1. Import Library

In [51]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

2.Import Datasets

In [52]:
```
data=pd.read_csv('jena_climate_2009_2016.csv')
data.head()
```

Out[52]:

	Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC)	rh (%)	VPmax (mbar)	VPact (mbar)	VPdef (mbar)	sh (g/kg)	H2OC (mmol/mol)	rho (g/m**3)	wv (m/s)	max. wv (m/s)	wd (deg)
0	01.01.2009 00:10:00	996.52	-8.02	265.40	-8.90	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
1	01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.80	0.72	1.50	136.1
2	01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.20	1.88	3.02	1310.24	0.19	0.63	171.6
3	01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.50	198.0
4	01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309.00	0.32	0.63	214.3

3.Data Undestanding

In [53]:
```
data=data[['Date Time','T (degC)']]
data
```

Out[53]:

	Date Time	T (degC)
0	01.01.2009 00:10:00	-8.02
1	01.01.2009 00:20:00	-8.41
2	01.01.2009 00:30:00	-8.51
3	01.01.2009 00:40:00	-8.31
4	01.01.2009 00:50:00	-8.27
...	...	...
420546	31.12.2016 23:20:00	-4.05
420547	31.12.2016 23:30:00	-3.35
420548	31.12.2016 23:40:00	-3.16
420549	31.12.2016 23:50:00	-4.23
420550	01.01.2017 00:00:00	-4.82

420551 rows × 2 columns

In [54]:
```
data=data[5::6]
data
```

Out[54]:

	Date Time	T (degC)
5	01.01.2009 01:00:00	-8.05
11	01.01.2009 02:00:00	-8.88
17	01.01.2009 03:00:00	-8.81
23	01.01.2009 04:00:00	-9.05
29	01.01.2009 05:00:00	-9.63
...	...	...
420521	31.12.2016 19:10:00	-0.98
420527	31.12.2016 20:10:00	-1.40
420533	31.12.2016 21:10:00	-2.75
420539	31.12.2016 22:10:00	-2.89
420545	31.12.2016 23:10:00	-3.93

In [55]:
```
data['Date Time']=pd.to_datetime(data['Date Time'])
data.head()
```

Out[55]:

	Date Time	T (degC)
5	2009-01-01 01:00:00	-8.05
11	2009-01-01 02:00:00	-8.88
17	2009-01-01 03:00:00	-8.81
23	2009-01-01 04:00:00	-9.05
29	2009-01-01 05:00:00	-9.63

In [56]:
```
data.index=data['Date Time']
```

In [57]:
```
del data['Date Time']
```

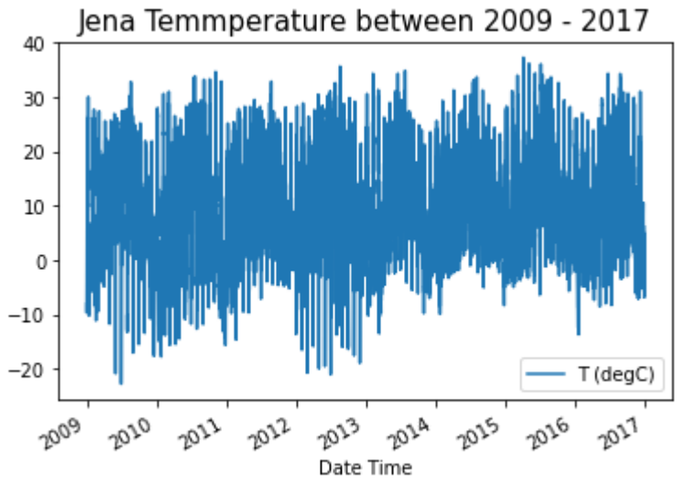
In [80]:
```
data.head()
```

Out[80]:

Date Time	T (degC)
2009-01-01 01:00:00	-8.05
2009-01-01 02:00:00	-8.88
2009-01-01 03:00:00	-8.81
2009-01-01 04:00:00	-9.05
2009-01-01 05:00:00	-9.63

3. Step 1 : Visualize your Time Series Data

In [59]:
```
import matplotlib.pyplot as plt
data.plot()
plt.title('Jena Temmperature between 2009 - 2017',size = 15)
plt.show()
```



In [ ]:
```
```

In [82]:
```
hourlydata=data
```

In [78]:
```
def jena_temp_to_x_y(data, window_size = 5):
    data_as_numpy = data.to_numpy()
    X = []
    y = []

    for i in range(len(data) - window_size):
        row = [temp for temp in data_as_numpy[i:i+window_size]] #2:6 -->2,3,4,5,6
        X.append(row)

        output = data_as_numpy[i+window_size] #2+5 = 7
        y.append(output)

    return np.array(X),np.array(y)
```

In [84]:
```
x,y=jena_temp_to_x_y(hourlydata)
```

In [85]:
```
x.shape,y.shape
```

Out[85]:
```
((70086, 5, 1), (70086, 1))
```

5. Model Building

Training Data

In [102]:
```
x_train=x[:60000]
y_train=y[:60000]
x_train.shape,y_train.shape
```

Out[102]:
```
((60000, 5, 1), (60000, 1))
```

Test Data

In [90]:
```
x_test=x[65000:70000]
y_test=y[65000:70000]
x_test.shape,y_test.shape
```

Out[90]:
```
((5000, 5, 1), (5000, 1))
```

3.Val data

In [91]:
```
x_val=x[60000:65000]
y_val=y[60000:65000]
x_val.shape,y_val.shape
```

Out[91]:
```
((5000, 5, 1), (5000, 1))
```

6. Model Training

In [95]:
```
from keras.models import Sequential
from keras.layers import Flatten,Dense,InputLayer,LSTM
```

In [98]:
```
model=Sequential()
model.add(InputLayer(input_shape=(5,1)))
model.add(LSTM(units=64))
model.add(Dense(units=8,activation="relu"))
model.add(Dense(units=1,activation="linear"))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 64)                16896

 dense_2 (Dense)             (None, 8)                 520

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 17,425
Trainable params: 17,425
Non-trainable params: 0
_________________________________________________________________
```

In [99]:
```
model.compile(optimizer='adam',loss="mse")
```

In [100]:
```
model.fit(x,y,batch_size=32,epochs=10,validation_data=(x_val,y_val))
```

```
Epoch 1/10
2191/2191 [==============================] - 31s 10ms/step - loss: 3.6193 - val_loss: 0.5212
Epoch 2/10
2191/2191 [==============================] - 21s 10ms/step - loss: 0.6614 - val_loss: 0.5816
Epoch 3/10
2191/2191 [==============================] - 33s 15ms/step - loss: 0.6562 - val_loss: 0.5062
Epoch 4/10
2191/2191 [==============================] - 26s 12ms/step - loss: 0.6471 - val_loss: 0.5005
Epoch 5/10
2191/2191 [==============================] - 28s 13ms/step - loss: 0.6426 - val_loss: 0.5259
Epoch 6/10
2191/2191 [==============================] - 26s 12ms/step - loss: 0.6400 - val_loss: 0.4901
Epoch 7/10
2191/2191 [==============================] - 28s 13ms/step - loss: 0.6352 - val_loss: 0.4936
Epoch 8/10
2191/2191 [==============================] - 26s 12ms/step - loss: 0.6330 - val_loss: 0.4949
Epoch 9/10
2191/2191 [==============================] - 25s 11ms/step - loss: 0.6327 - val_loss: 0.4823
Epoch 10/10
2191/2191 [==============================] - 25s 11ms/step - loss: 0.6286 - val_loss: 0.4939
```

Out[100]:
```
<keras.callbacks.History at 0x273a5cb7fa0>
```

1. For Train

In [104]:
```
y_train_pred=model.predict(x_train)
y_train_pred
```

Out[104]:
```
array([[-10.099517],
       [ -9.894124],
       [ -8.958533],
       ...,
       [ 12.408278],
       [ 15.623963],
       [ 16.945244]], dtype=float32)
```

In [108]:
```
train_output=pd.DataFrame({"actualo/p":y_train.flatten(),"predicto/p":y_train_pred.flatten()})
train_output
```

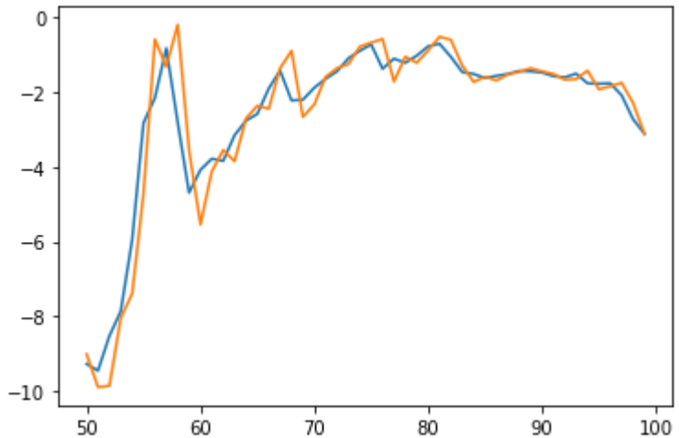
Out[108]:

	actualo/p	predicto/p
0	-9.67	-10.099517
1	-9.17	-9.894124
2	-8.10	-8.958533
3	-7.66	-7.395808
4	-7.04	-7.337616
...	...	...
59995	6.07	6.369117
59996	9.88	7.160706
59997	13.53	12.408278
59998	15.43	15.623963
59999	15.54	16.945244

60000 rows × 2 columns

In [110]:
```
plt.plot(train_output['actualo/p'][50:100])
plt.plot(train_output['predicto/p'][50:100])
```

Out[110]:
```
[<matplotlib.lines.Line2D at 0x274420de6d0>]
```



2. For Validation

In [116]:
```
y_val_pred=model.predict(x_val)
y_val_pred
```

Out[116]:
```
array([[16.048946],
       [13.458795],
       [12.980708],
       ...,
       [16.921562],
       [15.879438],
       [14.983412]], dtype=float32)
```

In [117]:
```
val_output=pd.DataFrame({"actual_val":y_val.flatten(),"prediction_val":y_val_pred.flatten()})
val_output
```

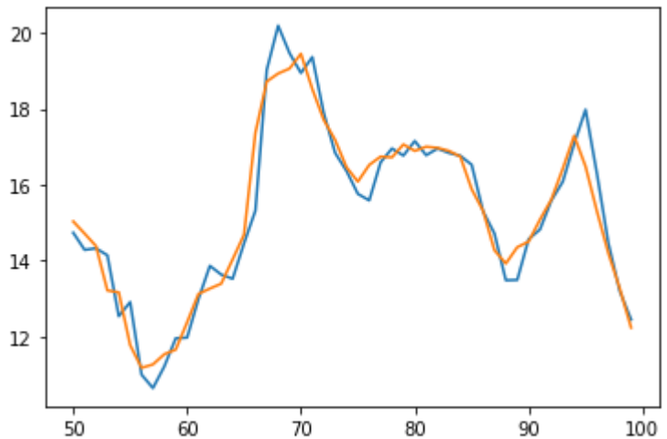
Out[117]:

	actual_val	prediction_val
0	14.02	16.048946
1	13.67	13.458795
2	12.27	12.980708
3	11.19	11.349584
4	10.85	10.372636
...	...	...
4995	18.27	17.512848
4996	17.85	17.296803
4997	16.65	16.921562
4998	15.85	15.879438
4999	15.09	14.983412

5000 rows × 2 columns

In [118]:
```
plt.plot(val_output['prediction_val'][50:100])
plt.plot(val_output['actual_val'][50:100])
```

Out[118]:
```
[<matplotlib.lines.Line2D at 0x27450141f70>]
```



3. For Test Data

In [111]:
```
y_test_pred=model.predict(x_test)
y_test_pred
```

Out[111]:
```
array([[14.312548 ],
       [13.188397 ],
       [12.898082 ],
       ...,
       [ 4.5498824],
       [ 4.6064568],
       [ 4.8349695]], dtype=float32)
```

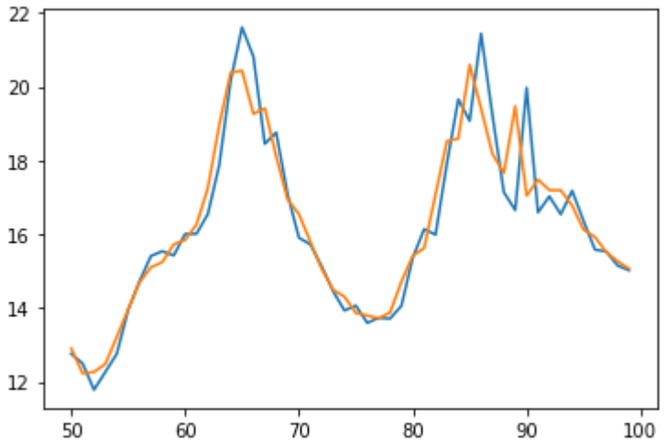
In [120]:
```
test_output=pd.DataFrame({"actual":y_test.flatten(),"prediction":y_test_pred.flatten()})
test_output.head()
```

Out[120]:

	actual	prediction
0	13.99	14.312548
1	13.46	13.188397
2	12.93	12.898082
3	12.43	12.497968
4	12.17	12.105231

In [121]:
```
plt.plot(test_output['prediction'][50:100])
plt.plot(test_output['actual'][50:100])
```

Out[121]:
```
[<matplotlib.lines.Line2D at 0x274685df1f0>]
```



That's the power of LSTMs and GRUs.