PANDAS FOUNDATION

INGESTION AND INSPECTION

Importing the libraries

In [7]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [8]:
```
worldpopulation=pd.read_csv ('C://Users//paul//Documents//STUTERN PANDAS PROJECT/
```

In [9]:
```
worldpopulation
```

Out[9]:

	CountryName	CountryCode	Year	Total Population	Urban population (% of total)
0	Arab World	ARB	1960	92495902	31.285384
1	Caribbean small states	CSS	1960	4190810	31.597490
2	Central Europe and the Baltics	CEB	1960	91401583	44.507921
3	East Asia & Pacific (all income levels)	EAS	1960	1042475394	22.471132
4	East Asia & Pacific (developing only)	EAP	1960	896492991	16.917679
...	...	...	...	...	...
13369	Virgin Islands (U.S.)	VIR	2014	104170	95.203000
13370	West Bank and Gaza	WBG	2014	4294682	75.026000
13371	Yemen, Rep.	YEM	2014	26183676	34.027000
13372	Zambia	ZMB	2014	15721343	40.472000
13373	Zimbabwe	ZWE	2014	15245855	32.501000

13374 rows × 5 columns

Inspecting data (Head, Tail)

In [10]:
```
worldpopulation.head()
```

Out[10]:

	CountryName	CountryCode	Year	Total Population	Urban population (% of total)
0	Arab World	ARB	1960	92495902	31.285384
1	Caribbean small states	CSS	1960	4190810	31.597490
2	Central Europe and the Baltics	CEB	1960	91401583	44.507921
3	East Asia & Pacific (all income levels)	EAS	1960	1042475394	22.471132
4	East Asia & Pacific (developing only)	EAP	1960	896492991	16.917679

In [11]:
```
worldpopulation.tail()
```

Out[11]:

	CountryName	CountryCode	Year	Total Population	Urban population (% of total)
13369	Virgin Islands (U.S.)	VIR	2014	104170	95.203
13370	West Bank and Gaza	WBG	2014	4294682	75.026
13371	Yemen, Rep.	YEM	2014	26183676	34.027
13372	Zambia	ZMB	2014	15721343	40.472
13373	Zimbabwe	ZWE	2014	15245855	32.501

Answer;

for the first and last row ;First: 1960, 92495902.0; Last: 2014, 15245855.0.

NumPy and pandas working together

Creating np_vals (array of DataFrame values)

In [12]:
```
np_vals = worldpopulation[['Total Population', 'Urban population (% of total)']].
```

In [13]:
```
np_vals
```

Out[13]:
```
array([[9.24959020e+07, 3.12853842e+01],
       [4.19081000e+06, 3.15974898e+01],
       [9.14015830e+07, 4.45079211e+01],
       ...,
       [2.61836760e+07, 3.40270000e+01],
       [1.57213430e+07, 4.04720000e+01],
       [1.52458550e+07, 3.25010000e+01]])
```

Creating Log 10 values

In [14]:
```
np_valslog10 = np.log10(np_vals)
```

In [15]:
```
np_valslog10
```

Out[15]:
```
array([[7.96612249, 1.49534149],
       [6.62229797, 1.49965258],
       [7.96095372, 1.64843731],
       ...,
       [7.41803062, 1.53182366],
       [7.19648964, 1.60715467],
       [7.18315178, 1.51189672]])
```

DataFrame data types

In [16]:
```
worldpopulation.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13374 entries, 0 to 13373
Data columns (total 5 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   CountryName                   13374 non-null  object 
 1   CountryCode                   13374 non-null  object 
 2   Year                          13374 non-null  int64  
 3   Total Population              13374 non-null  int64  
 4   Urban population (% of total) 13374 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 522.5+ KB
```

Zip lists to build a DataFrame

In [17]:
```
countries = [ 'United States', 'Russia', 'Germany']
```

In [18]:
```
total = [1127, 726, 578]
```

In [19]:
```
list_keys = [ 'countries', 'total']
```

In [20]:
```
list_values = [countries, total]
```

In [21]:
```
 zipped = list(zip(list_keys, list_values))
```

In [22]:
```
zipped
```

Out[22]:
```
[('countries', ['United States', 'Russia', 'Germany']),
 ('total', [1127, 726, 578])]
```

```
In [23]: data = dict(zipped)
```

```
In [24]: df = pd.DataFrame(data)
```

```
In [25]: print(df)
```

```
       countries  total
0  United States   1127
1         Russia    726
2        Germany    578
```

Labeling your data

```
In [26]: df=pd.read_csv ('C://Users//paul//Documents//STUTERN PANDAS PROJECT//Billboards.c
```

Create a list of new column labels with 'year', 'artist', 'song', 'chart weeks', and assign it to list_labels. Assign your list of labels to df.columns.

```
In [27]: df.columns= ['year','artist','song', 'chart weeks']
```

```
In [28]: df
```

Out[28]:

	year	artist	song	chart weeks
0	29225	KC and the Sunshine Band	Please Don't Go	1
1	29239	Michael Jackson	Rock with You	4
2	29267	Captain & Tennille	Do That to Me One More Time	1
3	29274	Queen	Crazy Little Thing Called Love	4
4	29302	Pink Floyd	Another Brick in the Wall (Part 2)	4
5	29330	Blondie	Call Me? (1980)	6
6	29372	Lipps, Inc.	Funkytown	4
7	29400	Paul McCartney	Coming Up (Live at Glasgow)	3
8	29421	Billy Joel	It's Still Rock and Roll to Me	2
9	29435	Olivia Newton-John	Magic	4
10	29463	Christopher Cross	Sailing	1
11	29470	Diana Ross	Upside Down	4
12	29498	Queen	Another One Bites the Dust	3
13	29519	Barbra Streisand	Woman in Love	3
14	29540	Kenny Rogers	Lady	6
15	29582	John Lennon	(Just Like) Starting Over	5

Building DataFrames with broadcasting

Make a string object with the value 'PA' and assign it to state. Construct a dictionary with 2 key:value pairs: 'state':state and 'city':cities. Construct a pandas DataFrame from the dictionary you created and assign it to df.

In [29]:
```
 cities = ['city', 'city', 'city', 'city', 'city', 'city', 'city']
```

In [30]:
```
data = {'city':  cities, 'state': 'PA'}
```

In [31]:
```
df = pd.DataFrame(data)
```

In [32]:
```
df
```

Out[32]:

	city	state
0	city	PA
1	city	PA
2	city	PA
3	city	PA
4	city	PA
5	city	PA
6	city	PA

Reading a flat file

In [39]:
```
df_b ='C://Users//paul//Documents//STUTERN PANDAS PROJECT//World Bank World Devel
```

In [40]:
```
worldpopulation_b = pd.read_csv(df_b)
```

```
In [41]: worldpopulation_b
```

Out[41]:

	CountryName	CountryCode	Year	Total Population	Urban population (% of total)
0	Arab World	ARB	1960	92495902	31.285384
1	Caribbean small states	CSS	1960	4190810	31.597490
2	Central Europe and the Baltics	CEB	1960	91401583	44.507921
3	East Asia & Pacific (all income levels)	EAS	1960	1042475394	22.471132
4	East Asia & Pacific (developing only)	EAP	1960	896492991	16.917679
...	...	...	...	...	...
13369	Virgin Islands (U.S.)	VIR	2014	104170	95.203000
13370	West Bank and Gaza	WBG	2014	4294682	75.026000
13371	Yemen, Rep.	YEM	2014	26183676	34.027000
13372	Zambia	ZMB	2014	15721343	40.472000
13373	Zimbabwe	ZWE	2014	15245855	32.501000

13374 rows × 5 columns

In [42]:
```
# Create a list of the new column labels: new_labels
new_labels = ['country', 'country code', 'year', 'population', 'urban population

# # Read in the file, specifying the header and names parameters
worldpopulation_c = pd.read_csv(df_b, header=0, names = new_labels)

# check
worldpopulation_c
```

Out[42]:

	country	country code	year	population	urban population as percentage of total
0	Arab World	ARB	1960	92495902	31.285384
1	Caribbean small states	CSS	1960	4190810	31.597490
2	Central Europe and the Baltics	CEB	1960	91401583	44.507921
3	East Asia & Pacific (all income levels)	EAS	1960	1042475394	22.471132
4	East Asia & Pacific (developing only)	EAP	1960	896492991	16.917679
...	...	...	...	...	...
13369	Virgin Islands (U.S.)	VIR	2014	104170	95.203000
13370	West Bank and Gaza	WBG	2014	4294682	75.026000
13371	Yemen, Rep.	YEM	2014	26183676	34.027000
13372	Zambia	ZMB	2014	15721343	40.472000
13373	Zimbabwe	ZWE	2014	15245855	32.501000

13374 rows × 5 columns

Delimiters, headers, and extensions

In [49]:
```
# get the file path
messy = 'C://Users//paul//Documents//STUTERN PANDAS PROJECT//messy.csv'

# read the messy file as a csv
df_c = pd.read_csv(messy)
```

In [50]:

```
# Read in the file with the correct parameters: df2
df_d = pd.read_csv(messy, delimiter=' ', header = 3, comment = '#')

#check
df_d.head()
```

Out[50]:

	name	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov
0	IBM	156.08	160.01	159.81	165.22	172.25	167.15	164.75	152.77	145.36	146.11	137.21
1	MSFT	45.51	43.08	42.13	43.47	47.53	45.96	45.61	45.51	43.56	48.70	53.88
2	GOOGLE	512.42	537.99	559.72	540.50	535.24	532.92	590.09	636.84	617.93	663.59	735.39
3	APPLE	110.64	125.43	125.97	127.29	128.76	127.81	125.34	113.39	112.80	113.36	118.16

In [51]:

```
# Save the cleaned up DataFrame to a CSV file without the index
df_d.to_csv('file_clean', index=False)

# Save the cleaned up DataFrame to an excel file without the index
df_d.to_excel('clean.xlsx', index=False)
```

Plotting series using pandas

In [53]:

```
# Austin temperature data
temperature_data = 'C://Users//paul//Documents//STUTERN PANDAS PROJECT//2010 Aust

# read in the data set
austin = pd.read_csv(temperature_data)
```

In [54]:

```
austin
```

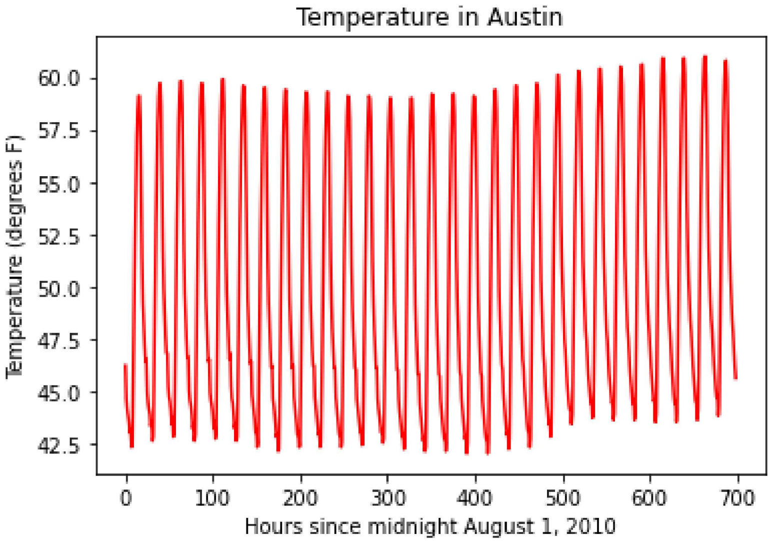
Out[54]:

	Temperature	DewPoint	Pressure	Date
0	46.2	37.5	1.0	20100101 00:00
1	44.6	37.1	1.0	20100101 01:00
2	44.1	36.9	1.0	20100101 02:00
3	43.8	36.9	1.0	20100101 03:00
4	43.5	36.8	1.0	20100101 04:00
...	...	...	...	...
8754	51.1	38.1	1.0	20101231 19:00
8755	49.0	37.9	1.0	20101231 20:00
8756	47.9	37.9	1.0	20101231 21:00
8757	46.9	37.9	1.0	20101231 22:00
8758	46.2	37.7	1.0	20101231 23:00

8759 rows × 4 columns

In [55]:

```
# temperature column
temp = austin.iloc[:700, 0]

# Create a plot with color='red'
temp.plot(color = 'red')

# title
plt.title('Temperature in Austin')

# Specify the x-axis
plt.xlabel('Hours since midnight August 1, 2010')

# Specify the y-axis label
plt.ylabel('Temperature (degrees F)')

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



Plotting with DataFrames

In [56]:

```
# lets get only the temperature column
temp = austin.iloc[:800, 0]
temp
```

Out[56]:

```
0      46.2
1      44.6
2      44.1
3      43.8
4      43.5
       ...
795    46.2
796    45.7
797    45.1
798    45.5
799    44.6
Name: Temperature, Length: 800, dtype: float64
```

In [60]:
```
# Get just the numeric columns
austin_col = austin.iloc[0:700, 0:3]
```

In [61]:
```
austin_col
```

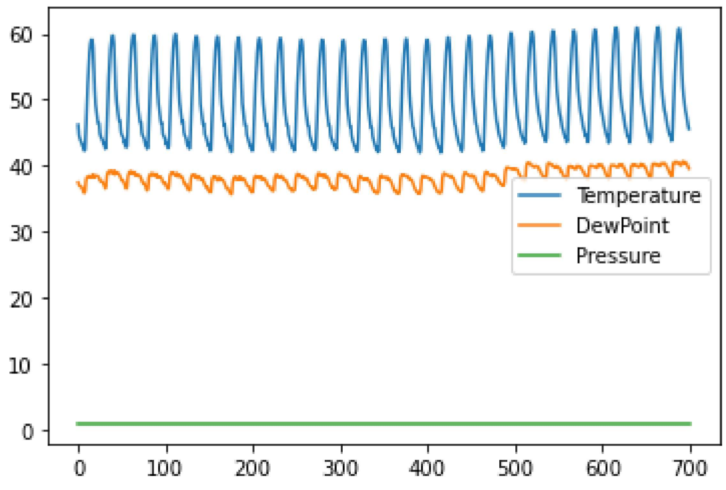
Out[61]:

	Temperature	DewPoint	Pressure
0	46.2	37.5	1.0
1	44.6	37.1	1.0
2	44.1	36.9	1.0
3	43.8	36.9	1.0
4	43.5	36.8	1.0
...	...	...	...
695	48.4	40.6	1.0
696	47.8	40.3	1.0
697	46.9	40.2	1.0
698	46.2	39.9	1.0
699	45.6	39.6	1.0

700 rows × 3 columns

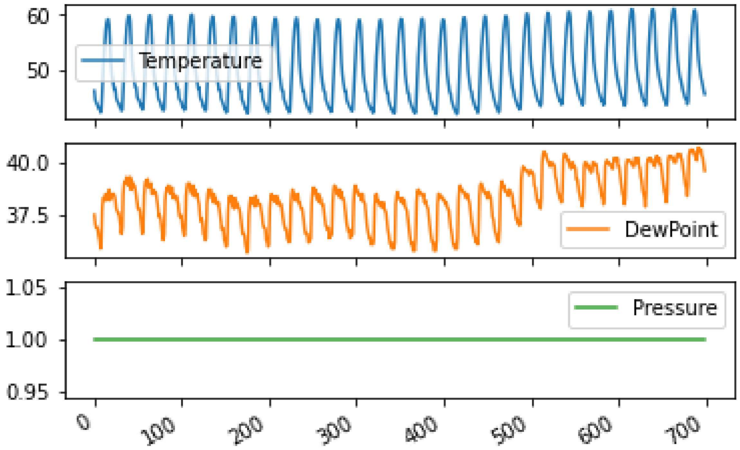
In [62]:
```
# plot all columns in same figure
austin_col.plot()
```

Out[62]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x13b5c9e0160>
```



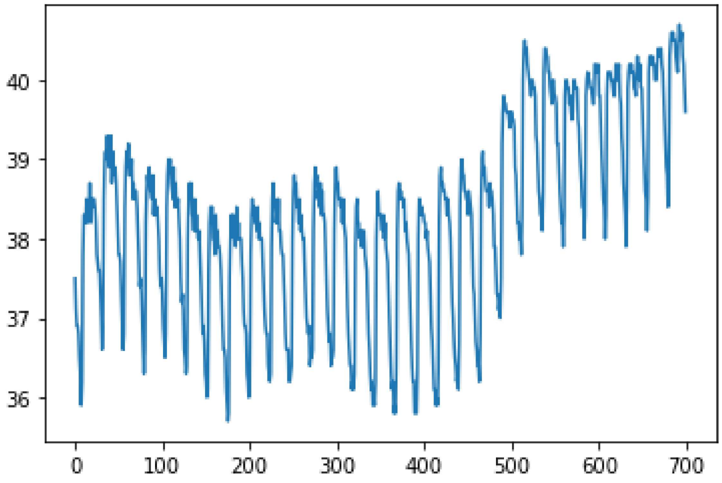
In [63]:

```
# make separate subplots for each column
austin_col.plot(subplots=True)
plt.show()
```

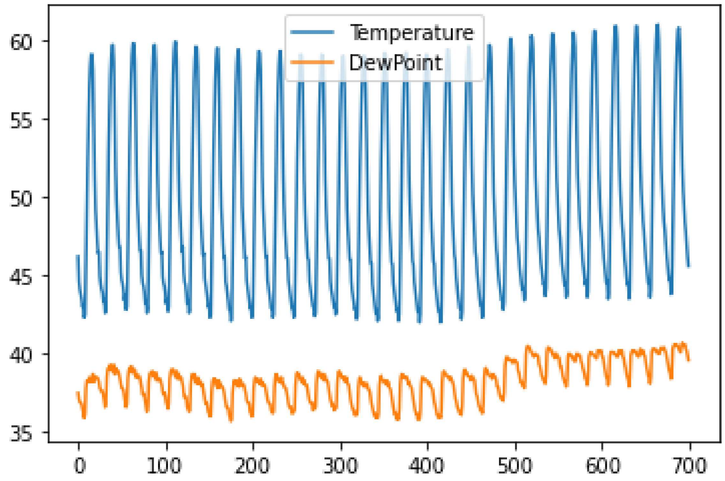


In [65]:

```
# plot just one columns data
austin_col['DewPoint'].plot()
plt.show()
```



In [66]:
```
# plot two columns data
austin_col[['Temperature', 'DewPoint']].plot()
plt.show()
```



Exploratory Data Analysis

In [68]:

```
# lets transpose the messy data we cleaned in order to get the stock prices data
df_e = df_d.transpose()
df_e
```

Out[68]:

	0	1	2	3
name	IBM	MSFT	GOOGLE	APPLE
Jan	156.08	45.51	512.42	110.64
Feb	160.01	43.08	537.99	125.43
Mar	159.81	42.13	559.72	125.97
Apr	165.22	43.47	540.5	127.29
May	172.25	47.53	535.24	128.76
Jun	167.15	45.96	532.92	127.81
Jul	164.75	45.61	590.09	125.34
Aug	152.77	45.51	636.84	113.39
Sep	145.36	43.56	617.93	112.8
Oct	146.11	48.7	663.59	113.36
Nov	137.21	53.88	735.39	118.16
Dec	137.96	55.4	755.35	111.73

In [69]:

```
# assign the new column names
df_e.rename(columns= {0:'IBM', 1:'MSFT', 2:'GOOGLE', 3:'APPLE'}, inplace = True)

# check
df_e
```

Out[69]:

	IBM	MSFT	GOOGLE	APPLE
name	IBM	MSFT	GOOGLE	APPLE
Jan	156.08	45.51	512.42	110.64
Feb	160.01	43.08	537.99	125.43
Mar	159.81	42.13	559.72	125.97
Apr	165.22	43.47	540.5	127.29
May	172.25	47.53	535.24	128.76
Jun	167.15	45.96	532.92	127.81
Jul	164.75	45.61	590.09	125.34
Aug	152.77	45.51	636.84	113.39
Sep	145.36	43.56	617.93	112.8
Oct	146.11	48.7	663.59	113.36
Nov	137.21	53.88	735.39	118.16
Dec	137.96	55.4	755.35	111.73

In [70]:

```
# Change the index column to a column series
df_e.reset_index(inplace=True)

# check
df_e
```

Out[70]:

	index	IBM	MSFT	GOOGLE	APPLE
0	name	IBM	MSFT	GOOGLE	APPLE
1	Jan	156.08	45.51	512.42	110.64
2	Feb	160.01	43.08	537.99	125.43
3	Mar	159.81	42.13	559.72	125.97
4	Apr	165.22	43.47	540.5	127.29
5	May	172.25	47.53	535.24	128.76
6	Jun	167.15	45.96	532.92	127.81
7	Jul	164.75	45.61	590.09	125.34
8	Aug	152.77	45.51	636.84	113.39
9	Sep	145.36	43.56	617.93	112.8
10	Oct	146.11	48.7	663.59	113.36
11	Nov	137.21	53.88	735.39	118.16
12	Dec	137.96	55.4	755.35	111.73

In [71]:

```
# rename the index column to months
df_e.rename(columns= {'index': 'Month'}, inplace = True)

# check
df_e
```

Out[71]:

	Month	IBM	MSFT	GOOGLE	APPLE
0	name	IBM	MSFT	GOOGLE	APPLE
1	Jan	156.08	45.51	512.42	110.64
2	Feb	160.01	43.08	537.99	125.43
3	Mar	159.81	42.13	559.72	125.97
4	Apr	165.22	43.47	540.5	127.29
5	May	172.25	47.53	535.24	128.76
6	Jun	167.15	45.96	532.92	127.81
7	Jul	164.75	45.61	590.09	125.34
8	Aug	152.77	45.51	636.84	113.39
9	Sep	145.36	43.56	617.93	112.8
10	Oct	146.11	48.7	663.59	113.36
11	Nov	137.21	53.88	735.39	118.16
12	Dec	137.96	55.4	755.35	111.73

In [72]:

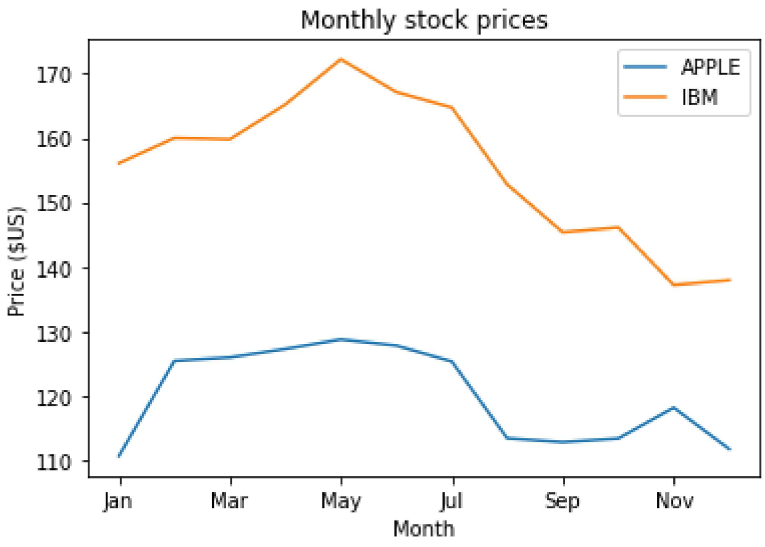
```
# delete the first row that repeats the header row
df_e.drop(0, axis=0, inplace=True)

# check
df_e
```

Out[72]:

	Month	IBM	MSFT	GOOGLE	APPLE
1	Jan	156.08	45.51	512.42	110.64
2	Feb	160.01	43.08	537.99	125.43
3	Mar	159.81	42.13	559.72	125.97
4	Apr	165.22	43.47	540.5	127.29
5	May	172.25	47.53	535.24	128.76
6	Jun	167.15	45.96	532.92	127.81
7	Jul	164.75	45.61	590.09	125.34
8	Aug	152.77	45.51	636.84	113.39
9	Sep	145.36	43.56	617.93	112.8
10	Oct	146.11	48.7	663.59	113.36
11	Nov	137.21	53.88	735.39	118.16
12	Dec	137.96	55.4	755.35	111.73

In [74]:

```
# plot stock prices data
y_columns = ['APPLE', 'IBM']

# plot months against stock prices
df_e.plot(x='Month', y=y_columns)

# give the plot a title
plt.title('Monthly stock prices')

# Add the y-axis label
plt.ylabel('Price ($US)')

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



In [ ]: