PyTorch

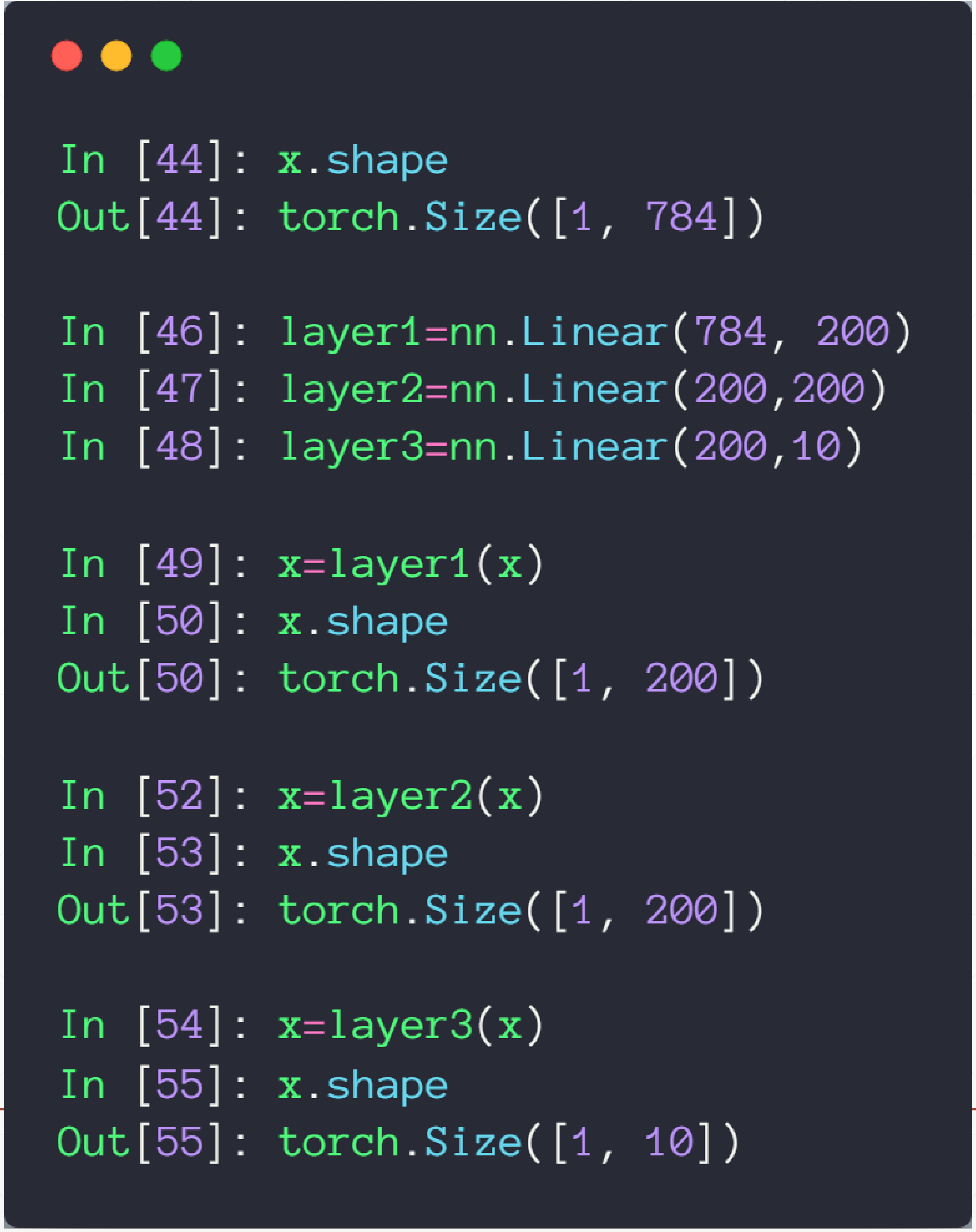
全军出击：全连接层

主讲人：龙良曲

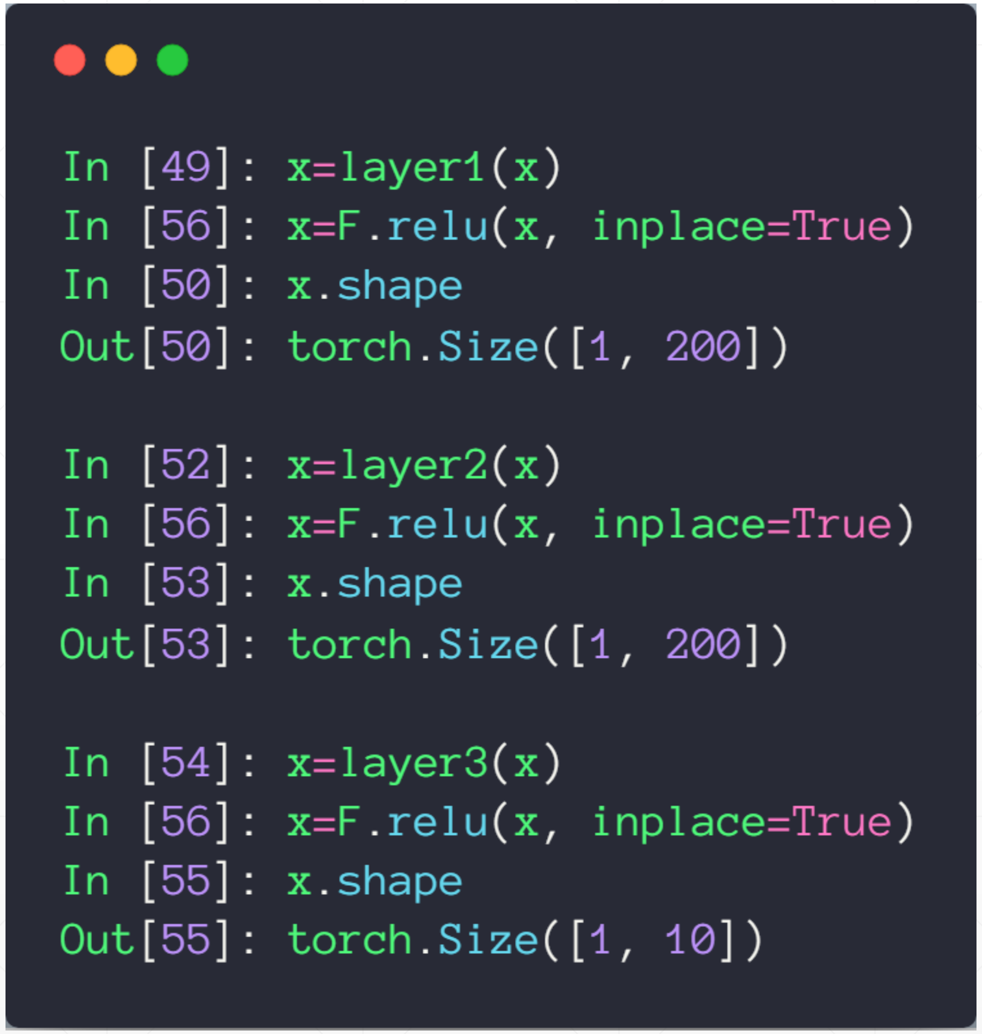
I know nothing

Be practical

nn.Linear


```
In [44]: x.shape
Out[44]: torch.Size([1, 784])

In [46]: layer1=nn.Linear(784, 200)
In [47]: layer2=nn.Linear(200,200)
In [48]: layer3=nn.Linear(200,10)

In [49]: x=layer1(x)
In [50]: x.shape
Out[50]: torch.Size([1, 200])

In [52]: x=layer2(x)
In [53]: x.shape
Out[53]: torch.Size([1, 200])

In [54]: x=layer3(x)
In [55]: x.shape
Out[55]: torch.Size([1, 10])
```

relu?

