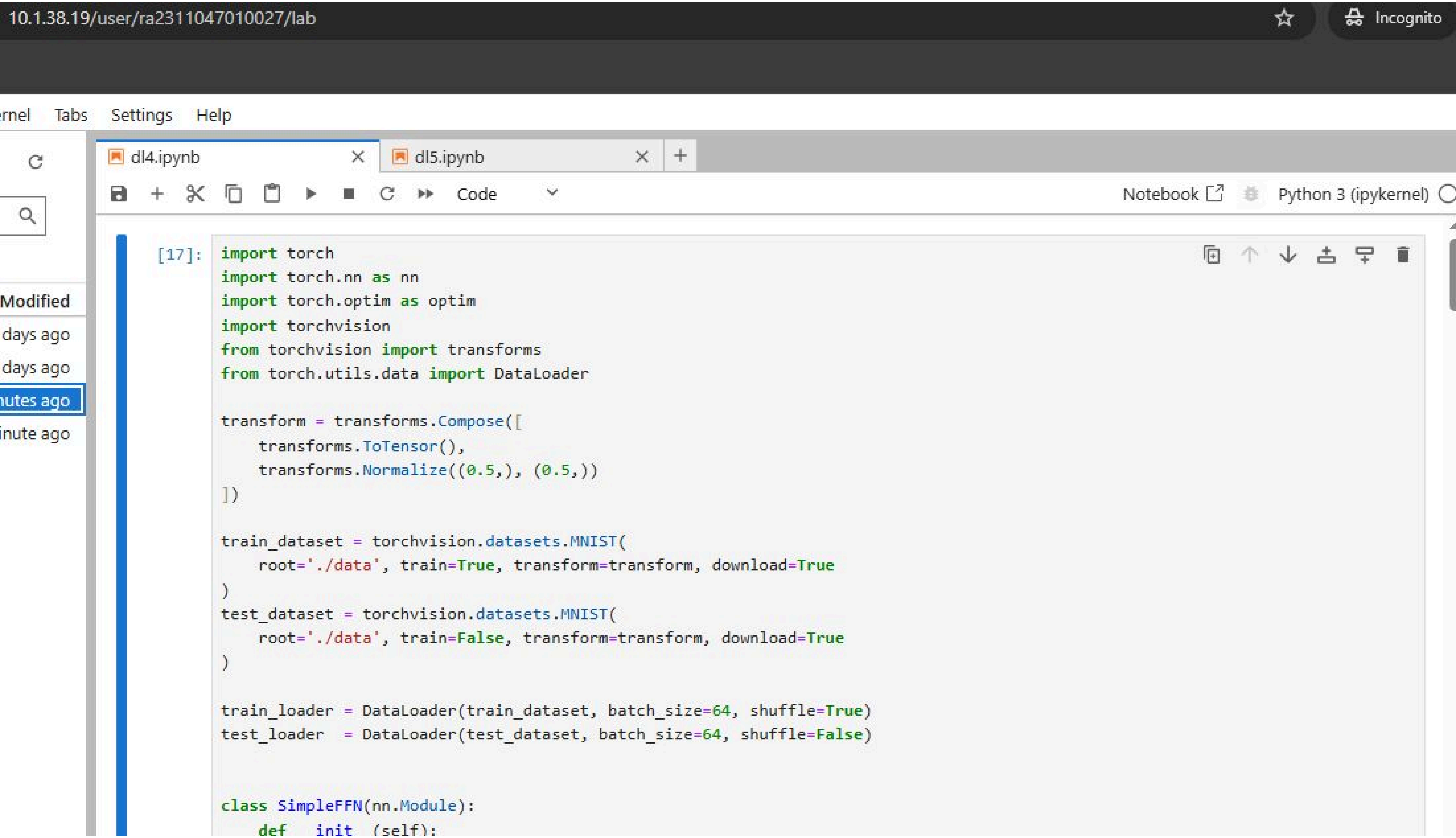
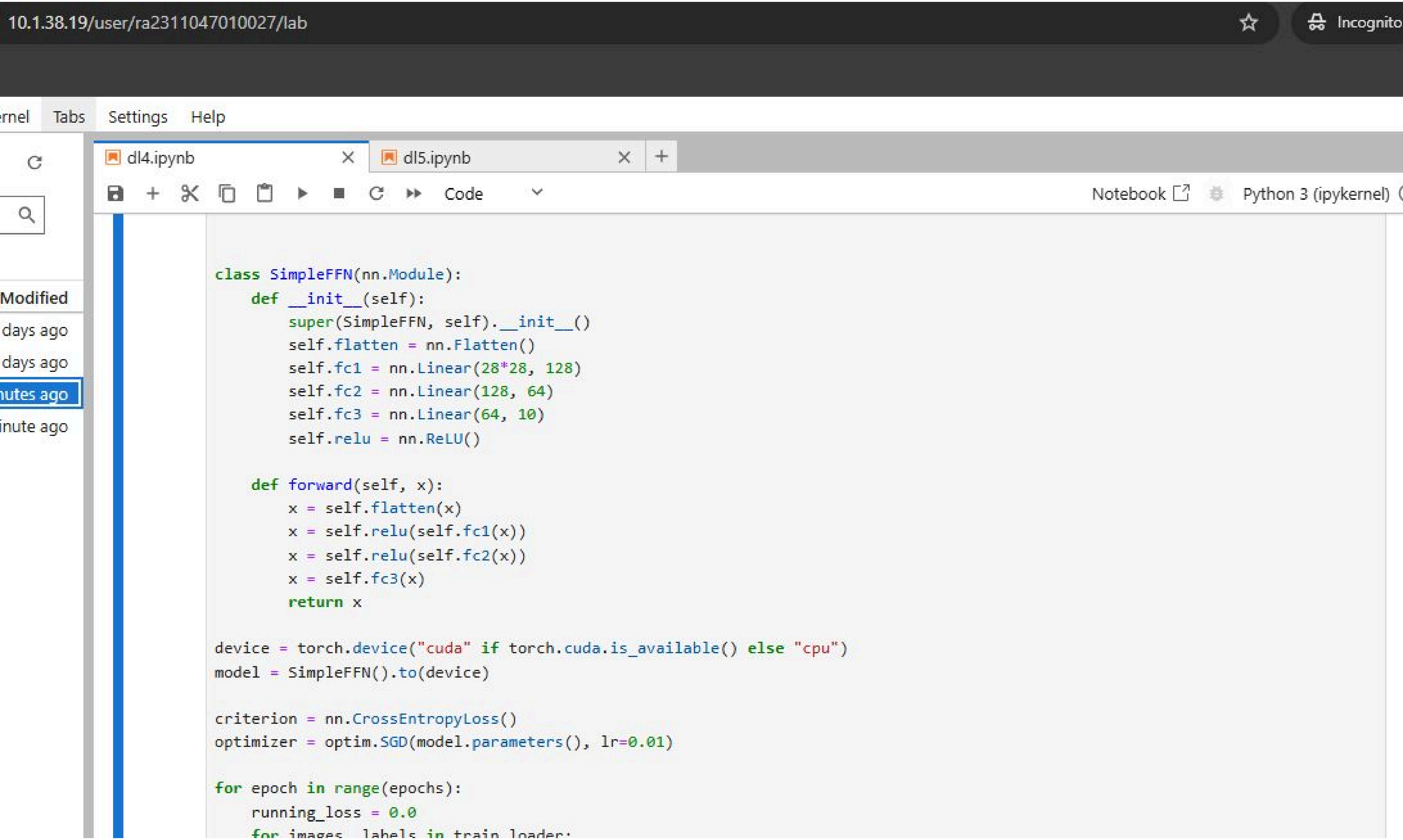
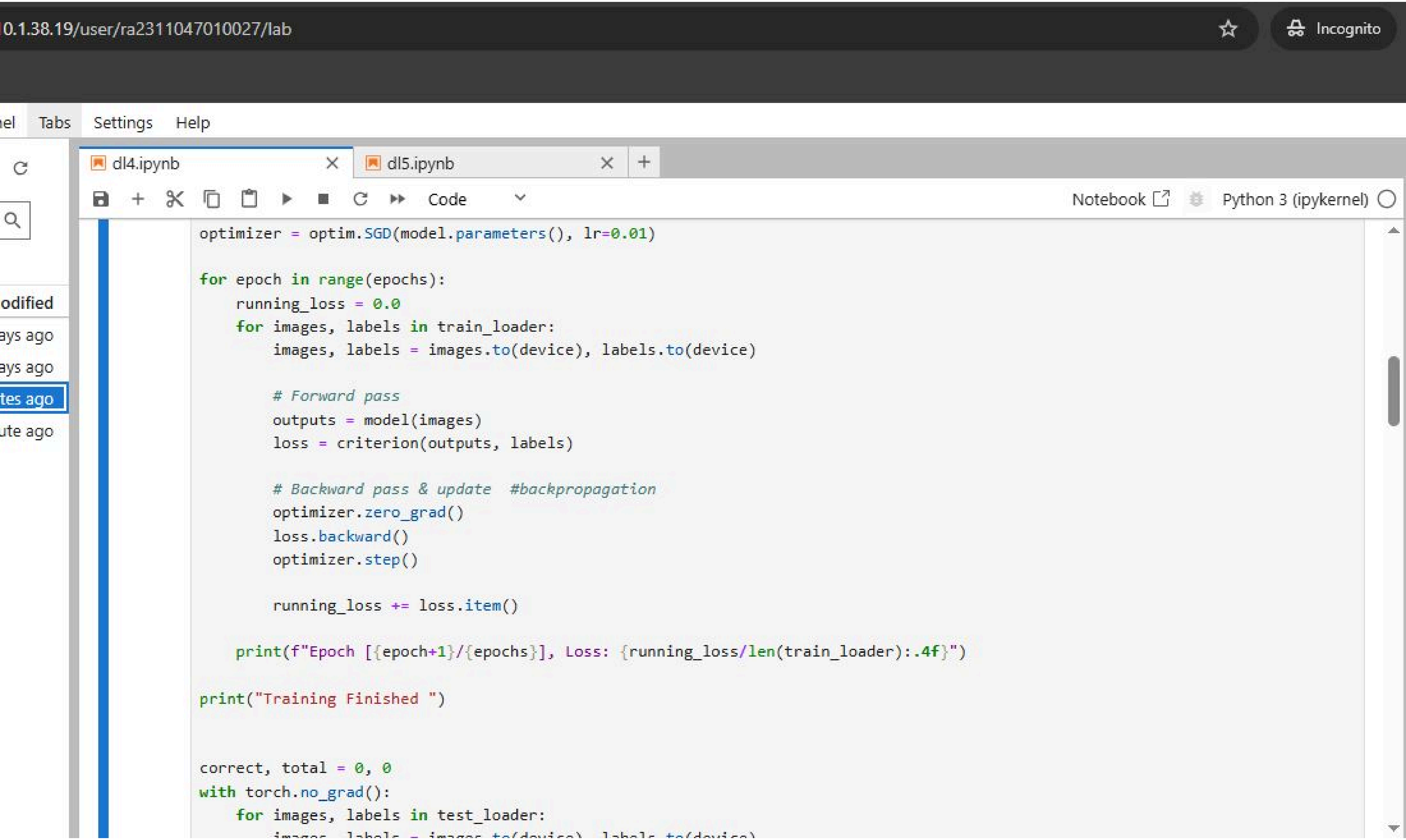


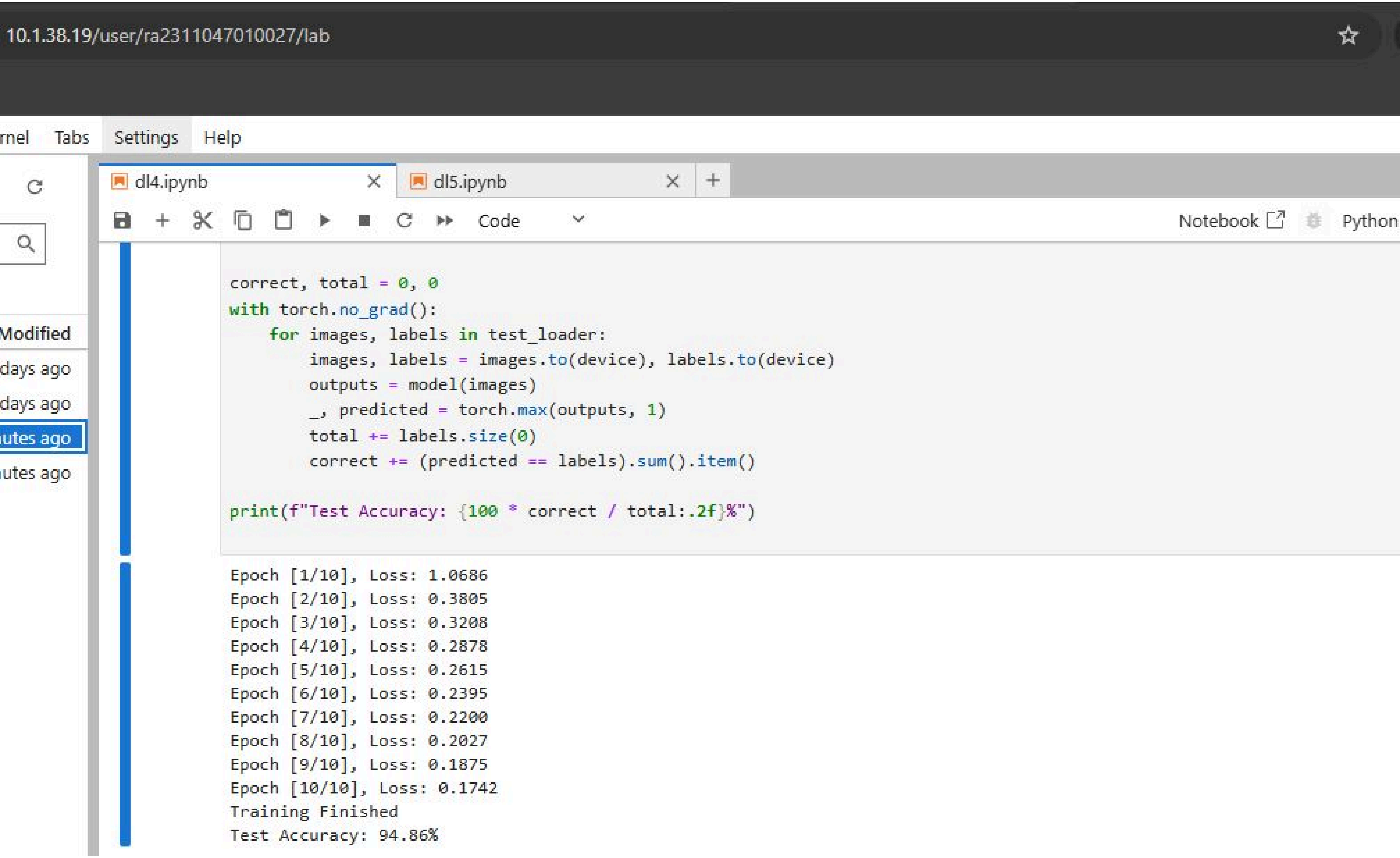
```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_dataset = torchvision.datasets.MNIST(
    root='./data', train=True, transform=transform, download=True
)
test_dataset = torchvision.datasets.MNIST(
    root='./data', train=False, transform=transform, download=True
)

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader  = DataLoader(test_dataset, batch_size=64, shuffle=False)


class SimpleFFN(nn.Module):
    def __init__(self):
```



```
class SimpleFFN(nn.Module):
    def __init__(self):
        super(SimpleFFN, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(28*28, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 10)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.flatten(x)
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = SimpleFFN().to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

for epoch in range(epochs):
    running_loss = 0.0
    for images, labels in train_loader:
```



```
optimizer = optim.SGD(model.parameters(), lr=0.01)

for epoch in range(epochs):
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        # Forward pass
        outputs = model(images)
        loss = criterion(outputs, labels)

        # Backward pass & update  #backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{epochs}], Loss: {running_loss/len(train_loader):.4f}")

print("Training Finished ")


correct, total = 0, 0
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
```



```
correct, total = 0, 0
with torch.no_grad():
    for images, labels in test_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f"Test Accuracy: {100 * correct / total:.2f}%")
```

```
Epoch [1/10], Loss: 1.0686
Epoch [2/10], Loss: 0.3805
Epoch [3/10], Loss: 0.3208
Epoch [4/10], Loss: 0.2878
Epoch [5/10], Loss: 0.2615
Epoch [6/10], Loss: 0.2395
Epoch [7/10], Loss: 0.2200
Epoch [8/10], Loss: 0.2027
Epoch [9/10], Loss: 0.1875
Epoch [10/10], Loss: 0.1742
Training Finished
Test Accuracy: 94.86%
```

