In [ ]:
```
#https://www.kaggle.com/code/colinpbowen/starter-climate-change-earth-surface-e24bc90c-4
#https://www.kdnuggets..com/2019/03/beginners-guide-linear-regression-python-scikit-learn.html
```

In [1]:
```
#Library and data importing
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [13]:
```
ls
```

```
GlobalLandTemperaturesByCountry.csv  archive.numbers*
GlobalTemperatures.csv               dsc650/
Untitled.ipynb
```

In [10]:
```
data_country = pd.read_csv('GlobalLandTemperaturesByCountry.csv')
```

In [94]:
```
data_denmark
```

Out[94]:

	dt	AverageTemperature	AverageTemperatureUncertainty	Country
142493	1823-01-01	-29.446	3.646	Denmark
142494	1823-02-01	-31.746	3.438	Denmark
142495	1823-03-01	-28.439	2.757	Denmark
142496	1823-04-01	-21.488	3.379	Denmark
142497	1823-05-01	-12.925	2.935	Denmark
...	...	...	...	...
144777	2013-05-01	-12.048	0.929	Denmark
144778	2013-06-01	-3.101	1.129	Denmark
144779	2013-07-01	-0.900	1.023	Denmark
144780	2013-08-01	-3.782	0.835	Denmark
144781	2013-09-01	NaN	NaN	Denmark

2289 rows × 4 columns

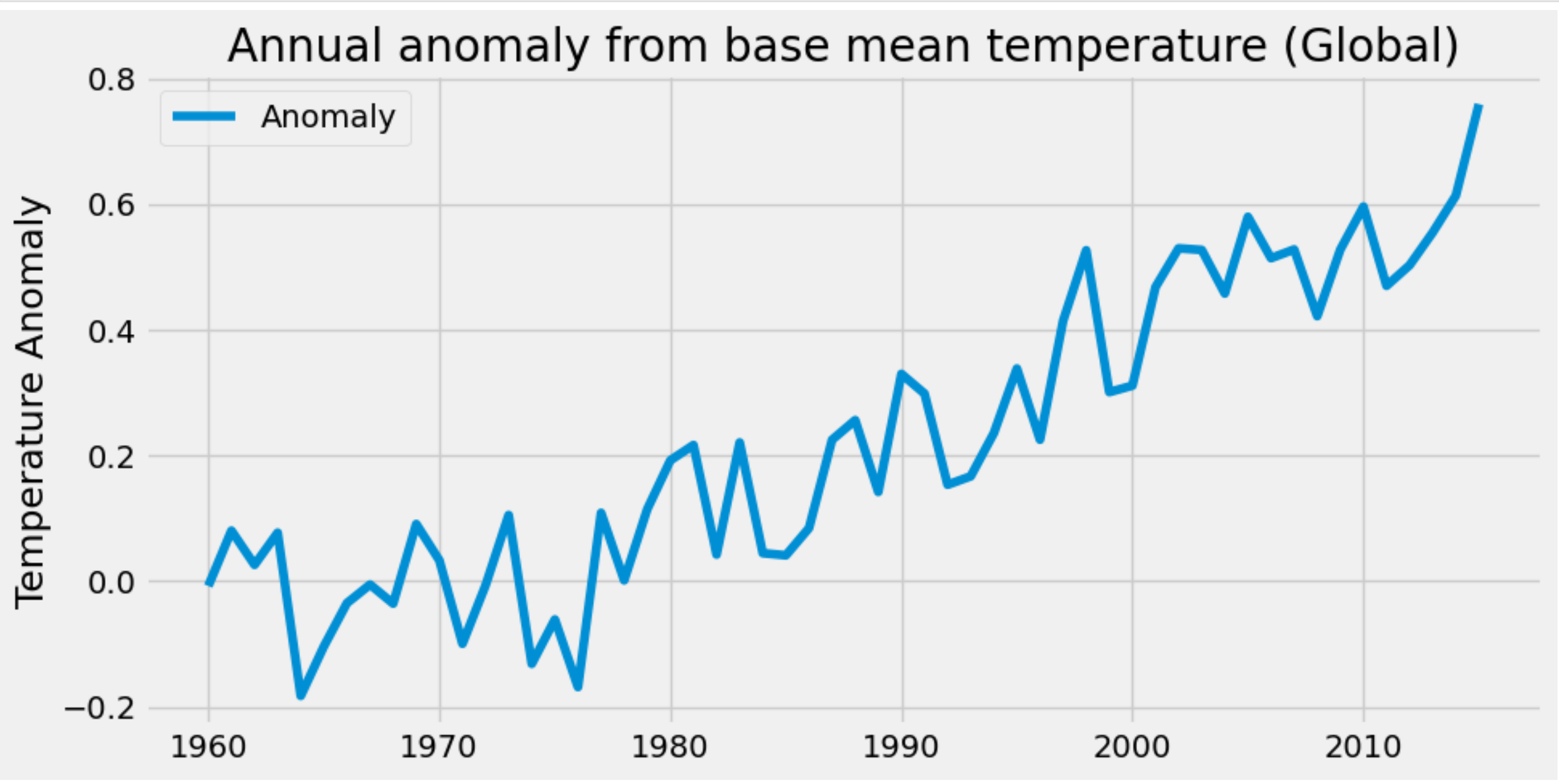
In [24]:
```
data_country = pd.read_csv("GlobalLandTemperaturesByCountry.csv")
data_denmark = data_country[data_country["Country"] == "Denmark"].copy()
data_denmark["dt"] = pd.to_datetime(data_denmark["dt"])

data_global = pd.read_csv("GlobalTemperatures.csv")
data_global["dt"] = pd.to_datetime(data_global["dt"])
co2_ppm = pd.read_csv("archive.csv")
```

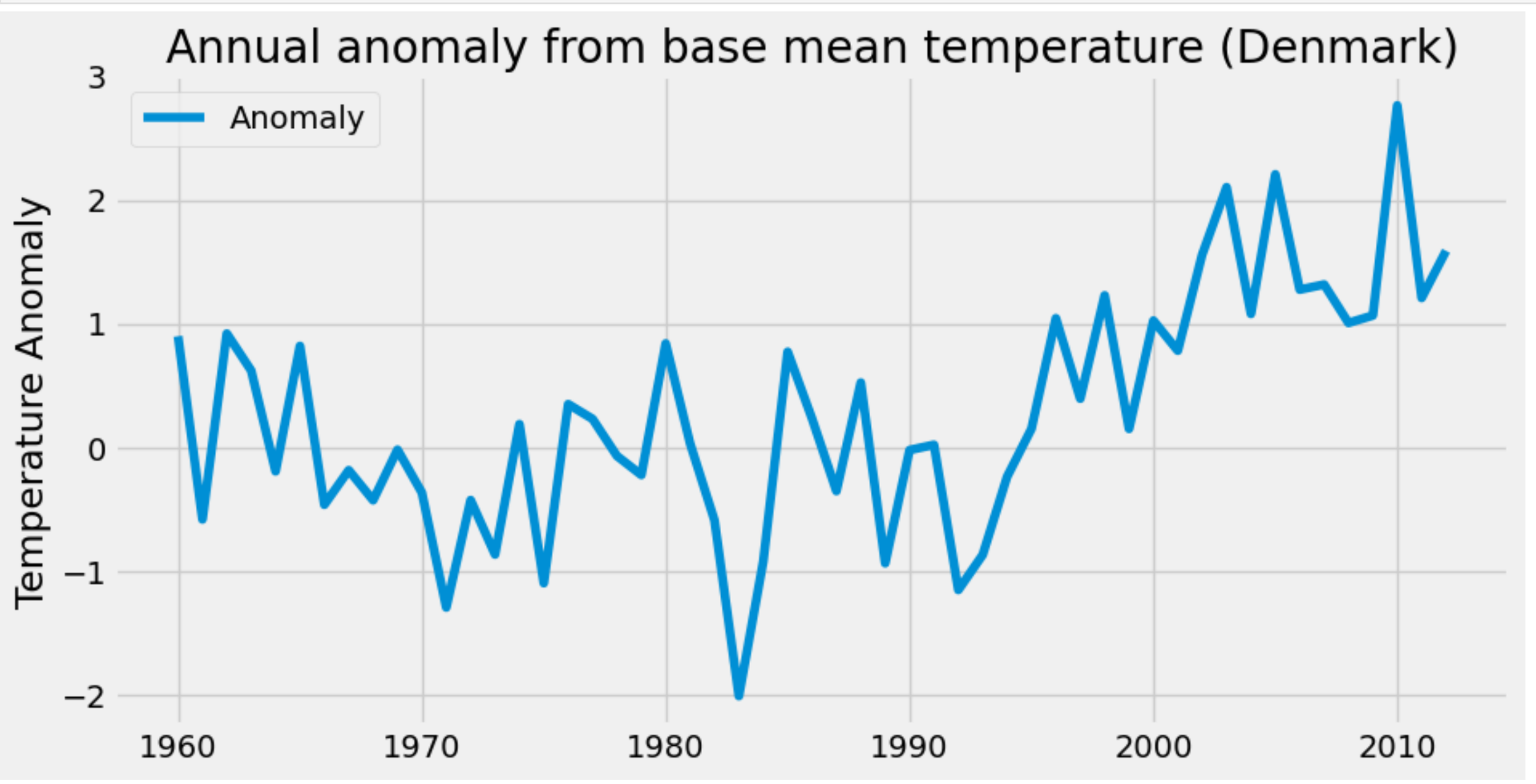
In [25]:
```
annual_mean_global = data_global.groupby(data_global["dt"].dt.year).mean()
reference_temperature_global = annual_mean_global.loc[1951:1980].mean()["LandAndOceanAverageTemperature"]
annual_mean_global["Anomaly"] = annual_mean_global["LandAndOceanAverageTemperature"] - reference_temperature_global
```

In [29]:
```
annual_mean_denmark = data_denmark.groupby(data_denmark["dt"].dt.year).mean()
reference_temperature_denmark = annual_mean_denmark.loc[1951:1980].mean()["AverageTemperature"]
annual_mean_denmark["Anomaly"] = annual_mean_denmark["AverageTemperature"] - reference_temperature_denmark
```

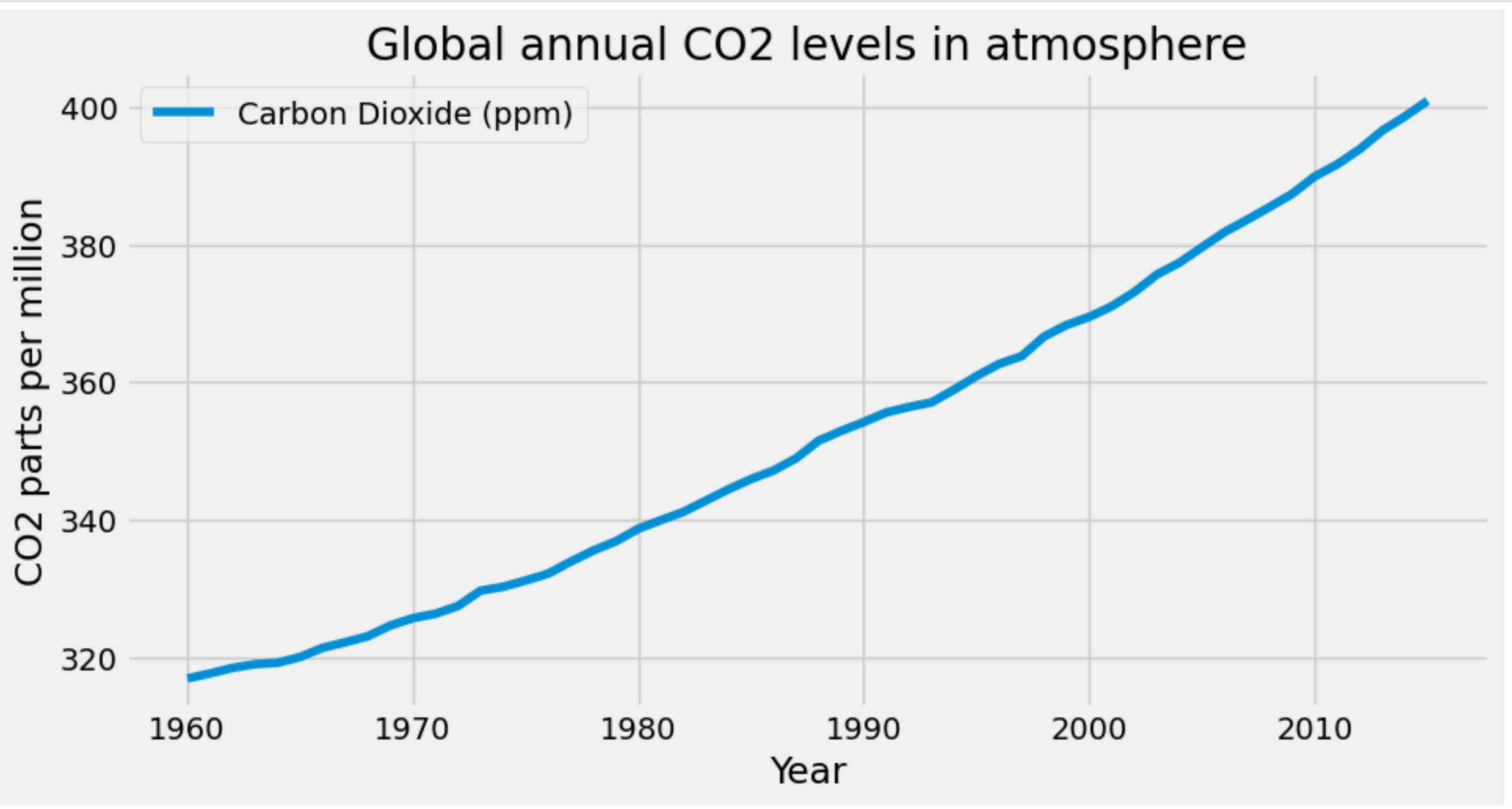
In [30]:
```
plt.figure()
plt.style.use("fivethirtyeight")
annual_mean_global.loc[1960:2015]["Anomaly"].plot(figsize = (10,5), grid=True, legend=True)
plt.title("Annual anomaly from base mean temperature (Global)")
plt.xlabel('')
plt.ylabel('Temperature Anomaly')
plt.show()
```



In [31]:
```
plt.figure()
plt.style.use("fivethirtyeight")
annual_mean_denmark.loc[1960:2012]["Anomaly"].plot(figsize = (10,5), grid=True, legend=True)
plt.title("Annual anomaly from base mean temperature (Denmark)")
plt.xlabel('')
plt.ylabel('Temperature Anomaly')
plt.show()
```



In [32]:
```
plt.figure()
plt.style.use("fivethirtyeight")
annual_co2_ppm = co2_ppm.groupby(co2_ppm["Year"]).mean()
annual_co2_ppm.loc[1960:2015]["Carbon Dioxide (ppm)"].plot(figsize = (10,5), grid=True, legend=True)
plt.title("Global annual CO2 levels in atmosphere")
plt.ylabel("CO2 parts per million")
plt.show()
```



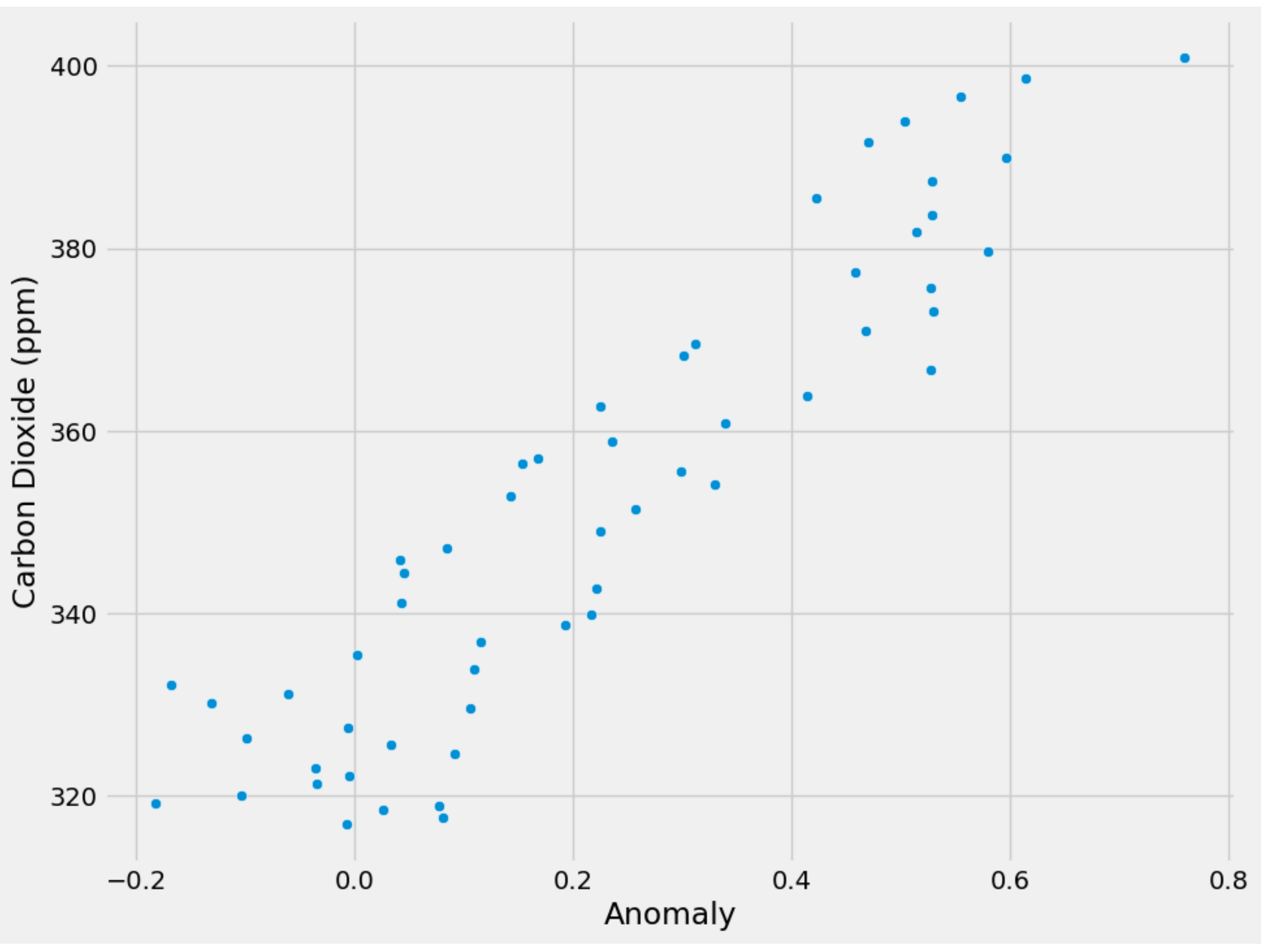
In [33]:
```
annual_co2_temp = pd.merge(annual_mean_global.loc[1960:2015], annual_co2_ppm.loc[1960:2015], left_index=True, right_index=True)
