This code connects to my pre-existing neural network model that I have created. It feeds in users' inputs into the model and evaluates them.

```
import torch
from torchvision import transforms
import torch.nn as nn
import torch.nn.functional as F
from PIL import Image
from tkinter import *
from PIL import Image, ImageDraw
from tkinter import Tk
from PIL import Image
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, 64)
        self.fc4 = nn.Linear(64, 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return F.log_softmax(x, dim=1)

def save():
    image.save("image.png")
    img = Image.open("image.png")
    resized_img = img.resize((28,28))
    resized_img.save("resized.png")

def evaluate():
    save()
    img = Image.open("resized.png")
    path = "model/training.pt"


    model = Net()
    model.load_state_dict(torch.load(path))
    model.eval()
    # test

    transform = transforms.ToTensor()
    tensor_array = transform(img) # this is a tensor

    with torch.no_grad():
        x = tensor_array
        output = model(x.view(-1,784))
        for i in enumerate(output):

            widget = Label(canvas, text=f'Predicted: {torch.argmax(i[1])}', fg='black', bg='white')
            widget.place(x=5,y=280)

            print(torch.argmax(i[1]),) # Should print out what number it thinks.
            break

def draw(arg):
    x,y,x1,y1 = (arg.x-1), (arg.y-1), (arg.x+1), (arg.y+1)
    canvas.create_oval(x,y,x1,y1, fill="white",width=30)
    draw.line([x,y,x1,y1],fill="white",width=30)
    evaluate()

def clear():
    canvas.delete("all")
    draw.rectangle((0,0,500,500),"black")
    save()

width, height = 300,300
app = Tk()

canvas = Canvas(app,bg="white",width=width,height=height)
canvas.pack(expand=YES,fill=BOTH)
canvas.bind("<B1-Motion>", draw)


image = Image.new("RGB", (width,height), (0,0,0))
draw = ImageDraw.Draw(image)

button=Button(text="Evaluate",command=evaluate)
button.pack()
button=Button(text="Clear",command=clear)
button.pack()


app.mainloop()
```

This code is used to create the actual model for the main program.

```
import matplotlib.pyplot as plt
import torch
from torchvision import transforms, datasets
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from PIL import Image
```

```
# Download files (requires internet connection)
train = datasets.MNIST('', train=True,
download=True,transform=transforms.Compose([transforms.ToTensor()]))
test = datasets.MNIST('', train=False,
download=True,transform=transforms.Compose([transforms.ToTensor()]))
trainset = torch.utils.data.DataLoader(train, batch_size=100,
shuffle=True)
testset = torch.utils.data.DataLoader(test, batch_size=100,
shuffle=False)

path = "model/training.pt"

# Predefined model
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(28*28, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, 64)
        self.fc4 = nn.Linear(64, 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return F.log_softmax(x, dim=1)

net = Net()
print(net) # Prints the neural net layout

model = Net()
model.load_state_dict(torch.load(path))
model.eval()

loss_function = nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

cout = int()
class Trainer():

    # Changing the range will affect the accuracy of the program.
    for epoch in range(100):
        for data in trainset:
            X, y = data
            net.zero_grad()
            output = net(X.view(-1,784)
            loss = F.nll_loss(output, y)
            loss.backward()
            optimizer.step()
        print(f"{loss} \n Saved! Ran {cout} time(s)")  # the higher the loss, the better
        torch.save(net.state_dict(),path)
        cout+=1
```