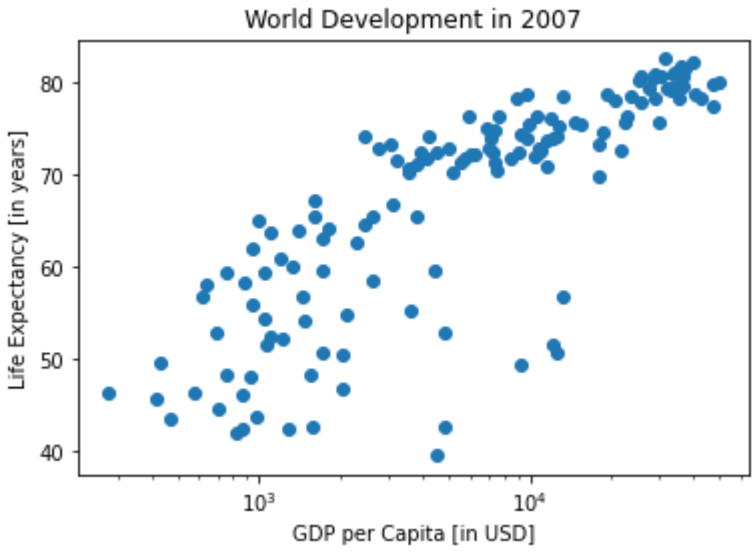
Ex5

I will use ScatterPlot because longer answers on exam questions lead to higher grades

Ex6

In [51]:

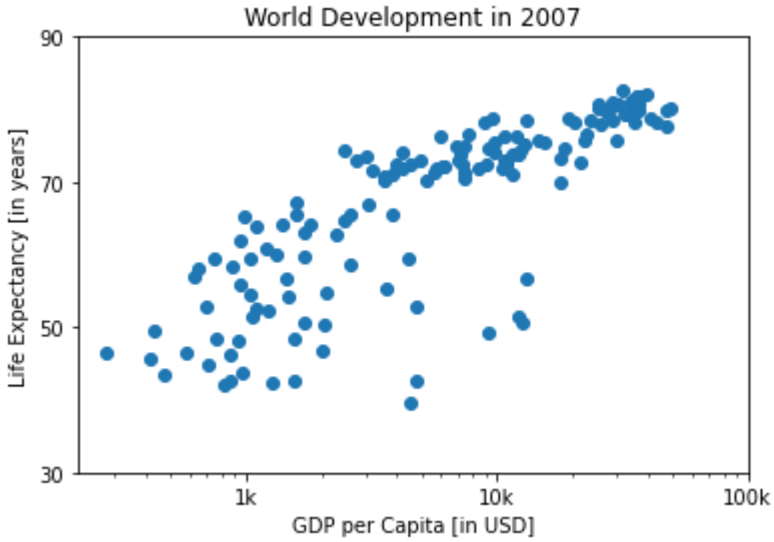
```
plt.scatter(gdp_cap, life_exp)
plt.xscale('log')

xlab = 'GDP per Capita [in USD]'
ylab = 'Life Expectancy [in years]'
title = 'World Development in 2007'

plt.xlabel(xlab)
plt.ylabel(ylab)
plt.title(title)
plt.show()
```



