```
import torch
from torchvision import models, transforms
from PIL import Image
from google.colab import files

# Upload image
print("Please upload your image:")
uploaded = files.upload()
image_path = list(uploaded.keys())[0]  # Get the uploaded image's name

# Load pretrained ResNet18 model
model = models.resnet18(pretrained=True)
#Purpose: Loads a pretrained ResNet18 model from torchvision.
#Explanation:
#resnet18 is a convolutional neural network with 18 layers, pretrained on
ImageNet (a large dataset with 1,000 classes).
#pretrained=True downloads the weights trained on ImageNet.
#The model is ready to classify images into one of the 1,000 ImageNet
classes.
model.eval()  # Set to evaluation mode
#In PyTorch, models have two modes: training and evaluation.
#eval() disables layers like dropout and batch normalization, which behave
differently during training, ensuring consistent predictions.

# Define image preprocessing
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225])
])
#transforms.Compose chains multiple transformations into a single
pipeline.
#transforms.Resize((224, 224)): Resizes the image to 224x224 pixels, the
input size expected by ResNet18.
#transforms.ToTensor(): Converts the PIL image (with pixel values in [0,
255]) to a PyTorch tensor (with values in [0, 1]) and changes the format
from (H, W, C) to (C, H, W).
#transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]): Normalizes the tensor's RGB channels using ImageNet's mean and
standard deviation, ensuring the input matches the data distribution the
model was trained on.
try:
  #This ensures the program doesn't crash if something goes wrong (e.g.,
invalid image file).
```

```
    # Load and preprocess image
    image = Image.open(image_path).convert("RGB")  # Ensure RGB format
    input_tensor = preprocess(image).unsqueeze(0)  # Add batch dimension

    # Make prediction
    with torch.no_grad():
        output = model(input_tensor)
        predicted_class = torch.argmax(output, dim=1).item()
        #torch.argmax(output, dim=1): Finds the index of the highest score
along dimension 1 (the class dimension), returning a tensor of shape (1,).

    # Output result
    print(f"Predicted class index: {predicted_class}")

except FileNotFoundError:
    print("Error: Image could not be loaded")
except Exception as e:
    print(f"Error: {str(e)}")
```