```
print('NISHA')
212223230143
```

```
NISHA
212223230143
```

```
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

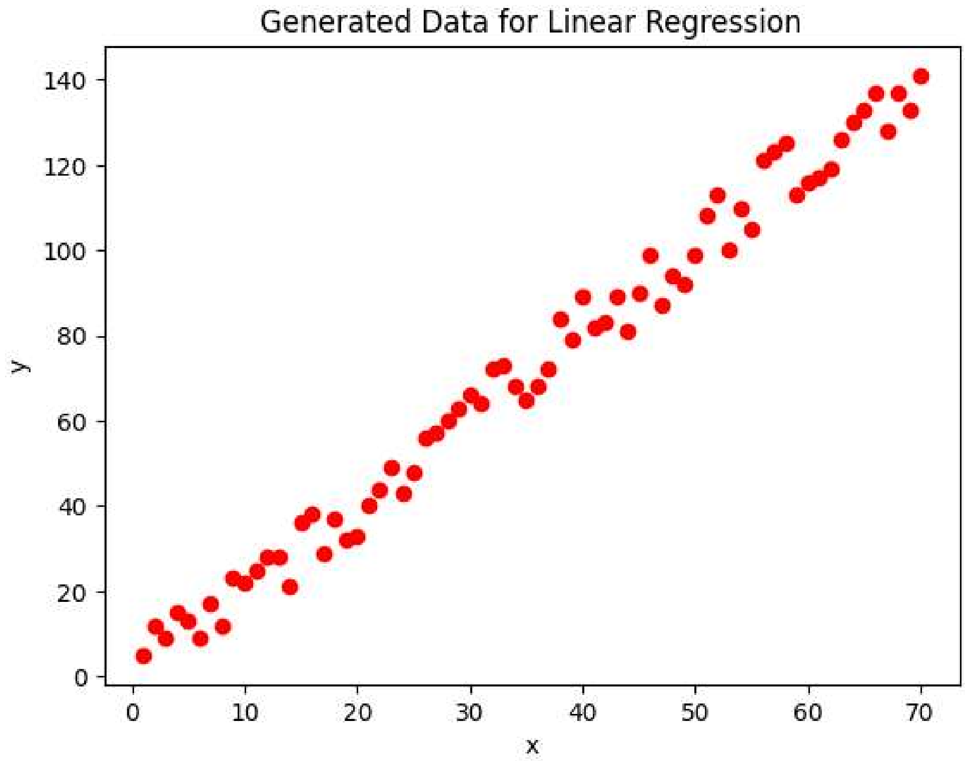
```
X = torch.linspace(1,70,70).reshape(-1,1)
```

```
torch.manual_seed(71)
e = torch.randint(-8,9,(70,1),dtype=torch.float)
# type e to check what are the numbers
```

```
y = 2*X + 1 + e
print(y.shape)
```

```
torch.Size([70, 1])
```

```
plt.scatter(X.numpy(), y.numpy(),color='red')  # Scatter plot of data points
plt.xlabel('x')
plt.ylabel('y')
plt.title('Generated Data for Linear Regression')
plt.show()
```



```
torch.manual_seed(59)
model = nn.Linear(1, 1)
print('Weight:', model.weight.item())
print('Bias:  ', model.bias.item())
```

```
Weight: 0.10597813129425049
Bias:   0.9637961387634277
```

```
print('NISHA')
212223230143
```

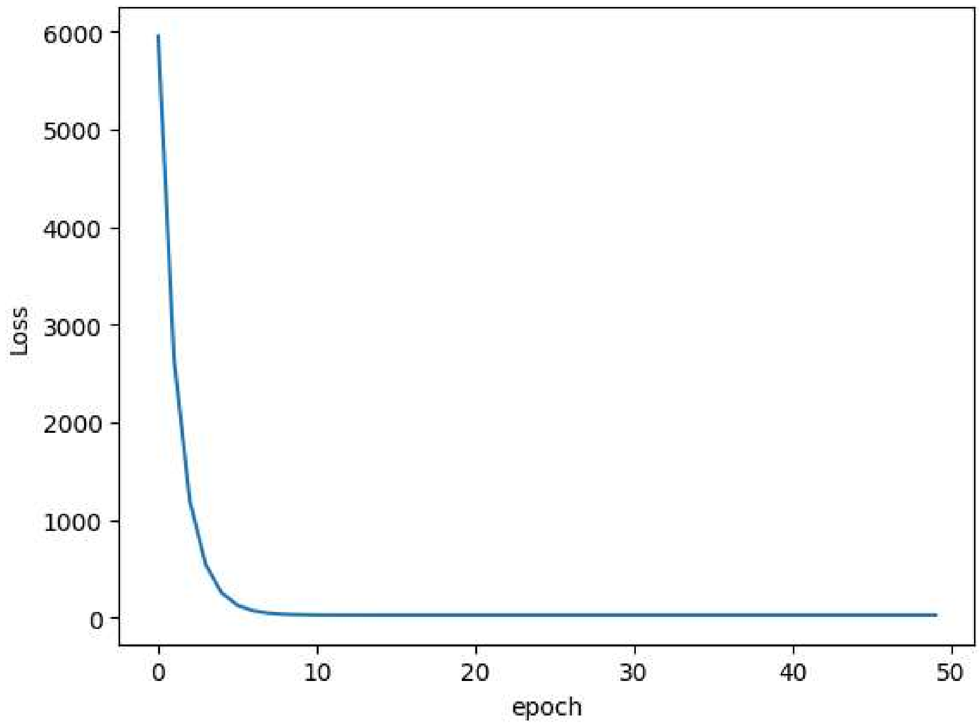
```
NISHA
212223230143
```