Ex7

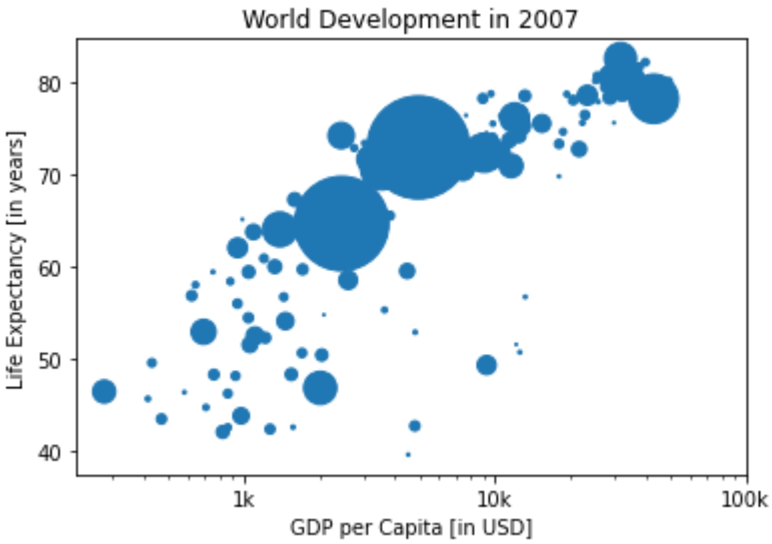
In [54]:

```
# Scatter plot
plt.scatter(gdp_cap, life_exp)
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')

# Definition of tick_val and tick_lab
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']

tick_yval = [30, 50, 70 , 90]
tick_ylab = ['30', '50', '70', '90']

# Adapt the ticks on the x-axis
plt.xticks(tick_val,tick_lab)
plt.yticks(tick_yval, tick_ylab)

# After customizing, display the plot
plt.show()
```



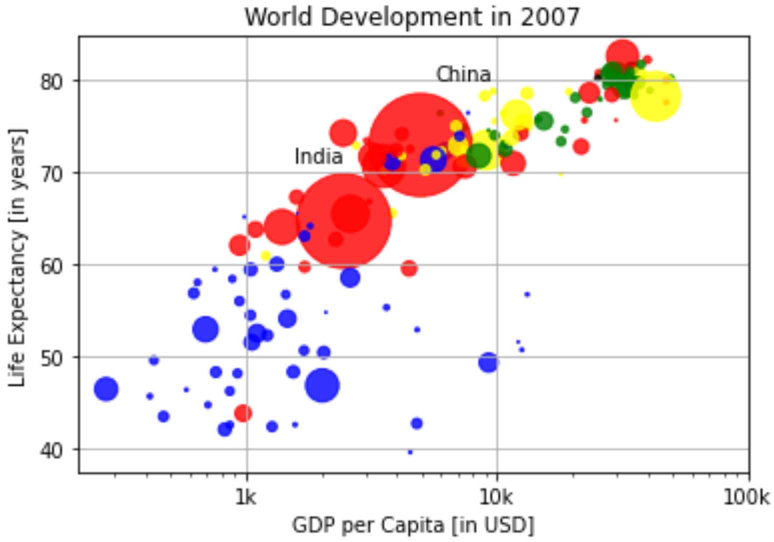
In [59]:

```
# Import numpy as np
import numpy as np

# Store pop as a numpy array: np_pop
np_pop = np.array(pop)

# Double np_pop
np_pop = np_pop * 2

# Update: set s argument to np_pop
plt.scatter(gdp_cap, life_exp, s = np_pop)

# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000, 10000, 100000],['1k', '10k', '100k'])

# Display the plot
plt.show()
```



In [68]:

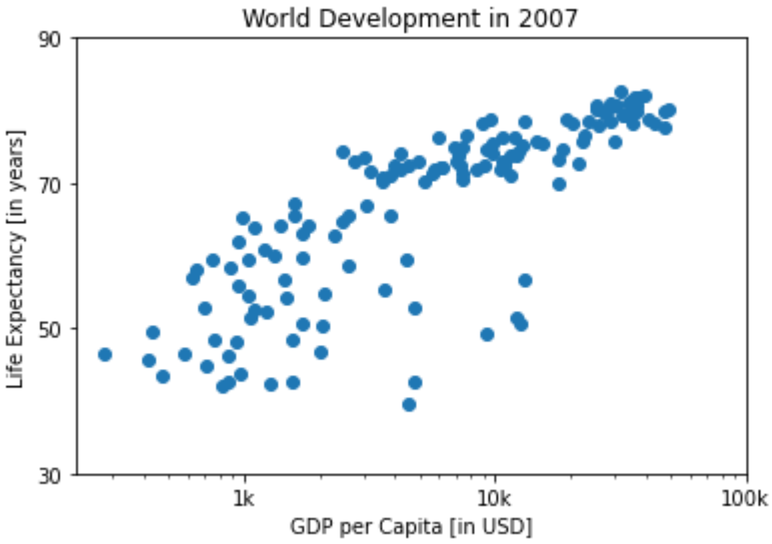
```
col = ['red', 'green', 'blue', 'blue', 'yellow', 'black', 'green', 'red',

# Scatter plot
plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2, c = col, alp
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000,10000,100000], ['1k','10k','100k'])
# Additional customizations
plt.text(1550, 71, 'India')
plt.text(5700, 80, 'China')
# Add grid() call
plt.grid(1)
# Show the plot
plt.show()
```

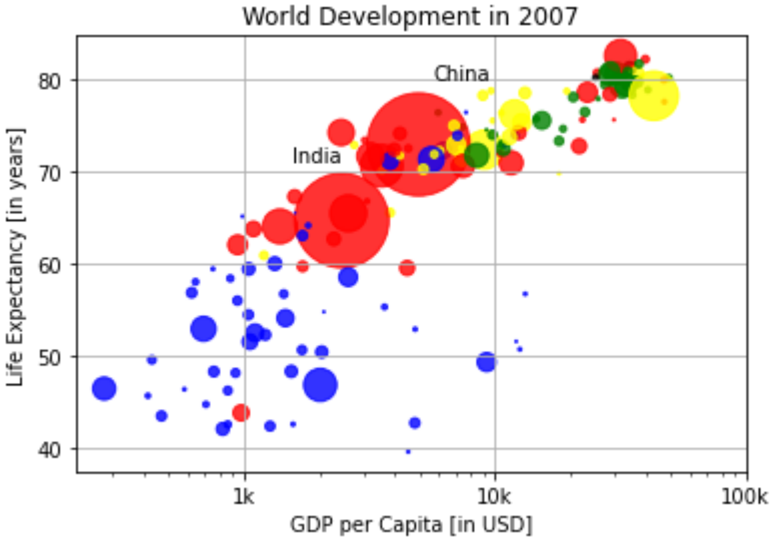


Ex8

In [70]:

```
# Scatter plot
plt.scatter(gdp_cap, life_exp)
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')

# Definition of tick_val and tick_lab
tick_val = [1000,10000,100000]
tick_lab = ['1k','10k','100k']

tick_yval = [30, 50, 70 , 90]
tick_ylab = ['30', '50', '70', '90']

# Adapt the ticks on the x-axis
plt.xticks(tick_val,tick_lab)
plt.yticks(tick_yval, tick_ylab)

# After customizing, display the plot
plt.show()


col = ['red', 'green', 'blue', 'blue', 'yellow', 'black', 'green', 'red',

# Scatter plot
plt.scatter(x = gdp_cap, y = life_exp, s = np.array(pop) * 2, c = col, alph
# Previous customizations
plt.xscale('log')
plt.xlabel('GDP per Capita [in USD]')
plt.ylabel('Life Expectancy [in years]')
plt.title('World Development in 2007')
plt.xticks([1000,10000,100000], ['1k','10k','100k'])
# Additional customizations
plt.text(1550, 71, 'India')
plt.text(5700, 80, 'China')
# Add grid() call
plt.grid(1)
# Show the plot
plt.show()
```





the first one is true. (o The countries in blue, corresponding to Africa, have both low life expectancy and a low GDP per capita.)

```
In [72]:
```

In [19]:

```python
import pandas as pd
brics = pd.read_csv('brics.csv', index_col = 0)

#brics['area'] will give all area data
brics.loc['BR': 'IN', 'capital' : 'area'] # will give only six data
#brics.iloc[0:3, 1:3] will give only six data with the row name and column
#brics['area'] > 8 will give all boolean
```

Out[19]:

```
BR     True
RU     True
IN    False
CH     True
SA    False
Name: area, dtype: bool
```

In [21]:

```python
brics[['area', 'country']]
```

Out[21]:

	area	country
BR	8.516	Brazil
RU	17.100	Russia
IN	3.286	India
CH	9.597	China
SA	1.221	South Africa

In [25]:

```python
result = brics['area'] > 8
brics[result]
```

Out[25]:

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.4
RU	Russia	Moscow	17.100	143.5
CH	China	Beijing	9.597	1357.0

In [28]:

```python
brics[brics['area']>8]
```

Out[28]:

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.4
RU	Russia	Moscow	17.100	143.5
CH	China	Beijing	9.597	1357.0

Ex9

In [34]:
```
brics[brics['population'] >= 200]  [['country', 'population']]
```

Out[34]:
```
    country  population
BR   Brazil       200.4
IN    India      1252.0
CH    China      1357.0
```

In [37]:
```
import numpy as np
area810 = np.logical_and(brics['area'] >8, brics['area'] < 10)
brics[area810]
```

Out[37]:
```
    country   capital   area  population
BR   Brazil  Brasilia  8.516       200.4
CH    China   Beijing  9.597      1357.0
```

Ex10

In [54]:
```
#brics[np.logical_or(brics['population'] > 1000, brics['area'] < 8)]

brics[(brics['area'] < 8) | (brics['population'] > 1000 )]   [['capital']]
```

Out[54]:
```
      capital
IN  New Delhi
CH    Beijing
SA   Pretoria
```

Ex11

In [62]:

```
#load car.csv
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)
cpc = cars[['cars_per_cap']]
many_cars = (cpc['cars_per_cap'] > 500)

car_maniac = cars[many_cars]
car_maniac.head()
```

Out[62]:

	cars_per_cap	country	drives_right
US	809	United States	True
AUS	731	Australia	False
JAP	588	Japan	False

Ex12

In [68]:

```
cpc100500 = (cars['cars_per_cap'] >= 100) & (cars['cars_per_cap'] <= 500)
cars[cpc100500]
```

Out[68]:

	cars_per_cap	country	drives_right
RU	200	Russia	True

Loop Over DataFrame

In [70]:

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)
# Iterate over rows of cars
for key in cars:
    print(key)
```

```
cars_per_cap
country
drives_right
```

In [71]:

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)
# Iterate over rows of cars
for key,value in cars.iterrows():
    print(key)
    print(value)
```

```
US
cars_per_cap              809
country         United States
drives_right             True
Name: US, dtype: object
AUS
cars_per_cap          731
country         Australia
drives_right        False
Name: AUS, dtype: object
JAP
cars_per_cap      588
country         Japan
drives_right    False
Name: JAP, dtype: object
IN
cars_per_cap       18
country         India
drives_right    False
Name: IN, dtype: object
RU
cars_per_cap       200
country         Russia
drives_right      True
Name: RU, dtype: object
MOR
cars_per_cap         70
country         Morocco
drives_right       True
Name: MOR, dtype: object
EG
cars_per_cap       45
country         Egypt
drives_right     True
Name: EG, dtype: object
```

In [73]:

```
# Import cars data
import pandas as pd
cars = pd.read_csv('cars.csv', index_col = 0)
# Adapt for loop
for lab,row in cars.iterrows():
    print(lab +": "+ str(row['cars_per_cap']))
```

```
US: 809
AUS: 731
JAP: 588
IN: 18
RU: 200
MOR: 70
EG: 45
```

In [118]:

```
## Import cars data
#import pandas as pd
#cars = pd.read_csv('cars.csv', index_col = 0)
# Code for loop that adds COUNTRY column
##for lab,row in cars.iterrows():
  ##  cars.loc[lab, "COUNTRY"] = row["country"].upper()
# Print cars
##print(cars)
##
```

In [ ]:

```

```

Ex13

In [95]:

```
for lab, row in brics.iterrows():

    brics["name_length"] = brics["country"].apply(len)

print(brics)
```

```
         country    capital    area  population  name_length
BR        Brazil   Brasilia   8.516      200.40            6
RU        Russia     Moscow  17.100      143.50            6
IN         India  New Delhi   3.286     1252.00            5
CH         China    Beijing   9.597     1357.00            5
SA  South Africa   Pretoria   1.221       52.98           12
```

Ex14

In [224]:

```
cars["COUNTRY"] = cars["country"].apply(str.upper)
print(cars)
```

```
     cars_per_cap        country  drives_right        COUNTRY        Coun
try
US            809  United States          True  UNITED STATES  UNITED STA
TES
AUS           731      Australia         False      AUSTRALIA      AUSTRA
LIA
JAP           588          Japan         False          JAPAN          JA
PAN
IN             18          India         False          INDIA          IN
DIA
RU            200         Russia          True         RUSSIA         RUS
SIA
MOR            70        Morocco          True        MOROCCO        MORO
CCO
EG             45          Egypt          True          EGYPT          EG
YPT
```

