Homework 03

Name: McClain Thiel

Student ID: 3034003600

Note: Please print the output of each question in a new cell below your code

Numpy Introduction

1a) Create two numpy arrays (a and b). a should be all integers between 25-35 (inclusive), and b should be ten evenly spaced numbers between 1-6. Print all the results below and store them seperately:

i) Cube (i.e. raise to the power of 3) all the elements in both arrays (element-wise)
ii) Add both the cubed arrays (e.g., [1,2] + [3,4] = [4,6])
iii) Sum the elements with even indices of the added array.
iv) Take the square root of the added array (element-wise square root)

In [284]:
```
# your code here
import numpy as np
import pandas as pd

a = np.array(range(25,35))
b = np.arange(1,6,.5)  #just to make the arrays the same size
print(a)
print(b)

a1 = a ** 3
b1 = b ** 3

print(a1)
print(b1)


c = a1 + b1 #size 11 array and size 10 array but it specifically says in
clusive
print(c)

d = c[::2].sum()
print(d)

f = np.sqrt(c)
print(f)
```

```
[25 26 27 28 29 30 31 32 33 34]
[1.  1.5 2.  2.5 3.  3.5 4.  4.5 5.  5.5]
[15625 17576 19683 21952 24389 27000 29791 32768 35937 39304]
[  1.       3.375    8.      15.625   27.      42.875   64.      91.125 125.
 166.375]
[15626.     17579.375 19691.     21967.625 24416.     27042.875 29855.
 32859.125 36062.     39470.375]
125650.0
[125.00399994 132.58723543 140.32462364 148.21479346 156.25619988
 164.44717997 172.7859948  181.27086087 189.89997367 198.67152539]
```

1b) Append b to a, reshape the appended array so that it is a 4x5, 2d array and store the results in a variable called m. Print m.

In [285]:
```
# your code here
m = np.vstack([a[:5], a[5:], b[:5], b[5:]])
m.T
```

Out[285]:
```
array([[25. , 30. ,  1. ,  3.5],
       [26. , 31. ,  1.5,  4. ],
       [27. , 32. ,  2. ,  4.5],
       [28. , 33. ,  2.5,  5. ],
       [29. , 34. ,  3. ,  5.5]])
```

1c) Extract the third and the fourth column of the m matrix. Store the resulting 4x2 matrix in a new variable called m2. Print m2.

In [286]:
```
# your code here
m2 = m[:, [3,4]]
m2
```

Out[286]:
```
array([[28. , 29. ],
       [33. , 34. ],
       [ 2.5,  3. ],
       [ 5. ,  5.5]])
```

1d) Take the dot product of m2 and m store the results in a matrix called m3. Print m3. Note that Dot product of two matrices A.B =A^TB

In [287]:
```
# your code here
m3 = a.T * b
m3
```

Out[287]:
```
array([ 25.,  39.,  54.,  70.,  87., 105., 124., 144., 165., 187.])
```

Numpy conditions

2a) Create a numpy array called 'f' where the values are cosine(x) for x from 0 to pi with 50 equally spaced values in f

- Print f
- Use condition on the array and print an array that is True when f >= 1/2 and False when f < 1/2
- Create and print an array sequence that has only those values where f>= 1/2

In [288]:
```
# your code here
temp = np.arange(0.0, np.pi, np.pi/50, dtype=float)
f = np.cos(temp)
np.where(f >= .5, True, False)
```

Out[288]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False])
```

NumPy and 2 Variable Prediction

Let 'x' be the number of miles a person drives per day and 'y' be the dollars spent on buying car fuel (per day).

We have created 2 numpy arrays each of size 100 that represent x and y.
x (number of miles) ranges from 1 to 10 with a uniform noise of (0,1/2)
y (money spent in dollars) will be from 1 to 20 with a uniform noise (0,1)

In [289]:
```
# seed the random number generator with a fixed value
import numpy as np
np.random.seed(500)

x = np.linspace(1,10,100)+ np.random.uniform(low=0,high=.5,size=100)
y = np.linspace(1,20,100)+ np.random.uniform(low=0,high=1,size=100)
print('x = ',x)
print('y = ',y)
```

```
x =  [ 1.34683976  1.12176759  1.51512398  1.55233174  1.40619168  1.65
075498
  1.79399331  1.80243817  1.89844195  2.00100023  2.3344038   2.2242487
2
  2.24914511  2.36268477  2.49808849  2.8212704   2.68452475  2.6822942
7
  3.09511169  2.95703884  3.09047742  3.2544361   3.41541904  3.4088637
5
  3.50672677  3.74960644  3.64861355  3.7721462   3.56368566  4.0109270
1
  4.15630694  4.06088549  4.02517179  4.25169402  4.15897504  4.2683533
3
  4.32520644  4.48563164  4.78490721  4.84614839  4.96698768  5.1875425
9
  5.29582013  5.32097781  5.0674106   5.47601124  5.46852704  5.6453745
2
  5.49642807  5.89755027  5.68548923  5.76276141  5.94613234  6.1813571
3
  5.96522091  6.0275473   6.54290191  6.4991329   6.74003765  6.8180980
7
  6.50611821  6.91538752  7.01250925  6.89905417  7.31314433  7.2047229
7
  7.1043621   7.48199528  7.58957227  7.61744354  7.6991707   7.8543682
2
  8.03510784  7.80787781  8.22410224  7.99366248  8.40581097  8.2891379
2
  8.45971515  8.54227144  8.6906456   8.61856507  8.83489887  8.6630965
8
  8.94837987  9.20890222  8.9614749   8.92608294  9.13231416  9.5588989
6
  9.61488451  9.54252979  9.42015491  9.90952569 10.00659591 10.0250426
5
 10.07330937  9.93489915 10.0892334  10.36509991]
y =  [ 1.6635012   2.0214592   2.10816052  2.26016496  1.96287558  2.95
54635
  3.02881887  3.33565296  2.75465779  3.4250107   3.39670148  3.3937776
7
  3.78503343  4.38293049  4.32963586  4.03925039  4.73691868  4.3009839
9
  4.8416329   4.78175957  4.99765787  5.31746817  5.76844671  5.9372374
9
  5.72811642  6.70973615  6.68143367  6.57482731  7.17737603  7.5486325
2
  7.30221419  7.3202573   7.78023884  7.91133365  8.2765417   8.6920328
1
  8.78219865  8.45897546  8.89094715  8.81719921  8.87106971  9.6619256
2
  9.4020625   9.85990783  9.60359778 10.07386266 10.6957995  10.6672191
6
 11.18256285 10.57431836 11.46744716 10.94398916 11.26445259 12.0975482
8
 12.11988037 12.121557   12.17613693 12.43750193 13.00912372 12.8640719
4
 13.24640866 12.76120085 13.11723062 14.07841099 14.19821707 14.2728900
1
 14.30624942 14.63060835 14.2770918  15.0744923  14.45261619 15.1189731
3
```

```
 15.2378667  15.27203124 15.32491892 16.01095271 15.71250558 16.2948850
6
 16.70618934 16.56555394 16.42379457 17.18144744 17.13813976 17.6961362
5
 17.37763019 17.90942839 17.90343733 18.01951169 18.35727914 18.1684126
9
 18.61813748 18.66062754 18.81217983 19.44995194 19.7213867  19.7196672
6
 19.78961904 19.64385088 20.69719809 20.07974319]
```

3a) Find Expected value of x and the expected value of y

In [290]:
```
# your code here
print(np.mean(x))
print(np.mean(y))
```

```
5.782532541587923
11.012981683344968
```

3b) Find variance of distributions of x and y

In [291]:
```
# your code here
print(np.var(x))
print(np.var(y))
```

```
7.03332752947585
30.113903575509635
```

3c) Find co-variance of x and y.

In [292]:
```
# your code here
print(np.cov(x,y))
print(np.cov(x,y)[1][0])
```

```
[[ 7.10437124 14.65774383]
 [14.65774383 30.41808442]]
14.657743832803437
```

3d) Assuming that number of dollars spent in car fuel is only dependant on the miles driven, by a linear relationship. Write code that uses a linear predictor to calculate a predicted value of y for each x i.e y_predicted = f(x) = y0+mx. (Do not use Machine learning libraries)

In [293]:
```
# your code here
m,b = np.polyfit(x,y,1)
print(m,b)
y_predicted = m*x + b
```

```
2.063200715971324 -0.9175435965867221
```

3e) Predict y for each value in x, put the error into an array called y_error

In [294]:
```
# your code here
y_error = [y_predicted for b in x ][0]
y_error
```

Out[294]:
```
array([ 1.86125717,  1.39688809,  2.20846128,  2.28522836,  1.98371207,
        2.48829527,  2.78382468,  2.80124813,  2.9993232 ,  3.21092152,
        3.8988    ,  3.67152796,  3.7228942 ,  3.9571493 ,  4.23651436,
        4.9033035 ,  4.62116978,  4.61656787,  5.46829307,  5.18342105,
        5.45873164,  5.79701128,  6.12915141,  6.11562653,  6.31753758,
        6.81864709,  6.61027849,  6.86515115,  6.43505522,  7.35780389,
        7.65775187,  7.46087825,  7.38719373,  7.85455455,  7.66325667,
        7.88892606,  8.00622544,  8.33721481,  8.95468038,  9.08103323,
        9.33034895,  9.78539799, 10.00879629, 10.06070164,  9.53754157,
       10.38056671, 10.36512531, 10.72999716, 10.42269073, 11.25028634,
       10.81276185, 10.97218988, 11.35052091, 11.83583685, 11.38990445,
       11.51849632, 12.58177632, 12.49147206, 12.98850691, 13.14956122,
       12.50588416, 13.35028889, 13.5506705 , 13.31658991, 14.17094102,
       13.947246  , 13.74018137, 14.51931443, 14.74126735, 14.79877137,
       14.96739089, 15.28759454, 15.66049665, 15.1916755 , 16.05043004,
       15.57498655, 16.42533161, 16.18461169, 16.53654675, 16.70687695,
       17.01300263, 16.86428603, 17.31062607, 16.95616347, 17.54476017,
       18.08227006, 17.57177784, 17.49875711, 17.92425351, 18.80438359,
       18.91989301, 18.77061069, 18.51812677, 19.5277969 , 19.72807224,
       19.76613158, 19.8657155 , 19.58014745, 19.89856998, 20.4677379
7])
```

3f) Write code that calculates the root mean square error(RMSE), that is root of average of y-error squared__

In [295]:
```
# your code here
np.sqrt(np.mean((y_error-y)**2))
```

Out[295]:
```
0.4176777236685612
```

Pandas Introduction

Reading File

In [296]:
```
# Load required modules
import pandas as pd
import numpy as np
```

Read the CSV file called 'data3.csv' into a dataframe called df.

Data description

- File location: https://bcourses.berkeley.edu/files/74463396/download?download_frd=1 (https://bcourses.berkeley.edu/files/74463396/download?download_frd=1)
- Data source: http://www.fao.org/nr/water/aquastat/data/query/index.html?*lang=en (http://www.fao.org/nr/water/aquastat/data/query/index.html?*lang=en)
- Data, units:
- GDP, current USD (CPI adjusted)
- NRI, mm/yr
- Population density, inhab/km^2
- Total area of the country, 1000 ha = 10km^2
- Total Population, unit 1000 inhabitants

In [297]:
```
# your code here
file = "data3.csv"
data = pd.read_csv(file)
```

4a) Display the first 10 rows of the dataframe

In [298]:
```
# your code here
data.head(10)
```

Out[298]:

	Area	Area Id	Variable Name	Variable Id	Year	Value	Symbol	Other
0	Argentina	9.0	Total area of the country	4100.0	1962.0	278040.0	E	NaN
1	Argentina	9.0	Total area of the country	4100.0	1967.0	278040.0	E	NaN
2	Argentina	9.0	Total area of the country	4100.0	1972.0	278040.0	E	NaN
3	Argentina	9.0	Total area of the country	4100.0	1977.0	278040.0	E	NaN
4	Argentina	9.0	Total area of the country	4100.0	1982.0	278040.0	E	NaN
5	Argentina	9.0	Total area of the country	4100.0	1987.0	278040.0	E	NaN
6	Argentina	9.0	Total area of the country	4100.0	1992.0	278040.0	E	NaN
7	Argentina	9.0	Total area of the country	4100.0	1997.0	278040.0	E	NaN
8	Argentina	9.0	Total area of the country	4100.0	2002.0	278040.0	E	NaN
9	Argentina	9.0	Total area of the country	4100.0	2007.0	278040.0	E	NaN

4b) Display the column names.

In [299]:
```
# your code here
data.columns
```

Out[299]:
```
Index(['Area', 'Area Id', 'Variable Name', 'Variable Id', 'Year', 'Valu
e',
       'Symbol', 'Other'],
      dtype='object')
```

4c) Use iloc to display the first 3 rows and first 4 columns.

In [300]:
```
# your code here
data.iloc[0:3, 0:4]
```

Out[300]:

	Area	Area Id	Variable Name	Variable Id
0	Argentina	9.0	Total area of the country	4100.0
1	Argentina	9.0	Total area of the country	4100.0
2	Argentina	9.0	Total area of the country	4100.0

Data Preprocessing

5a) Find all the rows that have 'NaN' in the 'Symbol' column. Display first 5 rows.

Hint : You might have to use a condition (mask)

In [301]:
```
# your code here
#null_col = data.columns[data.isnull().any()]
print(data[data['Symbol'].isnull()][null_columns].head(5))
```

```
                      Area  Area Id Variable Name  Variable Id  Year  V
alue  \
390                    NaN      NaN           NaN          NaN   NaN
NaN
391      E - External data      NaN           NaN          NaN   NaN
NaN
392  I - AQUASTAT estimate      NaN           NaN          NaN   NaN
NaN
393     K - Aggregate data      NaN           NaN          NaN   NaN
NaN
394      L - Modelled data      NaN           NaN          NaN   NaN
NaN

    Symbol  Other
390    NaN    NaN
391    NaN    NaN
392    NaN    NaN
393    NaN    NaN
394    NaN    NaN
```

5b) Now, we will try to get rid of the NaN valued rows and columns. First, drop the column 'Other' which only has 'NaN' values. Then drop all other rows that have any column with a value 'NaN'. Then display the last 5 rows of the dataframe.

In [302]:
```
# your code here
data = data.drop(columns = 'Other')
```

In [303]:
```
null_row = data[data.isnull().any(axis=1)]
null_row.index
data = data.drop(null_row.index)
data.tail(5)
```

Out[303]:

	Area	Area Id	Variable Name	Variable Id	Year	Value	Symbol
385	United States of America	231.0	National Rainfall Index (NRI)	4472.0	1981.0	949.2	E
386	United States of America	231.0	National Rainfall Index (NRI)	4472.0	1984.0	974.6	E
387	United States of America	231.0	National Rainfall Index (NRI)	4472.0	1992.0	1020.0	E
388	United States of America	231.0	National Rainfall Index (NRI)	4472.0	1996.0	1005.0	E
389	United States of America	231.0	National Rainfall Index (NRI)	4472.0	2002.0	938.7	E

6a) For our analysis we do not want all the columns in our dataframe. Lets drop all the redundant columns/ features.

Drop columns: Area Id, Variable Id, Symbol. Save the new dataframe as df1. Display the first 5 rows of the new dataframe.

In [304]:
```
# your code here
df1 = data.drop(columns = ['Area Id', 'Variable Id', 'Symbol'])
df1.head(5)
```

Out[304]:

	Area	Variable Name	Year	Value
0	Argentina	Total area of the country	1962.0	278040.0
1	Argentina	Total area of the country	1967.0	278040.0
2	Argentina	Total area of the country	1972.0	278040.0
3	Argentina	Total area of the country	1977.0	278040.0
4	Argentina	Total area of the country	1982.0	278040.0

6b) Display all the unique values in your new dataframe for columns: Area, Variable Name, Year.

In [305]:
```
# your code here
print(df1["Area"].unique())
print(df1["Variable Name"].unique())
print(df1["Year"].unique())
```

```
['Argentina' 'Australia' 'Germany' 'Iceland' 'Ireland' 'Sweden'
 'United States of America']
['Total area of the country' 'Total population' 'Population density'
 'Gross Domestic Product (GDP)' 'National Rainfall Index (NRI)']
[1962. 1967. 1972. 1977. 1982. 1987. 1992. 1997. 2002. 2007. 2012. 201
4.
 2015. 1963. 1970. 1974. 1978. 1984. 1990. 1964. 1981. 1985. 1996. 200
1.
 1969. 1973. 1979. 1993. 1971. 1975. 1986. 1991. 1998. 2000. 1965. 198
3.
 1988. 1995.]
```

6c) Convert the year column to pandas datetime. Convert the 'Year' column float values to pandas datetime objects, where each year is represented as the first day of that year. Also display the column and datatype for 'Year' after conversion. For eg: 1962.0 will be represented as 1962-01-01¶

In [306]:
```
df1['month'] = 1
df1['day'] = 1
df1['Year'] = pd.to_datetime(df1.drop(columns =  ['Area','Value','Variab
le Name']))
df1 = df1.drop(columns = ['month','day'])
df1
```

Out[306]:

	Area	Variable Name	Year	Value
0	Argentina	Total area of the country	1962-01-01	2.780400e+05
1	Argentina	Total area of the country	1967-01-01	2.780400e+05
2	Argentina	Total area of the country	1972-01-01	2.780400e+05
3	Argentina	Total area of the country	1977-01-01	2.780400e+05
4	Argentina	Total area of the country	1982-01-01	2.780400e+05
5	Argentina	Total area of the country	1987-01-01	2.780400e+05
6	Argentina	Total area of the country	1992-01-01	2.780400e+05
7	Argentina	Total area of the country	1997-01-01	2.780400e+05
8	Argentina	Total area of the country	2002-01-01	2.780400e+05
9	Argentina	Total area of the country	2007-01-01	2.780400e+05
10	Argentina	Total area of the country	2012-01-01	2.780400e+05
11	Argentina	Total area of the country	2014-01-01	2.780400e+05
12	Argentina	Total population	1962-01-01	2.128800e+04
13	Argentina	Total population	1967-01-01	2.293200e+04
14	Argentina	Total population	1972-01-01	2.478300e+04
15	Argentina	Total population	1977-01-01	2.687900e+04
16	Argentina	Total population	1982-01-01	2.899400e+04
17	Argentina	Total population	1987-01-01	3.132600e+04
18	Argentina	Total population	1992-01-01	3.365500e+04
19	Argentina	Total population	1997-01-01	3.583400e+04
20	Argentina	Total population	2002-01-01	3.788900e+04
21	Argentina	Total population	2007-01-01	3.997000e+04
22	Argentina	Total population	2012-01-01	4.209500e+04
23	Argentina	Total population	2015-01-01	4.341700e+04
24	Argentina	Population density	1962-01-01	7.656000e+00
25	Argentina	Population density	1967-01-01	8.248000e+00
26	Argentina	Population density	1972-01-01	8.913000e+00
27	Argentina	Population density	1977-01-01	9.667000e+00
28	Argentina	Population density	1982-01-01	1.043000e+01
29	Argentina	Population density	1987-01-01	1.127000e+01
...	...	...	...	...
360	United States of America	Population density	1972-01-01	2.214000e+01
361	United States of America	Population density	1977-01-01	2.317000e+01
362	United States of America	Population density	1982-01-01	2.430000e+01

	Area	Variable Name	Year	Value
363	United States of America	Population density	1987-01-01	2.549000e+01
364	United States of America	Population density	1992-01-01	2.678000e+01
365	United States of America	Population density	1997-01-01	2.834000e+01
366	United States of America	Population density	2002-01-01	2.995000e+01
367	United States of America	Population density	2007-01-01	3.132000e+01
368	United States of America	Population density	2012-01-01	3.202000e+01
369	United States of America	Population density	2015-01-01	3.273000e+01
370	United States of America	Gross Domestic Product (GDP)	1962-01-01	6.050000e+11
371	United States of America	Gross Domestic Product (GDP)	1967-01-01	8.620000e+11
372	United States of America	Gross Domestic Product (GDP)	1972-01-01	1.280000e+12
373	United States of America	Gross Domestic Product (GDP)	1977-01-01	2.090000e+12
374	United States of America	Gross Domestic Product (GDP)	1982-01-01	3.340000e+12
375	United States of America	Gross Domestic Product (GDP)	1987-01-01	4.870000e+12
376	United States of America	Gross Domestic Product (GDP)	1992-01-01	6.540000e+12
377	United States of America	Gross Domestic Product (GDP)	1997-01-01	8.610000e+12
378	United States of America	Gross Domestic Product (GDP)	2002-01-01	1.100000e+13
379	United States of America	Gross Domestic Product (GDP)	2007-01-01	1.450000e+13
380	United States of America	Gross Domestic Product (GDP)	2012-01-01	1.620000e+13
381	United States of America	Gross Domestic Product (GDP)	2015-01-01	1.790000e+13
382	United States of America	National Rainfall Index (NRI)	1965-01-01	9.285000e+02
383	United States of America	National Rainfall Index (NRI)	1969-01-01	9.522000e+02
384	United States of America	National Rainfall Index (NRI)	1974-01-01	1.008000e+03
385	United States of America	National Rainfall Index (NRI)	1981-01-01	9.492000e+02
386	United States of America	National Rainfall Index (NRI)	1984-01-01	9.746000e+02
387	United States of America	National Rainfall Index (NRI)	1992-01-01	1.020000e+03
388	United States of America	National Rainfall Index (NRI)	1996-01-01	1.005000e+03
389	United States of America	National Rainfall Index (NRI)	2002-01-01	9.387000e+02

390 rows × 4 columns

Extract specific statistics from the preprocessed data:

7a) Create a dataframe 'dftemp' to store rows where Area is 'Iceland'. Display the dataframe.

In [319]:
```
# your code here
dftemp = (df1[df1['Area'] == "Iceland"])
dftemp
```

Out[319]:

	Area	Variable Name	Year	Value
166	Iceland	Total area of the country	1962-01-01	1.030000e+04
167	Iceland	Total area of the country	1967-01-01	1.030000e+04
168	Iceland	Total area of the country	1972-01-01	1.030000e+04
169	Iceland	Total area of the country	1977-01-01	1.030000e+04
170	Iceland	Total area of the country	1982-01-01	1.030000e+04
171	Iceland	Total area of the country	1987-01-01	1.030000e+04
172	Iceland	Total area of the country	1992-01-01	1.030000e+04
173	Iceland	Total area of the country	1997-01-01	1.030000e+04
174	Iceland	Total area of the country	2002-01-01	1.030000e+04
175	Iceland	Total area of the country	2007-01-01	1.030000e+04
176	Iceland	Total area of the country	2012-01-01	1.030000e+04
177	Iceland	Total area of the country	2014-01-01	1.030000e+04
178	Iceland	Total population	1962-01-01	1.826000e+02
179	Iceland	Total population	1967-01-01	1.974000e+02
180	Iceland	Total population	1972-01-01	2.099000e+02
181	Iceland	Total population	1977-01-01	2.221000e+02
182	Iceland	Total population	1982-01-01	2.331000e+02
183	Iceland	Total population	1987-01-01	2.469000e+02
184	Iceland	Total population	1992-01-01	2.599000e+02
185	Iceland	Total population	1997-01-01	2.728000e+02
186	Iceland	Total population	2002-01-01	2.869000e+02
187	Iceland	Total population	2007-01-01	3.054000e+02
188	Iceland	Total population	2012-01-01	3.234000e+02
189	Iceland	Total population	2015-01-01	3.294000e+02
190	Iceland	Population density	1962-01-01	1.773000e+00
191	Iceland	Population density	1967-01-01	1.917000e+00
192	Iceland	Population density	1972-01-01	2.038000e+00
193	Iceland	Population density	1977-01-01	2.156000e+00
194	Iceland	Population density	1982-01-01	2.263000e+00
195	Iceland	Population density	1987-01-01	2.397000e+00
196	Iceland	Population density	1992-01-01	2.523000e+00
197	Iceland	Population density	1997-01-01	2.649000e+00
198	Iceland	Population density	2002-01-01	2.785000e+00
199	Iceland	Population density	2007-01-01	2.965000e+00

	Area	Variable Name	Year	Value
200	Iceland	Population density	2012-01-01	3.140000e+00
201	Iceland	Population density	2015-01-01	3.198000e+00
202	Iceland	Gross Domestic Product (GDP)	1962-01-01	2.849165e+08
203	Iceland	Gross Domestic Product (GDP)	1967-01-01	6.212260e+08
204	Iceland	Gross Domestic Product (GDP)	1972-01-01	8.465069e+08
205	Iceland	Gross Domestic Product (GDP)	1977-01-01	2.226539e+09
206	Iceland	Gross Domestic Product (GDP)	1982-01-01	3.232804e+09
207	Iceland	Gross Domestic Product (GDP)	1987-01-01	5.565384e+09
208	Iceland	Gross Domestic Product (GDP)	1992-01-01	7.138788e+09
209	Iceland	Gross Domestic Product (GDP)	1997-01-01	7.596126e+09
210	Iceland	Gross Domestic Product (GDP)	2002-01-01	9.161798e+09
211	Iceland	Gross Domestic Product (GDP)	2007-01-01	2.129384e+10
212	Iceland	Gross Domestic Product (GDP)	2012-01-01	1.419452e+10
213	Iceland	Gross Domestic Product (GDP)	2015-01-01	1.659849e+10
214	Iceland	National Rainfall Index (NRI)	1967-01-01	8.160000e+02
215	Iceland	National Rainfall Index (NRI)	1971-01-01	9.632000e+02
216	Iceland	National Rainfall Index (NRI)	1975-01-01	1.010000e+03
217	Iceland	National Rainfall Index (NRI)	1981-01-01	9.326000e+02
218	Iceland	National Rainfall Index (NRI)	1986-01-01	9.685000e+02
219	Iceland	National Rainfall Index (NRI)	1991-01-01	1.095000e+03
220	Iceland	National Rainfall Index (NRI)	1997-01-01	9.932000e+02
221	Iceland	National Rainfall Index (NRI)	1998-01-01	9.234000e+02

7b) Print the years when the National Rainfall Index (NRI) was greater than 900 and less than 950 in Iceland. Use the dataframe you created in the previous question 'dftemp'.

In [329]:

```
# your code here
temp2 = (dftemp[dftemp['Variable Name'] == "National Rainfall Index (NR
I)"])
print((temp2[temp2['Value'].between(900, 950)]))
```

```
        Area                  Variable Name        Year  Value
217  Iceland  National Rainfall Index (NRI) 1981-01-01  932.6
221  Iceland  National Rainfall Index (NRI) 1998-01-01  923.4
```

US statistics:

8a) Create a new DataFrame called `df_usa` that only contains values where 'Area' is equal to 'United States of America'. Set the indices to be the 'Year' column (Use .set_index( )). Display the dataframe head.

In [335]:
```
# your code here
df_usa = (df1[df1['Area'] == "United States of America"]).set_index('Yea
r')
df_usa.head()
```

Out[335]:

	Area	Variable Name	Value
Year			
1962-01-01	United States of America	Total area of the country	962909.0
1967-01-01	United States of America	Total area of the country	962909.0
1972-01-01	United States of America	Total area of the country	962909.0
1977-01-01	United States of America	Total area of the country	962909.0
1982-01-01	United States of America	Total area of the country	962909.0

8b) Pivot the DataFrame so that the unique values in the column 'Variable Name' becomes the columns. The DataFrame values should be the ones in the the 'Value' column. Save it in df_usa. Display the dataframe head.

In [340]:
```
# your code here
df_usa = df_usa.pivot(columns = "Variable Name", values = "Value")
df_usa.head()
```

Out[340]:

Variable Name	Gross Domestic Product (GDP)	National Rainfall Index (NRI)	Population density	Total area of the country	Total population
Year					
1962-01-01	6.050000e+11	NaN	19.93	962909.0	191861.0
1965-01-01	NaN	928.5	NaN	NaN	NaN
1967-01-01	8.620000e+11	NaN	21.16	962909.0	203713.0
1969-01-01	NaN	952.2	NaN	NaN	NaN
1972-01-01	1.280000e+12	NaN	22.14	962909.0	213220.0

8c) Rename new columns to ['GDP','NRI','PD','Area','Population'] and display the head.

In [342]:
```
# your code here
df_usa = df_usa.rename(index=str, columns={"Gross Domestic Product (GD
P)": "GDP",
                                   "National Rainfall Index (NRI)": "NRI"
,
                                   "Population density" : "PD",
                                   'Total area of the country': "Area",
                                   'Total population': 'Population'
                                  })
df_usa.head()
```

Out[342]:

Variable Name	GDP	NRI	PD	Area	Population
Year					
1962-01-01 00:00:00	6.050000e+11	NaN	19.93	962909.0	191861.0
1965-01-01 00:00:00	NaN	928.5	NaN	NaN	NaN
1967-01-01 00:00:00	8.620000e+11	NaN	21.16	962909.0	203713.0
1969-01-01 00:00:00	NaN	952.2	NaN	NaN	NaN
1972-01-01 00:00:00	1.280000e+12	NaN	22.14	962909.0	213220.0

8d) Replace all 'Nan' values in df_usa with 0. Display the head of the dataframe.

In [345]:
```
# your code here
df_usa = df_usa.fillna(0)
df_usa.head()
```

Out[345]:

Variable Name	GDP	NRI	PD	Area	Population
Year					
1962-01-01 00:00:00	6.050000e+11	0.0	19.93	962909.0	191861.0
1965-01-01 00:00:00	0.000000e+00	928.5	0.00	0.0	0.0
1967-01-01 00:00:00	8.620000e+11	0.0	21.16	962909.0	203713.0
1969-01-01 00:00:00	0.000000e+00	952.2	0.00	0.0	0.0
1972-01-01 00:00:00	1.280000e+12	0.0	22.14	962909.0	213220.0

Note: Use df_usa

9a) Multiply the 'Area' column for all countries by 10 (so instead of 1000 ha, the unit becomes 100 ha = 1km^2). Display the dataframe head.

In [347]:
```
# your code here
df_usa['Area'] = df_usa['Area']*10
df_usa.head()
```

Out[347]:

Variable Name	GDP	NRI	PD	Area	Population
Year					
1962-01-01 00:00:00	6.050000e+11	0.0	19.93	9629090.0	191861.0
1965-01-01 00:00:00	0.000000e+00	928.5	0.00	0.0	0.0
1967-01-01 00:00:00	8.620000e+11	0.0	21.16	9629090.0	203713.0
1969-01-01 00:00:00	0.000000e+00	952.2	0.00	0.0	0.0
1972-01-01 00:00:00	1.280000e+12	0.0	22.14	9629090.0	213220.0

9b) Create a new column in df_usa called 'GDP/capita' and populate it with the calculated GDP per capita. Round the results to two decimal points. Display the dataframe head.

GDP per capita = (GDP / Population)

In [352]:
```
# your code here
df_usa['GDP/capita'] = df_usa['GDP']/df_usa['Population']
df_usa['GDP/capita'] = df_usa['GDP/capita'].round(2)
df_usa.head()
```

Out[352]:

Variable Name	GDP	NRI	PD	Area	Population	GDP/capita
Year						
1962-01-01 00:00:00	6.050000e+11	0.0	19.93	9629090.0	191861.0	3153324.54
1965-01-01 00:00:00	0.000000e+00	928.5	0.00	0.0	0.0	NaN
1967-01-01 00:00:00	8.620000e+11	0.0	21.16	9629090.0	203713.0	4231443.26
1969-01-01 00:00:00	0.000000e+00	952.2	0.00	0.0	0.0	NaN
1972-01-01 00:00:00	1.280000e+12	0.0	22.14	9629090.0	213220.0	6003189.19

9c) Find the maximum value of the 'NRI' column in the US (using pandas methods). What year does the max value occur? Display the values.

In [354]:
```
# your code here
print(df_usa['NRI'].max())
df_usa.loc[df_usa['NRI'].idxmax()]
```

```
1020.0
```

Out[354]:
```
Variable Name
GDP           6.540000e+12
NRI           1.020000e+03
PD            2.678000e+01
Area          9.629090e+06
Population    2.579080e+05
GDP/capita    2.535788e+07
Name: 1992-01-01 00:00:00, dtype: float64
```

In [ ]: