```python
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
```

```python
# 1. 데이터 생성 (y = 2x + 3 + noise)
torch.manual_seed(42)
X = torch.linspace(0, 10, 100).unsqueeze(1)   # 입력 (100 x 1)
y = 2 * X + 3 + torch.randn(X.size()) * 2     # 출력 (100 x 1)
```

```python
# 2. 모델 정의 (Linear Regression)
class LinearRegressionModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.linear = nn.Linear(1, 1)  # 입력 1차원 → 출력 1차원

    def forward(self, x):
        return self.linear(x)

model = LinearRegressionModel()
```

```python
# 3. 손실 함수 & 옵티마이저
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

```python
# 4. 학습 루프
epochs = 200
for epoch in range(epochs):
    # Forward
    y_pred = model(X)
    loss = criterion(y_pred, y)

    # Backward
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    if (epoch+1) % 20 == 0:
        print(f"Epoch {epoch+1}/{epochs}, Loss: {loss.item():.4f}")
```

```python
# 5. 결과 시각화
predicted = model(X).detach()

plt.scatter(X.numpy(), y.numpy(), label="Data")
plt.plot(X.numpy(), predicted.numpy(), color="red", label="Fitted Line")
plt.legend()
plt.show()
```

```
Epoch 20/200, Loss: 6.7266
Epoch 40/200, Loss: 6.2077
Epoch 60/200, Loss: 5.7824
Epoch 80/200, Loss: 5.4339
Epoch 100/200, Loss: 5.1483
Epoch 120/200, Loss: 4.9143
Epoch 140/200, Loss: 4.7225
Epoch 160/200, Loss: 4.5653
Epoch 180/200, Loss: 4.4365
Epoch 200/200, Loss: 4.3309
```