```
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)
```

```
X
```

```
tensor([[ 1.],
        [ 2.],
        [ 3.],
        [ 4.],
        [ 5.],
        [ 6.],
        [ 7.],
        [ 8.],
        [ 9.],
        [10.],
        [11.],
        [12.],
        [13.],
        [14.],
        [15.],
        [16.],
        [17.],
        [18.],
        [19.],
        [20.],
        [21.],
        [22.],
        [23.],
        [24.],
        [25.],
        [26.],
        [27.],
        [28.],
        [29.],
        [30.],
        [31.],
        [32.],
        [33.],
        [34.],
        [35.],
        [36.],
        [37.],
        [38.],
        [39.],
        [40.],
        [41.],
        [42.],
        [43.],
        [44.],
        [45.],
        [46.],
        [47.],
        [48.],
        [49.],
        [50.],
        [51.],
        [52.],
        [53.],
        [54.],
        [55.],
        [56.],
        [57.],
        [58.],
        [59.],
        [60.],
        [61.],
        [62.],
        [63.],
        [64.],
        [65.],
        [66.],
        [67.],
        [68.],
        [69.],
        [70.]])
```

```
e = torch.randint(-8, 9, (70, 1), dtype=torch.float)
y = 2 * X + 1 + e
print(y.shape)
```

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

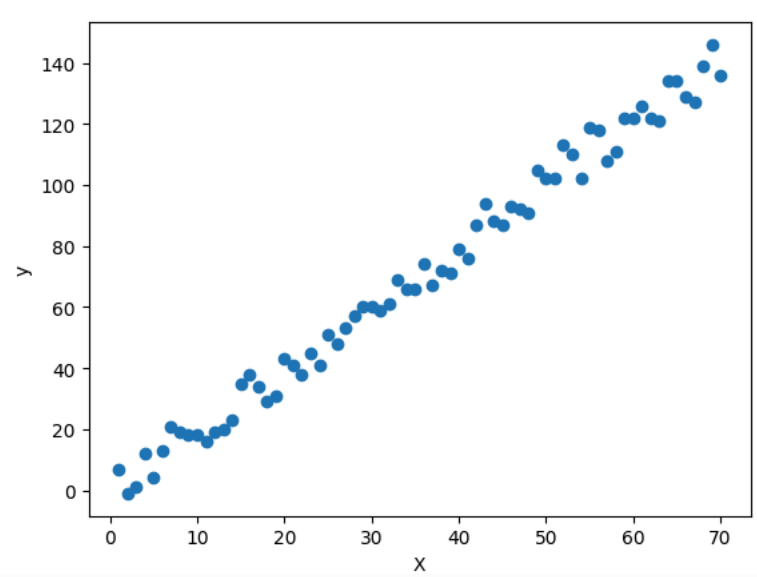
```
print(e.sum)
```

```
<built-in method sum of Tensor object at 0x7bd760046210>
```

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

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

```
plt.scatter(X.numpy(),y.numpy())
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```




```
torch.manual_seed(59)
```

```
# Defining the model class
model = nn.Linear(1, 1)  # Linear regression model with 1 input and 1 output

# Accessing weight and bias directly
print('Weight:', model.weight.item())
print('Bias:  ', model.bias.item())
```

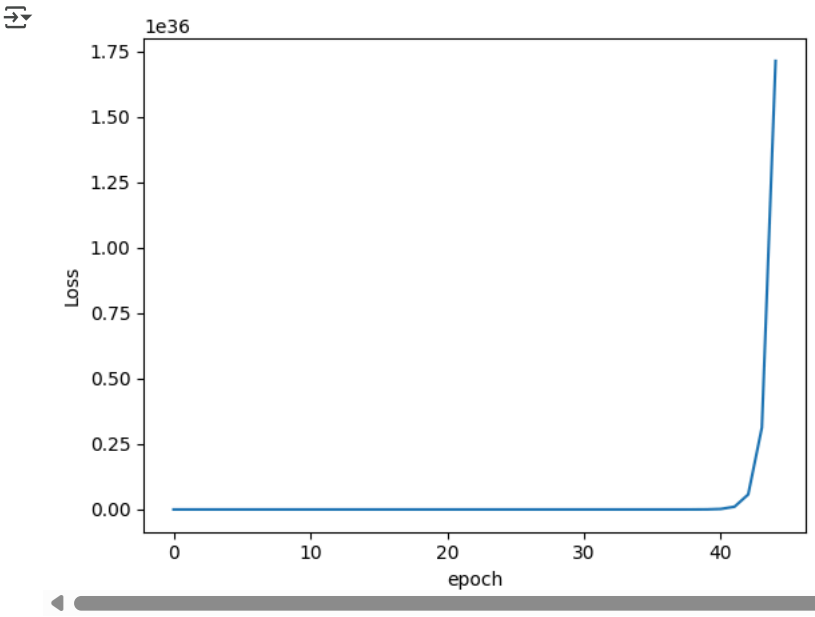
```
Weight: 0.10597813129425049
Bias:   0.9637961387634277
```

```
loss_function = nn.MSELoss()  # Mean Squared Error (MSE) loss

optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
```

```
eepochs = 50  # Number of training iterations
losses = []  # List to store loss values

for epoch in range(1, epochs + 1):  # Start from 1 to 50
    optimizer.zero_grad()  # Clear previous gradients
    y_pred = model(X)  # Forward pass
    loss = loss_function(y_pred, y)  # Compute loss
    losses.append(loss.item())  # Store loss value

    loss.backward()  # Compute gradients
    optimizer.step()  # Update weights

    # Print loss, weight, and bias for EVERY epoch (1 to 50)
    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: 5870.00781250  weight: 6.35281420  bias: 1.09531558
epoch:  2  loss: 31997.91015625  weight: -8.25538158  bias: 0.78304660
epoch:  3  loss: 174881.89062500  weight: 25.90614700  bias: 1.50858414
epoch:  4  loss: 956262.06250000  weight: -53.98085785  bias: -0.19279826
epoch:  5  loss: 5229343.00000000  weight: 132.83587646  bias: 3.78120017
epoch:  6  loss: 28597258.00000000  weight: -304.03698730  bias: -5.51673794
epoch:  7  loss: 156387808.00000000  weight: 717.59503174  bias: 16.22189331
epoch:  8  loss: 855227392.00000000  weight: -1671.50256348  bias: -34.61883163
epoch:  9  loss: 4676924928.00000000  weight: 3915.42773438  bias: 84.26806641
epoch: 10  loss: 25576378368.00000000  weight: -9149.66992188  bias: -193.75488281
epoch: 11  loss: 139867881472.00000000  weight: 21403.20312500  bias: 456.40020752
epoch: 12  loss: 764885991424.00000000  weight: -50045.02343750  bias: -1063.99914551
epoch: 13  loss: 4182881075200.00000000  weight: 117037.44531250  bias: 2491.46655273
epoch: 14  loss: 22874649788416.00000000  weight: -273686.75000000  bias: -5823.03417969
epoch: 15  loss: 125093069979648.00000000  weight: 640026.12500000  bias: 13620.51269531
epoch: 16  loss: 684088298045440.00000000  weight: -1496701.50000000  bias: -31848.44531250
epoch: 17  loss: 3741027915530240.00000000  weight: 3500059.25000000  bias: 74481.19531250
epoch: 18  loss: 20458305767866368.00000000  weight: -8184920.00000000  bias: -174171.84375000
epoch: 19  loss: 111878993208147968.00000000  weight: 19140532.00000000  bias: 407305.93750000
epoch: 20  loss: 611825169421303808.00000000  weight: -44760336.00000000  bias: -952486.37500000
epoch: 21  loss: 3345847418427015168.00000000  weight: 104672552.00000000  bias: 2227402.75000000
epoch: 22  loss: 18297215989890154496.00000000  weight: -244777936.00000000  bias: -5208803.50000000
epoch: 23  loss: 100060791802964213760.00000000  weight: 572415936.00000000  bias: 12180847.00000000
epoch: 24  loss: 547195974896677027840.00000000  weight: -1338600832.00000000  bias: -28485054.00000000
epoch: 25  loss: 2992413394706605539328.00000000  weight: 3130333184.00000000  bias: 66612568.00000000
epoch: 26  loss: 16364418091701975908352.00000000  weight: -7320318976.00000000  bias: -155774320.00000000
epoch: 27  loss: 894910191882798712750008.00000000  weight: 17118648320.00000000  bias: 364279872.00000000
epoch: 28  loss: 4893936334849892008591360.00000000  weight: -40032149504.00000000  bias: -851872768.00000000
epoch: 29  loss: 2676315311514278214762496.00000000  weight: 93615620096.00000000  bias: 1992113408.00000000
epoch: 30  loss: 1463578740450472028746547 2.00000000  weight: -218921205760.00000000  bias: -4658580480.00000000
epoch: 31  loss: 8003778465142320399594291 2.00000000  weight: 511949733888.00000000  bias: 10894143488.00000000
epoch: 32  loss: 4376974442986967337214361 6.00000000  weight: -1197200113664.00000000  bias: -25476077568.00000000
epoch: 33  loss: 2393607304854110770238062592.00000000  weight: 2799665414144.00000000  bias: 59576082432.00000000
epoch: 34  loss: 1308976000985622769543885619 2.00000000  weight: -6547047972864.00000000  bias: -139319328768.00000000
epoch: 35  loss: 7158309508094777173922021376 0.00000000  weight: 15310346256384.00000000  bias: 325799706624.00000000
epoch: 36  loss: 3914618540811625805328280453 12.00000000  weight: -35803417804800.00000000  bias: -761886539776.00000000
epoch: 37  loss: 2140761761248944143196231827456.00000000  weight: 83726679670784.00000000  bias: 1781679915008.00000000
epoch: 38  loss: 1170704179116529135792830598 3488.00000000  weight: -195795764641792.00000000  bias: -4166477938688.00000000
epoch: 39  loss: 6402151984262697847141785665 5360.00000000  weight: 457870592180224.00000000  bias: 9743355609088.00000000
epoch: 40  loss: 3501103333480684825336112532 68480.00000000  weight: -10707352797839 36.00000000  bias: -22784946208768.00000000
epoch: 41  loss: 1914624525727108858162361493 618688.00000000  weight: 2503926205120512.00000000  bias: 53282835791872.00000000
epoch: 42  loss: 1047037924797201425397067726 9790720.00000000  weight: -5855458837397504.00000000  bias: -124602487406592.00000000
epoch: 43  loss: 5725866346627376522335867457 5032320.00000000  weight: 13693053671833600.00000000  bias: 291384271568896.00000000
epoch: 44  loss: 3131266530831507147379307101 33243904.00000000  weight: -32021352013627392.00000000  bias: -681405352771584.00000000
epoch: 45  loss: 1712374755878598095391484855 753637888.00000000  weight: 74882282359357440.00000000  bias: 1593473400569856.00000000
epoch: 46  loss:        inf  weight: -175113032202977280.00000000  bias: -3726356038811648.00000000
epoch: 47  loss:        inf  weight: 409503725556596736.00000000  bias: 8714122497622016.00000000
epoch: 48  loss:        inf  weight: -957628860621389824.00000000  bias: -20378071483809792.00000000
epoch: 49  loss:        inf  weight: 2239425497509396480.00000000  bias: 47654337186365440.00000000
epoch: 50  loss:        inf  weight: -5236921656294768640.00000000  bias: -111440202169319424.00000000
epoch: 51  loss:        inf  weight: 12246599104192315392.00000000  bias: 260604115711688704.00000000
epoch: 52  loss:        inf  weight: -2863881004389079 4496.00000000  bias: -609425691452112896.00000000
epoch: 53  loss:        inf  weight: 669721724395569807 36.00000000  bias: 1425148763894185984.00000000
epoch: 54  loss:        inf  weight: -156615175132227305472.00000000  bias: -3332726121539043328.00000000
epoch: 55  loss:        inf  weight: 366246338049767112704.00000000  bias: 7793617693629743104.00000000
epoch: 56  loss:        inf  weight: -856471073972765589504.00000000  bias: -18225461861061492736.00000000
epoch: 57  loss:        inf  weight: 2002867144194529427456.00000000  bias: 42620435198178754560.00000000
epoch: 58  loss:        inf  weight: -4683727286916666621952.00000000  bias: -99668389297150492672.00000000
```

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

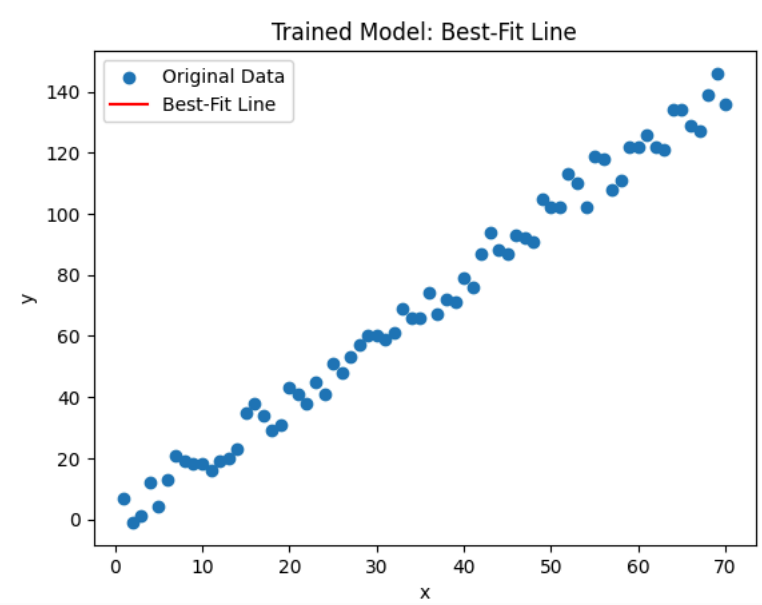


```
# Automatically determine x-range
x1 = torch.tensor([X.min().item(), X.max().item()])

# Extract model parameters
w1, b1 = model.weight.item(), model.bias.item()

# Compute y1 (predicted values)
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: nan, Final Bias: nan
X range: [ 1. 70.]
Predicted Y values: [nan nan]
```

```
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()
```




```
torch.save(model.state_dict(),'Sanafathimampregression.pt')
```