In [142]:

```
years = [2011,2012,2013,2014,2015,2016,2017,2018,2019,2020]
durations = [103,101,99,100,100,95,95,96,93,90]

# Create a dictionary with the two lists
movie_dict = {"years":years,"durations":durations}

# Print the dictionary
movie_dict

import pandas as pd

# Create a DataFrame from the dictionary
durations_df = pd.DataFrame(movie_dict)

# Print the DataFrame
print(durations_df) # or just durations_df
durations_df
```

```
   years  durations
0   2011        103
1   2012        101
2   2013         99
3   2014        100
4   2015        100
5   2016         95
6   2017         95
7   2018         96
8   2019         93
9   2020         90
```

Out[142]:

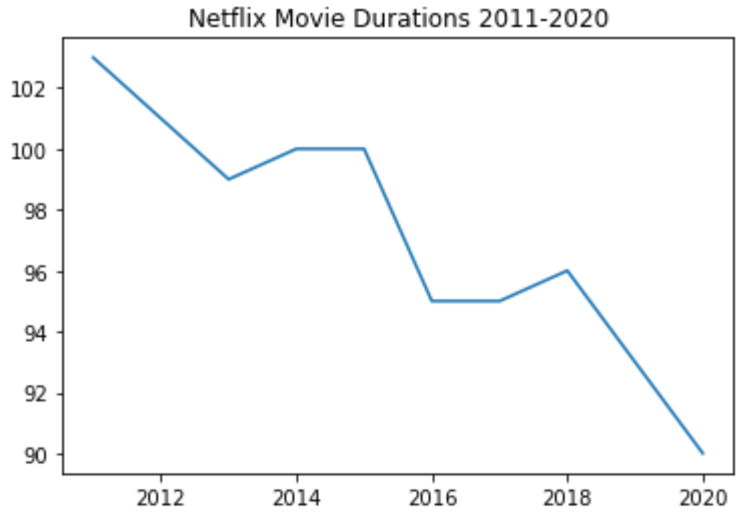
	years	durations
0	2011	103
1	2012	101
2	2013	99
3	2014	100
4	2015	100
5	2016	95
6	2017	95
7	2018	96
8	2019	93
9	2020	90

In [143]:

```
# Import matplotlib.pyplot under its usual alias and create a figure
import matplotlib.pyplot as plt
fig = plt.figure()

# Draw a line plot of release_years and durations
plt.plot(durations_df['years'], durations_df['durations'])

# Create a title
plt.title('Netflix Movie Durations 2011-2020')
```

Out[143]:

```
Text(0.5, 1.0, 'Netflix Movie Durations 2011-2020')
```



In [185]:

```
# Read in the CSV as a DataFrame
netflix = pd.read_csv(r'netflix_data.csv')

# Subset the DataFrame for type "Movie"
netflix_df_movies_only = netflix_df.query('type == "Movie"')

# Select only the columns of interest
netflix_movies_col_subset = netflix_df_movies_only[['title','country','gen

# Print the first five rows of the new DataFrame
netflix_movies_col_subset.head()
```

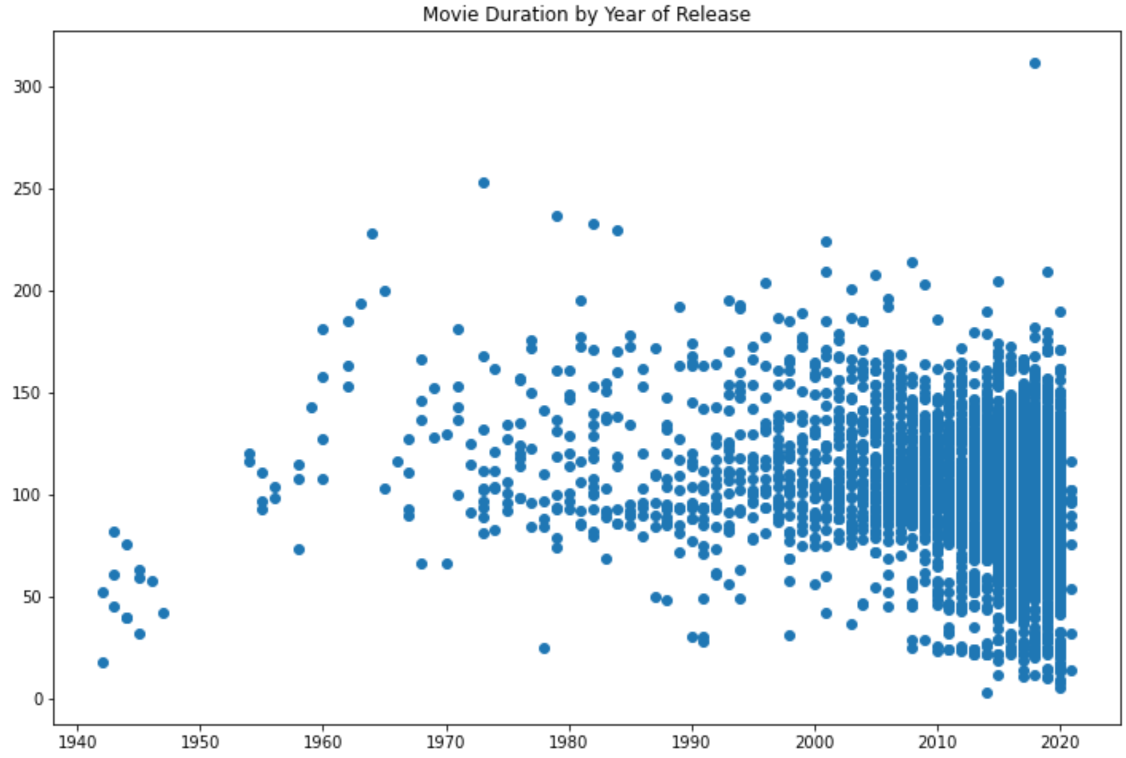
Out[185]:

	title	country	genre	release_year	duration
1	7:19	Mexico	Dramas	2016	93
2	23:59	Singapore	Horror Movies	2011	78
3	9	United States	Action	2009	80
4	21	United States	Dramas	2008	123
6	122	Egypt	Horror Movies	2019	95

Let’s do a scatter plot

In [186]:

```
# Create a figure and increase the figure size
fig = plt.figure(figsize=(12,8))

# Create a scatter plot of duration versus year
plt.scatter(netflix_movies_col_subset.release_year, netflix_movies_col_sub

# Create a title
plt.title("Movie Duration by Year of Release")

# Show the plot
plt.show()
```



In [188]:

```
# Filter for durations shorter than 60 minutes
short_movies = netflix_movies_col_subset[netflix_movies_col_subset['durati
#short_movies = netflix_movies_col_subset.query('duration < 60')

# Print the first 10 rows of short_movies
short_movies.head(10)
```

Out[188]:

	title	country	genre	release_year	duration
35	#Rucker50	United States	Documentaries	2016	56
55	100 Things to do Before High School	United States	Uncategorized	2014	44
67	13TH: A Conversation with Oprah Winfrey & Ava ...	NaN	Uncategorized	2017	37
101	3 Seconds Divorce	Canada	Documentaries	2018	53
146	A 3 Minute Hug	Mexico	Documentaries	2019	28
162	A Christmas Special: Miraculous: Tales of Lady...	France	Uncategorized	2016	22
171	A Family Reunion Christmas	United States	Uncategorized	2019	29
177	A Go! Go! Cory Carson Christmas	United States	Children	2020	22
178	A Go! Go! Cory Carson Halloween	NaN	Children	2020	22
179	A Go! Go! Cory Carson Summer Camp	NaN	Children	2020	21

In [199]:

```
# Define an empty list
colors = []

# Iterate over rows of netflix_movies_col_subset
for lab, row in netflix_movies_col_subset.iterrows() :
    if row['genre'] == "Children" :
        colors.append("red")
    elif row['genre'] == "Documentaries" :
        colors.append("blue")
    elif row['genre'] == "Stand-Up" :
        colors.append("green")
    else:
        colors.append("black")

# Inspect the first 10 values in your list
colors[:10]
```

Out[199]:

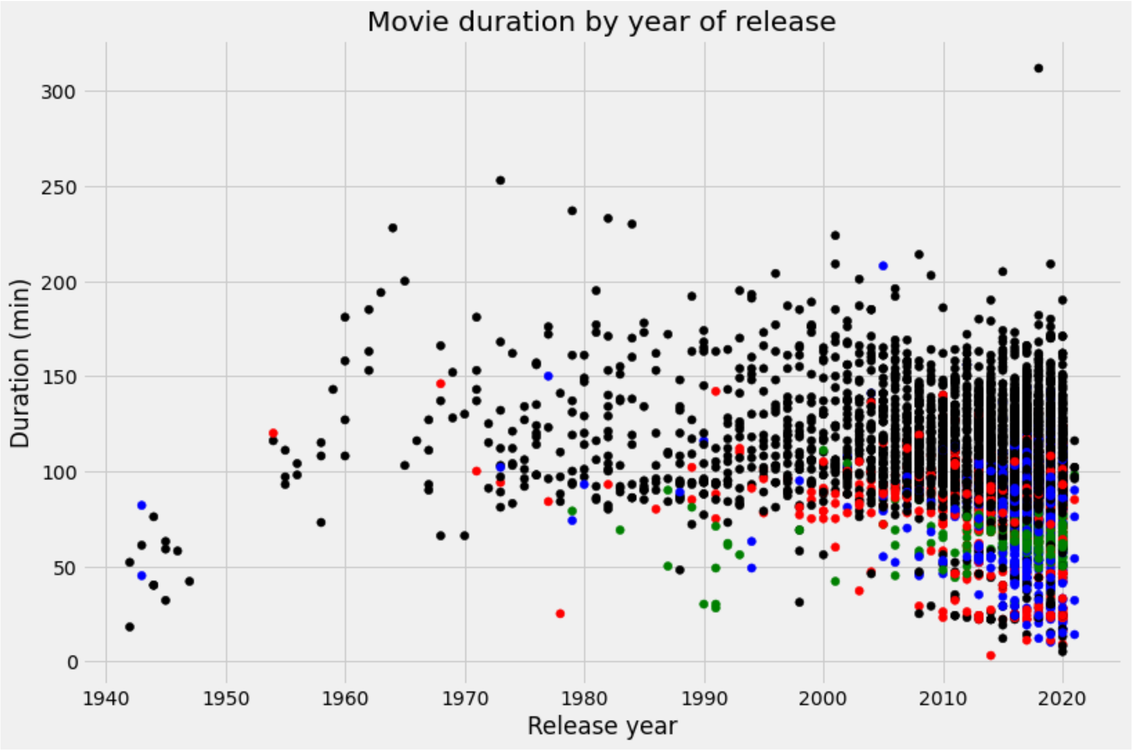
```
['black',
 'black',
 'black',
 'black',
 'black',
 'black',
 'black',
 'black',
 'black',
 'blue']
```

In [200]:

```
# Set the figure style and initalize a new figure
plt.style.use('fivethirtyeight')
fig = plt.figure(figsize=(12,8))

# Create a scatter plot of duration versus release_year
plt.scatter(netflix_movies_col_subset.release_year, netflix_movies_col_sub

# Create a title and axis labels
plt.title("Movie duration by year of release")
plt.xlabel("Release year")
plt.ylabel("Duration (min)")
```

Out[200]:

```
Text(0, 0.5, 'Duration (min)')
```



Task 1

In [223]:

```
netflix_df_country = netflix_movies_col_subset.query('country == "United S

movie = netflix_df_country[['title','country','genre','release_year','dura


movie.head()
```

Out[223]:

```
      title        country              genre  release_year  duration
 3        9  United States             Action          2009        80
 4       21  United States             Dramas          2008       123
 7      187  United States             Dramas          1997       119
10     1922  United States             Dramas          2017       103
14     3022  United States  Independent Movies          2019        91
```

In [ ]: