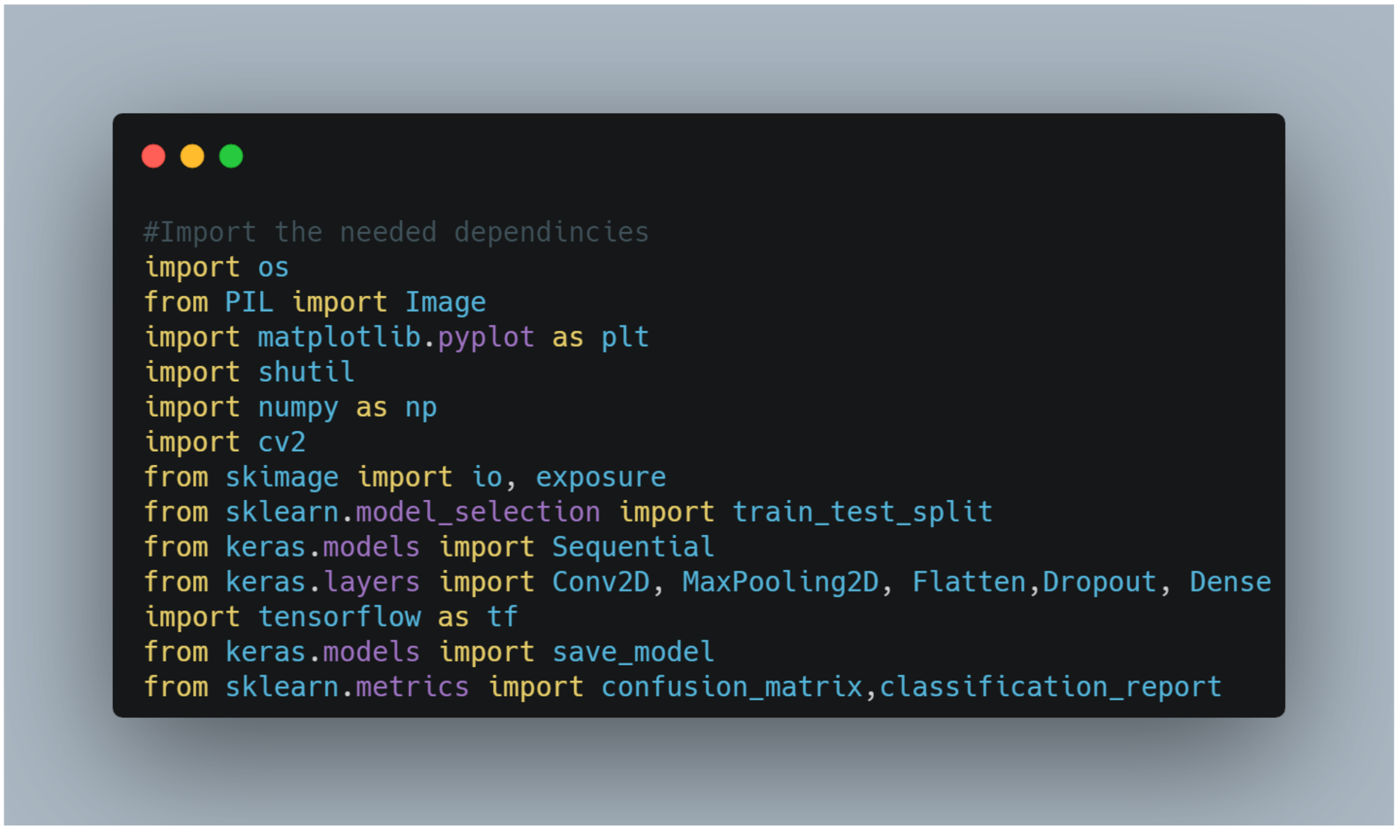
blnkOCR-Digits

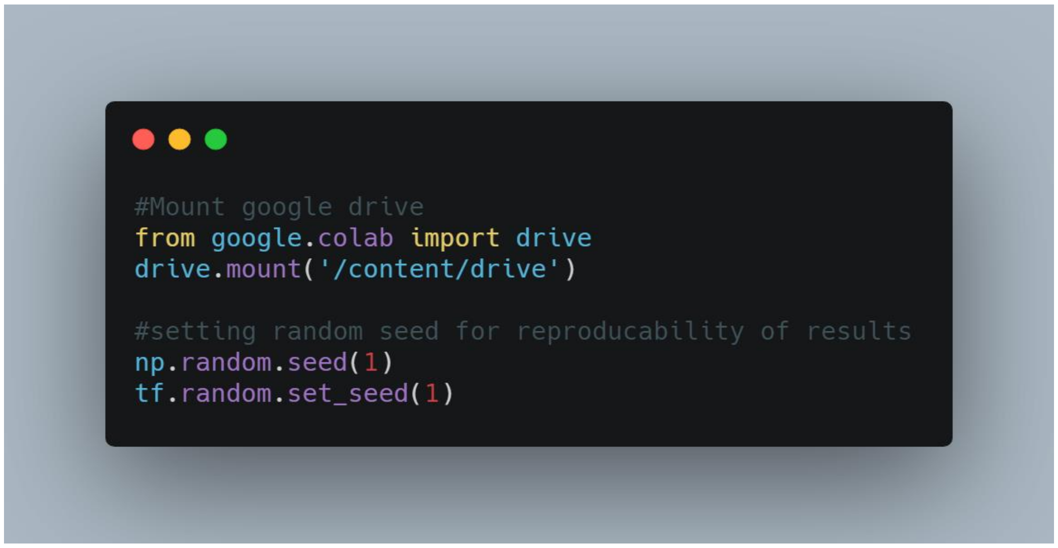
• Import Dependencies



```python
#Import the needed dependincies
import os
from PIL import Image
import matplotlib.pyplot as plt
import shutil
import numpy as np
import cv2
from skimage import io, exposure
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten,Dropout, Dense
import tensorflow as tf
from keras.models import save_model
from sklearn.metrics import confusion_matrix,classification_report
```

1- os is used in creating directory, joining paths of directories and file names.
2- Image is used to open images using the file path as input.
3- Matplotlib is used for plotting the digit images and plotting the confusion matrix of the test set.
4- shutil is used to move the images from their directory to the new directory that will contain all the images.
5- numpy is used to convert lists to np.array.
6- cv2 is used to convert the BGRA to grayscale.
7- skimage is used in normalizaing images and contrast enhancements of images.
8- sklearn.model_selection is used to split the images and labels to train and test labels.
9- keras is used to import the layers to create our model.
10- tensorflow is imported to generate random seed for reproducibility of results.
11- keras.models to save our model to reuse it again while predicting new images.
12- sklearn.metrics is used to create confusion matrix and classification report for testset results.

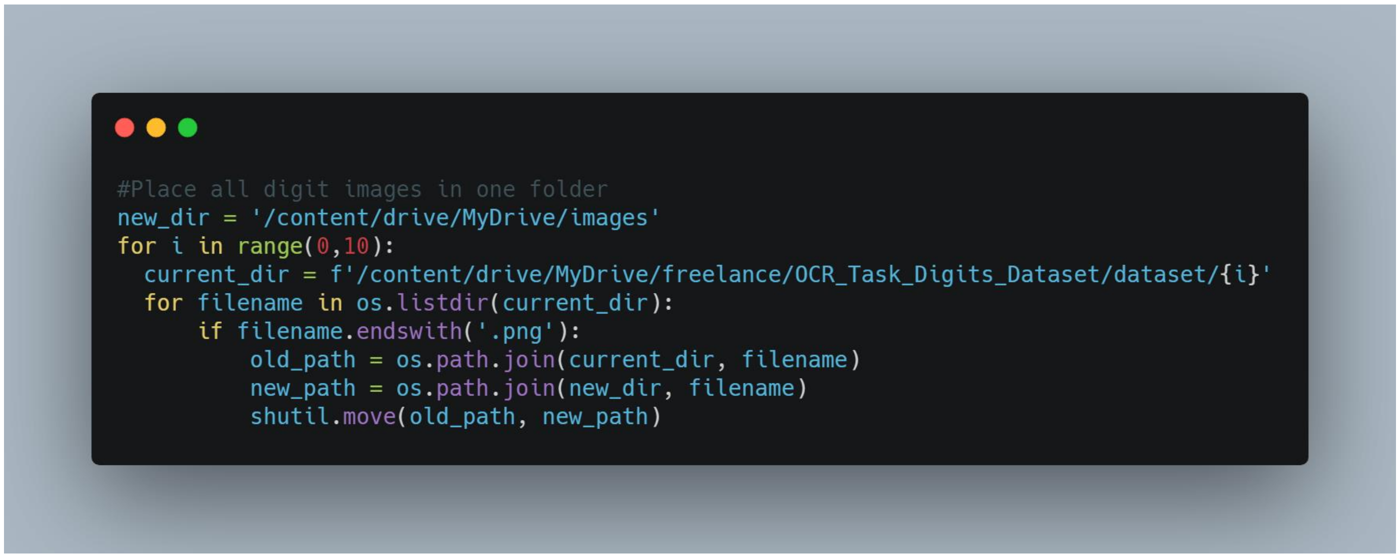
• Mounting Drive & creating seed



```
#Mount google drive
from google.colab import drive
drive.mount('/content/drive')

#setting random seed for reproducability of results
np.random.seed(1)
tf.random.set_seed(1)
```

-Mounting google drive to use it as the data source.
-np.random.seed sets the random seed for the NumPy random number generator.
-tf.random.set_seed sets the random seed for the TensorFlow random number generator.
-Using random seed for reproducibility of results.

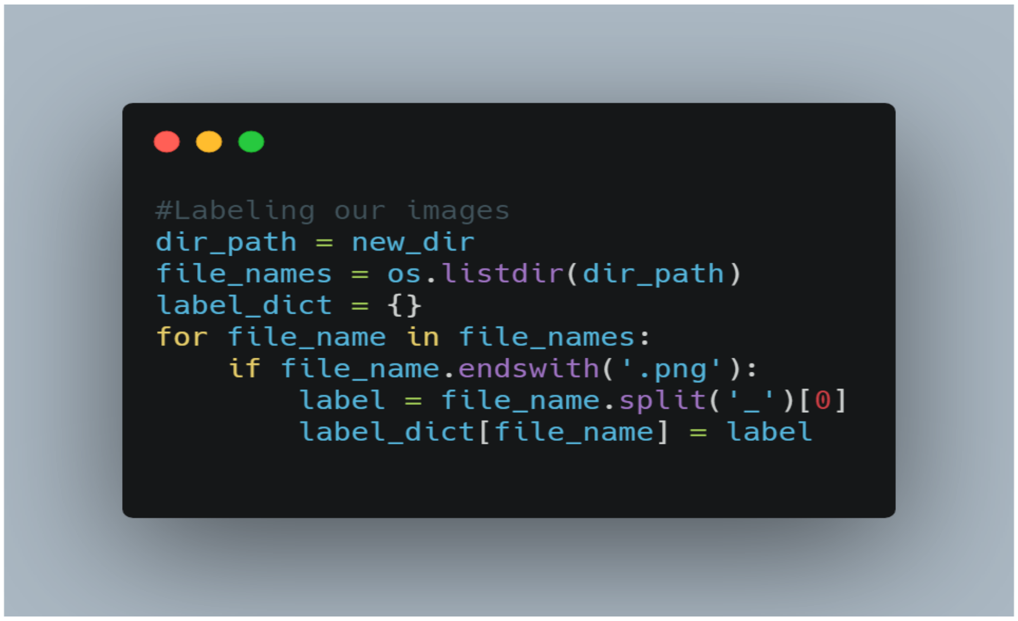
• Joining images directories



```
#Place all digit images in one folder
new_dir = '/content/drive/MyDrive/images'
for i in range(0,10):
  current_dir = f'/content/drive/MyDrive/freelance/OCR_Task_Digits_Dataset/dataset/{i}'
  for filename in os.listdir(current_dir):
      if filename.endswith('.png'):
          old_path = os.path.join(current_dir, filename)
          new_path = os.path.join(new_dir, filename)
          shutil.move(old_path, new_path)
```

-We have 10 directories each directory contains images for a specific digit.
-We want to sum up all the images into one directory to preprocess and then applying our model on them.
-Firstly, we defined the path for the new directory.
-We iterate over our 10 directories and check images inside, then move the images from their current dir to the new dir.

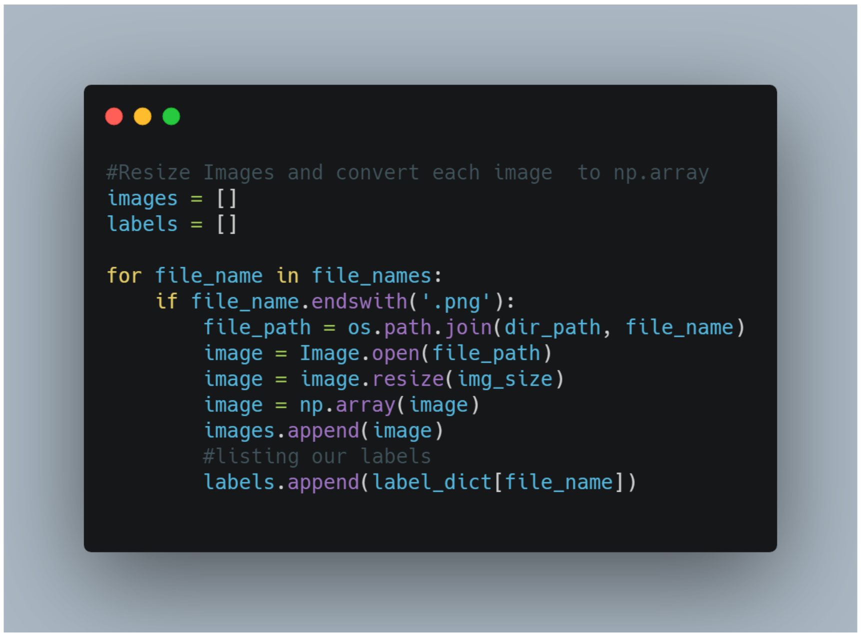
• Creating labels



```python
#Labeling our images
dir_path = new_dir
file_names = os.listdir(dir_path)
label_dict = {}
for file_name in file_names:
    if file_name.endswith('.png'):
        label = file_name.split('_')[0]
        label_dict[file_name] = label
```

-Iterate over every image in the directory and check that is a png image.
-We can see a pattern in our dataset that each image is named by its label then '_' then a number.
-Split the images names by '_' and taken the first partition as the image label.
-Create a dictionary that their keys are the image's names, and the values are the labels of the images.

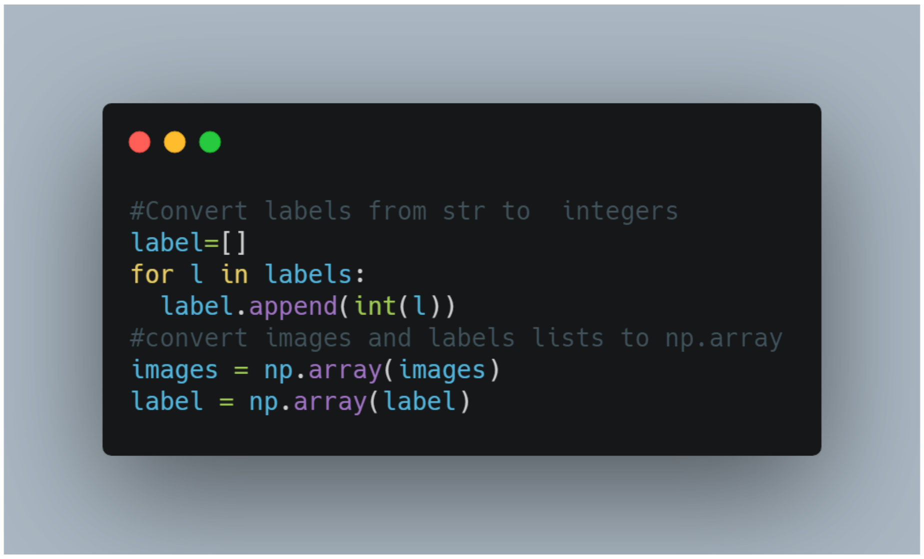
• Listing Images &labels



```python
#Resize Images and convert each image  to np.array
images = []
labels = []

for file_name in file_names:
    if file_name.endswith('.png'):
        file_path = os.path.join(dir_path, file_name)
        image = Image.open(file_path)
        image = image.resize(img_size)
        image = np.array(image)
        images.append(image)
        #listing our labels
        labels.append(label_dict[file_name])
```

-Open the images by their file paths and resize them to 28x28 (good size for CNN model).
-Convert the image pixels to np.array, create list for images and append all the images in the images list.
-Iterate over the image's labels and appending them in the labels list.

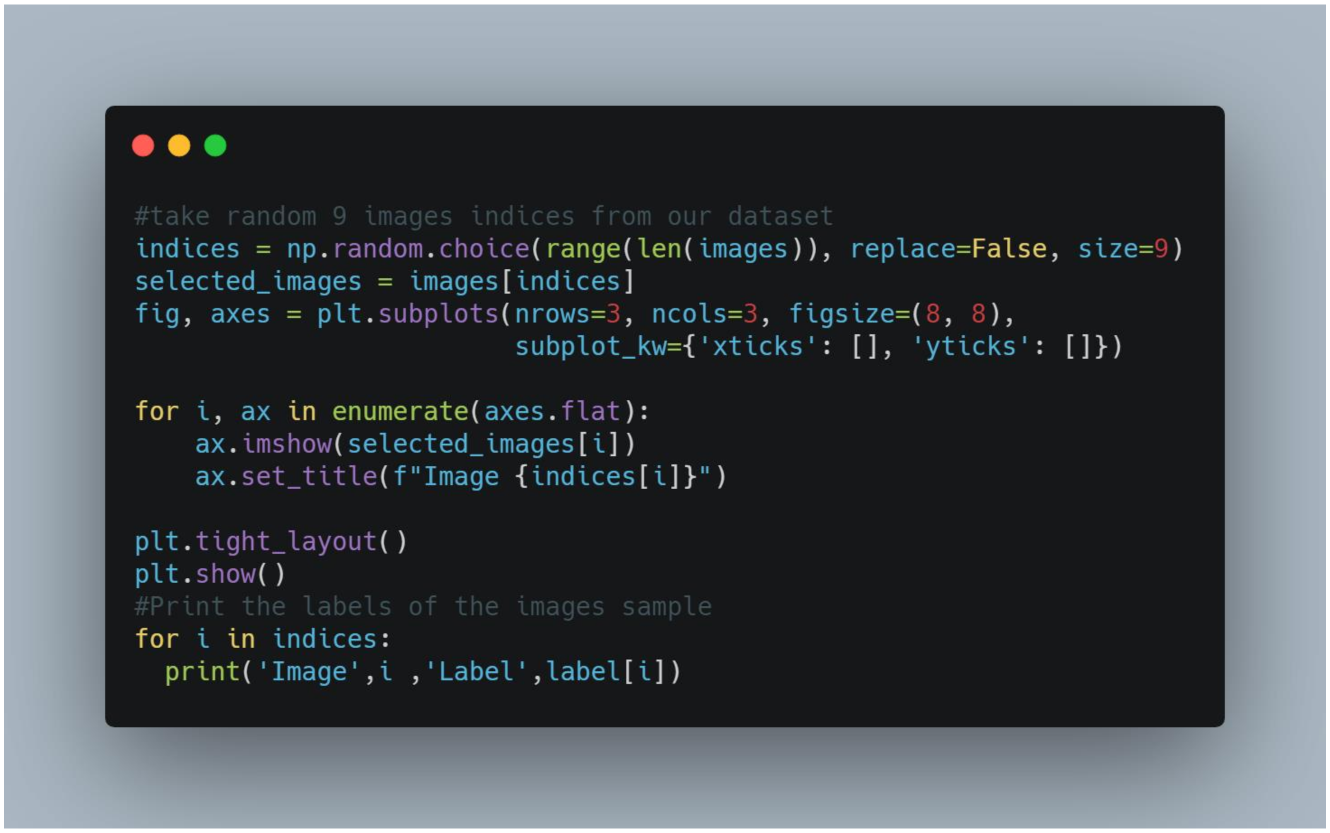
•Modifying the lists



```python
#Convert labels from str to  integers
label=[]
for l in labels:
  label.append(int(l))
#convert images and labels lists to np.array
images = np.array(images)
label = np.array(label)
```

-Convert the images and labels list to np.array because only integer scalar arrays can be converted to a scalar index.
-Convert the labels from string to integer.

•Plot Sample of Images

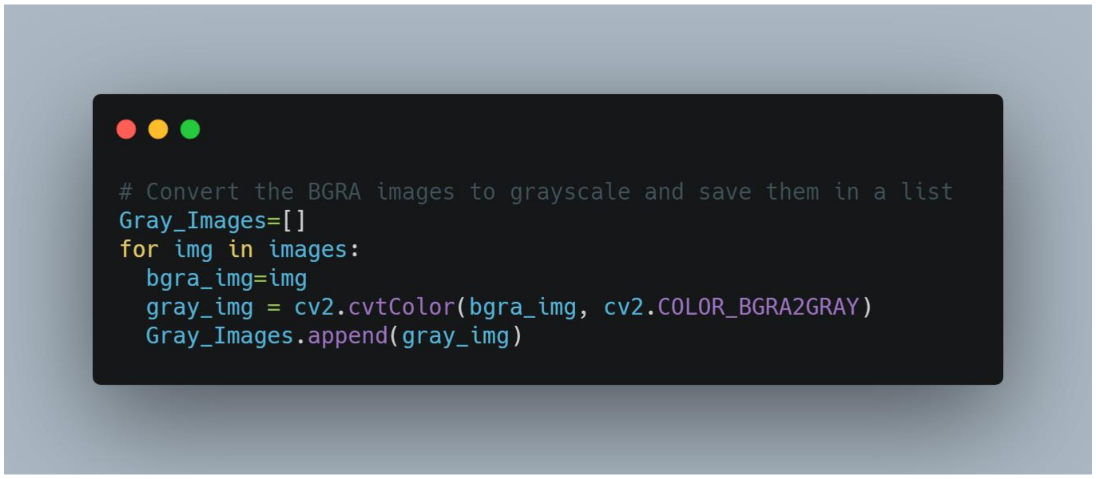


```python
#take random 9 images indices from our dataset
indices = np.random.choice(range(len(images)), replace=False, size=9)
selected_images = images[indices]
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8),
                         subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(selected_images[i])
    ax.set_title(f"Image {indices[i]}")

plt.tight_layout()
plt.show()
#Print the labels of the images sample
for i in indices:
  print('Image',i ,'Label',label[i])
```

-Select 9 random images from the data set and plot them to see the images before preprocessing.
-Print the labels for the sample.

•Image Preprocessing

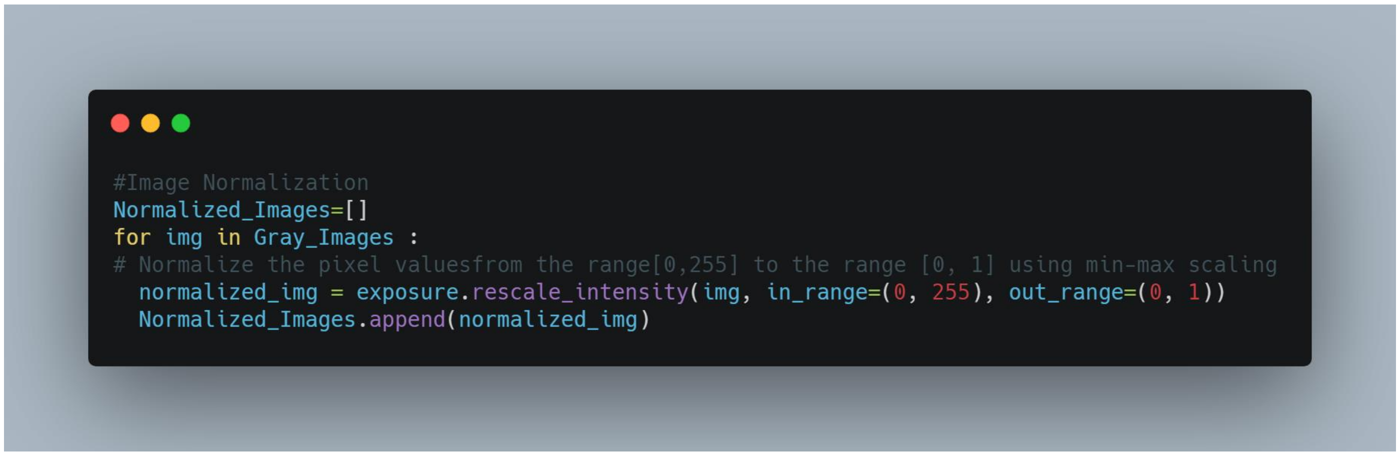
Convert to grayscale



```
# Convert the BGRA images to grayscale and save them in a list
Gray_Images=[]
for img in images:
  bgra_img=img
  gray_img = cv2.cvtColor(bgra_img, cv2.COLOR_BGRA2GRAY)
  Gray_Images.append(gray_img)
```

-Iterate over the images and convert from BGRA to grayscale images.
-Append the gray images in the Gray_Images list.

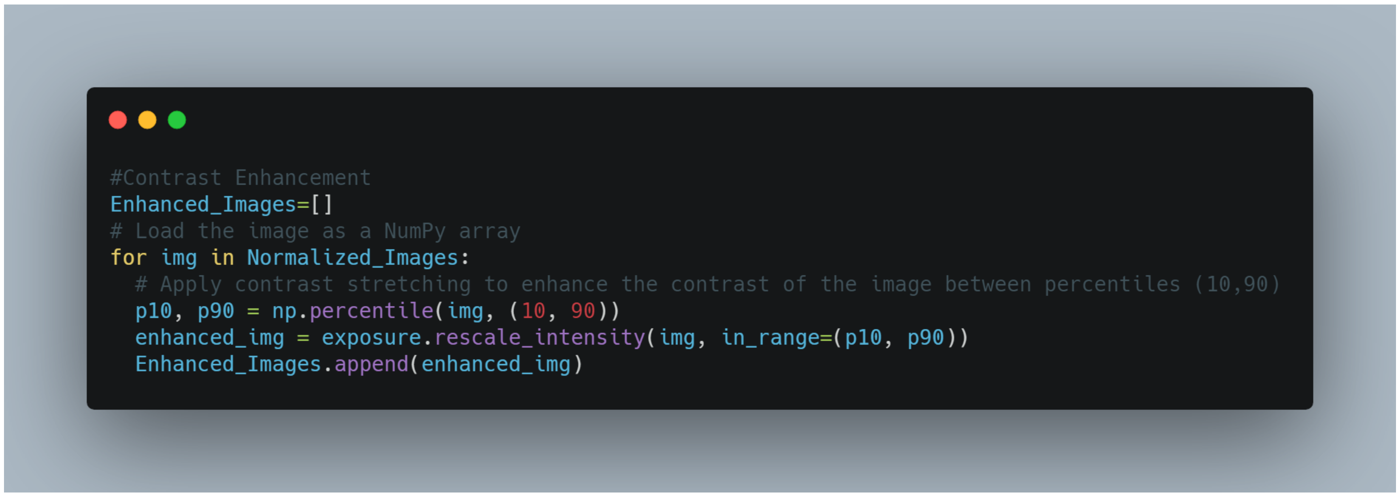
Normalization



```
#Image Normalization
Normalized_Images=[]
for img in Gray_Images :
# Normalize the pixel valuesfrom the range[0,255] to the range [0, 1] using min-max scaling
  normalized_img = exposure.rescale_intensity(img, in_range=(0, 255), out_range=(0, 1))
  Normalized_Images.append(normalized_img)
```

-Iterate over the gray images and normalize them from their range from (0,255) to be from 0 to 1.
-Append the normalized images in the Normalized_Images list.

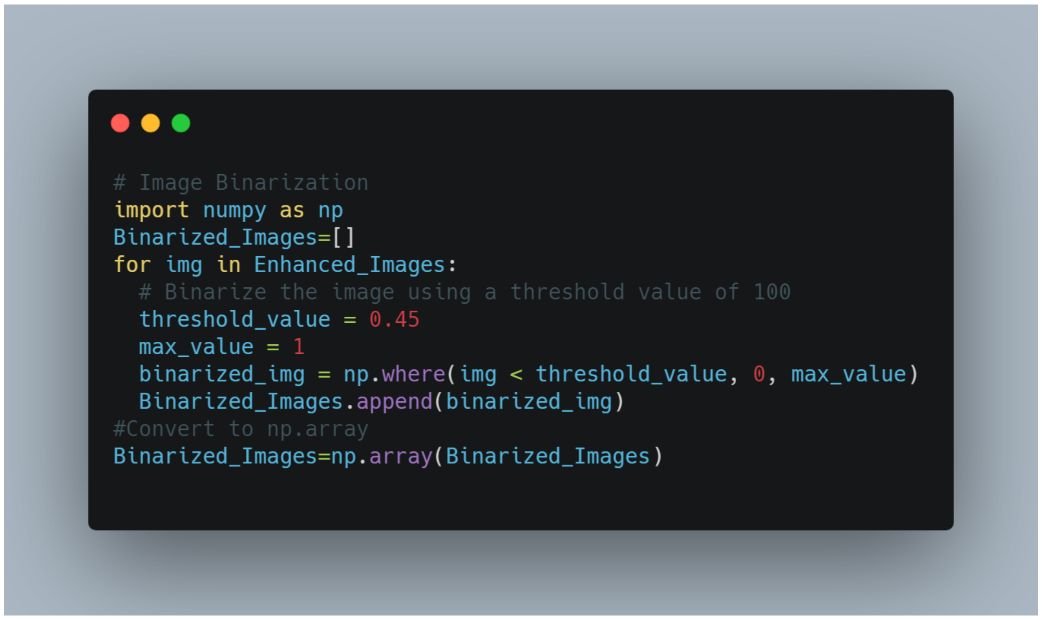
Contrast Enhancement



```python
#Contrast Enhancement
Enhanced_Images=[]
# Load the image as a NumPy array
for img in Normalized_Images:
  # Apply contrast stretching to enhance the contrast of the image between percentiles (10,90)
  p10, p90 = np.percentile(img, (10, 90))
  enhanced_img = exposure.rescale_intensity(img, in_range=(p10, p90))
  Enhanced_Images.append(enhanced_img)
```

-Iterate over the normalized images & enhance the contrast between 10th and 90th percentiles to improve images quality.
-Append the enhanced images in the Enhanced_Images list.

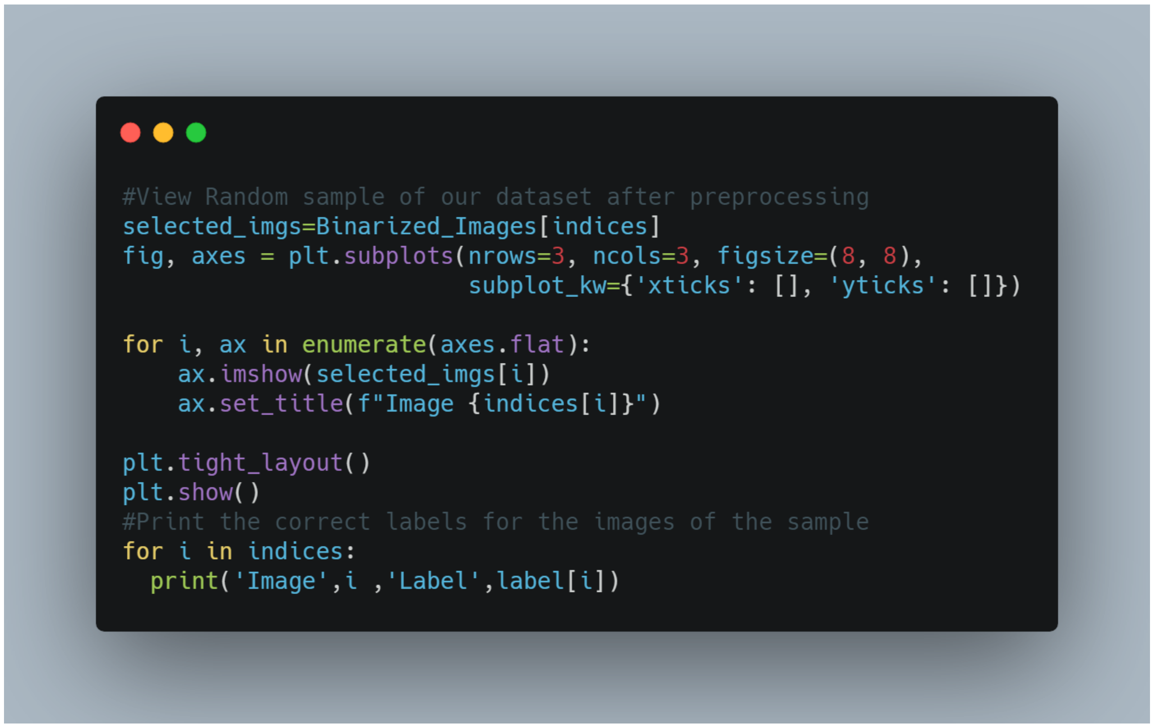
Image Binarization



```python
# Image Binarization
import numpy as np
Binarized_Images=[]
for img in Enhanced_Images:
  # Binarize the image using a threshold value of 100
  threshold_value = 0.45
  max_value = 1
  binarized_img = np.where(img < threshold_value, 0, max_value)
  Binarized_Images.append(binarized_img)
#Convert to np.array
Binarized_Images=np.array(Binarized_Images)
```

-Iterate over the enhanced images.
-Binarize the image with threshold 0.45 if pixel has value >0.45 then it will be 1 else it will be 0.
-Append the binarized images in the Binarized Images list.
-Binarized_Images list has the images after applying all the preprocessing (gray scaling, normalization, contrast enhancements, binarization).
-Convert the list to np.array.

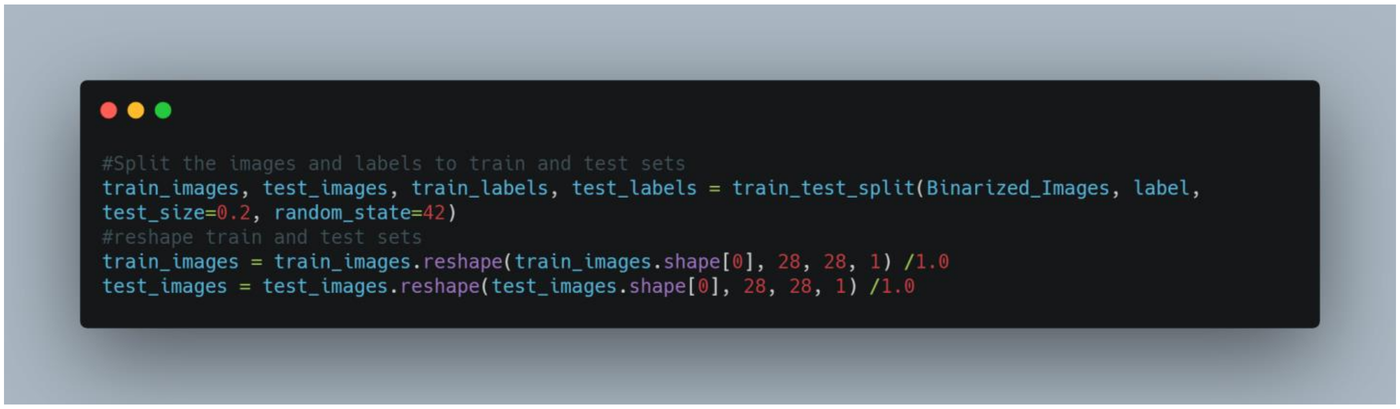
•Plot sample of images after preprocessing



```
#View Random sample of our dataset after preprocessing
selected_imgs=Binarized_Images[indices]
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(8, 8),
                         subplot_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):
    ax.imshow(selected_imgs[i])
    ax.set_title(f"Image {indices[i]}")

plt.tight_layout()
plt.show()
#Print the correct labels for the images of the sample
for i in indices:
  print('Image',i ,'Label',label[i])
```

-Plot random sample of images after preprocessing.
-Print the sample's labels.

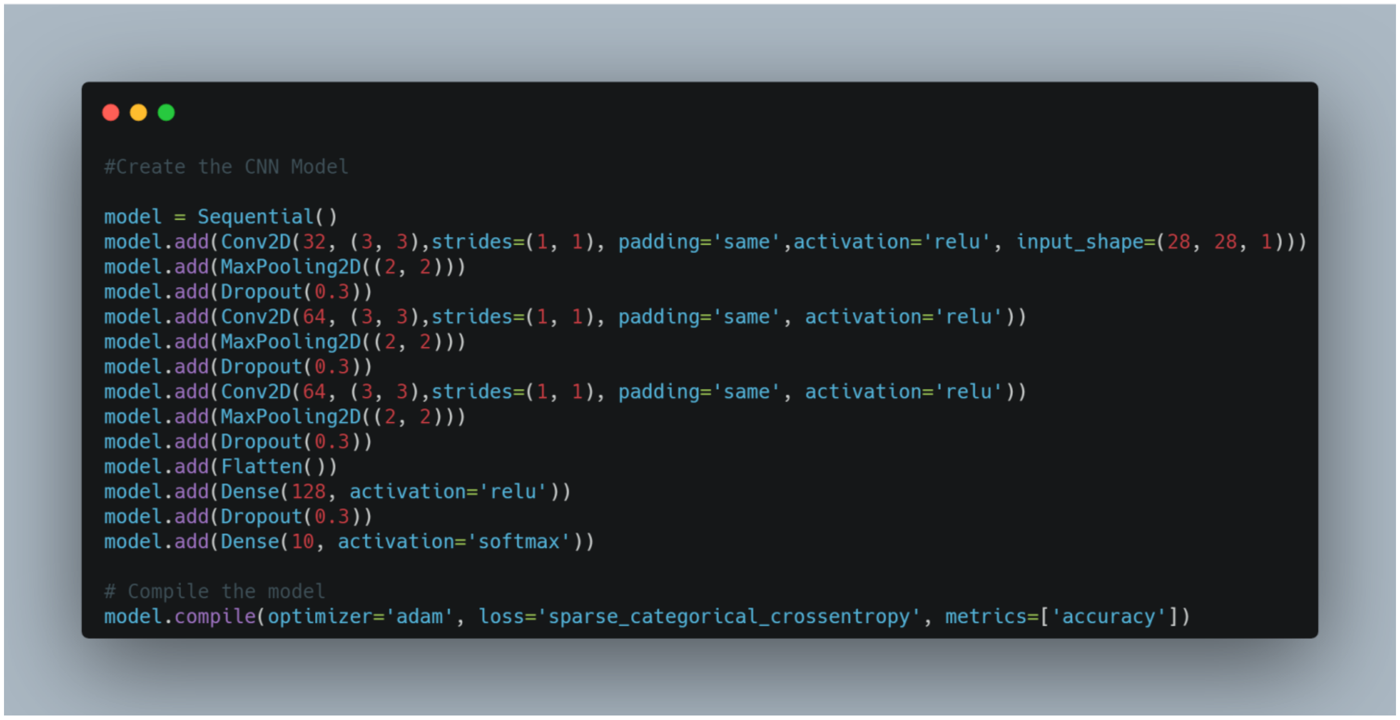
•Split and resize the dataset



```
#Split the images and labels to train and test sets
train_images, test_images, train_labels, test_labels = train_test_split(Binarized_Images, label,
test_size=0.2, random_state=42)
#reshape train and test sets
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1) /1.0
test_images = test_images.reshape(test_images.shape[0], 28, 28, 1) /1.0
```

-Split the dataset after preprocessing to train and test sets (80% training ,20%testing).
-random_state fixed to control randomness and for reproducibility of results.
-Reshaping the images in both train and test images:
-The first dimension represents the number of samples.
-The second and third dimensions (28, 28) represent the height and width of each image.
-The fourth dimension (1) represents the number of color channels in each image which is grayscale.
-We divide by 1.0 to get the image in black and white.

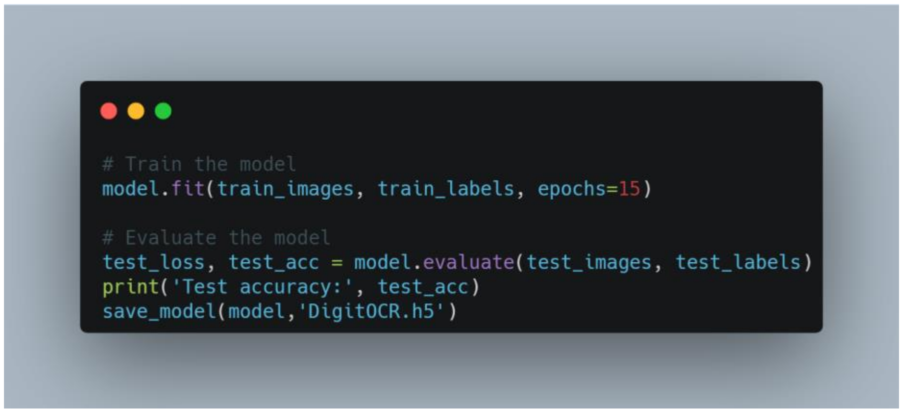
•Create CNN Model



```python
#Create the CNN Model

model = Sequential()
model.add(Conv2D(32, (3, 3),strides=(1, 1), padding='same',activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(64, (3, 3),strides=(1, 1), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(64, (3, 3),strides=(1, 1), padding='same', activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

-Create sequential model to add the need layers
-Convolution layer with 32 filter, size of filter is 3x3, strides = (1, 1): This specifies the step size of
the convolutional filter, padding='same' This specifies how the input is padded to ensure that the filter can be applied at the edges of the image. 'same' padding means that the output of the convolutional layer will have the same spatial dimensions as the input, activation ='relu' is used, which applies a simple thresholding operation to the output, setting any negative values to zero, input_shape (28,28,1) 28x28 is the size of image and 1 refers to 1 channel (grayscale).
-Maxpooling2d layer: Max pooling is a technique used in convolutional neural networks to reduce the spatial dimensions of the input feature maps while retaining the most important information.
-Dropout: regularization technique to avoid overfitting.
-We repeat the cycle twice again (Conv2d, Maxpooling, Dropout)
- Adding multiple Conv2D layers in a CNN model allows the network to learn increasingly complex and abstract features from the input data.
-Flatten layer is a type of layer in a neural network model that is used to convert the input data into a one-dimensional vector, and it is always the last layer in CNN model and the input for the fully connected layer.
-Dense layer is a type of layer in a neural network model that is used for fully connected layers. In a Dense layer, every neuron in the previous layer is connected to every neuron in the current layer, and each connection has a learnable weight
-Last layer is dense layer with 10 outputs that refers to the number of classes and with activation softmax that return the probability of the image for each class.
-Optimizer: This specifies the optimization algorithm used to update the weights of the neural network during training
-Loss: This specifies the loss function used to measure the difference between the predicted output of the model and the true target output during training.
-sparse_categorical_crossentropy is a common loss function used for multi-class classification problems where the target labels are integers.
-Metrics: This specifies the evaluation metric used to monitor the performance of the model during training and testing.

•Train and evaluate the model

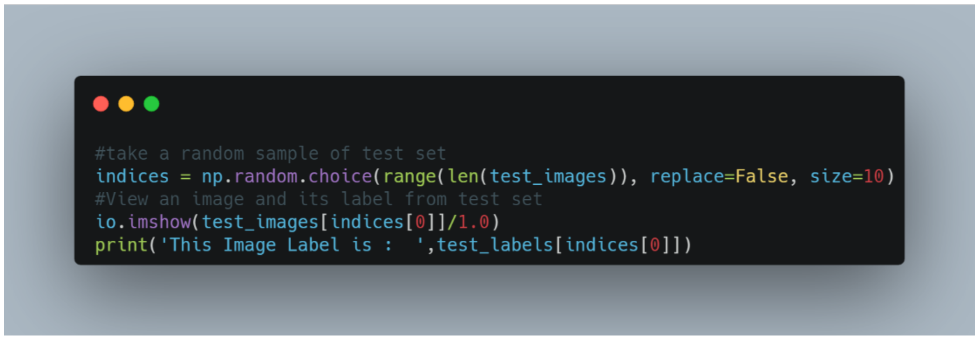


```
# Train the model
model.fit(train_images, train_labels, epochs=15)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
save_model(model,'DigitOCR.h5')
```

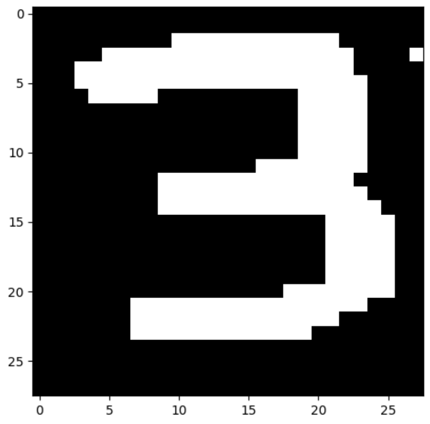
-model.fit is a function in Keras used to train a neural network model on a training dataset take the input argument train_images, and train labels and epochs is the number of complete pass through the entire dataset.
-model.evaluate :evaluate the trained model on the test dataset and return the test loss and the test accuracy.
-Save the model to reuse it again while predicting new images.

•View an image & a label from the test set

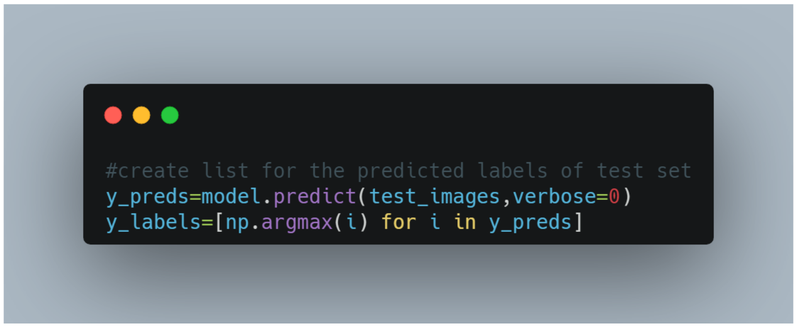
-Take a random image from test set and show it and print its label.



```
#take a random sample of test set
indices = np.random.choice(range(len(test_images)), replace=False, size=10)
#View an image and its label from test set
io.imshow(test_images[indices[0]]/1.0)
print('This Image Label is :  ',test_labels[indices[0]])
```



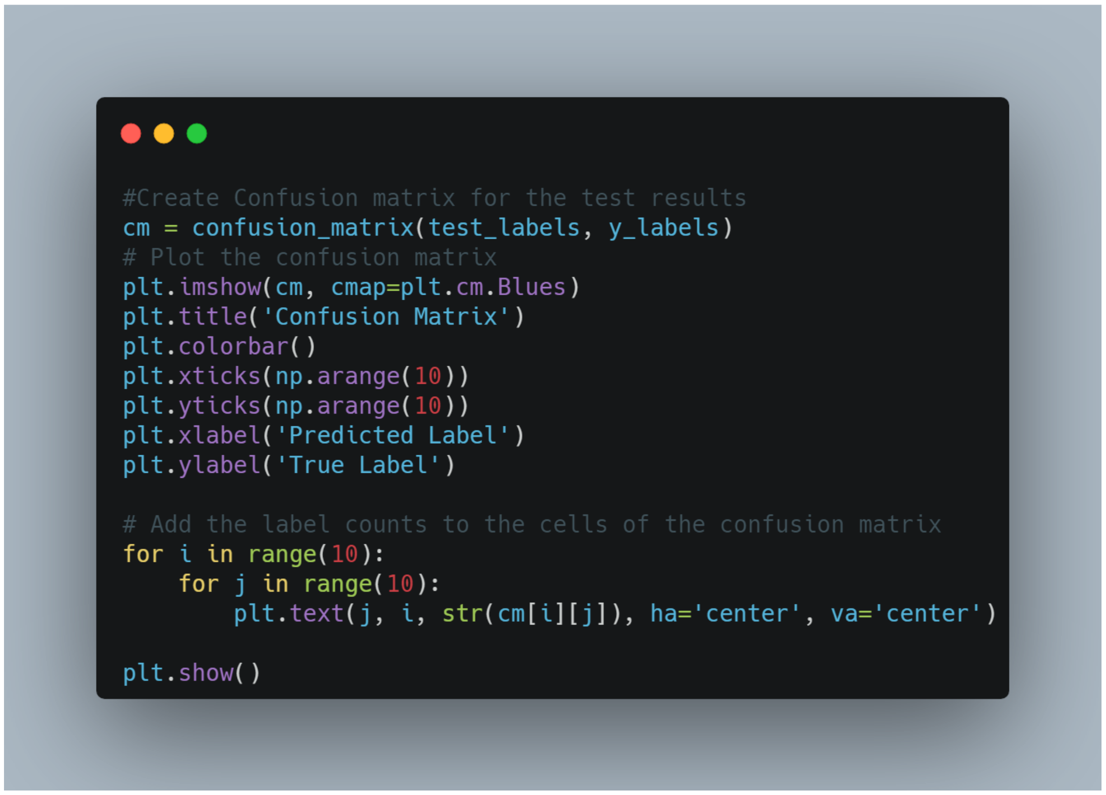
•Predict the test set images



```
#create list for the predicted labels of test set
y_preds=model.predict(test_images,verbose=0)
y_labels=[np.argmax(i) for i in y_preds]
```

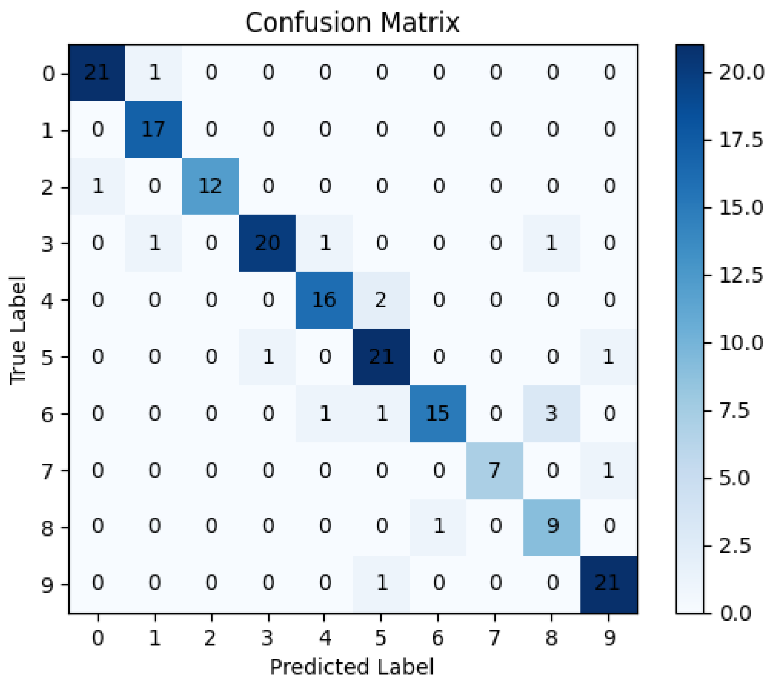
-Predict all the test set images.
-Save the results in list y_labels.

•Create and plot confusion matrix



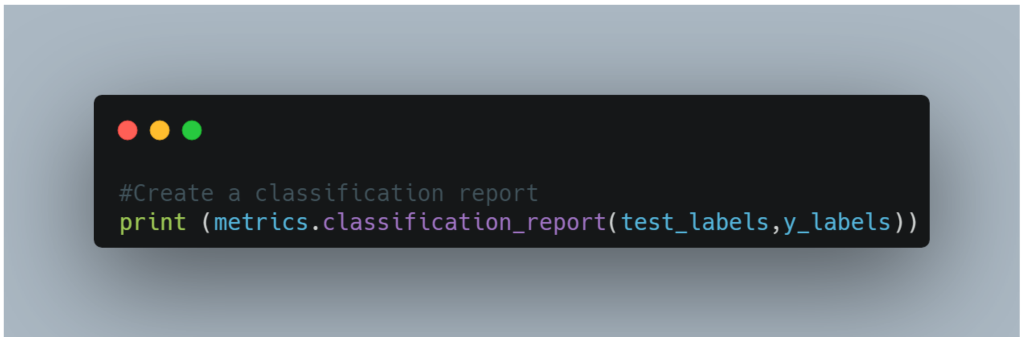
```
#Create Confusion matrix for the test results
cm = confusion_matrix(test_labels, y_labels)
# Plot the confusion matrix
plt.imshow(cm, cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xticks(np.arange(10))
plt.yticks(np.arange(10))
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

# Add the label counts to the cells of the confusion matrix
for i in range(10):
    for j in range(10):
        plt.text(j, i, str(cm[i][j]), ha='center', va='center')

plt.show()
```

-Create confusion matrix to compare between the predicted results and the actual results for the test set.



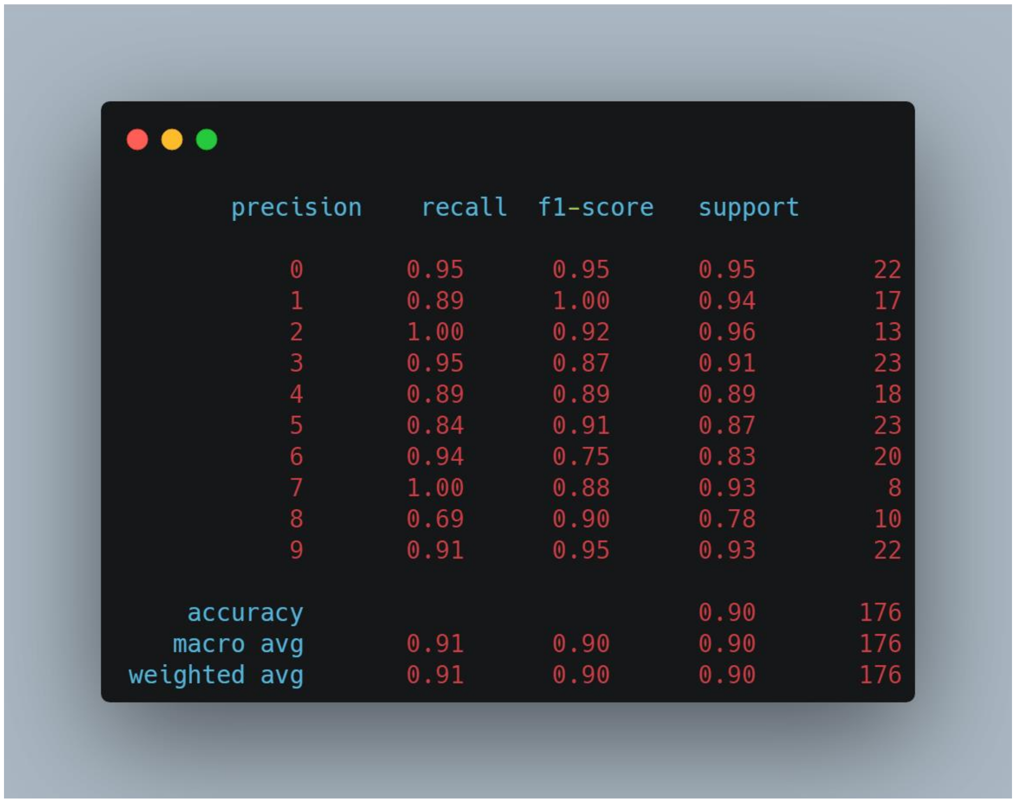
True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	21	1	0	0	0	0	0	0	0	0
1	0	17	0	0	0	0	0	0	0	0
2	1	0	12	0	0	0	0	0	0	0
3	0	1	0	20	1	0	0	0	1	0
4	0	0	0	0	16	2	0	0	0	0
5	0	0	0	1	0	21	0	0	0	1
6	0	0	0	0	1	1	15	0	3	0
7	0	0	0	0	0	0	0	7	0	1
8	0	0	0	0	0	0	1	0	9	0
9	0	0	0	0	0	1	0	0	0	21

•Create classification report



```
#Create a classification report
print (metrics.classification_report(test_labels,y_labels))
```

-**Output**:



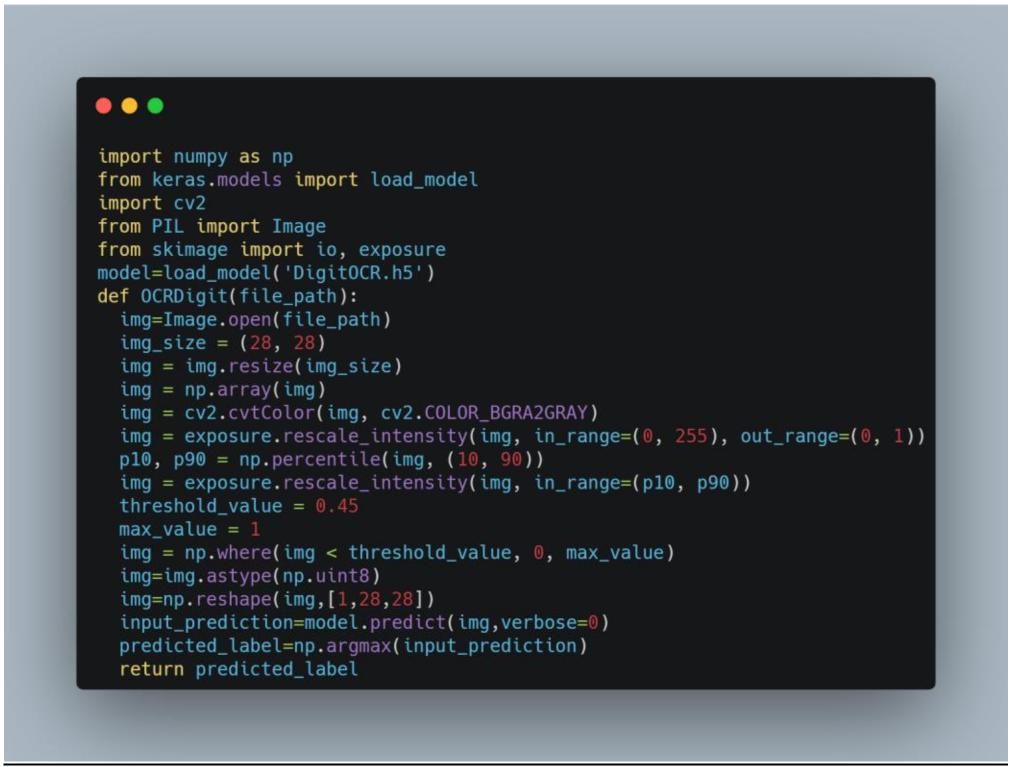
```
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        22
           1       0.89      1.00      0.94        17
           2       1.00      0.92      0.96        13
           3       0.95      0.87      0.91        23
           4       0.89      0.89      0.89        18
           5       0.84      0.91      0.87        23
           6       0.94      0.75      0.83        20
           7       1.00      0.88      0.93         8
           8       0.69      0.90      0.78        10
           9       0.91      0.95      0.93        22

    accuracy                           0.90       176
   macro avg       0.91      0.90      0.90       176
weighted avg       0.91      0.90      0.90       176
```

-As we can see the classification report gives us the precision, recall,F1-Score and accuracy .
-It is not the best results, but it is acceptable because the data set is small.

• Model prediction

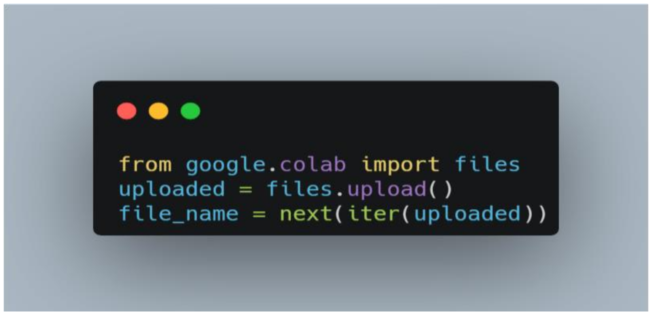
1- Using the Predict on OCR notebook.
2- Using the model deployed with Django.

-Predict by PredictonOCR notebook:



```python
import numpy as np
from keras.models import load_model
import cv2
from PIL import Image
from skimage import io, exposure
model=load_model('DigitOCR.h5')
def OCRDigit(file_path):
  img=Image.open(file_path)
  img_size = (28, 28)
  img = img.resize(img_size)
  img = np.array(img)
  img = cv2.cvtColor(img, cv2.COLOR_BGRA2GRAY)
  img = exposure.rescale_intensity(img, in_range=(0, 255), out_range=(0, 1))
  p10, p90 = np.percentile(img, (10, 90))
  img = exposure.rescale_intensity(img, in_range=(p10, p90))
  threshold_value = 0.45
  max_value = 1
  img = np.where(img < threshold_value, 0, max_value)
  img=img.astype(np.uint8)
  img=np.reshape(img,[1,28,28])
  input_prediction=model.predict(img,verbose=0)
  predicted_label=np.argmax(input_prediction)
  return predicted_label
```

1-You should have the 'DigitOCR' model in your directory.
2-import the dependencies needed in image preprocessing only.
3-load the model we just created ('DigitOCR.h5').
4-Create a function that take the file path as an input.
5-This function preprocess the image and then we apply the model after preprocessing the image.
6-the output of the softmax function is probability for each class that the image belong to it.
7-Use argmax to know which class has the biggest probability that the image belong to it.
8-return the predicted label.



```python
from google.colab import files
uploaded = files.upload()
file_name = next(iter(uploaded))
```

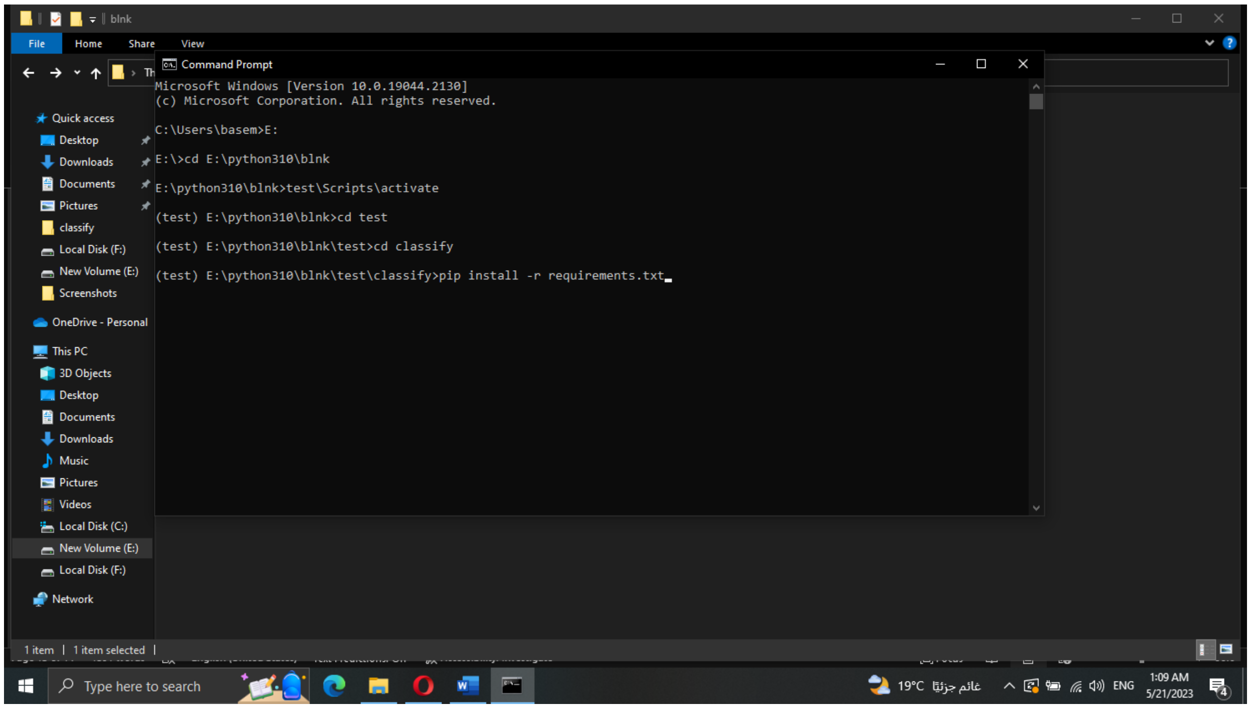
-This code is used to create 'choose file' button that can browse and choose an image to upload.
-Image's file path will be saved in file_name.
-Now when we call our function (OCRDigit) and pass the file_name :OCRDigit(file_name) it will return the predicted label.

-Predict by the model deployed with django:

-Firstly, I created a virtual environment
-Install the required dependencies from requirements.txt
-Create a Django project (django-admin createproject classify)
-Create a Django app(django-admin createapp classify_app)
-Create directory template that contain the index.html
-Copy our model 'OCRDigit.h5' in the same directory
-Open classify>setting.py, add the 'classify_app' in the installed apps, add 'template' in 'DIRS' in TEMPLATES
Add (STATICFILES_DIRS, STATIC_URL, MEDIA_URL, MEDIA_ROOT) at the end of the setting.py
-Open classify>urls.py, from classify_app import views
-Open classify_app>views.py, install the needed dependencies, load the model, create function makepredictions that take the file path as input and preprocess the image then return the predicted digit, create function index that take the request as an input to deal with the requests,it returns the predictions and file url if there is no errors,
it renders error message if no image selected or no file selected, else it will return the html file.
-Open template>index.html, write basic html code to have a choose file button to upload image and submit button to perform the prediction function and view the filename and the predicted digit

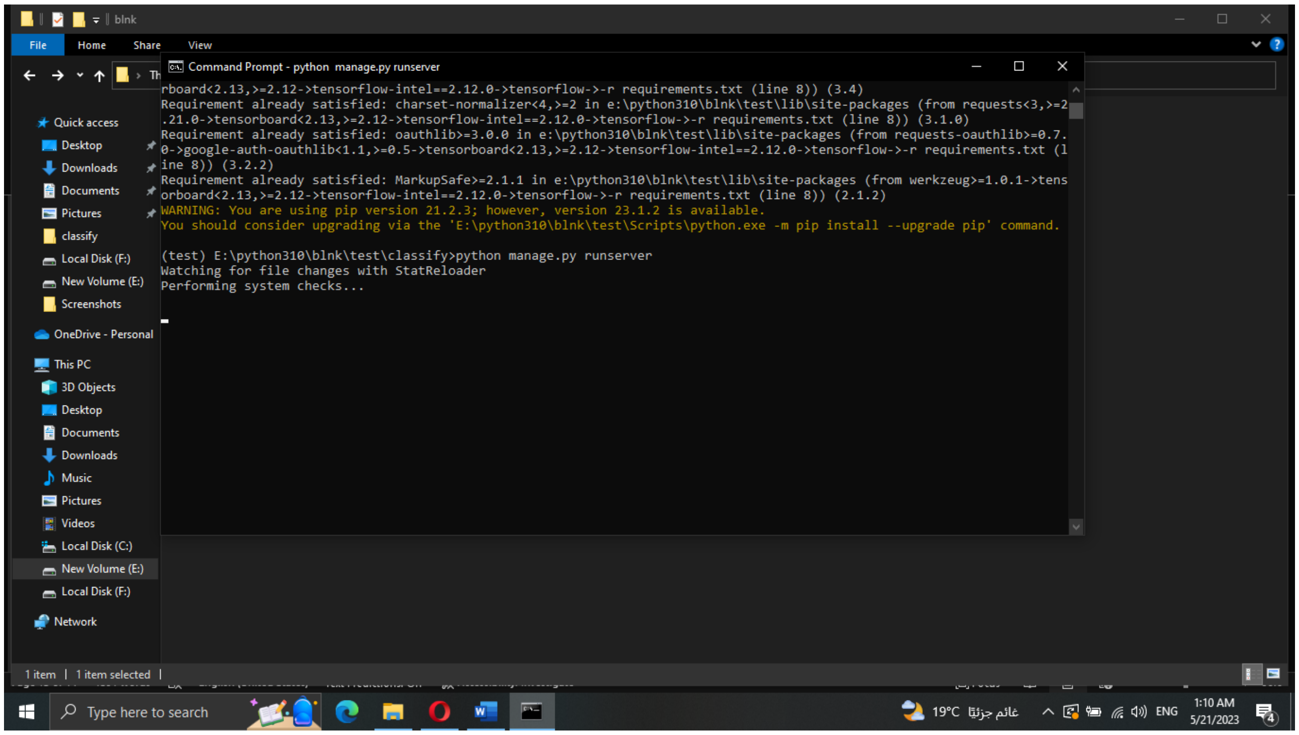
Steps to predict:

1-Extract classify.rar
2-pip install -r requirements.txt



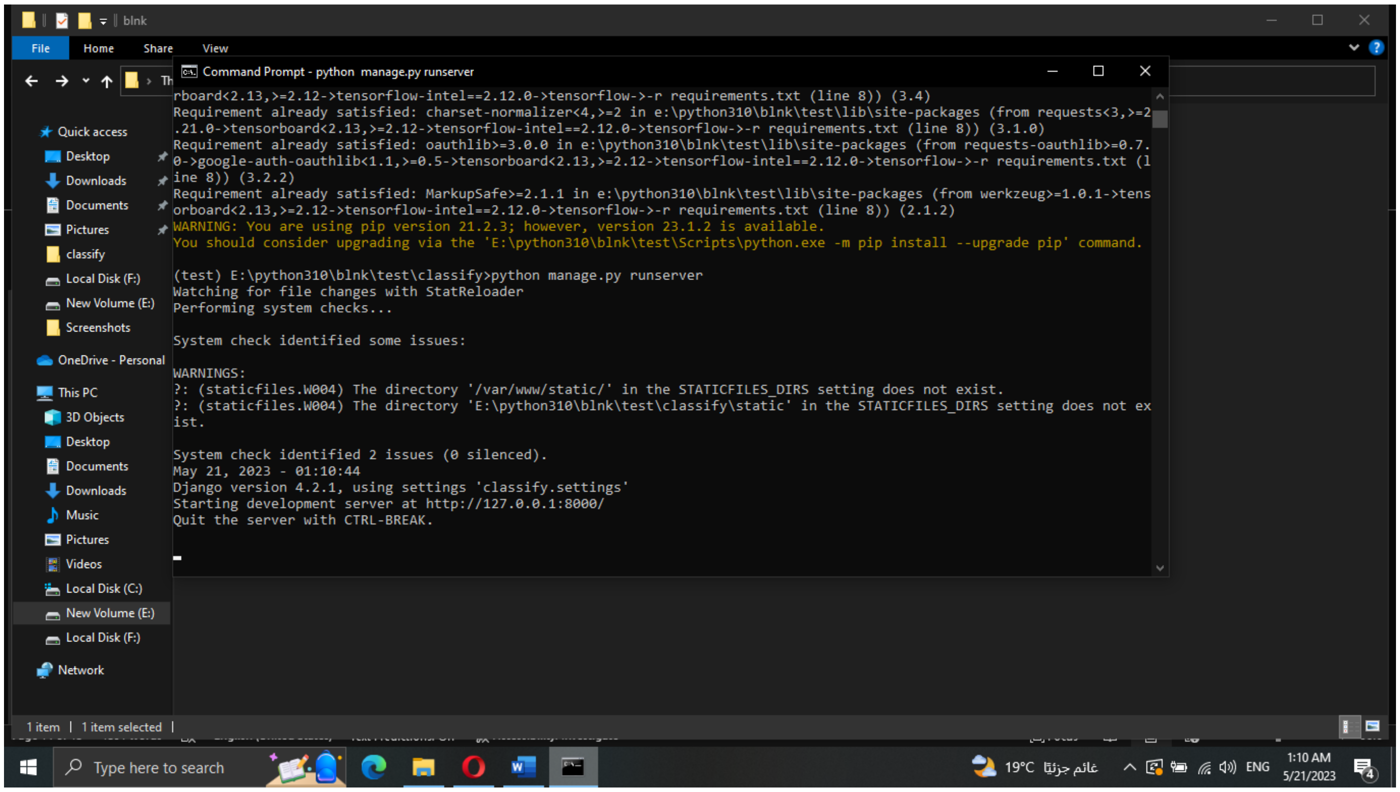
```
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\basem>E:

E:\>cd E:\python310\blnk

E:\python310\blnk>test\Scripts\activate

(test) E:\python310\blnk>cd test

(test) E:\python310\blnk\test>cd classify

(test) E:\python310\blnk\test\classify>pip install -r requirements.txt
```

3-python manage.py runserver



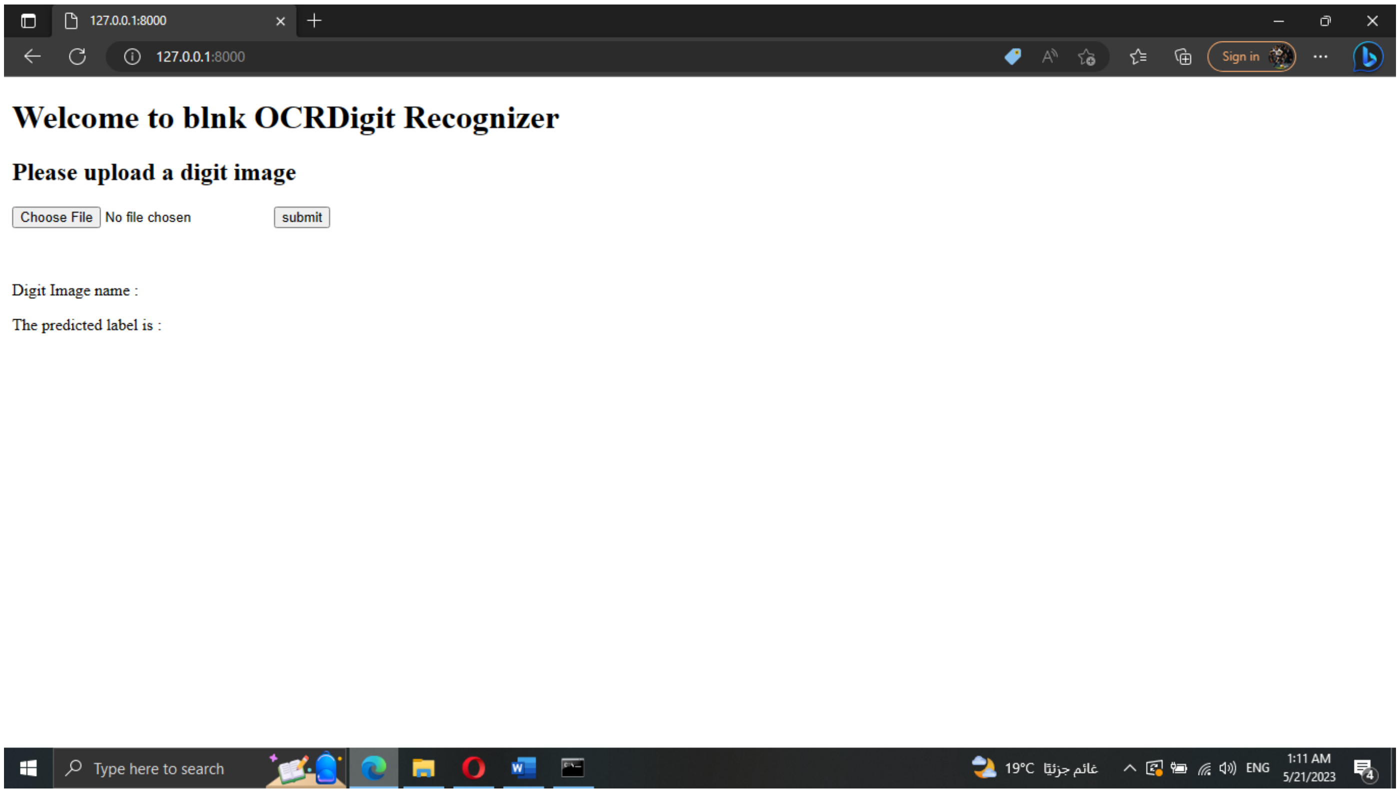
```
rboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (3.4)
Requirement already satisfied: charset-normalizer<4,>=2 in e:\python310\blnk\test\lib\site-packages (from requests<3,>=2
.21.0->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (3.1.0)
Requirement already satisfied: oauthlib>=3.0.0 in e:\python310\blnk\test\lib\site-packages (from requests-oauthlib>=0.7.
0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (l
ine 8)) (3.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in e:\python310\blnk\test\lib\site-packages (from werkzeug>=1.0.1->tens
orboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (2.1.2)
WARNING: You are using pip version 21.2.3; however, version 23.1.2 is available.
You should consider upgrading via the 'E:\python310\blnk\test\Scripts\python.exe -m pip install --upgrade pip' command.

(test) E:\python310\blnk\test\classify>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
```

4- Now you can go on http://127.0.0.1:8000/



```
rboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (3.4)
Requirement already satisfied: charset-normalizer<4,>=2 in e:\python310\blnk\test\lib\site-packages (from requests<3,>=2
.21.0->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (3.1.0)
Requirement already satisfied: oauthlib>=3.0.0 in e:\python310\blnk\test\lib\site-packages (from requests-oauthlib>=0.7.
0->google-auth-oauthlib<1.1,>=0.5->tensorboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (l
ine 8)) (3.2.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in e:\python310\blnk\test\lib\site-packages (from werkzeug>=1.0.1->tens
orboard<2.13,>=2.12->tensorflow-intel==2.12.0->tensorflow->-r requirements.txt (line 8)) (2.1.2)
WARNING: You are using pip version 21.2.3; however, version 23.1.2 is available.
You should consider upgrading via the 'E:\python310\blnk\test\Scripts\python.exe -m pip install --upgrade pip' command.

(test) E:\python310\blnk\test\classify>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
?: (staticfiles.W004) The directory '/var/www/static/' in the STATICFILES_DIRS setting does not exist.
?: (staticfiles.W004) The directory 'E:\python310\blnk\test\classify\static' in the STATICFILES_DIRS setting does not ex
ist.

System check identified 2 issues (0 silenced).
May 21, 2023 - 01:10:44
Django version 4.2.1, using settings 'classify.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

5-Our html page will appear



Welcome to blnk OCRDigit Recognizer

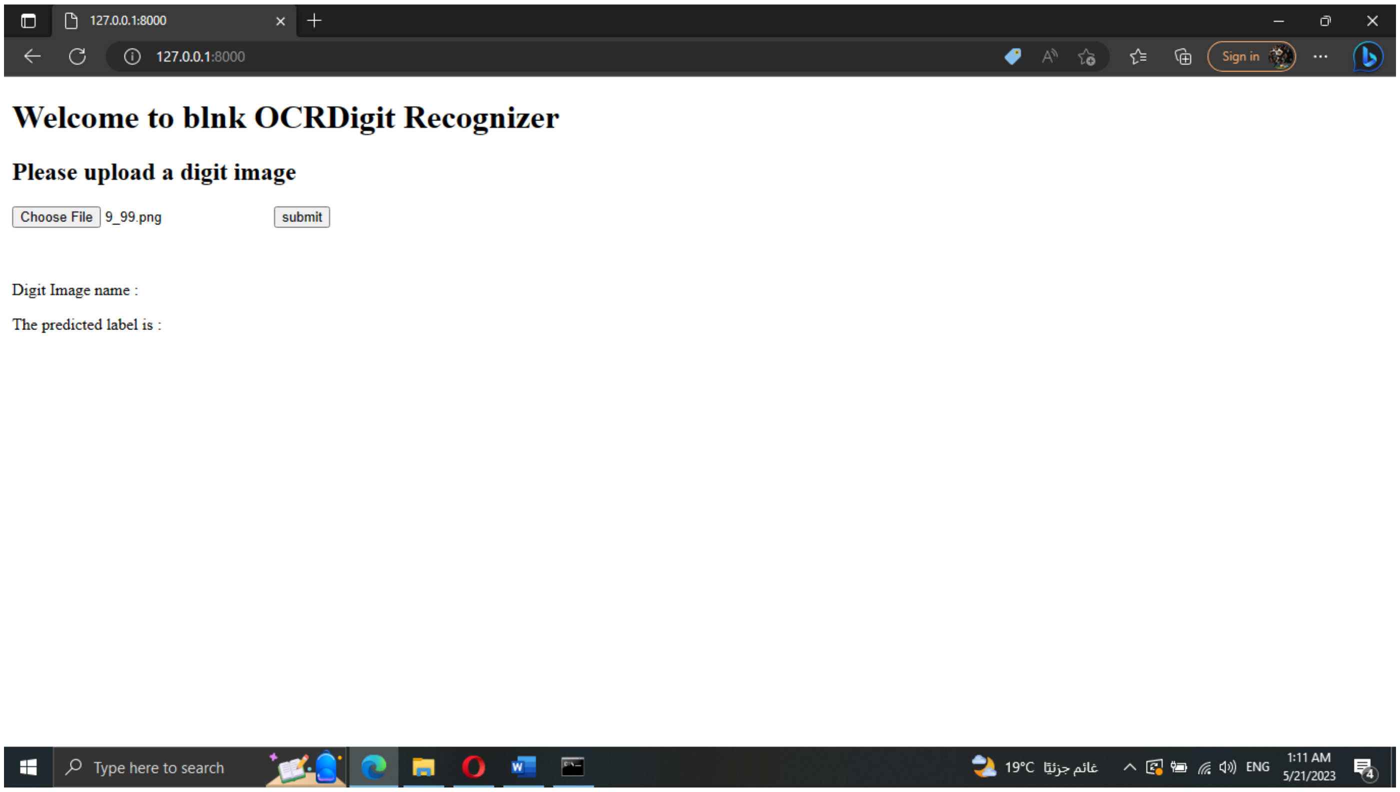
Please upload a digit image

Choose File No file chosen submit

Digit Image name :

The predicted label is :

6-Click on choose file, browse and select your digit image



Welcome to blnk OCRDigit Recognizer

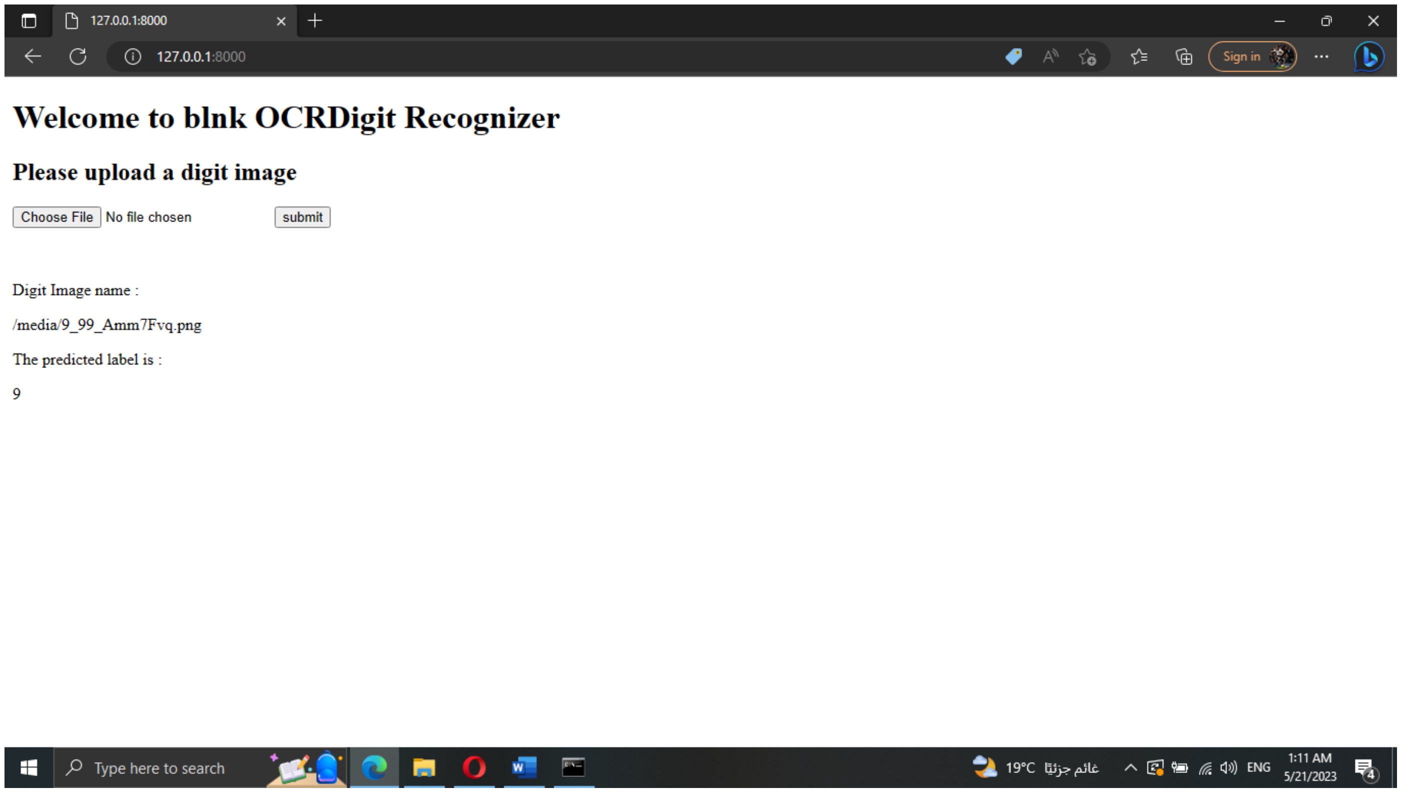
Please upload a digit image

Choose File 9_99.png submit

Digit Image name :

The predicted label is :

7-Click submit, the Digit Image name & the predicted label will appear



Welcome to blnk OCRDigit Recognizer

Please upload a digit image

Digit Image name :

/media/9_99_Amm7Fvq.png

The predicted label is :

9

My Github repo for this project : https://github.com/BasemRizk/blnk-OCR