In [ ]:
```
import pandas as pd
```

In [ ]:
```
%matplotlib inline
```

In [ ]:
```
# importing the matplotlib library as plt for data visualization
import matplotlib.pyplot as plt
```

In [ ]:
```
# reading the csv file into a pandas dataframe
nyc=pd.read_csv('ave_hi_nyc_jan_1895-2018.csv')
```

In [ ]:
```
# displaying the first five rows of the dataframe
nyc.head()
```

Out[ ]:

	Date	Value	Anomaly
0	189501	34.2	-3.2
1	189601	34.7	-2.7
2	189701	35.5	-1.9
3	189801	39.6	2.2
4	189901	36.4	-1.0

In [ ]:
```
# displaying the last five rows of the dataframe
nyc.tail()
```

Out[ ]:

	Date	Value	Anomaly
119	201401	35.5	-1.9
120	201501	36.1	-1.3
121	201601	40.8	3.4
122	201701	42.8	5.4
123	201801	38.7	1.3

In [ ]:
```
# displaying random 20 rows of the dataframe
nyc.sample(20)
```

Out[ ]:

	Date	Value	Anomaly
90	198501	33.0	-4.4
104	199901	40.8	3.4
1	189601	34.7	-2.7
103	199801	45.8	8.4
82	197701	26.1	-11.3
51	194601	40.2	2.8
40	193501	35.0	-2.4
74	196901	35.7	-1.7
107	200201	45.4	8.0
91	198601	39.3	1.9
19	191401	36.9	-0.5
78	197301	40.6	3.2
77	197201	40.5	3.1
100	199501	42.6	5.2
113	200801	42.4	5.0
92	198701	35.9	-1.5
48	194301	36.1	-1.3
35	193001	38.5	1.1
41	193601	34.3	-3.1
83	197801	32.3	-5.1

In [ ]:
```
# declaring the column names of the dataframe
nyc.columns =['Date','Temperature','Anomaly']
```

In [ ]:
```
# displaying the first five rows of the dataframe
nyc.head()
```

Out[ ]:

	Date	Temperature	Anomaly
0	189501	34.2	-3.2
1	189601	34.7	-2.7
2	189701	35.5	-1.9
3	189801	39.6	2.2
4	189901	36.4	-1.0

In [ ]:
```
# Takes the values of the Date column in the nyc dataframe and divides th
nyc.Date=nyc.Date.floordiv(100)
```

In [ ]:
```
# displaying the first five rows of the dataframe
nyc.head()
```

Out[ ]:

	Date	Temperature	Anomaly
0	1895	34.2	-3.2
1	1896	34.7	-2.7
2	1897	35.5	-1.9
3	1898	39.6	2.2
4	1899	36.4	-1.0

In [ ]:
```
# gets the dimensions of the nyc dataframe
nyc.shape
```

Out[ ]: (124, 3)

In [ ]:
```
# displays the last five rows of the nyc dataframe
nyc.tail()
```

Out[ ]:

	Date	Temperature	Anomaly
119	2014	35.5	-1.9
120	2015	36.1	-1.3
121	2016	40.8	3.4
122	2017	42.8	5.4
123	2018	38.7	1.3

In [ ]:
```
# importing the train_test_split function from the sklearn.model_selectio
from sklearn.model_selection import train_test_split
```

In [ ]:
```
nyc.Date.values.shape
```

Out[ ]: (124,)

In [ ]:
```
# splitting the nyc dataframe into training and testing sets. The trainin
X_train,X_test,y_train,y_test=train_test_split(nyc.Date.values.reshape(-1
```

In [ ]:
```
# This code accesses the shape of the X_train variable(dimension of the t
X_train.shape
```

Out[ ]: (93, 1)

In [ ]:
```
# This code accesses the shape of the X_test variable(dimension of the te
X_test.shape
```

Out[ ]: (31, 1)

In [ ]:
```
93+31
```

Out[ ]: 124

In [ ]:
```
93/124*100
```

Out[ ]: 75.0

In [ ]:
```
#Training the model
```

In [ ]:
```
# importing the LinearRegression class from the sklearn.linear_model libr
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
# creating an instance of the LinearRegression class
linear_regression=LinearRegression()
```

In [ ]:
```
# calling the linear_regression object instance.
linear_regression
```

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

In [ ]:
```
# training the model using the fit method of the linear_regression object
linear_regression.fit(X=X_train, y=y_train)
```

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

In [ ]:
```
#Equation M and C
```

In [ ]:
```
# This code accesses the slope of the linear regression model(coefficient
linear_regression.coef_
```

Out[ ]:
```
array([0.01939167])
```

In [ ]:
```
# intercept of the linear regression equation
linear_regression.intercept_
```

Out[ ]:
```
-0.30779820252656265
```

The equation: $Temperature = 0.01939167 * Date - 0.30779820252656265$

In [ ]:
```
# Testing the model
(0.01939167 * 2014) - 0.30779820252656265
```

Out[ ]:
```
38.74702517747344
```

In [ ]:
```
# Testing the Model
```

In [ ]:
```
# This code accesses the predicted values of the linear regression model
predicted = linear_regression.predict(X_test)
```

