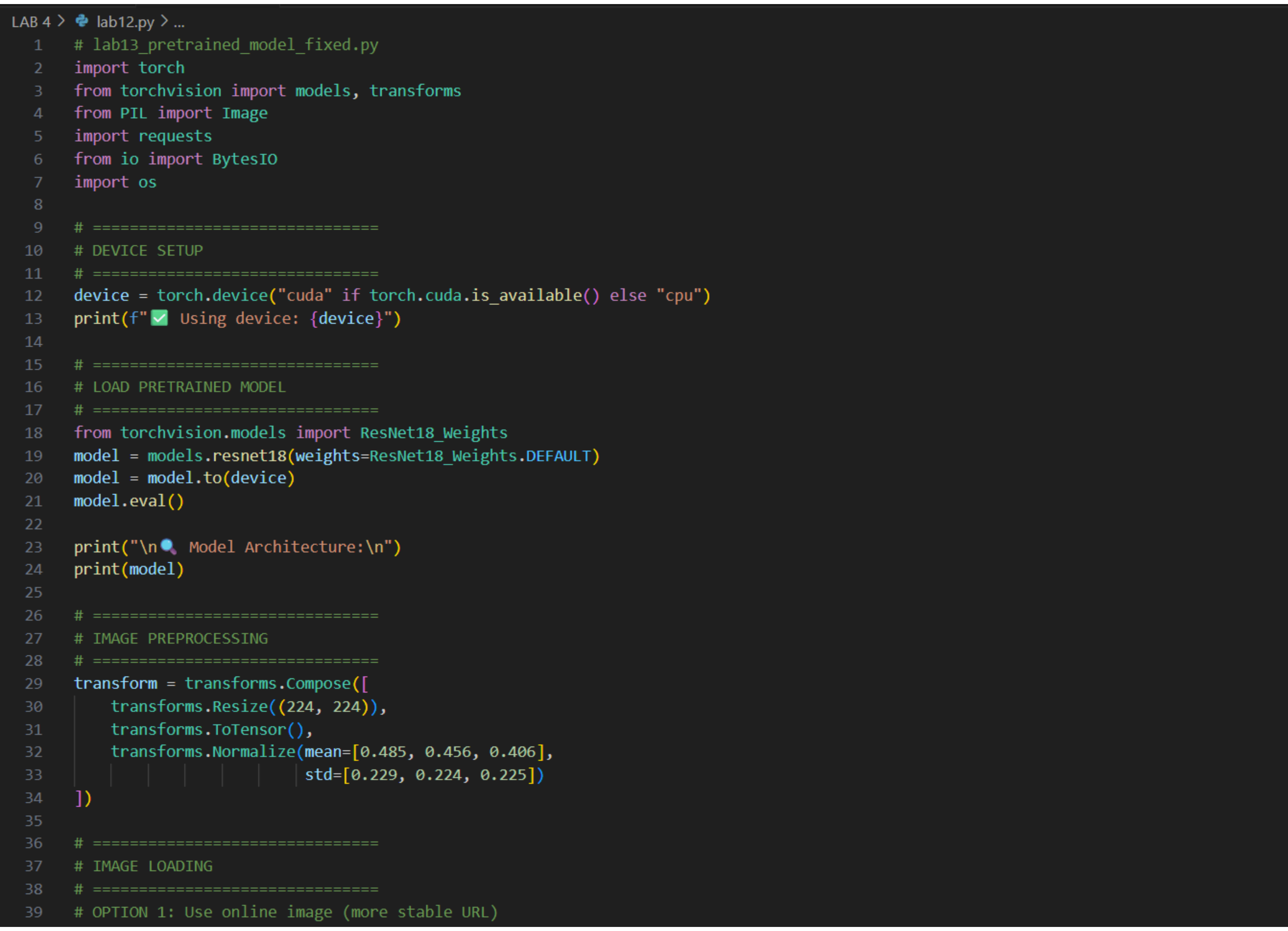
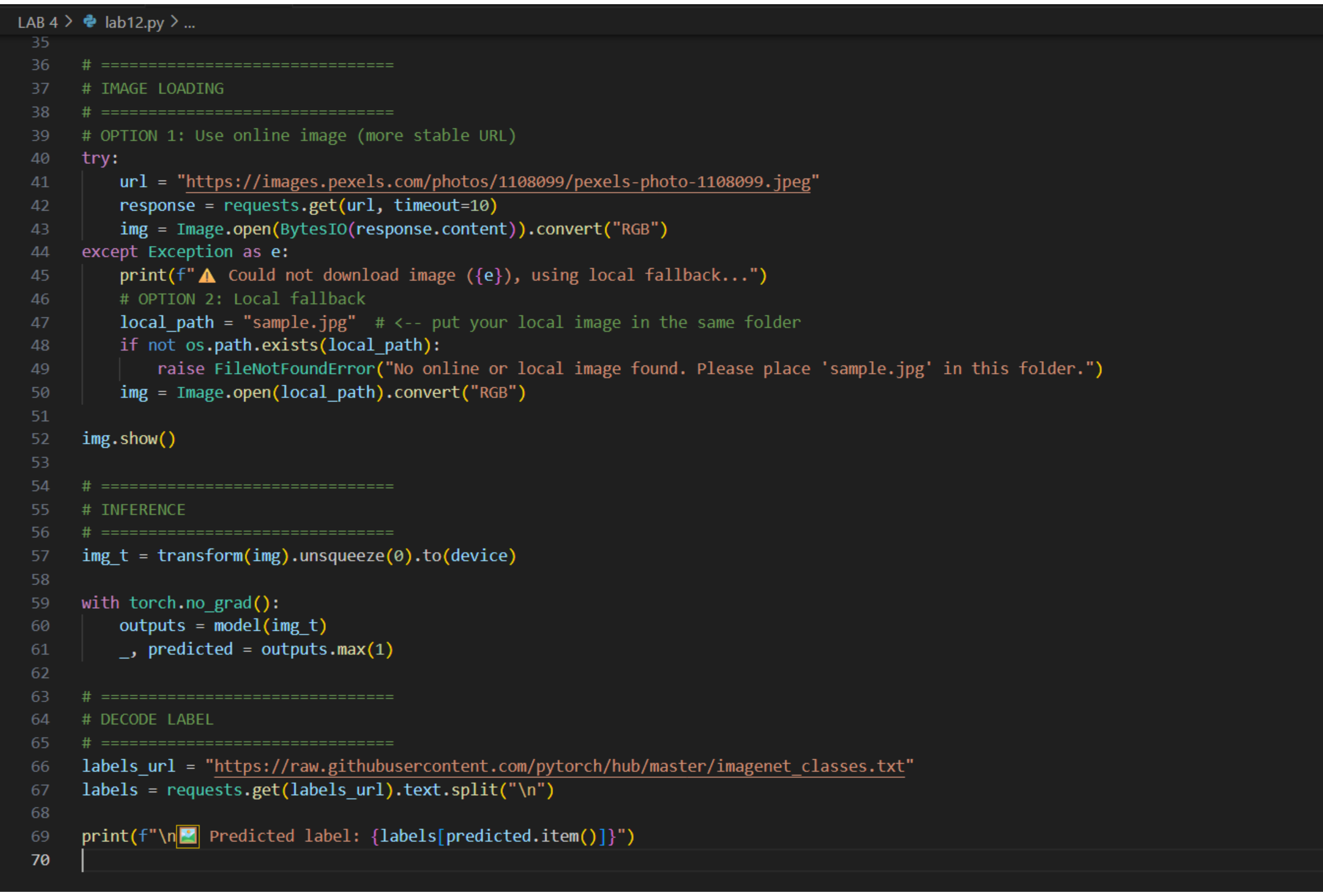

