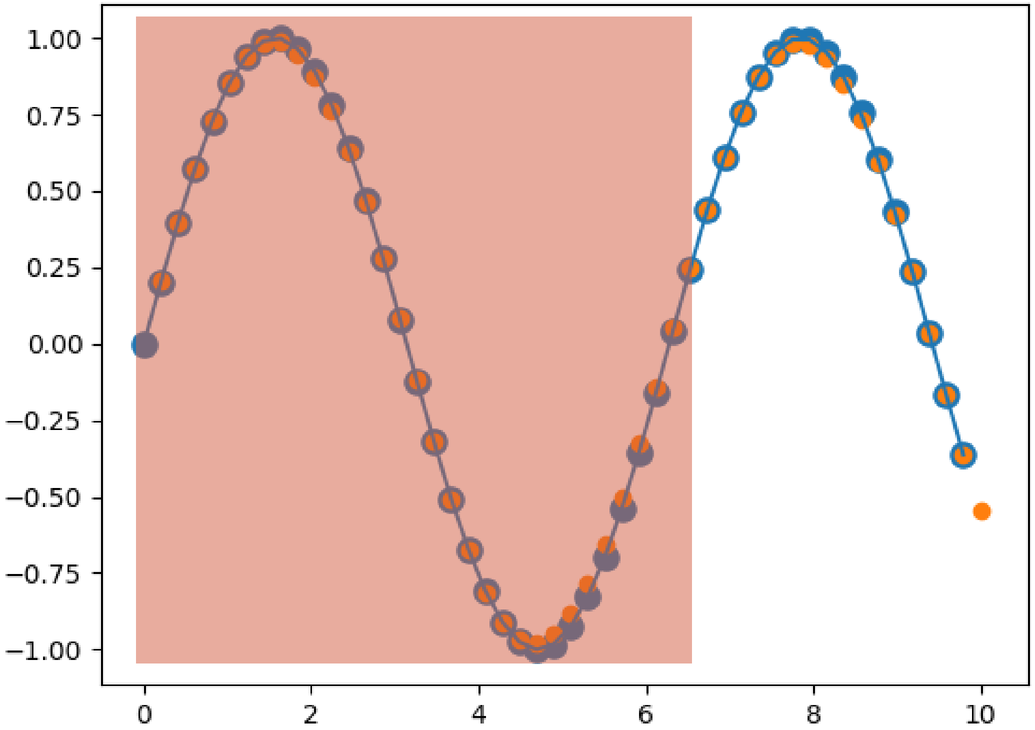
PyTorch

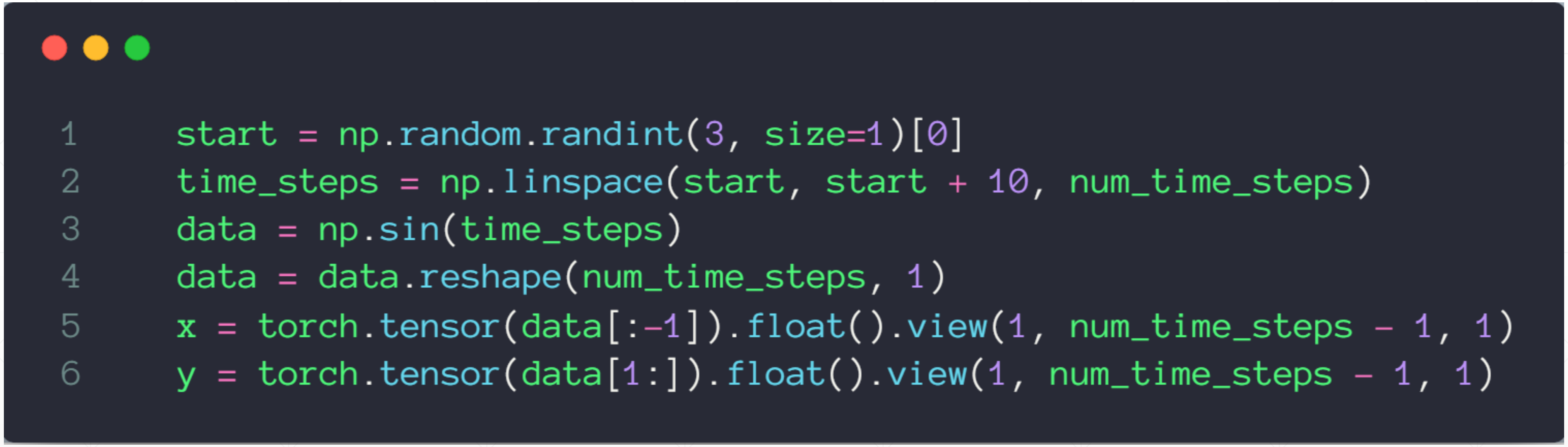
# 时间序列预测

主讲人：龙良曲

# Predict next

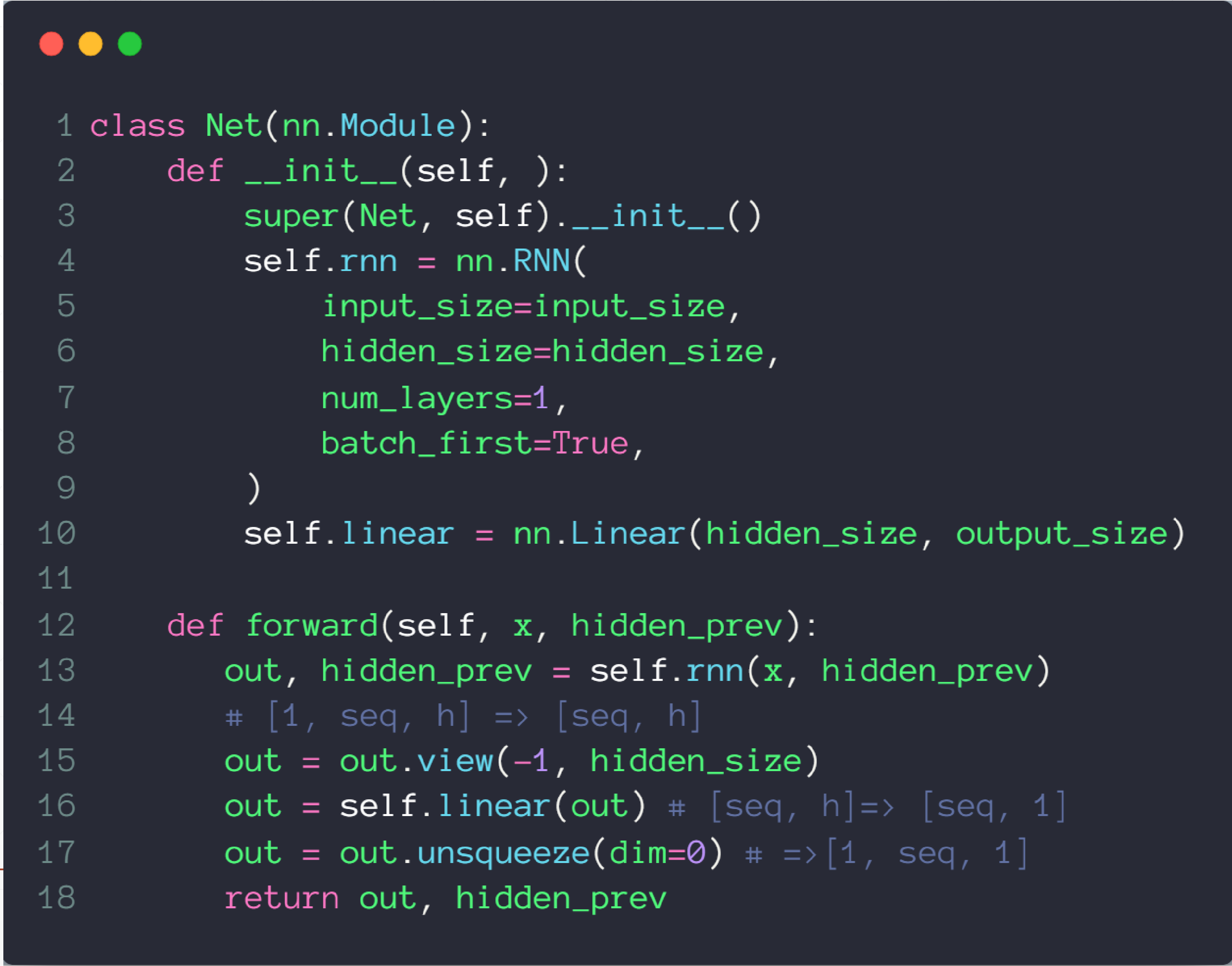


# Sample data



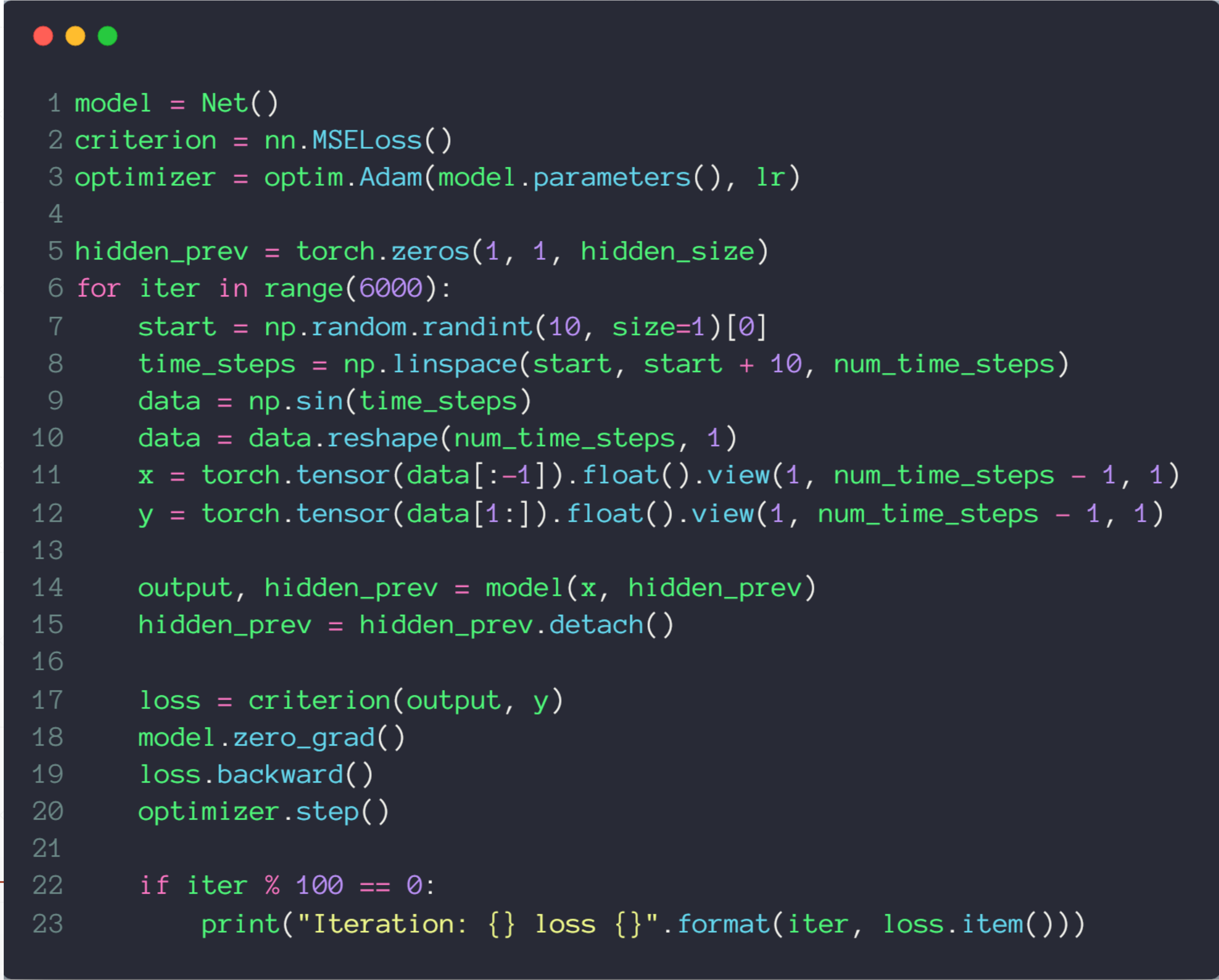
```python
start = np.random.randint(3, size=1)[0]
time_steps = np.linspace(start, start + 10, num_time_steps)
data = np.sin(time_steps)
data = data.reshape(num_time_steps, 1)
x = torch.tensor(data[:-1]).float().view(1, num_time_steps - 1, 1)
y = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)
```

# Network



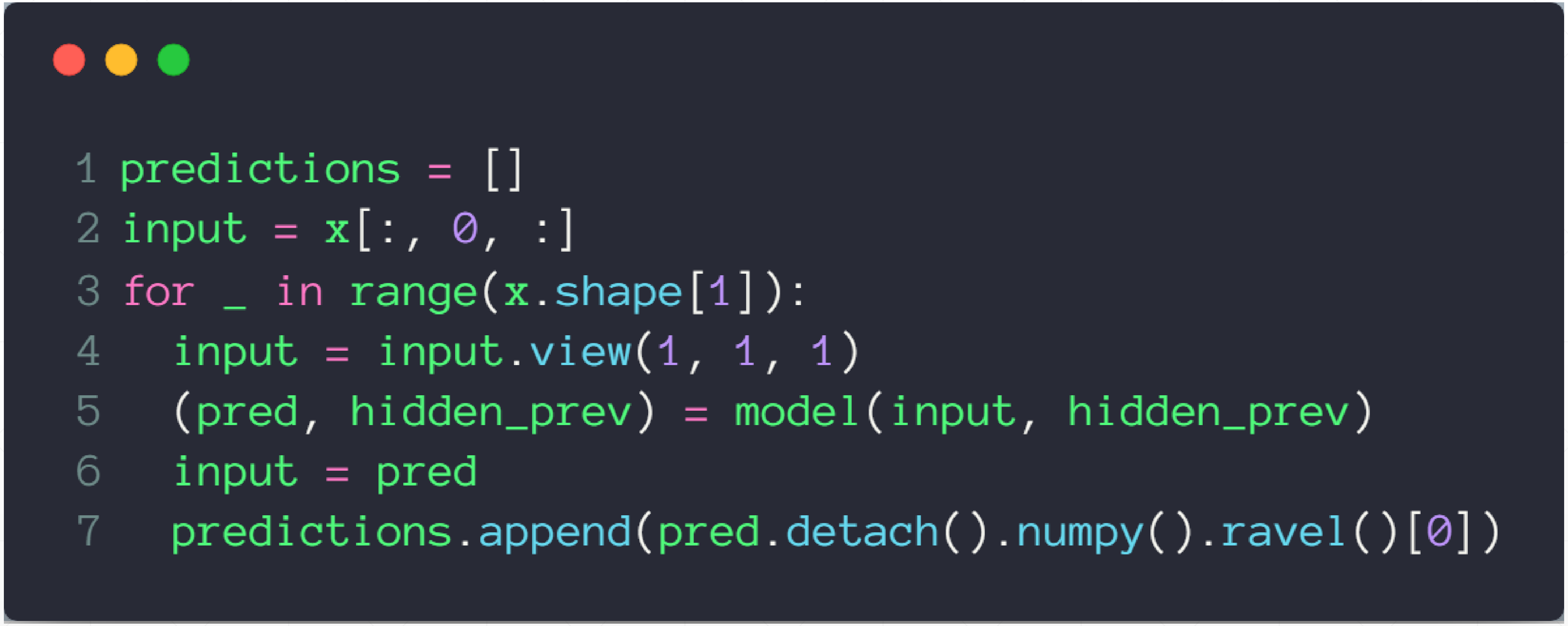
```python
class Net(nn.Module):
    def __init__(self, ):
        super(Net, self).__init__()
        self.rnn = nn.RNN(
            input_size=input_size,
            hidden_size=hidden_size,
            num_layers=1,
            batch_first=True,
        )
        self.linear = nn.Linear(hidden_size, output_size)

    def forward(self, x, hidden_prev):
       out, hidden_prev = self.rnn(x, hidden_prev)
       # [1, seq, h] => [seq, h]
       out = out.view(-1, hidden_size)
       out = self.linear(out) # [seq, h]=> [seq, 1]
       out = out.unsqueeze(dim=0) # =>[1, seq, 1]
       return out, hidden_prev
```

# Train

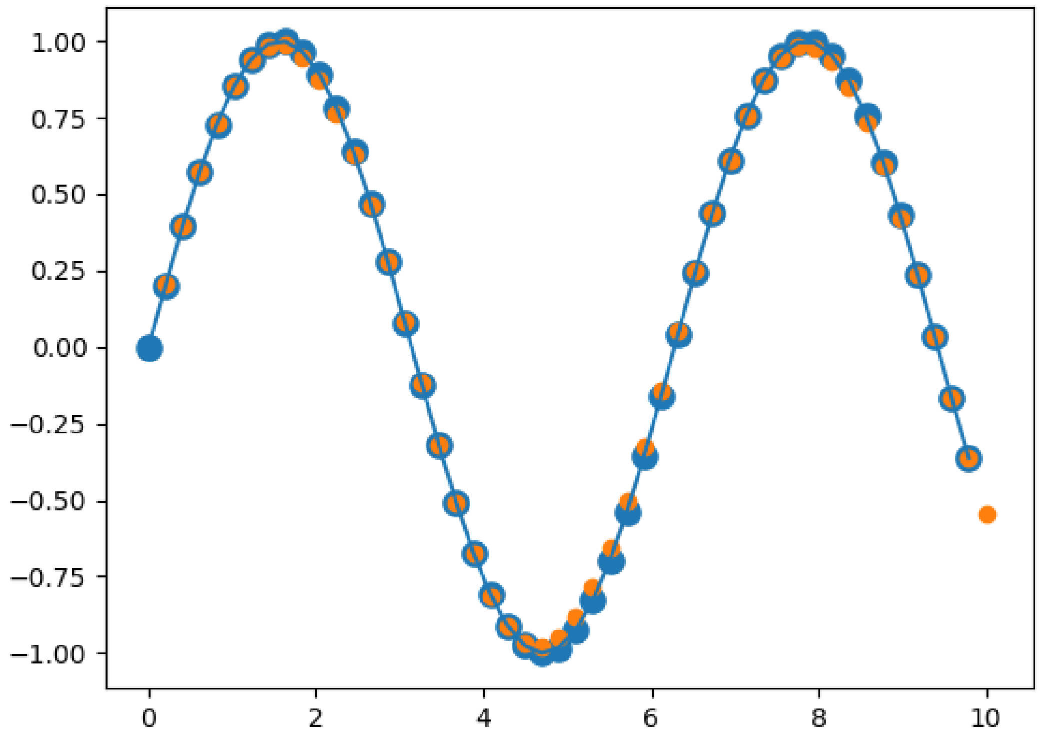


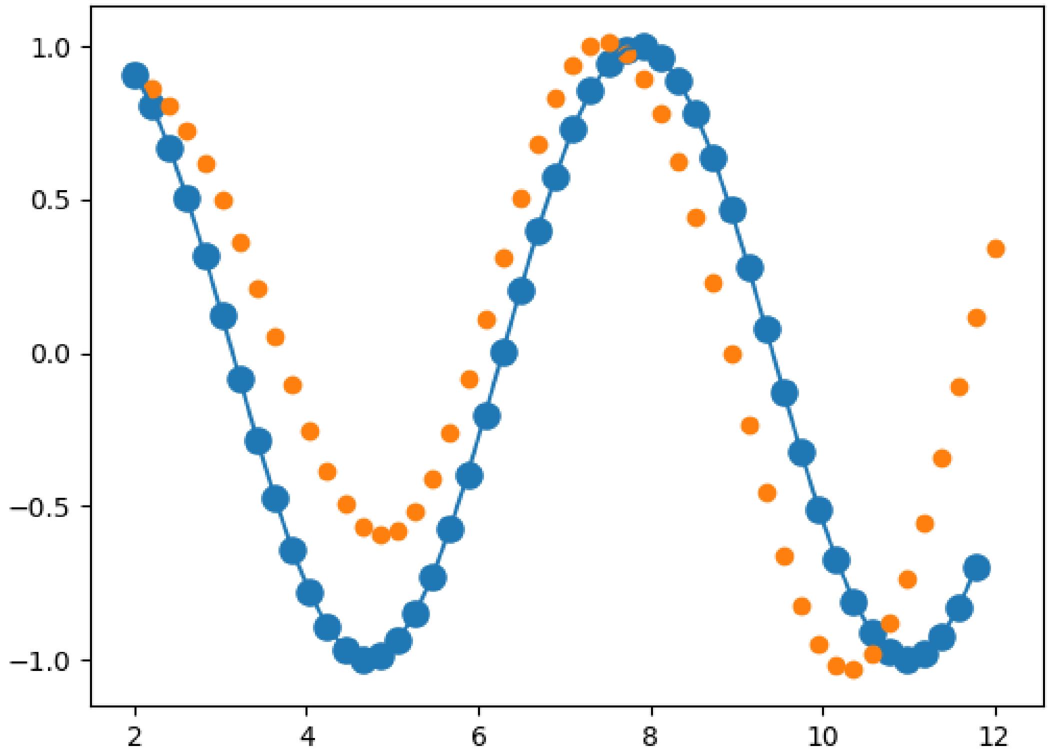
```
model = Net()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr)

hidden_prev = torch.zeros(1, 1, hidden_size)
for iter in range(6000):
    start = np.random.randint(10, size=1)[0]
    time_steps = np.linspace(start, start + 10, num_time_steps)
    data = np.sin(time_steps)
    data = data.reshape(num_time_steps, 1)
    x = torch.tensor(data[:-1]).float().view(1, num_time_steps - 1, 1)
    y = torch.tensor(data[1:]).float().view(1, num_time_steps - 1, 1)

    output, hidden_prev = model(x, hidden_prev)
    hidden_prev = hidden_prev.detach()

    loss = criterion(output, y)
    model.zero_grad()
    loss.backward()
    optimizer.step()

    if iter % 100 == 0:
        print("Iteration: {} loss {}".format(iter, loss.item()))
```

# Predict



```
predictions = []
input = x[:, 0, :]
for _ in range(x.shape[1]):
  input = input.view(1, 1, 1)
  (pred, hidden_prev) = model(input, hidden_prev)
  input = pred
  predictions.append(pred.detach().numpy().ravel()[0])
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10


1.0
0.5
0.0
−0.5
−1.0
2
4
6
8
10
12

# 下一课时

## RNN训练难题

# Thank You.