14/08/25

4 Bulid Simple feed forward neural network to recognize handwritten Charac

Aim : To build a simple feed forward neural network to recognize hand written characters

Objective :

1. Load and preprocess MNIST dataset
2. Design feed forward neural network with input, hidden and output layers
3. Train the network using SGD optimizer & cross entropy
4. Evaluate the model on the test dataset and measure accuracy.
5. Compare training Performance and accuracy.

Pseudocode :

```
START
1. Import necessary libraries (torch, torchvision, torch.nn, torch.optim)
2. Load MNIST dataset :
    • Convert image to tensors
    • Normalize to range [-1, 1]
    • Use DataLoader for batching
3. Define FeedForwardNN class :
    • Input layer : 784 neurons (28x28 image flattened)
    • Hidden layer : 128, 64 neurons with ReLU activation
    • Output layer : 10 neurons (digits 0-9)
4. Define loss function as CrossEntropyLoss
5. Define Optimizer as SGD with learning rate = 0.01, ~~momentum = 0~~
```

```
6. TRAINING LOOP:
  for each epoch:
    for each batch:
           - forward pass
           - Compute loss
           - Backpropagation
           - Update weights
    Print loss at each epoch
7. TESTING LOOP:
    - forward pass on test data
    - Compare predictions with true labels
    - Compute accuracy
8. Print final accuracy on test dataset
END
```

Observations:

→ The dataset Contained 60,000 training images and 10,000 test image

→ Input images were grayscale (28x28)

→ After 10 epochs, accuracy is above 95%.

→ SGD optimizer requires more epochs compared to Adam

→ Loss decreased gradually over epochs, shows effective learning.

Training:

Epoch	Loss
1	0.0767
2	0.0645
3	0.0586
4	0.0500
5	0.0445
6	0.0395
7	0.0364
8	0.0347
9	0.0301
10	0.0248

Testing:

Total accuracy on test dataset = 97.86 %

Result :

I Successfully implmented simple feed forward neural network to recognize handwritten Character.