LAB 4 > lab12.py > ...

```python
# lab13_pretrained_model_fixed.py
import torch
from torchvision import models, transforms
from PIL import Image
import requests
from io import BytesIO
import os

# ==============================
# DEVICE SETUP
# ==============================
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"✅ Using device: {device}")

# ==============================
# LOAD PRETRAINED MODEL
# ==============================
from torchvision.models import ResNet18_Weights
model = models.resnet18(weights=ResNet18_Weights.DEFAULT)
model = model.to(device)
model.eval()

print("\n🔍 Model Architecture:\n")
print(model)

# ==============================
# IMAGE PREPROCESSING
# ==============================
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# ==============================
# IMAGE LOADING
# ==============================
# OPTION 1: Use online image (more stable URL)
```


LAB 4 > lab12.py > ...

```python

# ==============================
# IMAGE LOADING
# ==============================
# OPTION 1: Use online image (more stable URL)
try:
    url = "https://images.pexels.com/photos/1108099/pexels-photo-1108099.jpeg"
    response = requests.get(url, timeout=10)
    img = Image.open(BytesIO(response.content)).convert("RGB")
except Exception as e:
    print(f"⚠ Could not download image ({e}), using local fallback...")
    # OPTION 2: Local fallback
    local_path = "sample.jpg"  # <-- put your local image in the same folder
    if not os.path.exists(local_path):
        raise FileNotFoundError("No online or local image found. Please place 'sample.jpg' in this folder.")
    img = Image.open(local_path).convert("RGB")

img.show()

# ==============================
# INFERENCE
# ==============================
img_t = transform(img).unsqueeze(0).to(device)

with torch.no_grad():
    outputs = model(img_t)
    _, predicted = outputs.max(1)

# ==============================
# DECODE LABEL
# ==============================
labels_url = "https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt"
labels = requests.get(labels_url).text.split("\n")

print(f"\n🖼 Predicted label: {labels[predicted.item()]}")

```