```
In [2]: import torch
```

```
In [3]: #Back propagation using torch
        x = torch.tensor(4.0,requires_grad=True)
```

```
In [4]: x
```

```
Out[4]: tensor(4., requires_grad=True)
```

```
In [8]: # Backpropagation using equation y = x^2  which is derivative y = 2x
        y=x**2
        y
```

```
Out[8]: tensor(16., grad_fn=<PowBackward0>)
```

```
In [10]: y.backward()
```

```
In [11]: print(x.grad) # y = 2*4
```

```
tensor(8.)
```

```
In [12]: lst = [[2.,3.,1.],[4.,5.,3.],[7.,6.,4.]]
         torch_input = torch.tensor(lst,requires_grad= True)
```

```
In [13]: torch_input
```

```
Out[13]: tensor([[2., 3., 1.],
                 [4., 5., 3.],
                 [7., 6., 4.]], requires_grad=True)
```

```
In [14]: #y=x**3+x**2
         y=torch_input**3+torch_input**2
```

```
In [15]: y
```

```
Out[15]: tensor([[ 12.,  36.,   2.],
                 [ 80., 150.,  36.],
                 [392., 252.,  80.]], grad_fn=<AddBackward0>)
```

```
In [16]: z = y.sum()
```

```
In [17]: z
```

```
Out[17]: tensor(1040., grad_fn=<SumBackward0>)
```

```
In [18]: z.backward()  # y = x**3 + x**2 ,## y= 3*x**2 + 2*x
```

In [19]:

```
torch_input.grad
```

Out[19]:

```
tensor([[ 16.,  33.,   5.],
        [ 56.,  85.,  33.],
        [161., 120.,  56.]])
```