```
import pandas as pd
data=pd.read_csv('/content/alpha1.csv')
data
```

	Input	Output
0	10	2
1	20	4
2	30	6
3	40	8
4	50	10
5	60	12
6	70	14
7	80	16
8	90	18
9	100	20
10	110	22
11	120	24
12	130	26
13	140	28
14	150	30
15	160	32
16	170	34
17	180	36
18	190	38
19	200	40

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
from os import X_OK
x=data[['Input']].values
y=data[['Output']].values
```

```
from os import X_OK
x=data[['Input']].values
y=data[['Output']].values
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=33)
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
```

```
import torch
x_train_tensor=torch.tensor(x_train,dtype=torch.float32)
y_train_tensor=torch.tensor(y_train,dtype=torch.float32).view(-1,1)
x_test_tensor=torch.tensor(x_test,dtype=torch.float32)
y_test_tensor=torch.tensor(y_test,dtype=torch.float32).view(-1,1)
```

```
import torch.nn as nn
import torch.optim as optim
class Neuralnet(nn.Module):
   def __init__(self):
        super().__init__()
        self.n1=nn.Linear(1,12)
        self.n2=nn.Linear(12,14)
        self.n3=nn.Linear(14,1)
        self.relu=nn.ReLU()
        self.history={'loss': []}
   def forward(self,x):
        x=self.relu(self.n1(x))
        x=self.relu(self.n2(x))
```

```
        x=self.n3(x)
        return x
```

```
Lahari_brain=Neuralnet()
criteria=nn.MSELoss()
optimizer=optim.RMSprop(Lahari_brain.parameters(),lr=0.001)
```

```
def train_model(Lahari_brain,x_train,y_train,criteria,optmizer,epochs=4000):
    for i in range(epochs):
        optimizer.zero_grad()
        loss=criteria(Lahari_brain(x_train),y_train)
        loss.backward()
        optimizer.step()

        Lahari_brain.history['loss'].append(loss.item())
        if i%200==0:
            print(f"Epoch [{i}/epochs], loss: {loss.item():.6f}")
```

```
train_model(Lahari_brain,x_train_tensor,y_train_tensor,criteria,optimizer)
```

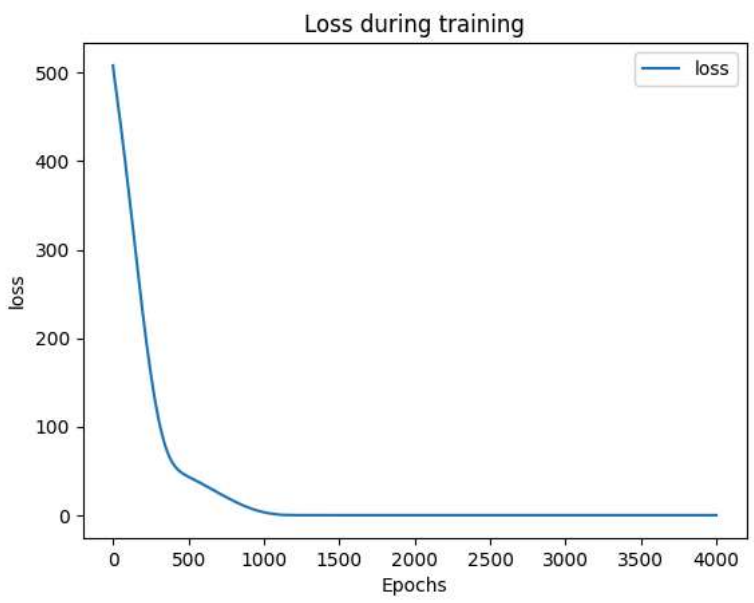
```
Epoch [0/epochs], loss: 508.032501
Epoch [200/epochs], loss: 224.298767
Epoch [400/epochs], loss: 57.715263
Epoch [600/epochs], loss: 34.557713
Epoch [800/epochs], loss: 16.057808
Epoch [1000/epochs], loss: 3.345932
Epoch [1200/epochs], loss: 0.077020
Epoch [1400/epochs], loss: 0.002397
Epoch [1600/epochs], loss: 0.002859
Epoch [1800/epochs], loss: 0.003176
Epoch [2000/epochs], loss: 0.002780
Epoch [2200/epochs], loss: 0.002750
Epoch [2400/epochs], loss: 0.002714
Epoch [2600/epochs], loss: 0.002762
Epoch [2800/epochs], loss: 0.002734
Epoch [3000/epochs], loss: 0.002769
Epoch [3200/epochs], loss: 0.002817
Epoch [3400/epochs], loss: 0.002779
Epoch [3600/epochs], loss: 0.002770
Epoch [3800/epochs], loss: 0.002820
```

```
with torch.no_grad():
  test_loss=criteria(Lahari_brain(x_test_tensor),y_test_tensor)
  print(f"Loss:{test_loss.item():.6f}")
```

```
Loss:0.107807
```

```
loss_df=pd.DataFrame(Lahari_brain.history)
```

```
import matplotlib.pyplot as plt
loss_df.plot()
plt.xlabel("Epochs")
plt.ylabel("loss")
plt.title("Loss during training")
plt.show()
```



```
x_n1_1=torch.tensor([[30]],dtype=torch.float32)
predict=Lahari_brain(torch.tensor(scaler.transform(x_n1_1),dtype=torch.float32)).item()
print(f"Prediction: {predict}")
```

```
Prediction: 5.964658737182617
```