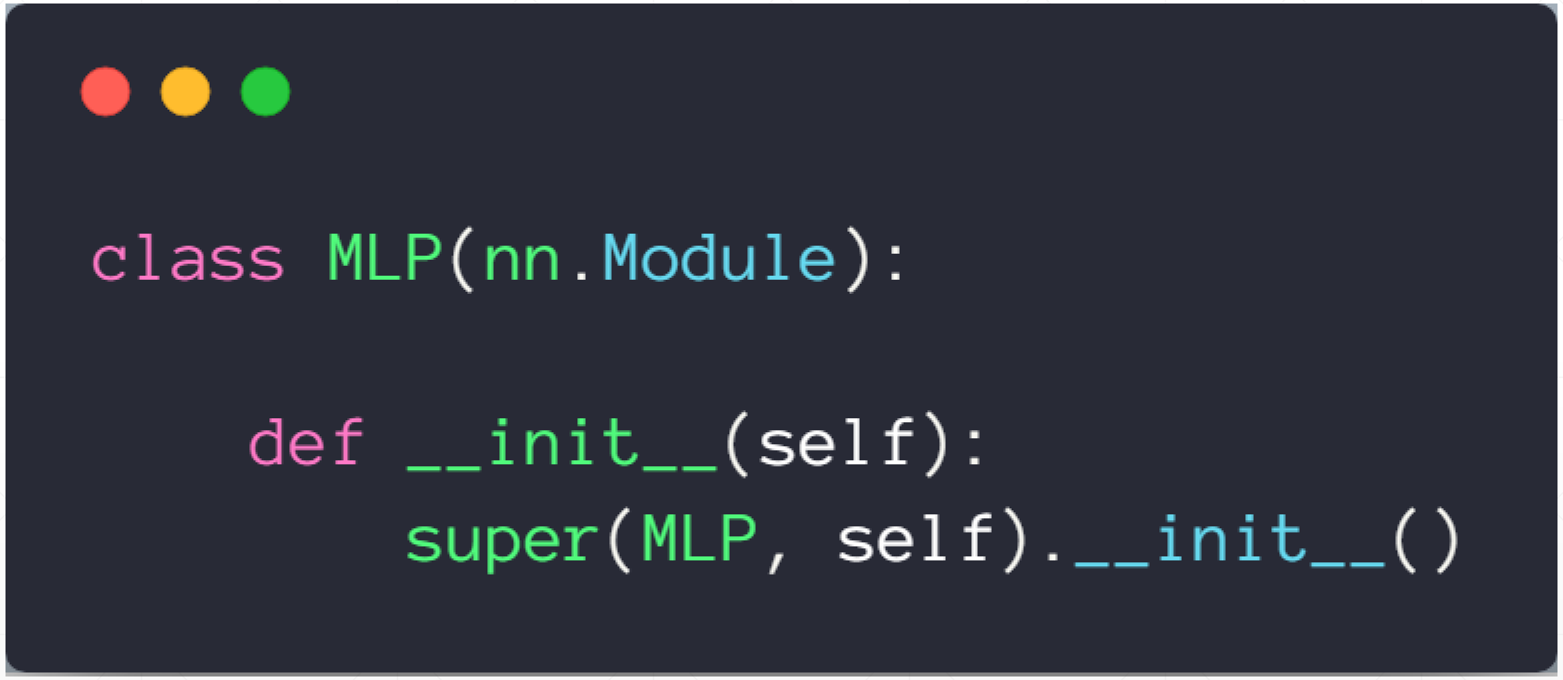


```
In [49]: x=layer1(x)
In [56]: x=F.relu(x, inplace=True)
In [50]: x.shape
Out[50]: torch.Size([1, 200])

In [52]: x=layer2(x)
In [56]: x=F.relu(x, inplace=True)
In [53]: x.shape
Out[53]: torch.Size([1, 200])

In [54]: x=layer3(x)
In [56]: x=F.relu(x, inplace=True)
In [55]: x.shape
Out[55]: torch.Size([1, 10])
```