In [ ]:
```
# expected values of the linear regression model on the y_test set
expected = y_test
```

In [ ]:
```
# This code snippet is iterating over two lists, predicted and expected,
for p,e in zip(predicted[::], expected[::]):
  print(f'Predicted: {p:.2f}, Expected: {e:.2f}')
```

```
Predicted: 37.86, Expected: 31.70
Predicted: 36.48, Expected: 35.50
Predicted: 37.93, Expected: 40.50
Predicted: 36.61, Expected: 29.80
Predicted: 36.75, Expected: 40.70
Predicted: 38.69, Expected: 34.80
Predicted: 36.44, Expected: 34.20
Predicted: 37.14, Expected: 38.20
Predicted: 37.62, Expected: 36.20
Predicted: 37.53, Expected: 42.50
Predicted: 37.00, Expected: 39.40
Predicted: 38.32, Expected: 40.90
Predicted: 37.20, Expected: 39.80
Predicted: 38.46, Expected: 40.80
Predicted: 36.56, Expected: 37.00
Predicted: 37.25, Expected: 45.70
Predicted: 38.18, Expected: 33.00
Predicted: 37.89, Expected: 29.90
Predicted: 38.15, Expected: 38.00
Predicted: 38.63, Expected: 42.40
Predicted: 38.05, Expected: 32.30
Predicted: 37.02, Expected: 33.80
Predicted: 37.12, Expected: 38.50
Predicted: 37.70, Expected: 37.80
Predicted: 36.73, Expected: 36.10
Predicted: 37.64, Expected: 33.80
Predicted: 37.56, Expected: 42.40
Predicted: 38.11, Expected: 30.60
Predicted: 36.87, Expected: 38.40
Predicted: 36.85, Expected: 42.30
Predicted: 36.94, Expected: 39.70
```

In [ ]:
```python
# This code snippet is a for loop that iterates over two lists, predicted
for p,e in zip(predicted[::], expected[::]):
  print(f'Predicted: {p:.2f}, Expected: {e:.2f}, Error: {e-p:.2f}')
```

```
Predicted: 37.86, Expected: 31.70, Error: -6.16
Predicted: 36.48, Expected: 35.50, Error: -0.98
Predicted: 37.93, Expected: 40.50, Error: 2.57
Predicted: 36.61, Expected: 29.80, Error: -6.81
Predicted: 36.75, Expected: 40.70, Error: 3.95
Predicted: 38.69, Expected: 34.80, Error: -3.89
Predicted: 36.44, Expected: 34.20, Error: -2.24
Predicted: 37.14, Expected: 38.20, Error: 1.06
Predicted: 37.62, Expected: 36.20, Error: -1.42
Predicted: 37.53, Expected: 42.50, Error: 4.97
Predicted: 37.00, Expected: 39.40, Error: 2.40
Predicted: 38.32, Expected: 40.90, Error: 2.58
Predicted: 37.20, Expected: 39.80, Error: 2.60
Predicted: 38.46, Expected: 40.80, Error: 2.34
Predicted: 36.56, Expected: 37.00, Error: 0.44
Predicted: 37.25, Expected: 45.70, Error: 8.45
Predicted: 38.18, Expected: 33.00, Error: -5.18
Predicted: 37.89, Expected: 29.90, Error: -7.99
Predicted: 38.15, Expected: 38.00, Error: -0.15
Predicted: 38.63, Expected: 42.40, Error: 3.77
Predicted: 38.05, Expected: 32.30, Error: -5.75
Predicted: 37.02, Expected: 33.80, Error: -3.22
Predicted: 37.12, Expected: 38.50, Error: 1.38
Predicted: 37.70, Expected: 37.80, Error: 0.10
Predicted: 36.73, Expected: 36.10, Error: -0.63
Predicted: 37.64, Expected: 33.80, Error: -3.84
Predicted: 37.56, Expected: 42.40, Error: 4.84
Predicted: 38.11, Expected: 30.60, Error: -7.51
Predicted: 36.87, Expected: 38.40, Error: 1.53
Predicted: 36.85, Expected: 42.30, Error: 5.45
Predicted: 36.94, Expected: 39.70, Error: 2.76
```

In [ ]:
```
# Mean Absolute Error (MAE)
```

In [ ]:
```
# importing the mean_absolute_error function from the sklearn.metrics lib
from sklearn.metrics import mean_absolute_error
```

In [ ]:
```
# printing the mean absolute error of the linear regression model
print("MAE", mean_absolute_error(expected,predicted))
```

```
MAE 3.450753706948741
```

In [ ]:
```
#RMSE - Root Mean Squared Error
```

In [ ]:
```
# importing the mean_squared_error function from the sklearn.metrics libr
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
# importing numpy as np for mathematical operations on arrays
import numpy as np
```

In [ ]:
```
# printing the root mean squared error of the linear regression model
print(f'RMSE',np.log(np.sqrt(mean_squared_error(expected, predicted)) ))
```

```
RMSE 1.4256936366057185
```

In [ ]:
```
# R Squared (R2)
```

In [ ]:
```
# importing the r2_score function from the sklearn.metrics library. Used
from sklearn.metrics import r2_score
```

In [ ]:
```
# r2 variable stores the r2 score function of the linear regression model
r2=r2_score(expected,predicted)
```

In [ ]:
```
r2
```

Out[ ]:
```
-0.033370346388810423
```

In [ ]:
```
# predict future model
```

In [ ]:
```
# The code you provided defines a lambda function named predict.
# The lambda function takes a single argument, x, and returns the result


predict=(lambda x: linear_regression.coef_ *x + linear_regression.interce
```

In [ ]:
```
# This code snippet calls the predict function and passes in the value 20
predict(2014)
```

Out[ ]:
```
array([38.74703181])
```

In [ ]:
```
# This code snippet calls the predict function and passes in the value 20
predict(2022)
```

Out[ ]:
```
array([38.9021652])
```

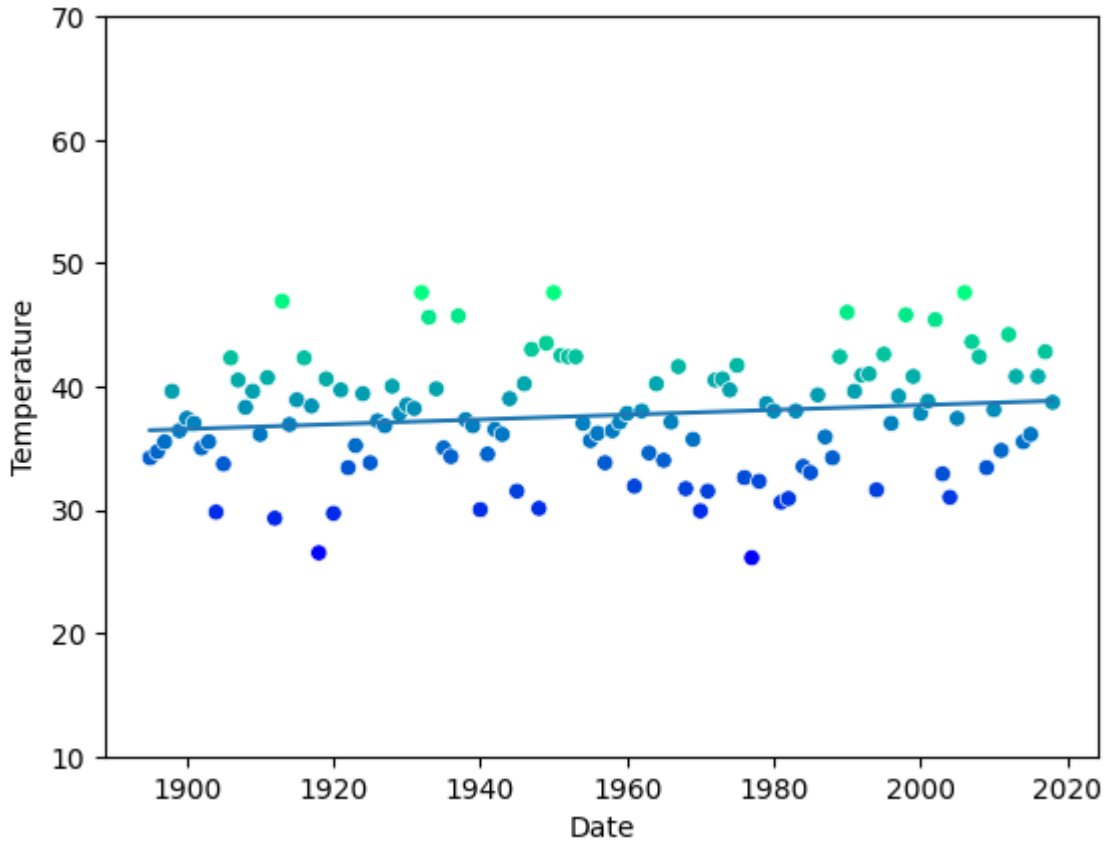
In [ ]:
```
# This code snippet calls the predict function and passes in the value 18
predict(1800)
```

Out[ ]:
```
array([34.59721373])
```

In [ ]:
```
# Visualizing the dataset with a Regression Line
```

In [ ]:
```
# importing the seaborn library as sns for data visualization
import seaborn as sns
```

In [ ]:
```
# This code snippet creates a scatter plot of the nyc dataframe's Date an
axes=sns.scatterplot(data=nyc, x='Date',y='Temperature',
                     hue='Temperature', palette='winter', legend=False)
axes.set_ylim(10,70)
x=np.array([min(nyc.Date.values),max(nyc.Date.values)])
y=predict(x)
line=plt.plot(x,y)
```



In [ ]:
```
# The x variable stores the minimum and maximum values of the nyc datafra
x
```

Out[ ]:
```
array([1895, 2018])
```

In [ ]:
```
# The y variable stores the predicted values of the linear regression mod
y
```

Out[ ]:
```
array([36.43942269, 38.82459851])
```

In [ ]: