In [1]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

In [2]:
```
# Load the data
data = pd.read_csv('temperatures.csv')
```

In [3]:
```
data.isna().sum()
```

Out[3]:
```
YEAR       0
JAN        0
FEB        0
MAR        0
APR        0
MAY        0
JUN        0
JUL        0
AUG        0
SEP        0
OCT        0
NOV        0
DEC        0
ANNUAL     0
JAN-FEB    0
MAR-MAY    0
JUN-SEP    0
OCT-DEC    0
dtype: int64
```

In [4]:
```
data.head()
```

Out[4]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNU
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.49	2
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.04	2
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.65	2
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.63	2
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.82	2

In [5]:
```
data.shape
```

Out[5]:
```
(117, 18)
```

In [6]:
```
data.info()
```

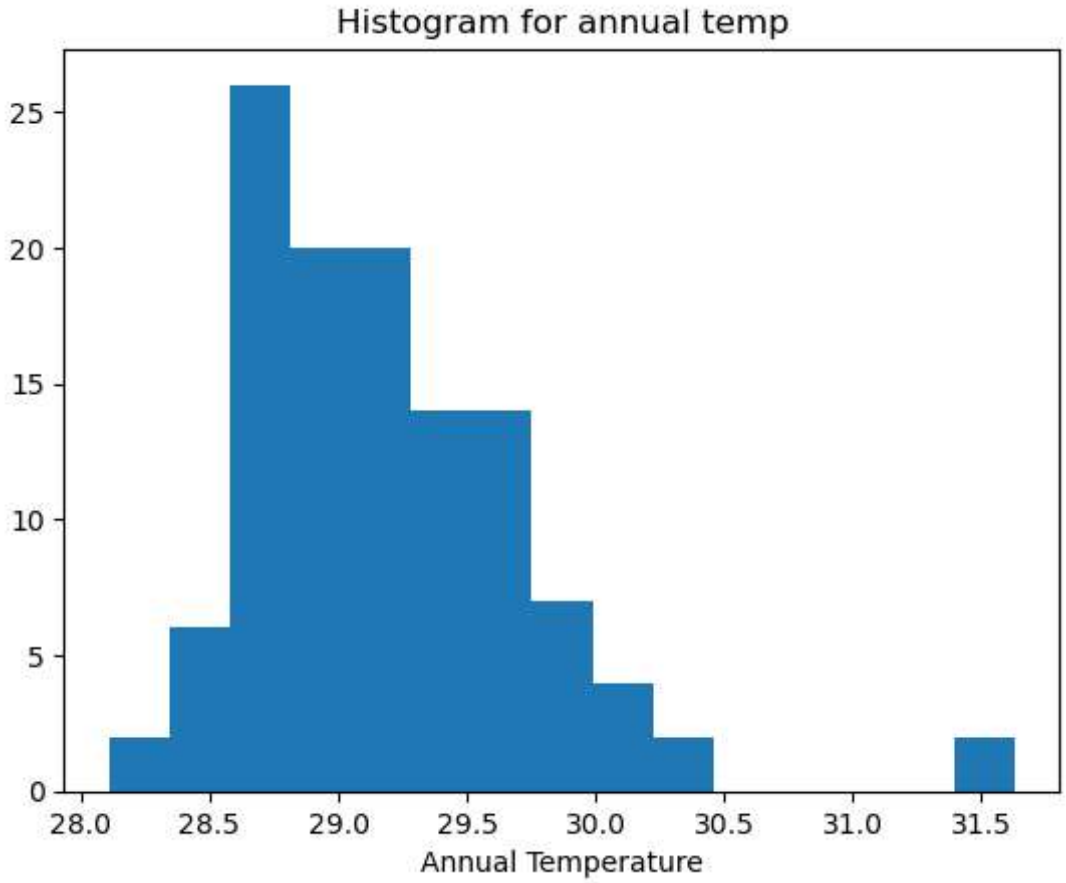
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117 entries, 0 to 116
Data columns (total 18 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   YEAR     117 non-null    int64  
 1   JAN      117 non-null    float64
 2   FEB      117 non-null    float64
 3   MAR      117 non-null    float64
 4   APR      117 non-null    float64
 5   MAY      117 non-null    float64
 6   JUN      117 non-null    float64
 7   JUL      117 non-null    float64
 8   AUG      117 non-null    float64
 9   SEP      117 non-null    float64
 10  OCT      117 non-null    float64
 11  NOV      117 non-null    float64
 12  DEC      117 non-null    float64
 13  ANNUAL   117 non-null    float64
 14  JAN-FEB  117 non-null    float64
 15  MAR-MAY  117 non-null    float64
 16  JUN-SEP  117 non-null    float64
 17  OCT-DEC  117 non-null    float64
dtypes: float64(17), int64(1)
memory usage: 16.6 KB
```

In [7]:
```
data.describe()
```

Out[7]:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	
count	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.
mean	1959.000000	23.687436	25.597863	29.085983	31.975812	33.565299	32.774274	31.
std	33.919021	0.834588	1.150757	1.068451	0.889478	0.724905	0.633132	0.
min	1901.000000	22.000000	22.830000	26.680000	30.010000	31.930000	31.100000	29.
25%	1930.000000	23.100000	24.780000	28.370000	31.460000	33.110000	32.340000	30.
50%	1959.000000	23.680000	25.480000	29.040000	31.950000	33.510000	32.730000	31.
75%	1988.000000	24.180000	26.310000	29.610000	32.420000	34.030000	33.180000	31.
max	2017.000000	26.940000	29.720000	32.620000	35.380000	35.840000	34.480000	32.

In [7]:
```
data.shape
plt.hist(data['ANNUAL'],bins = 15)
plt.xlabel('Annual Temperature')
plt.title('Histogram for annual temp')
plt.show()
```



In [8]:
```
X = data['YEAR'].values[:,None]
y = data['JAN']
```

In [19]:
```
X
```

```
       [1991],
       [1992],
       [1993],
       [1994],
       [1995],
       [1996],
       [1997],
       [1998],
       [1999],
       [2000],
       [2001],
       [2002],
       [2003],
       [2004],
       [2005],
       [2006],
       [2007],
       [2008],
       [2009],
       [2010],
```

In [11]:
```
y
```

Out[11]:
```
0      22.40
1      24.93
2      23.44
3      22.50
4      22.00
       ...
112    24.56
113    23.83
114    24.58
115    26.94
116    26.45
Name: JAN, Length: 117, dtype: float64
```

In [10]:
```
#Split the data into training and test datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,
random_state = 0)
```

In [11]:
```
y_test
```

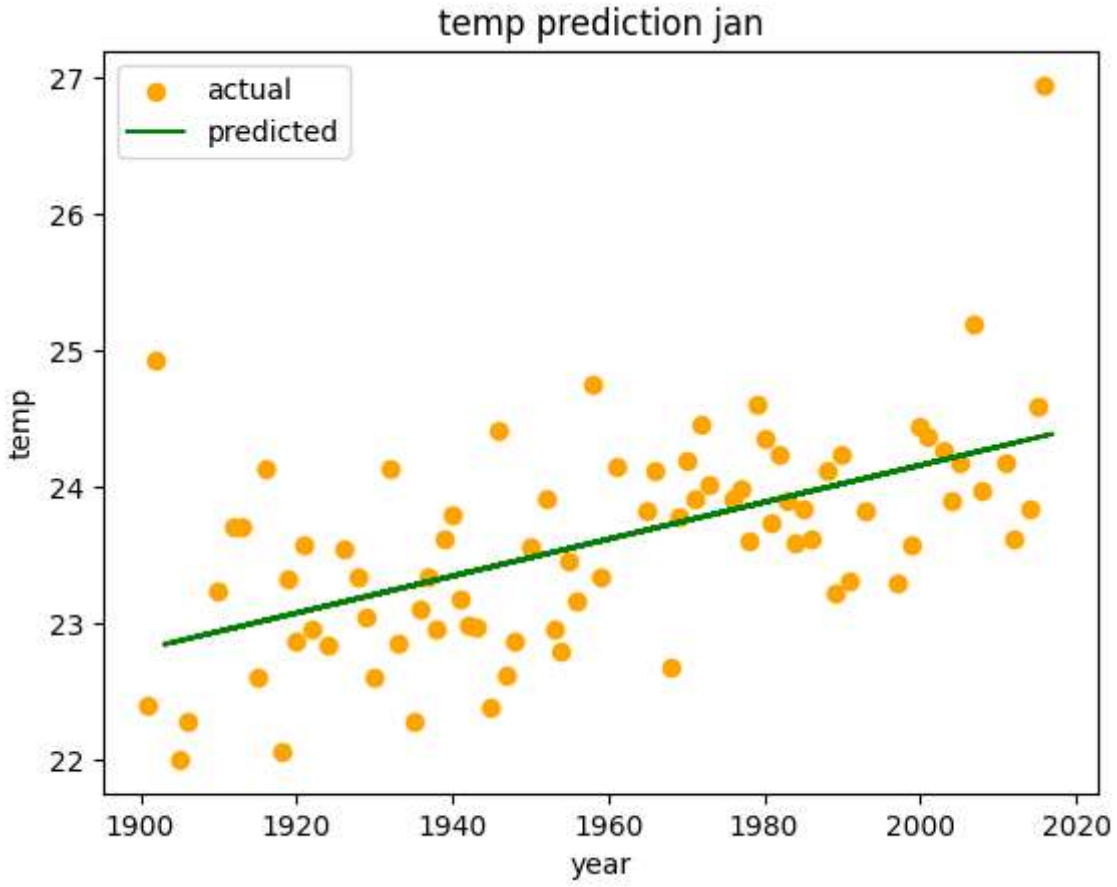
Out[11]:
```
10     23.22
59     23.78
95     25.18
91     23.84
7      23.57
86     23.81
26     23.23
112    24.56
22     23.25
94     24.44
101    24.56
2      23.44
50     24.36
24     22.56
116    26.45
62     22.90
93     24.67
74     23.15
61     22.89
73     23.54
16     23.68
108    25.27
13     24.42
43     23.17
105    25.66
33     22.76
30     24.57
56     22.98
48     24.31
8      22.67
97     23.95
66     23.72
109    24.89
3      22.50
63     23.06
6      24.46
Name: JAN, dtype: float64
```

In [12]:
```
# from sklearn.linear_model import train_test_split
model1 = LinearRegression()
model1.fit(X_train,y_train)
```

Out[12]:
```
▾ LinearRegression

LinearRegression()
```

In [13]:
```
y_test_predict = model1.predict(X_test)
y_train_predict = model1.predict(X_train)
```

In [16]:
```
plt.scatter(X_train,y_train,color = 'orange',label='actual')
plt.plot(X_test,y_test_predict,color='green',label='predicted')
plt.xlabel('year')
plt.ylabel('temp')
plt.title('temp prediction jan')
plt.legend()
plt.show()
```



In [14]:
```
y_predict = model1.predict(X_test)
y_predict
y_test
```

Out[14]:
```
10     23.22
59     23.78
95     25.18
91     23.84
7      23.57
86     23.81
26     23.23
112    24.56
22     23.25
94     24.44
101    24.56
2      23.44
50     24.36
24     22.56
116    26.45
62     22.90
93     24.67
74     23.15
61     22.89
73     23.54
16     23.68
108    25.27
13     24.42
43     23.17
105    25.66
33     22.76
30     24.57
56     22.98
48     24.31
8      22.67
97     23.95
66     23.72
109    24.89
3      22.50
63     23.06
6      24.46
Name: JAN, dtype: float64
```

In [15]:
```
from sklearn import metrics

r_square = metrics.r2_score(y_test, y_test_predict)
print('R-Square Error:', r_square)
```

```
R-Square Error: 0.27921723515312413
```

In [16]:
```
from sklearn import metrics

mse = metrics.mean_squared_error(y_test, y_test_predict)
print('Mean Squared Error:', mse)

rmse = np.sqrt(mse)
print('Root Mean Squared Error:', rmse)

mae = metrics.mean_absolute_error(y_test, y_test_predict)
print('Mean Absolute Error:', mae)
```

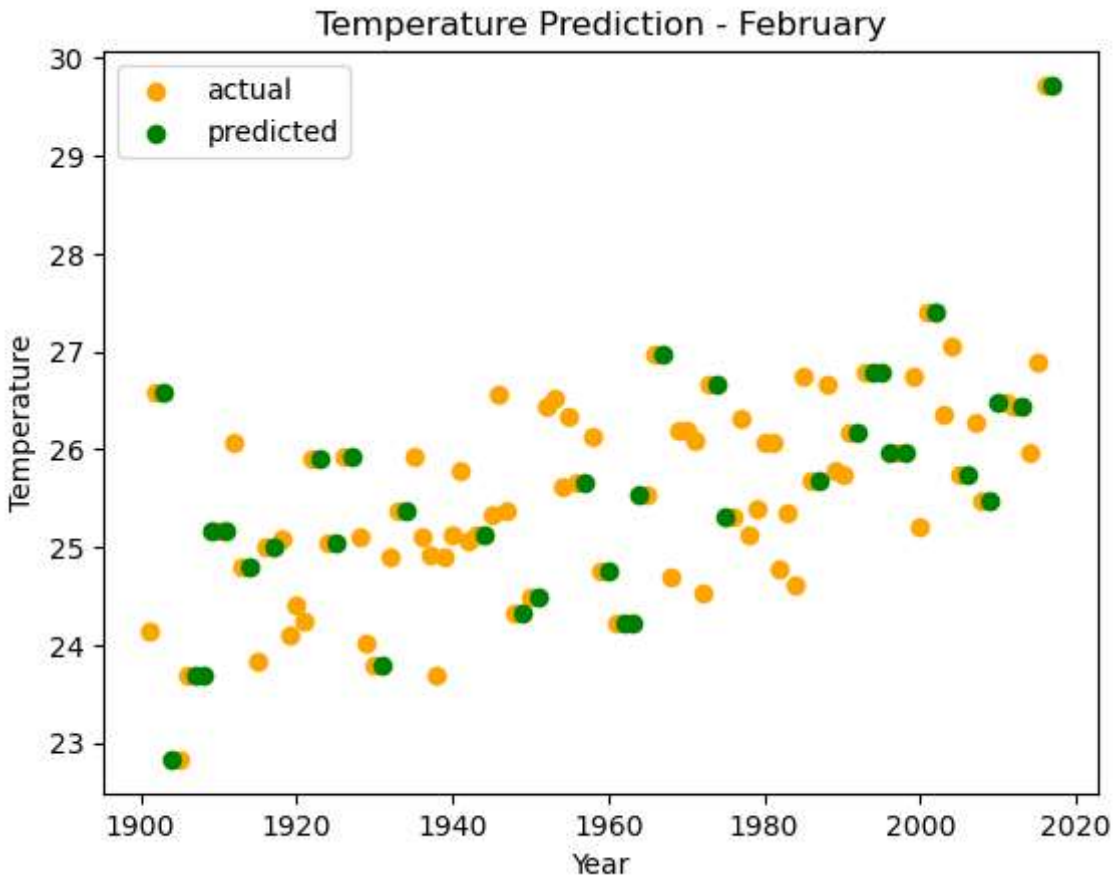
```
Mean Squared Error: 0.6080338203121165
Root Mean Squared Error: 0.7797652341006981
Mean Absolute Error: 0.6231302838065337
```

In [ ]:

In [24]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('temperatures.csv')

# Define a dictionary to map months to column names
month_columns = {
    'JAN': 'January',
    'FEB': 'February',
    'MAR': 'March',
    'APR': 'April',
    'MAY': 'May',
    'JUN': 'June',
    'JUL': 'July',
    'AUG': 'August',
    'SEP': 'September',
    'OCT': 'October',
    'NOV': 'November',
    'DEC': 'December'
}

# Define a dictionary to map months to their corresponding number
month_number_mapping = {
    1: 'JAN',
    2: 'FEB',
    3: 'MAR',
    4: 'APR',
    5: 'MAY',
    6: 'JUN',
    7: 'JUL',
    8: 'AUG',
    9: 'SEP',
    10: 'OCT',
    11: 'NOV',
    12: 'DEC'
}

# Get user input for month number
while True:
    try:
        month_number = int(input("Enter a month number (1-12): "))
        if month_number < 1 or month_number > 12:
            print("Please enter a valid month number between 1 and 12.")
        else:
            break
    except ValueError:
        print("Invalid input. Please enter a valid number.")

# Retrieve the corresponding month code and name using the dictionaries
month_code = month_number_mapping[month_number]
month_name = month_columns[month_code]
```

```
X = data['YEAR'].values.reshape(-1, 1)
y = data[month_code]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

model = DecisionTreeRegressor()
model.fit(X_train, y_train)

y_test_predict = model.predict(X_test)
y_train_predict = model.predict(X_train)

plt.scatter(X_train, y_train, color='orange', label='actual')
plt.scatter(X_test, y_test_predict, color='green', label='predicted')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title(f'Temperature Prediction - {month_name}')
plt.legend()
plt.show()

# Calculate and print evaluation metrics
r_square = metrics.r2_score(y_test, y_test_predict)
mse = metrics.mean_squared_error(y_test, y_test_predict)
rmse = np.sqrt(mse)
mae = metrics.mean_absolute_error(y_test, y_test_predict)

print(f'{month_name} - R-Square: {r_square:.2f}, MSE: {mse:.2f}, RMSE: {rmse:.
```

```
Enter a month number (1-12): 2
```



```
February - R-Square: 0.08, MSE: 1.66, RMSE: 1.29, MAE: 1.00
```

In [25]:
```python
from sklearn.tree import DecisionTreeRegressor

# Create a Decision Tree Regressor model
model = DecisionTreeRegressor()

# Fit the model on the training data
model.fit(X_train, y_train)
```

Out[25]:
```
▾ DecisionTreeRegressor

DecisionTreeRegressor()
```

In [26]:
```python
predictions = model.predict(X_test)
```

In [28]:
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error

# ... (your code to train and predict using DecisionTreeRegressor) ...

# Calculate evaluation metrics
mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, predictions)

print(f'MSE: {mse:.2f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}')
```

```
MSE: 1.66, RMSE: 1.29, MAE: 1.00
```

In [ ]:
```python

```