concisely

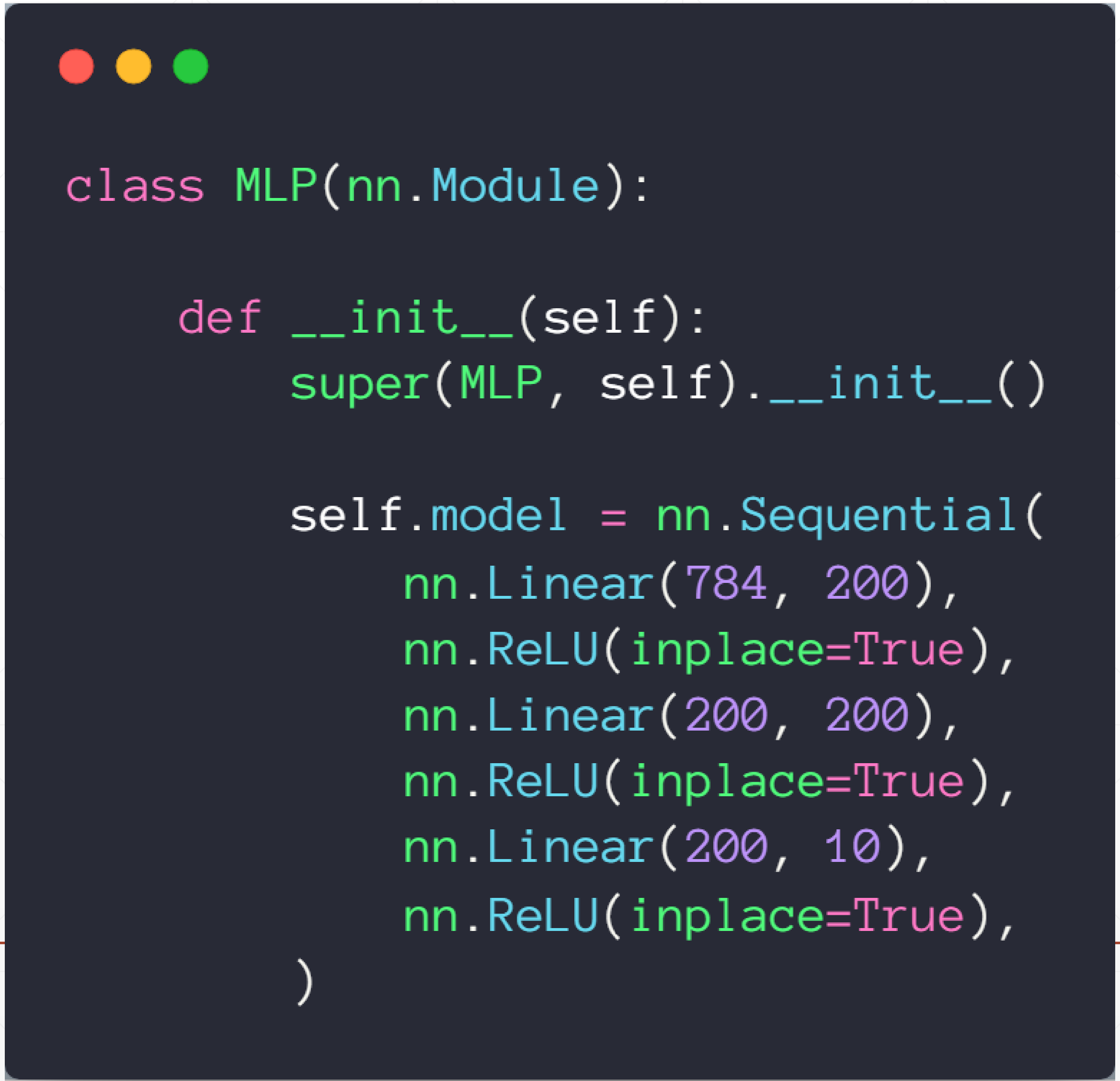
- inherit from nn.Module
- init layer in __init__
- implement forward()

Step1.



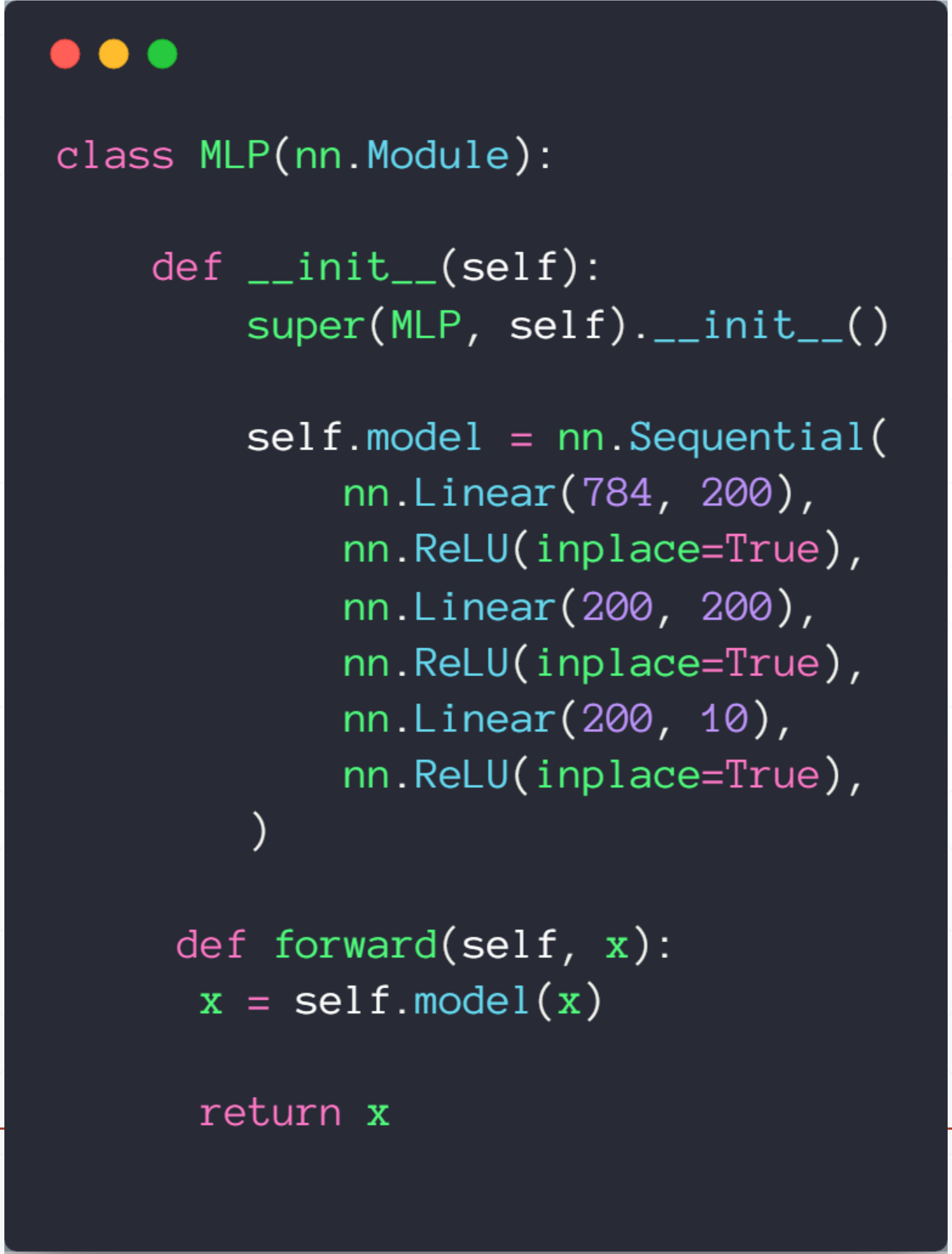
```python
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()
```

Step2.



```
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()

        self.model = nn.Sequential(
            nn.Linear(784, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 10),
            nn.ReLU(inplace=True),
        )
```

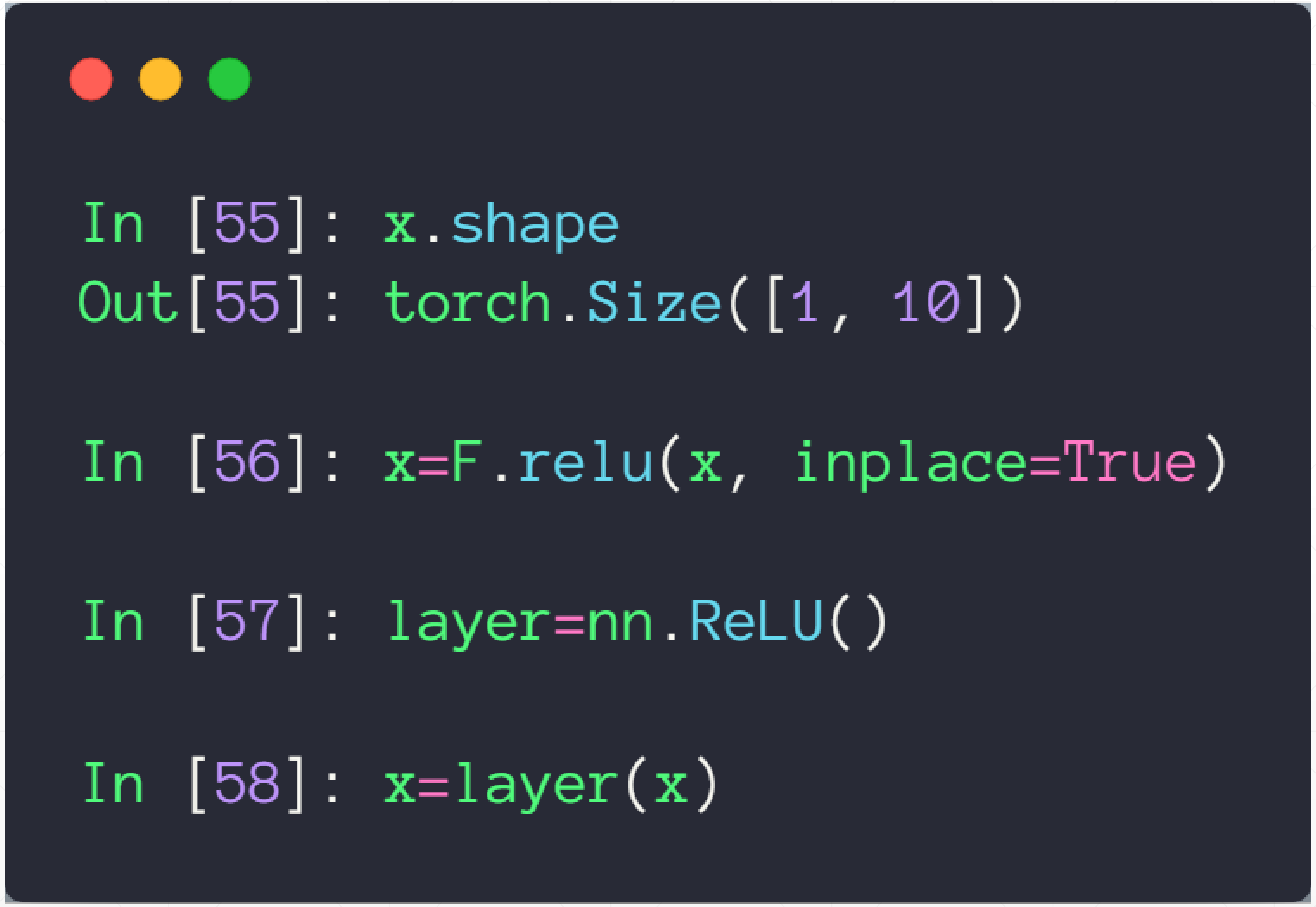
Step3.



```
class MLP(nn.Module):

    def __init__(self):
        super(MLP, self).__init__()

        self.model = nn.Sequential(
            nn.Linear(784, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 200),
            nn.ReLU(inplace=True),
            nn.Linear(200, 10),
            nn.ReLU(inplace=True),
        )

     def forward(self, x):
      x = self.model(x)

      return x
```

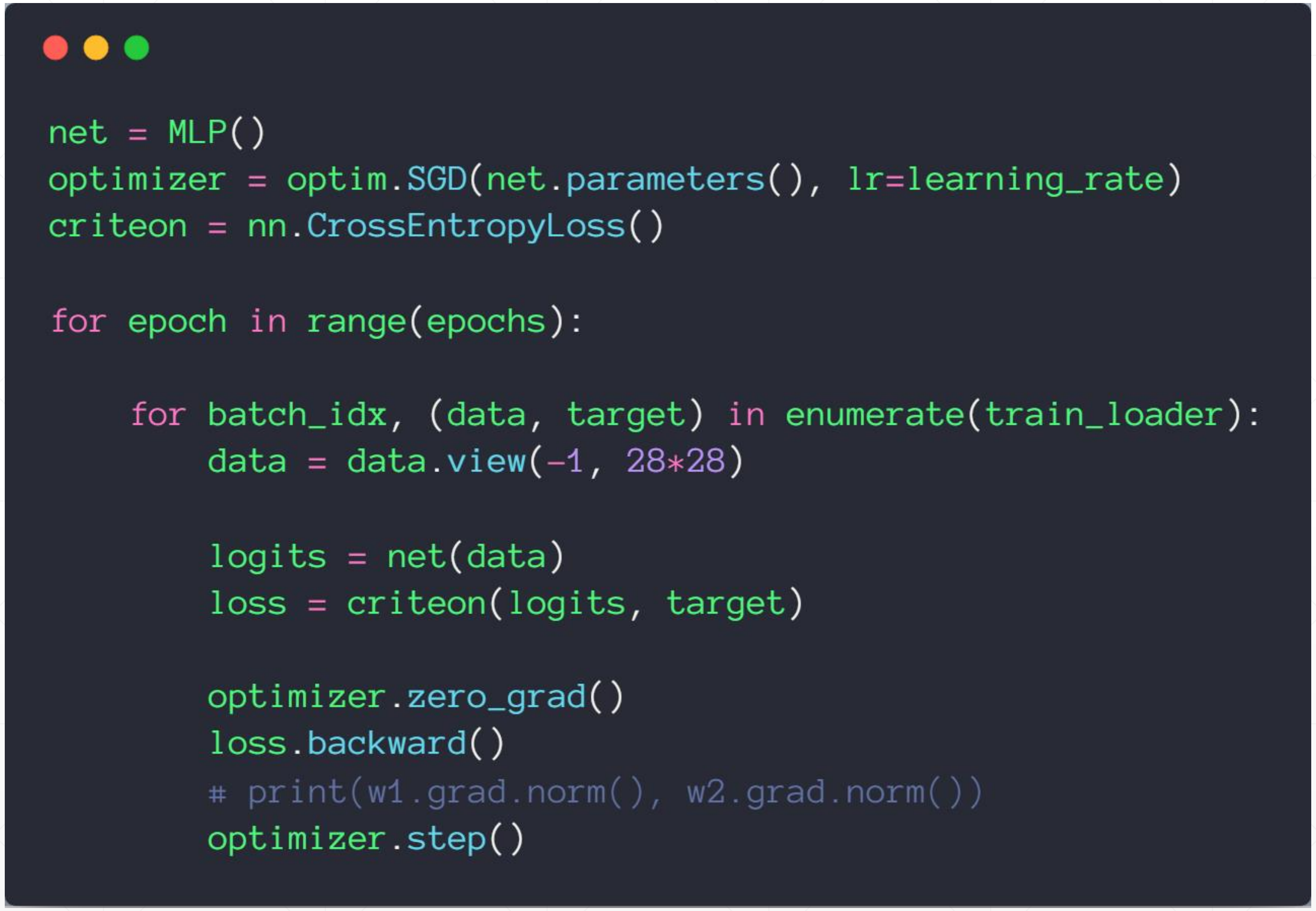
nn.ReLU v.s. F.relu()

- class-style API
- function-style API



```
In [55]: x.shape
Out[55]: torch.Size([1, 10])

In [56]: x=F.relu(x, inplace=True)

In [57]: layer=nn.ReLU()

In [58]: x=layer(x)
```

Train



```
net = MLP()
optimizer = optim.SGD(net.parameters(), lr=learning_rate)
criteon = nn.CrossEntropyLoss()

for epoch in range(epochs):

    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.view(-1, 28*28)

        logits = net(data)
        loss = criteon(logits, target)

        optimizer.zero_grad()
        loss.backward()
        # print(w1.grad.norm(), w2.grad.norm())
        optimizer.step()
```

下一课时

激活函数与GPU

Thank You.