```
loss_function = nn.MSELoss()

optimizer = torch.optim.SGD(model.parameters(), lr=0.0001)
```

```
epochs = 50
losses = []

for epoch in range(1, epochs + 1):
    optimizer.zero_grad()
    y_pred = model(X)
    loss = loss_function(y_pred, y)
    losses.append(loss.item())

    loss.backward()
    optimizer.step()
    print(f'epoch: {epoch:2}  loss: {loss.item():10.8f}  '
          f'weight: {model.weight.item():10.8f}  '
          f'bias: {model.bias.item():10.8f}')
```

```
epoch:  1  loss: 5954.00195312  weight: 0.73509312  bias: 0.97723663
epoch:  2  loss: 2655.30761719  weight: 1.15417695  bias: 0.98620772
epoch:  3  loss: 1191.49755859  weight: 1.43334889  bias: 0.99220157
epoch:  4  loss: 541.92523193  weight: 1.61931860  bias: 0.99621207
epoch:  5  loss: 253.67466736  weight: 1.74320173  bias: 0.99890137
epoch:  6  loss: 125.76227570  weight: 1.82572591  bias: 1.00071061
epoch:  7  loss: 69.00058746  weight: 1.88069904  bias: 1.00193357
epoch:  8  loss: 43.81228256  weight: 1.91731894  bias: 1.00276589
epoch:  9  loss: 32.63482285  weight: 1.94171286  bias: 1.00333810
epoch: 10  loss: 27.67477417  weight: 1.95796239  bias: 1.00373697
epoch: 11  loss: 25.47373009  weight: 1.96878660  bias: 1.00402045
epoch: 12  loss: 24.49699783  weight: 1.97599685  bias: 1.00422692
epoch: 13  loss: 24.06353760  weight: 1.98079956  bias: 1.00438225
epoch: 14  loss: 23.87118340  weight: 1.98399854  bias: 1.00450337
epoch: 15  loss: 23.78580666  weight: 1.98612916  bias: 1.00460184
epoch: 16  loss: 23.74789619  weight: 1.98754811  bias: 1.00468516
epoch: 17  loss: 23.73106384  weight: 1.98849285  bias: 1.00475836
epoch: 18  loss: 23.72358131  weight: 1.98912179  bias: 1.00482488
epoch: 19  loss: 23.72023773  weight: 1.98954046  bias: 1.00488687
epoch: 20  loss: 23.71874046  weight: 1.98981893  bias: 1.00494587
epoch: 21  loss: 23.71806335  weight: 1.99000406  bias: 1.00500286
epoch: 22  loss: 23.71775055  weight: 1.99012709  bias: 1.00505853
epoch: 23  loss: 23.71759033  weight: 1.99020863  bias: 1.00511336
epoch: 24  loss: 23.71750450  weight: 1.99026263  bias: 1.00516760
epoch: 25  loss: 23.71745491  weight: 1.99029815  bias: 1.00522137
epoch: 26  loss: 23.71741104  weight: 1.99032140  bias: 1.00527489
epoch: 27  loss: 23.71738243  weight: 1.99033654  bias: 1.00532830
epoch: 28  loss: 23.71734619  weight: 1.99034631  bias: 1.00538158
epoch: 29  loss: 23.71732521  weight: 1.99035239  bias: 1.00543475
epoch: 30  loss: 23.71729469  weight: 1.99035609  bias: 1.00548792
epoch: 31  loss: 23.71726418  weight: 1.99035811  bias: 1.00554097
epoch: 32  loss: 23.71723938  weight: 1.99035919  bias: 1.00559402
epoch: 33  loss: 23.71720505  weight: 1.99035943  bias: 1.00564706
epoch: 34  loss: 23.71717262  weight: 1.99035931  bias: 1.00570011
epoch: 35  loss: 23.71715355  weight: 1.99035883  bias: 1.00575316
epoch: 36  loss: 23.71712494  weight: 1.99035811  bias: 1.00580621
epoch: 37  loss: 23.71710014  weight: 1.99035728  bias: 1.00585926
epoch: 38  loss: 23.71706963  weight: 1.99035633  bias: 1.00591230
epoch: 39  loss: 23.71703720  weight: 1.99035537  bias: 1.00596535
epoch: 40  loss: 23.71701050  weight: 1.99035430  bias: 1.00601840
epoch: 41  loss: 23.71697998  weight: 1.99035323  bias: 1.00607145
epoch: 42  loss: 23.71695518  weight: 1.99035215  bias: 1.00612450
epoch: 43  loss: 23.71692657  weight: 1.99035096  bias: 1.00617754
epoch: 44  loss: 23.71689796  weight: 1.99034977  bias: 1.00623047
epoch: 45  loss: 23.71686935  weight: 1.99034870  bias: 1.00628340
epoch: 46  loss: 23.71684265  weight: 1.99034762  bias: 1.00633633
epoch: 47  loss: 23.71681023  weight: 1.99034643  bias: 1.00638926
epoch: 48  loss: 23.71678543  weight: 1.99034536  bias: 1.00644219
epoch: 49  loss: 23.71675301  weight: 1.99034429  bias: 1.00649512
epoch: 50  loss: 23.71673203  weight: 1.99034309  bias: 1.00654805
```

```
plt.plot(range(epochs), losses)
plt.ylabel('Loss')
plt.xlabel('epoch');
plt.show()
```

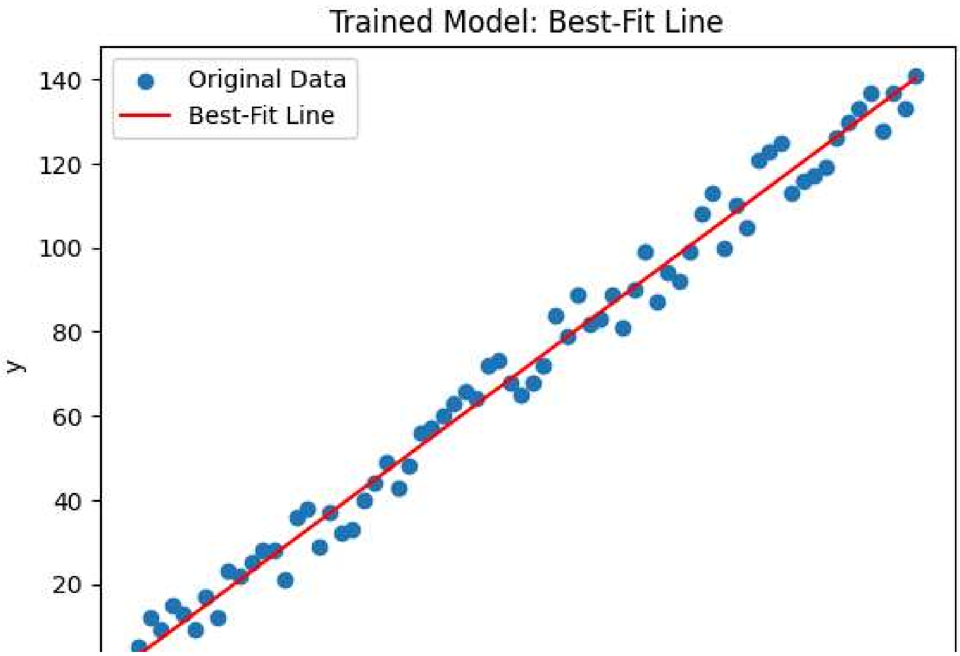


```
x1 = torch.tensor([X.min().item(), X.max().item()])


w1, b1 = model.weight.item(), model.bias.item()


y1 = x1 * w1 + b1
```

```


print(f'Final Weight: {w1:.8f}, Final Bias: {b1:.8f}')
print(f'X range: {x1.numpy()}')
print(f'Predicted Y values: {y1.numpy()}')
```

```
Final Weight: 1.99034309, Final Bias: 1.00654805
X range: [ 1. 70.]
Predicted Y values: [  2.996891 140.33057 ]
```

```
plt.scatter(X.numpy(), y.numpy(), label="Original Data")
plt.plot(x1.numpy(), y1.numpy(), 'r', label="Best-Fit Line")
plt.xlabel('x')
plt.ylabel('y')
plt.title('Trained Model: Best-Fit Line')
plt.legend()
plt.show()
```


Trained Model: Best-Fit Line
● Original Data
Best-Fit Line
y
140
120
100
80
60
40
20