annual_co2_temp = annual_co2_temp[["LandAndOceanAverageTemperature", "Anomaly", "Carbon Dioxide (ppm)"]].copy()
annual_co2_temp.corr()
```

Out[33]:

	LandAndOceanAverageTemperature	Anomaly	Carbon Dioxide (ppm)
LandAndOceanAverageTemperature	1.000000	1.000000	0.923603
Anomaly	1.000000	1.000000	0.923603
Carbon Dioxide (ppm)	0.923603	0.923603	1.000000

In [34]:
```
plt.figure(figsize=(10,8))
sns.scatterplot(x="Anomaly",y="Carbon Dioxide (ppm)", data=annual_co2_temp)
```

Out[34]:
```
<AxesSubplot:xlabel='Anomaly', ylabel='Carbon Dioxide (ppm)'>
```



In [71]:
```
y_data = annual_co2_temp["Carbon Dioxide (ppm)"]
x_data = annual_co2_temp["Anomaly"]
y_data
x_data
```

Out[71]:
```
dt
1960   -0.007631
1961    0.080369
1962    0.026036
1963    0.077119
1964   -0.182464
1965   -0.103797
1966   -0.034881
1967   -0.005381
1968   -0.035631
1969    0.090953
1970    0.033119
1971   -0.099547
1972   -0.006631
1973    0.105203
1974   -0.131464
1975   -0.060881
1976   -0.168464
1977    0.108703
1978    0.001369
1979    0.114619
1980    0.192286
1981    0.216619
1982    0.042369
1983    0.220703
1984    0.044619
1985    0.041119
1986    0.084453
1987    0.224953
1988    0.256203
1989    0.142036
1990    0.329786
1991    0.298453
1992    0.153453
1993    0.166869
1994    0.235453
1995    0.338286
1996    0.225119
1997    0.414286
1998    0.526453
1999    0.300786
2000    0.311119
2001    0.467953
2002    0.529619
2003    0.527036
2004    0.457703
2005    0.579703
2006    0.513953
2007    0.527786
2008    0.421703
2009    0.527619
2010    0.595953
2011    0.469953
2012    0.502786
2013    0.554869
2014    0.613453
2015    0.759036
Name: Anomaly, dtype: float64
```

In [72]:
```
import numpy as np
import pandas as pd
new_y_data = y_data.to_numpy()
new_x_data = x_data.to_numpy()
```

In [80]:
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
%matplotlib inline
```

In [73]:
```
from sklearn.model_selection import train_test_split
x_training_data, x_test_data, y_training_data, y_test_data = train_test_split(new_x_data, new_y_data, test_size = 0.3)
```

In [74]:
```
#from sklearn.linear_model import LogisticRegression
#model = LogisticRegression()
```

In [81]:
```
regressor = LinearRegression()
regressor.fit(x_training_data, y_training_data)
```

Out[81]:
```
LinearRegression()
```

In [82]:
```
#To retrieve the intercept:
print(regressor.intercept_)
```

```
[331.13881958]
```

In [83]:
```
#To retrieve the slope:
print(regressor.coef_)
```

```
[[94.63385194]]
```

In [87]:
```
x_test_data = x_test_data.reshape(-1, 1)
x_test_data
```

Out[87]:
```
array([[ 0.22511944],
       [ 0.52703611],
       [ 0.02603611],
       [-0.00663056],
       [ 0.14203611],
       [ 0.75903611],
       [ 0.55486944],
       [-0.03488056],
       [ 0.31111944],
       [ 0.10520278],
       [ 0.52778611],
       [ 0.21661944],
       [ 0.52645278],
       [ 0.09095278],
       [ 0.50278611],
       [ 0.30078611],
       [-0.18246389]])
```

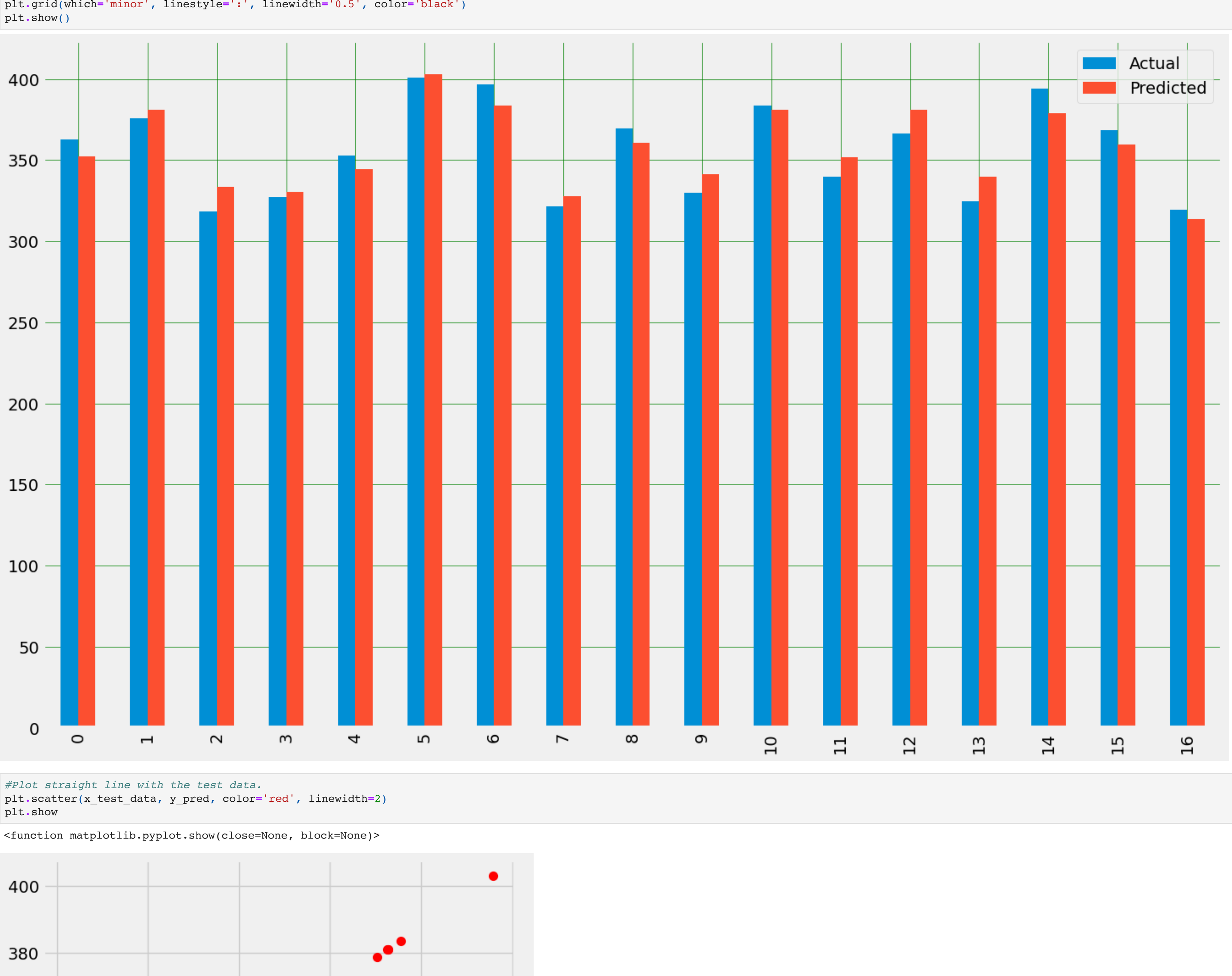
In [88]:
```
y_pred = regressor.predict(x_test_data)
```

In [90]:
```
df = pd.DataFrame({'Actual': y_test_data.flatten(), 'Predicted': y_pred.flatten()})
df
```

Out[90]:

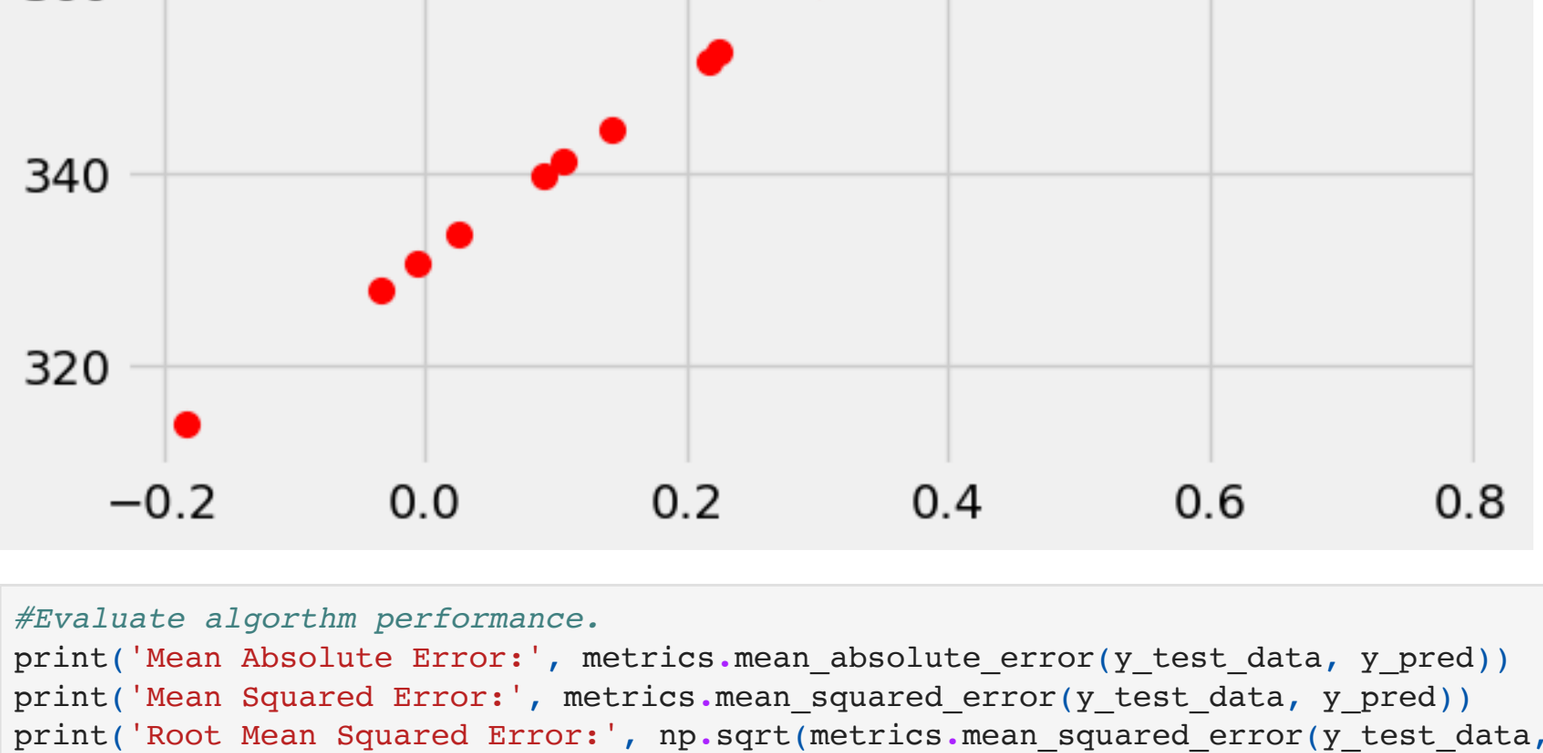
	Actual	Predicted
0	362.640833	352.442740
1	375.636667	381.014277
2	318.453333	333.602717
3	327.457500	330.511345
4	352.907500	344.580244
5	400.885000	402.969331
6	396.566667	383.648252
7	321.370000	327.837938
8	369.475000	360.581251
9	329.677500	341.094564
10	383.592500	381.085252
11	339.934167	351.638352
12	366.630000	380.959074
13	324.621667	339.746031
14	393.870833	378.719406
15	368.313333	359.603368
16	319.202222	313.871559

In [91]:
```
df1 =df.head(25)
df1.plot(kind='bar', figsize=(16, 10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```



In [92]:
```
#Plot straight line with the test data.
plt.scatter(x_test_data, y_pred, color='red', linewidth=2)
plt.show
```

Out[92]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [93]:
```
#Evaluate algorthm performance.
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test_data, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test_data, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test_data, y_pred)))
```

```
Mean Absolute Error: 9.220271263297734
Mean Squared Error: 104.39960113130051
Root Mean Squared Error: 10.217612300890092
```