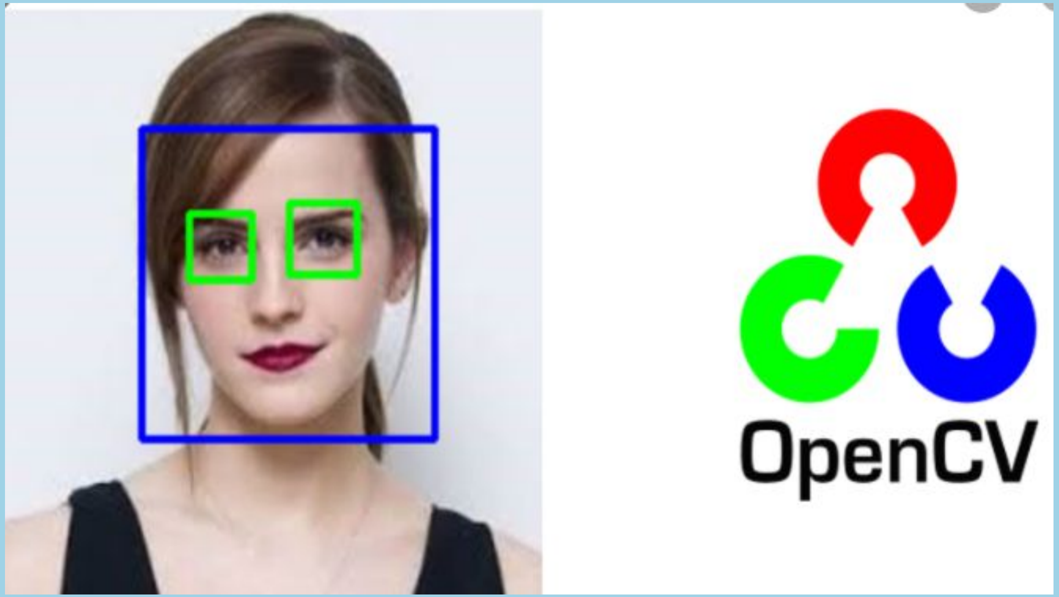
MASK DETECTION

SHIRIN KOUL

Idea: The concepts of deep learning will
Be used detect whether a person is wearing mask.

Methodology:

1. Detect face (using Open CV)
2. 2. Detect Mask



TECHNOLOGY :

1. Language used will be python.
2. Open CV for face detection.
3. TensorFlow
4. Keras
5. MobileNet

Learning

1. Basics of python
2. Variables
3. Lists
4. Tuples
5. Sets
6. Variables

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays

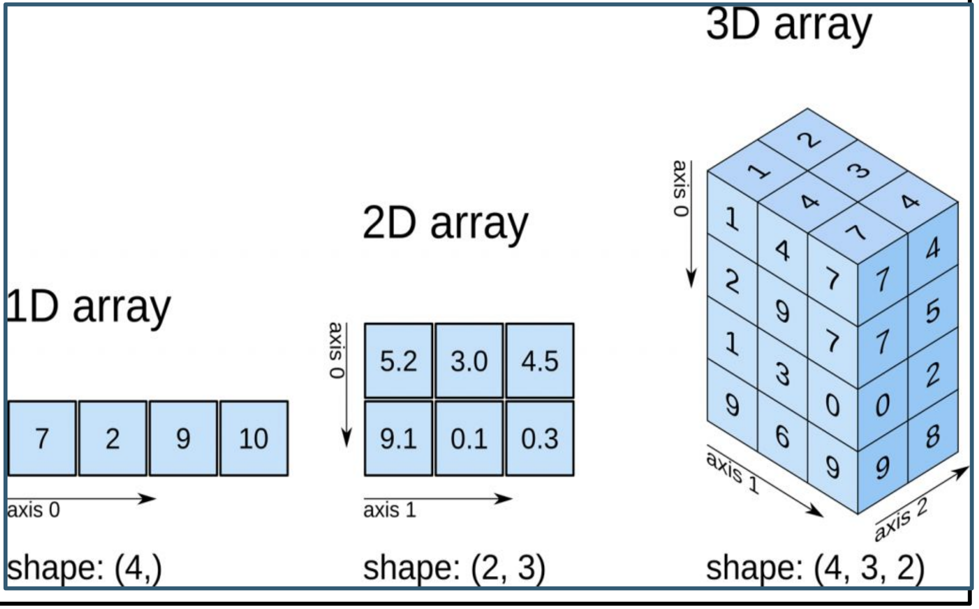
Numpy array has the various function, methods, and variables, to ease our task of matrix computation.

Advantages of using Numpy Arrays Over Python Lists:

consumes less memory.

fast as compared to the python List.

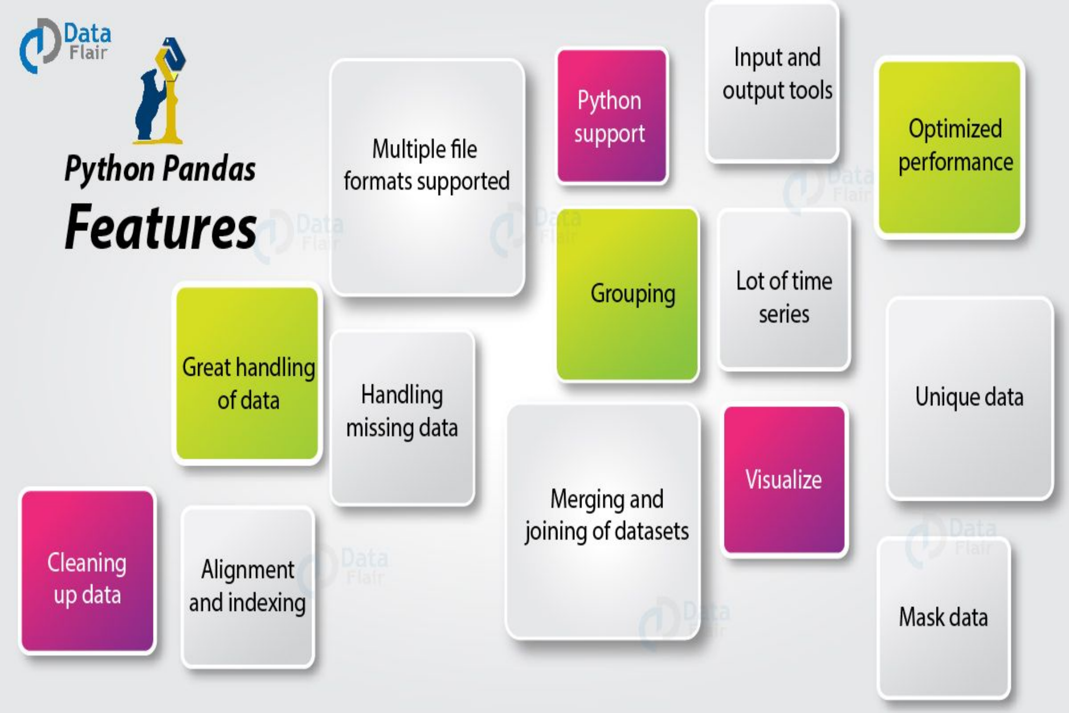
convenient to use.



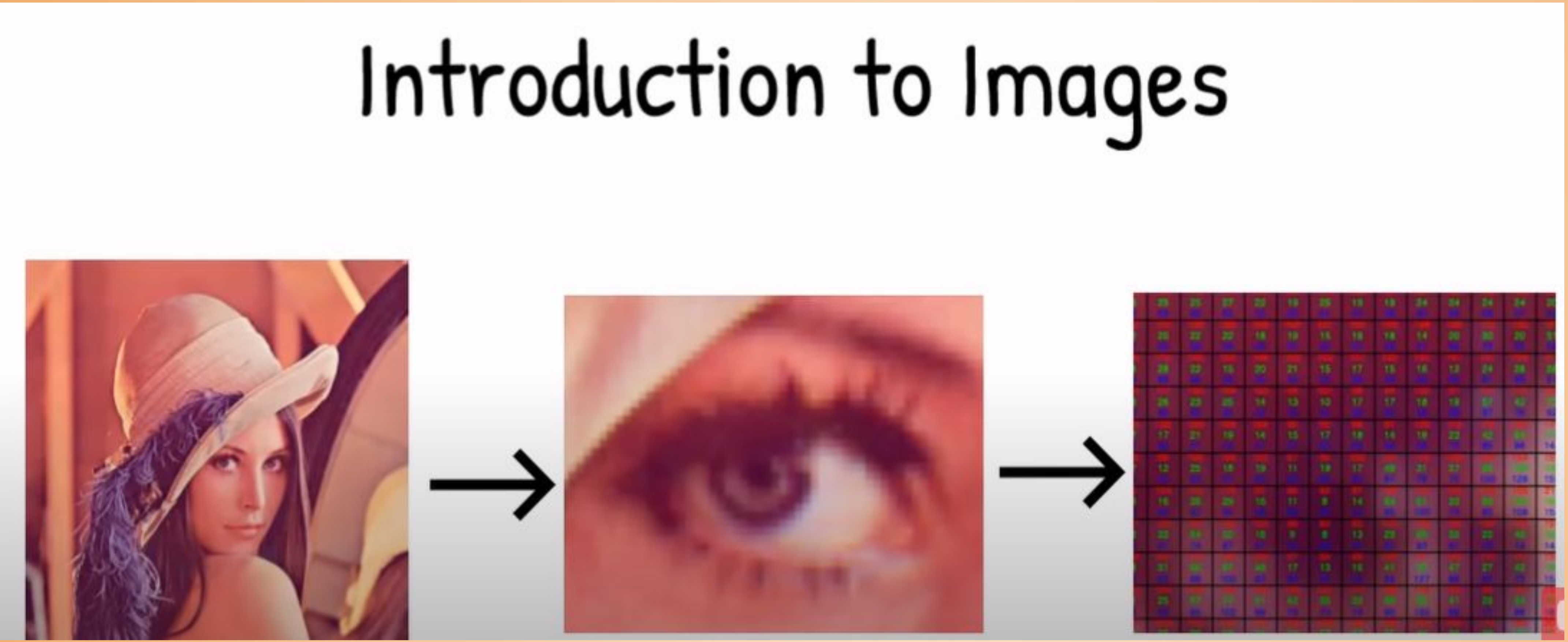
Pandas

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

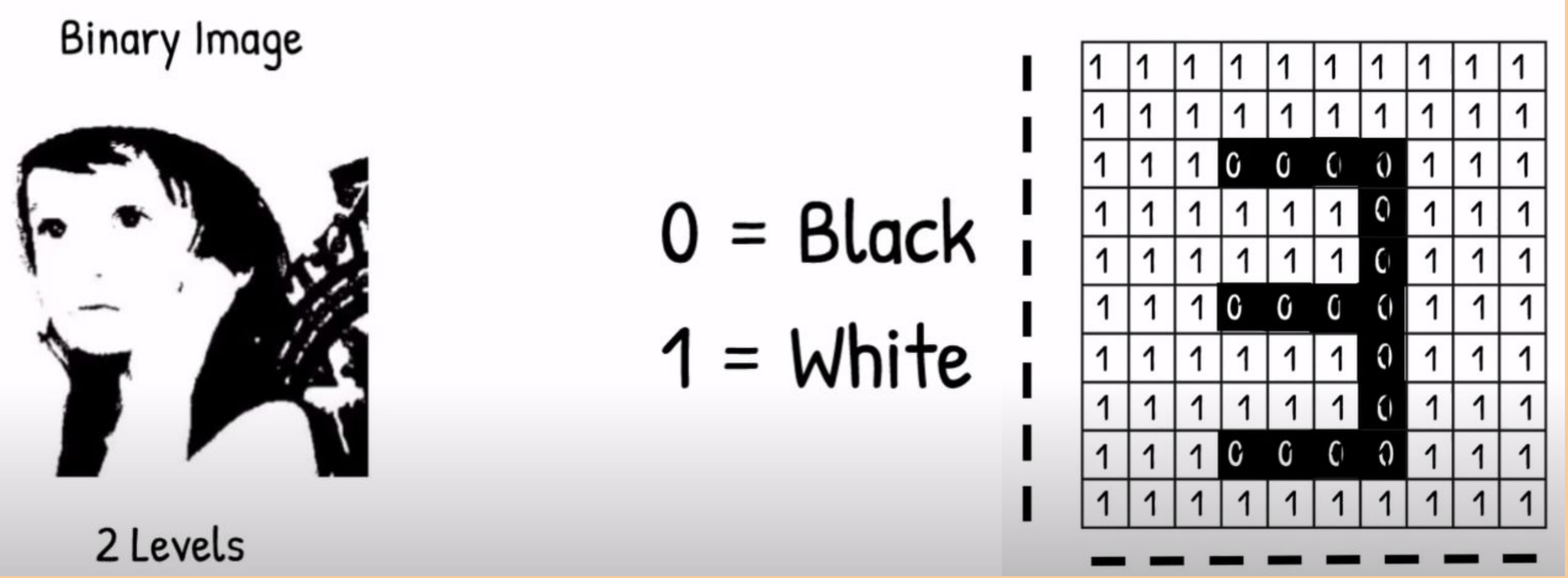
It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.



OPEN CV :



Introduction to Images



Binary Image

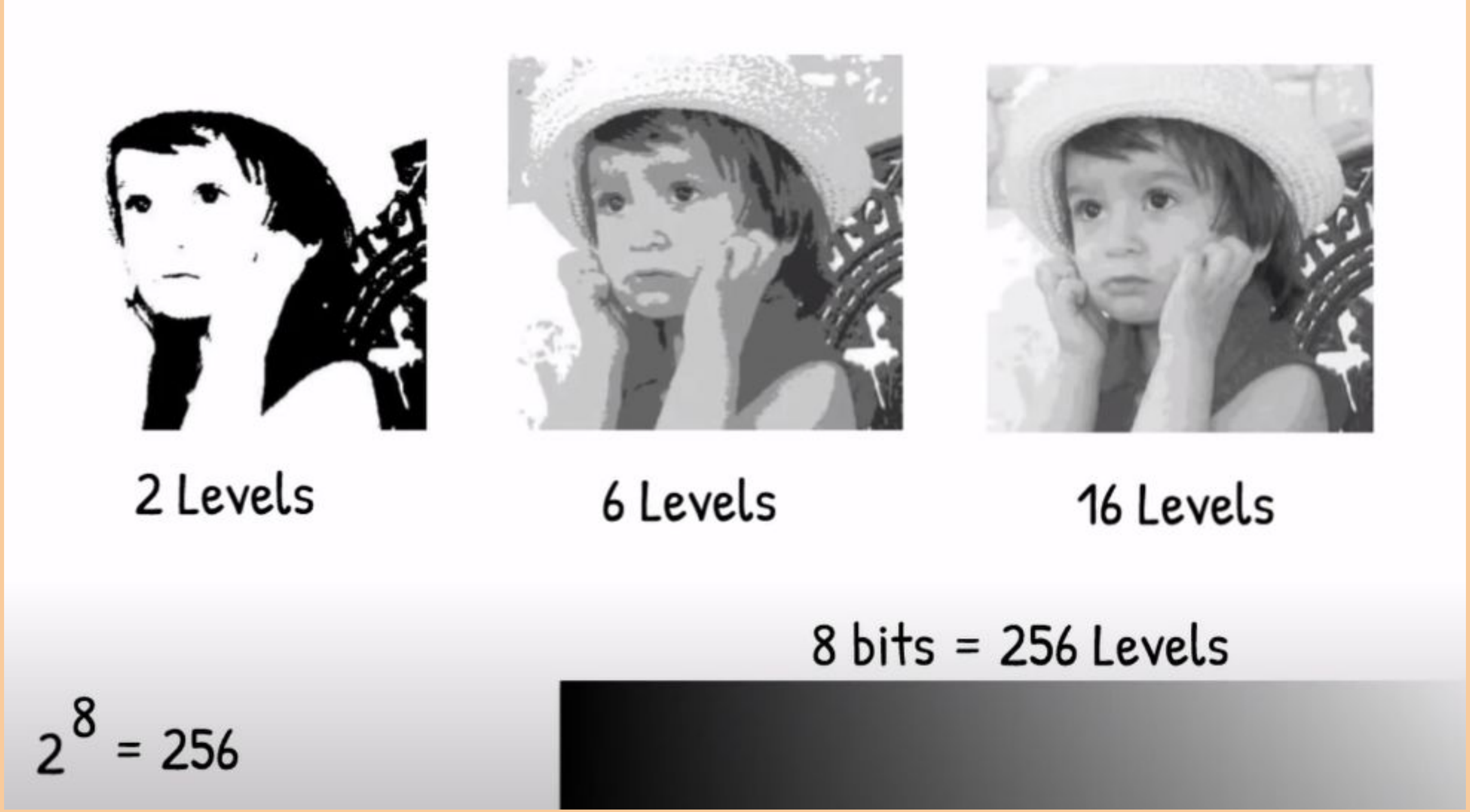
2 Levels

0 = Black

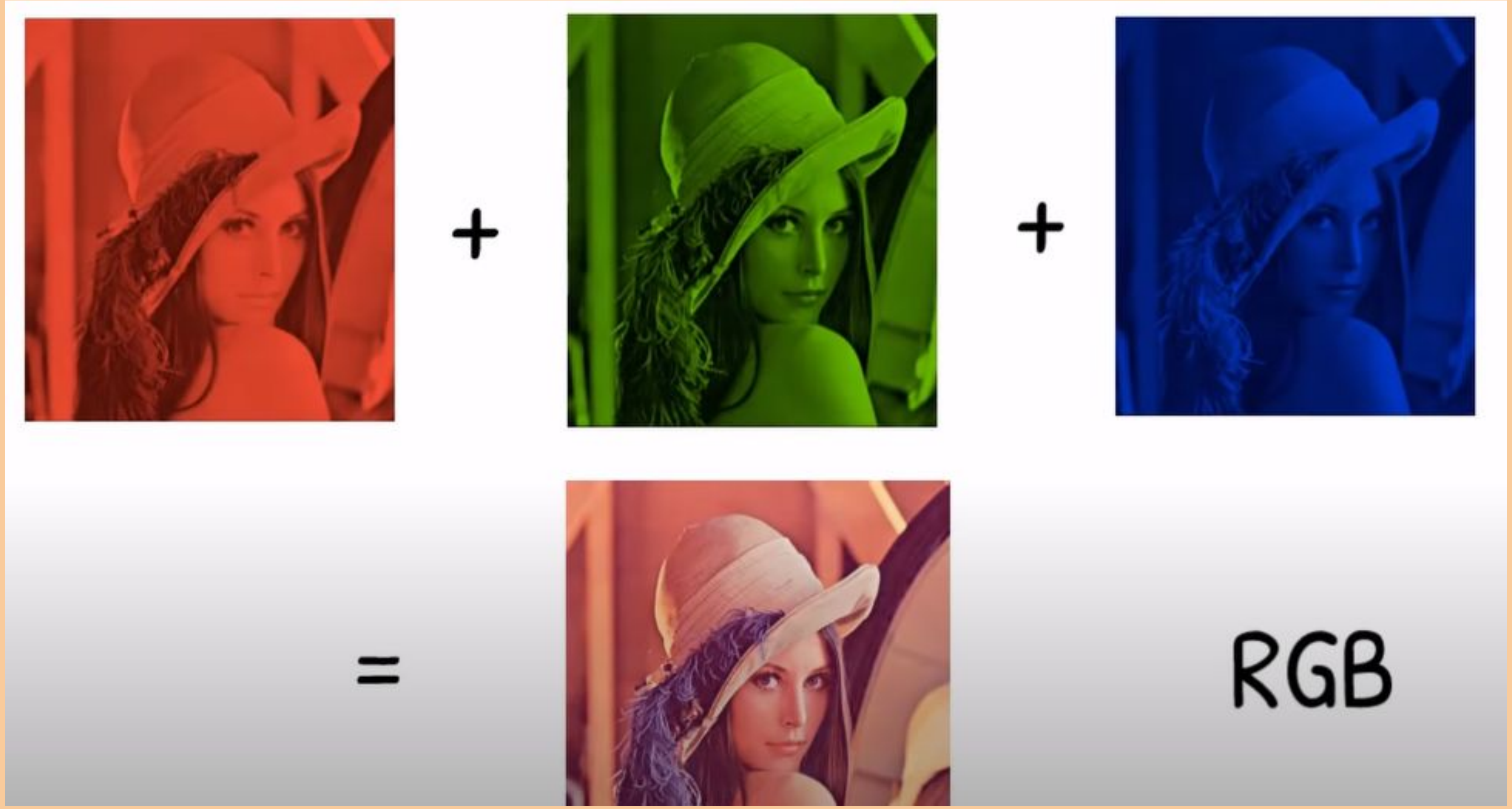
1 = White

1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1

Gray Scale Image- 8 bit or 256 levels



Colored Image has 3 levels. Each level is accountable for different colors amongst red blue and green which results in a colored image.

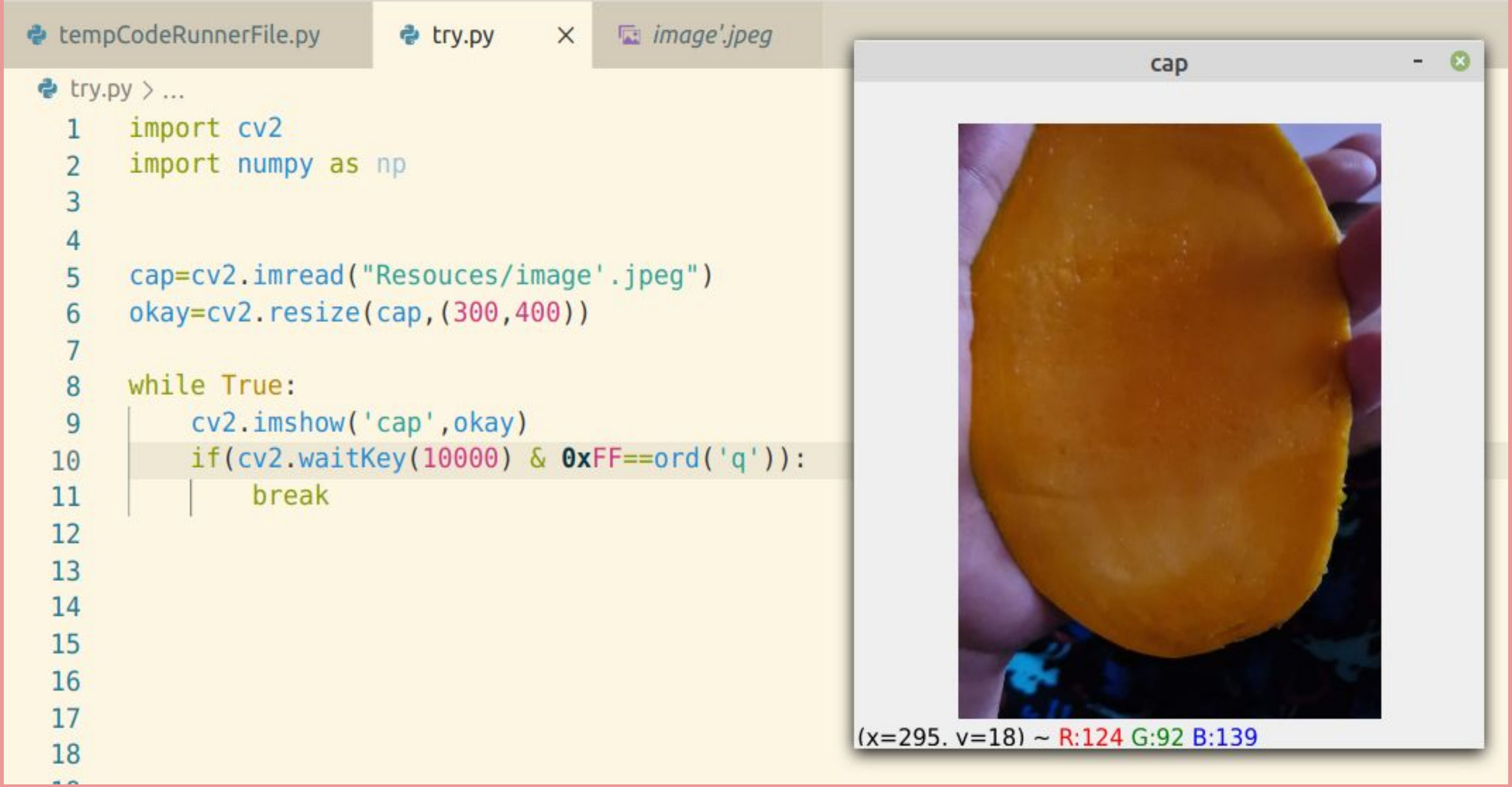


Images are stored in OpenCV as numpy nd array.

1. Viewing an image –

Its a 2 step process in OpenCV. Firstly, we read an image using opencv and then we show it using the function imshow().

We can also provide a parameter in imread function to directly convert it into grayscale image or keep as it is.



```
import cv2
import numpy as np


cap=cv2.imread("Resouces/image'.jpeg")
okay=cv2.resize(cap,(300,400))

while True:
    cv2.imshow('cap',okay)
    if(cv2.waitKey(10000) & 0xFF==ord('q')):
        break
```

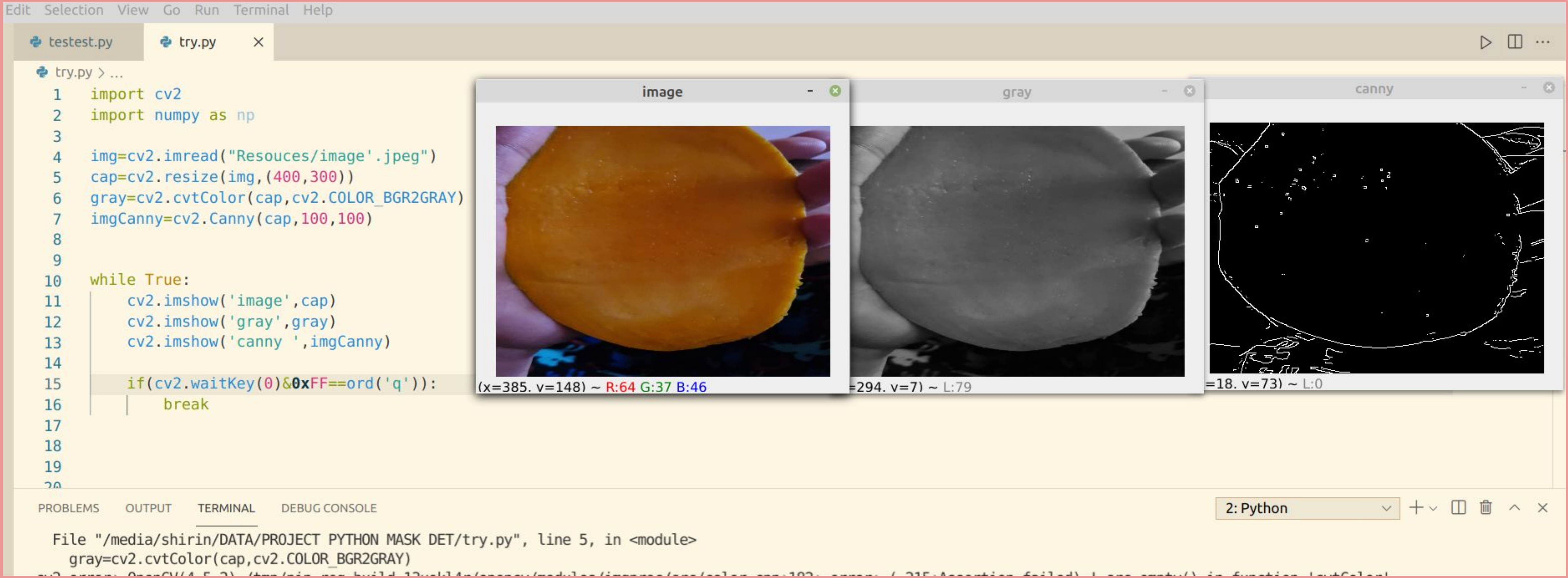
```
(x=295. v=18) ~ R:124 G:92 B:139
```

2. Video Capturing - Open CV is used in capturing video using webcam or can play any video which can also be edited using Opencv functions.

Video is basically a collection of images viewed really fast hence we use loop to display that.

3. Image Editing –Image can be transformed into different kind of image like colored into grey scale and vice versa or into Canny image or blurred image just by using a simple function cvtColor() where it takes the name of the image that needs to be converted as a parameter with the conversion it has to get.

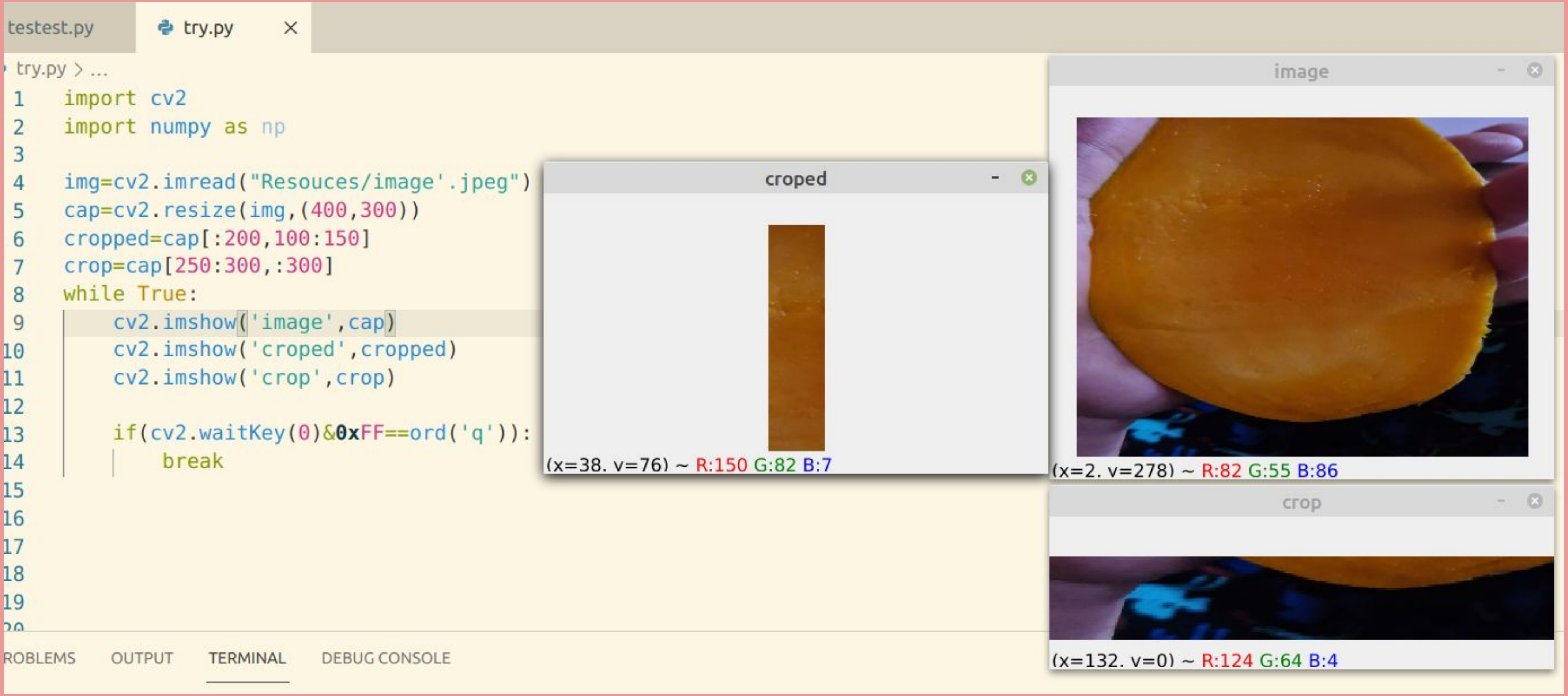
4. Resizing image- Function resize which takes the image and the final dimension as parameters in order to resize an image.



```python
import cv2
import numpy as np

img=cv2.imread("Resouces/image'.jpeg")
cap=cv2.resize(img,(400,300))
gray=cv2.cvtColor(cap,cv2.COLOR_BGR2GRAY)
imgCanny=cv2.Canny(cap,100,100)


while True:
    cv2.imshow('image',cap)
    cv2.imshow('gray',gray)
    cv2.imshow('canny ',imgCanny)

    if(cv2.waitKey(0)&0xFF==ord('q')):
        break
```

```
  File "/media/shirin/DATA/PROJECT PYTHON MASK DET/try.py", line 5, in <module>
    gray=cv2.cvtColor(cap,cv2.COLOR_BGR2GRAY)
cv2.error: OpenCV(4.5.2) /tmp/pip-req-build-12ukl4n/opencv/modules/imgproc/src/color.cpp:182: error: (-215:Assertion failed) !_src.empty() in function 'cvtColor'
```

5 . Cropping an Image



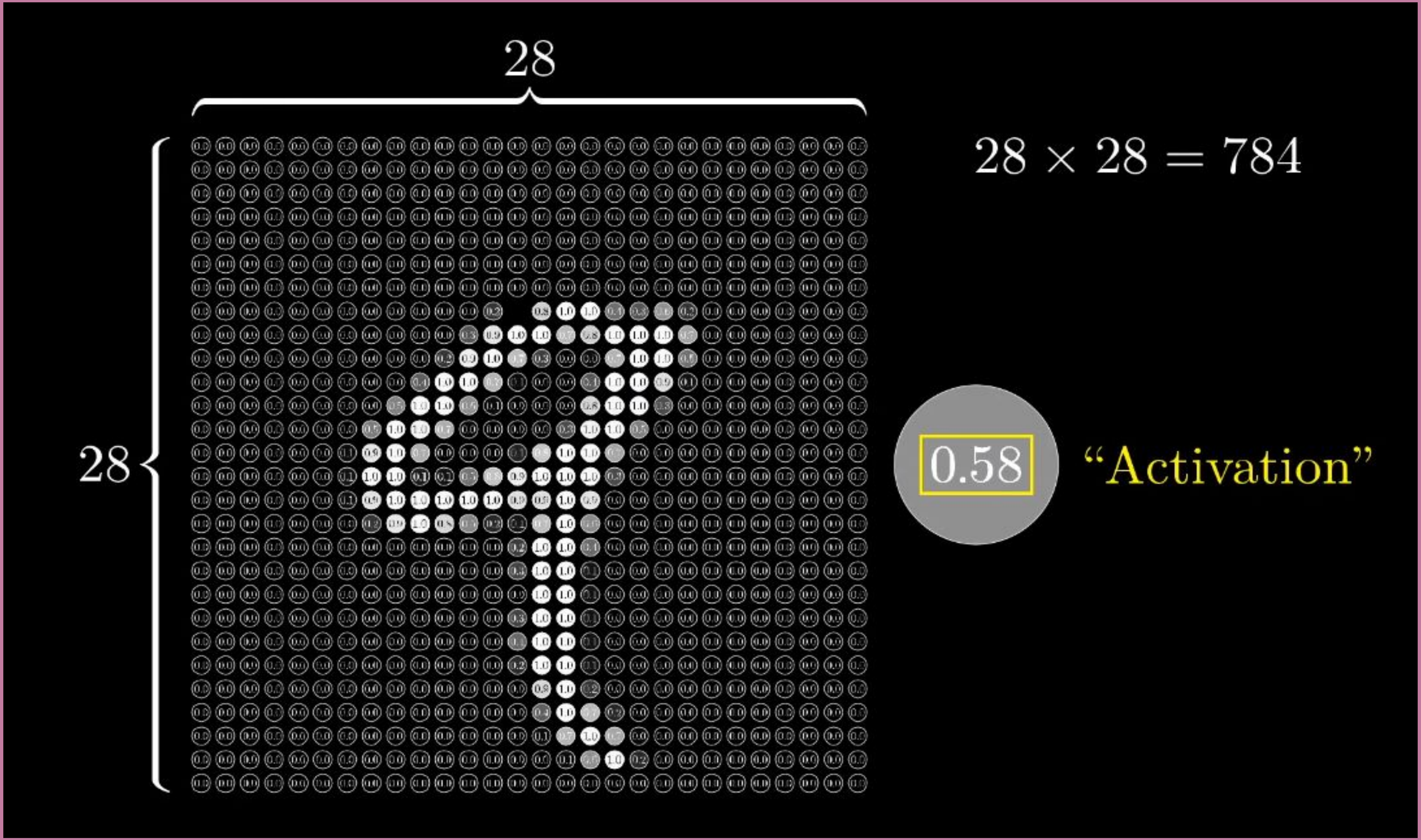
```
import cv2
import numpy as np

img=cv2.imread("Resouces/image'.jpeg")
cap=cv2.resize(img,(400,300))
cropped=cap[:200,100:150]
crop=cap[250:300,:300]
while True:
    cv2.imshow('image',cap)
    cv2.imshow('croped',cropped)
    cv2.imshow('crop',crop)

    if(cv2.waitKey(0)&0xFF==ord('q')):
        break
```

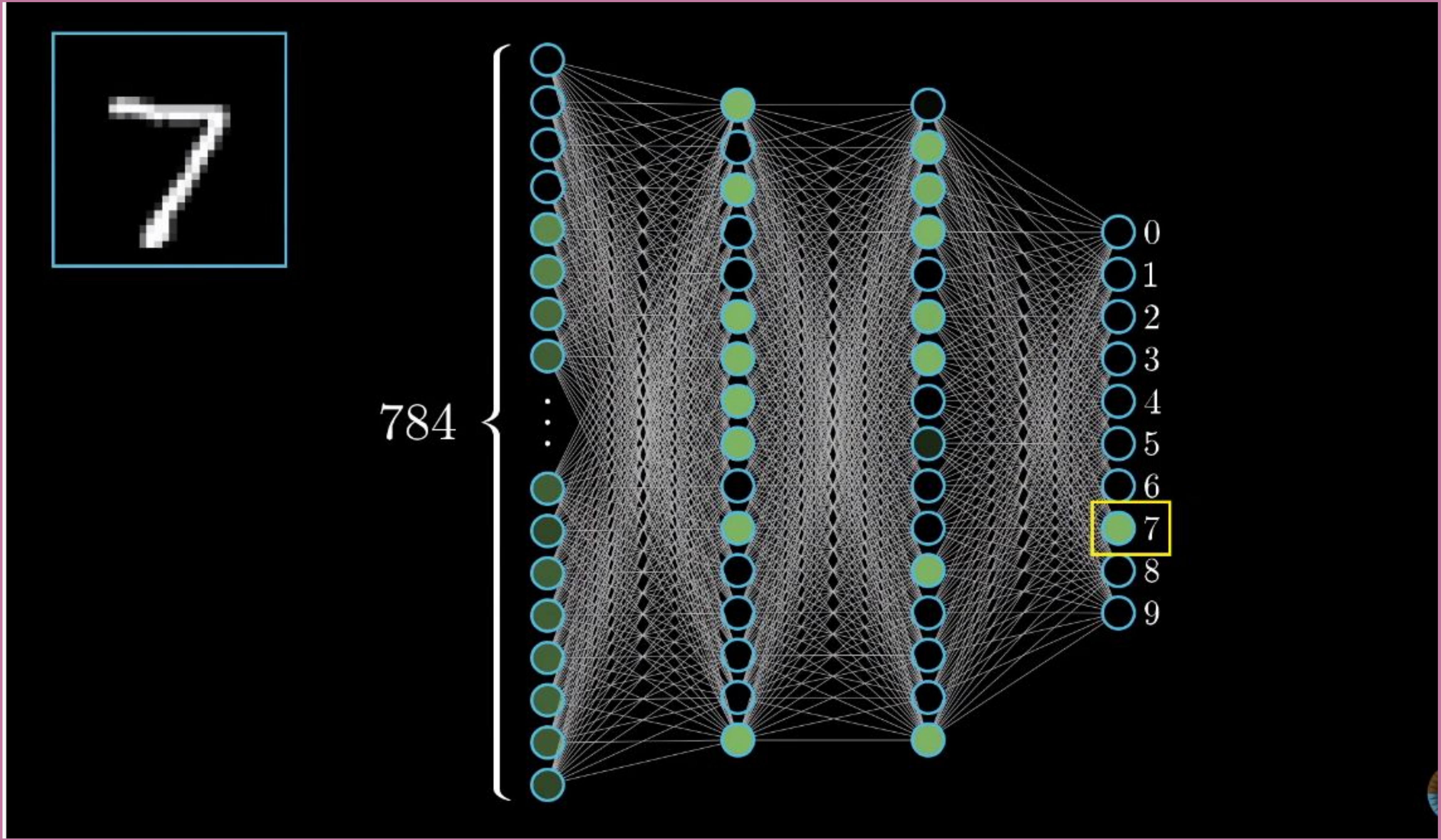
```
(x=38, y=76) ~ R:150 G:82 B:7
```

```
(x=2, y=278) ~ R:82 G:55 B:86
```

```
(x=132, y=0) ~ R:124 G:64 B:4
```

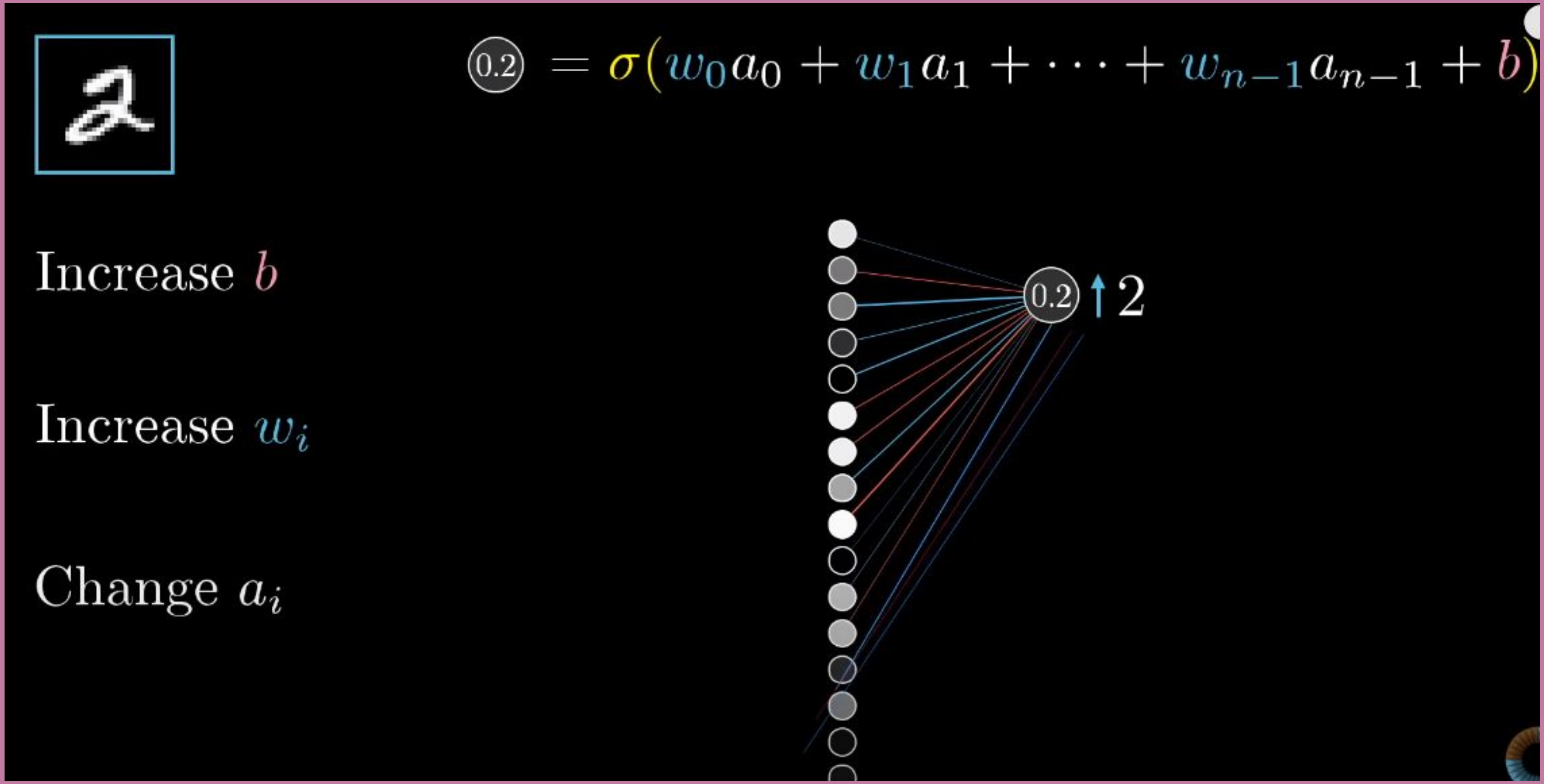
DEEP LEARNING



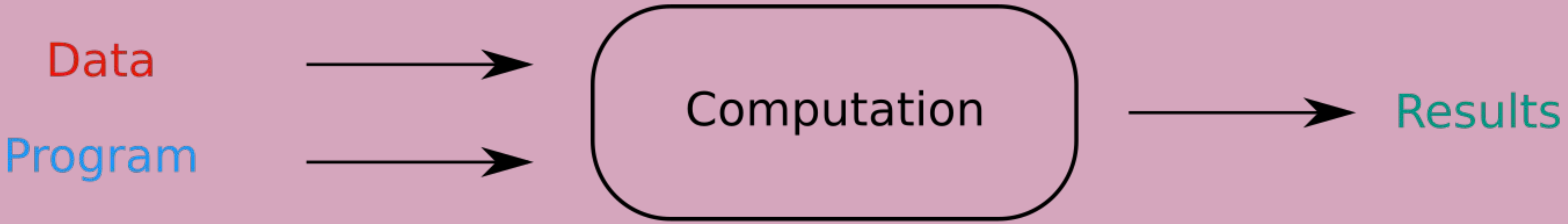

784
0
1
2
3
4
5
6
7
8
9

Cost function - The square of the difference between the output answer and the actual answer.
The aim is to reduce this cost function.

Back Propagation- Core algorithm behind how neural networks learn.


$0.2 = \sigma(w_0 a_0 + w_1 a_1 + \cdots + w_{n-1} a_{n-1} + b)$
Increase b
Increase w_i
Change a_i
0.2
↑ 2

Traditional programming



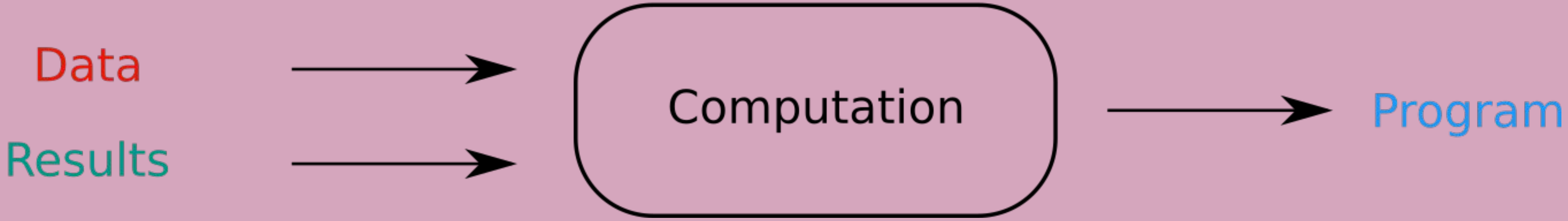
Data

Program

Computation

Results

Machine Learning Approach



Data

Results

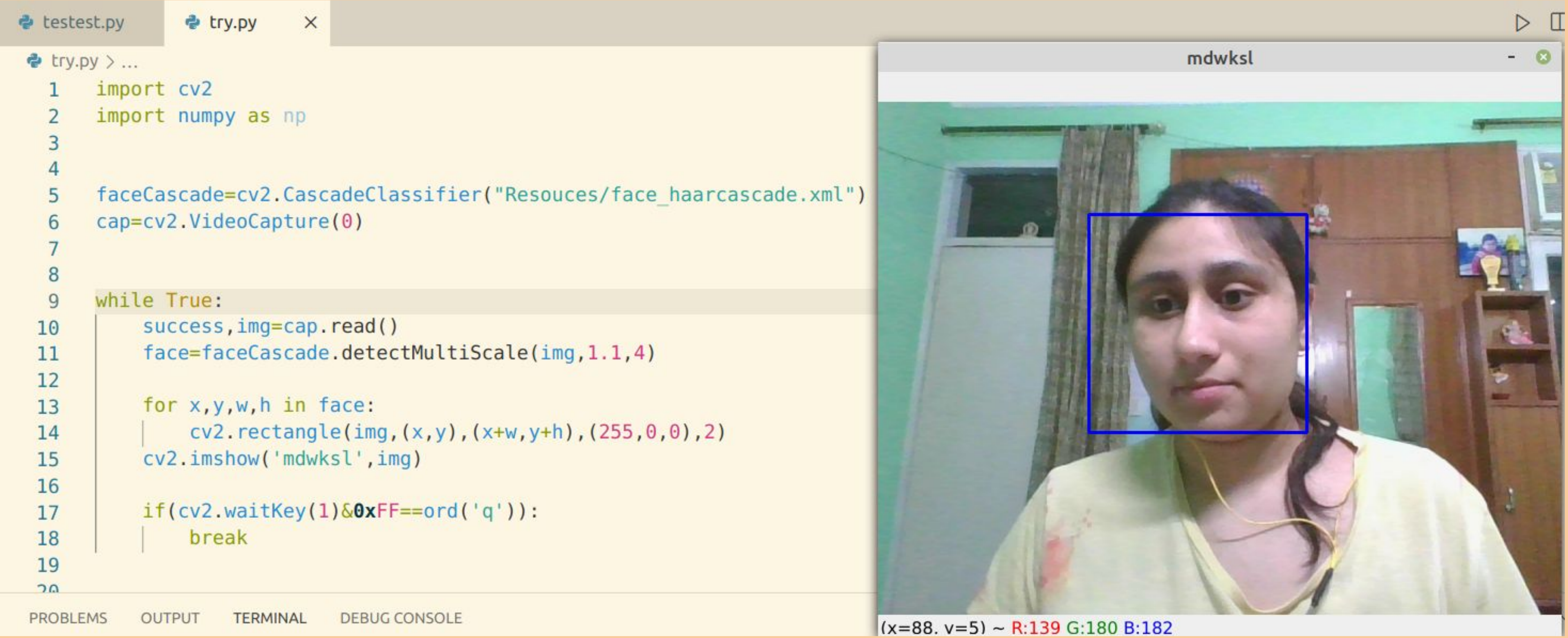
Computation

Program

IDEA

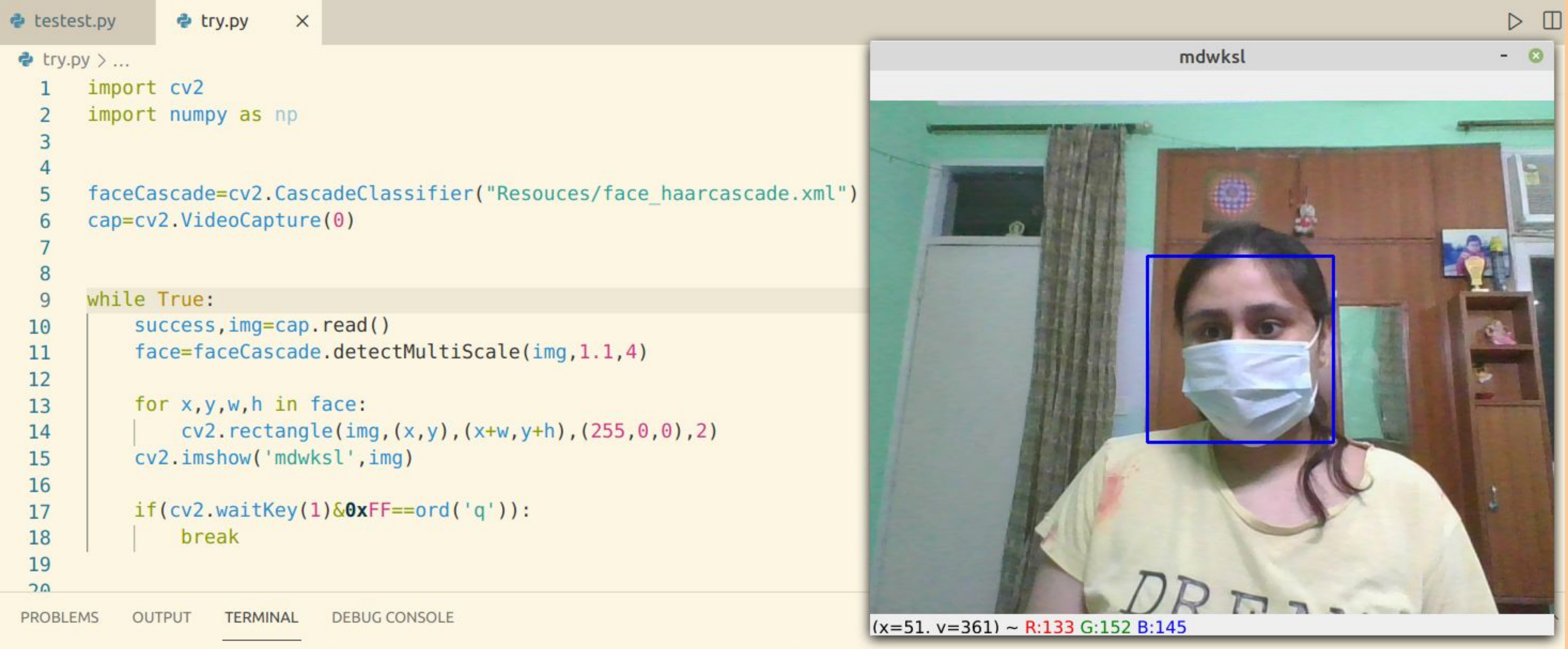
1. To check whether there is some face present .
2. If Yes
3. Pass the face cropped and resized to our trained model network.
4. If No-Repeat
5. Face to be detected using dnn module.

FACE DETECTION



```
import cv2
import numpy as np


faceCascade=cv2.CascadeClassifier("Resouces/face_haarcascade.xml")
cap=cv2.VideoCapture(0)


while True:
    success,img=cap.read()
    face=faceCascade.detectMultiScale(img,1.1,4)

    for x,y,w,h in face:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.imshow('mdwksl',img)

    if(cv2.waitKey(1)&0xFF==ord('q')):
        break

```

(x=88, y=5) ~ R:139 G:180 B:182



```python
import cv2
import numpy as np


faceCascade=cv2.CascadeClassifier("Resouces/face_haarcascade.xml")
cap=cv2.VideoCapture(0)


while True:
    success,img=cap.read()
    face=faceCascade.detectMultiScale(img,1.1,4)

    for x,y,w,h in face:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.imshow('mdwksl',img)

    if(cv2.waitKey(1)&0xFF==ord('q')):
        break

```

Training Model

The model was trained using data set from kaggle. 80% of the images were used to train the data and 20% were used for its testing.

There were 2 labels in data set: With Mask and Without Mask.

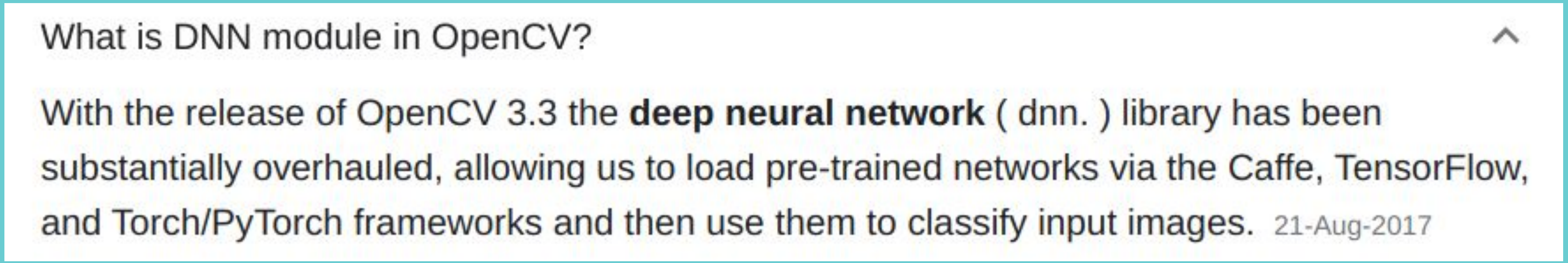
The images and labels were converted to numpy array and binary respectively.

Then a model was trained. A function was used to augment the images slightly which resulted in slightly different kinds of images while training of data which results in better accuracy .

After all the the model was finally saved and used.

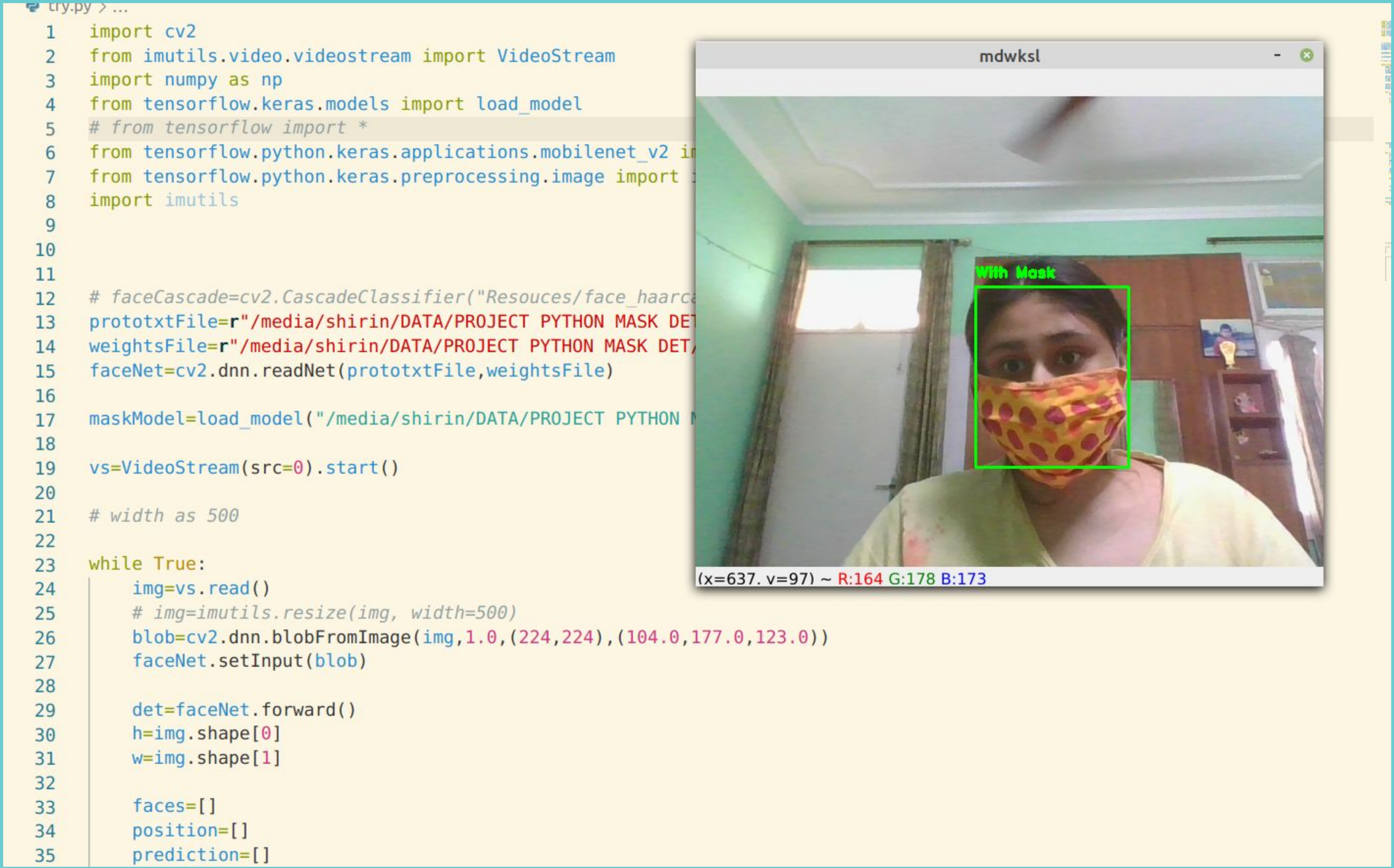
WHY DNN MODULE OF OPEN CV?

1. Better accuracy in real time .
2. Introduced in Open CV in 2017



What is DNN module in OpenCV?

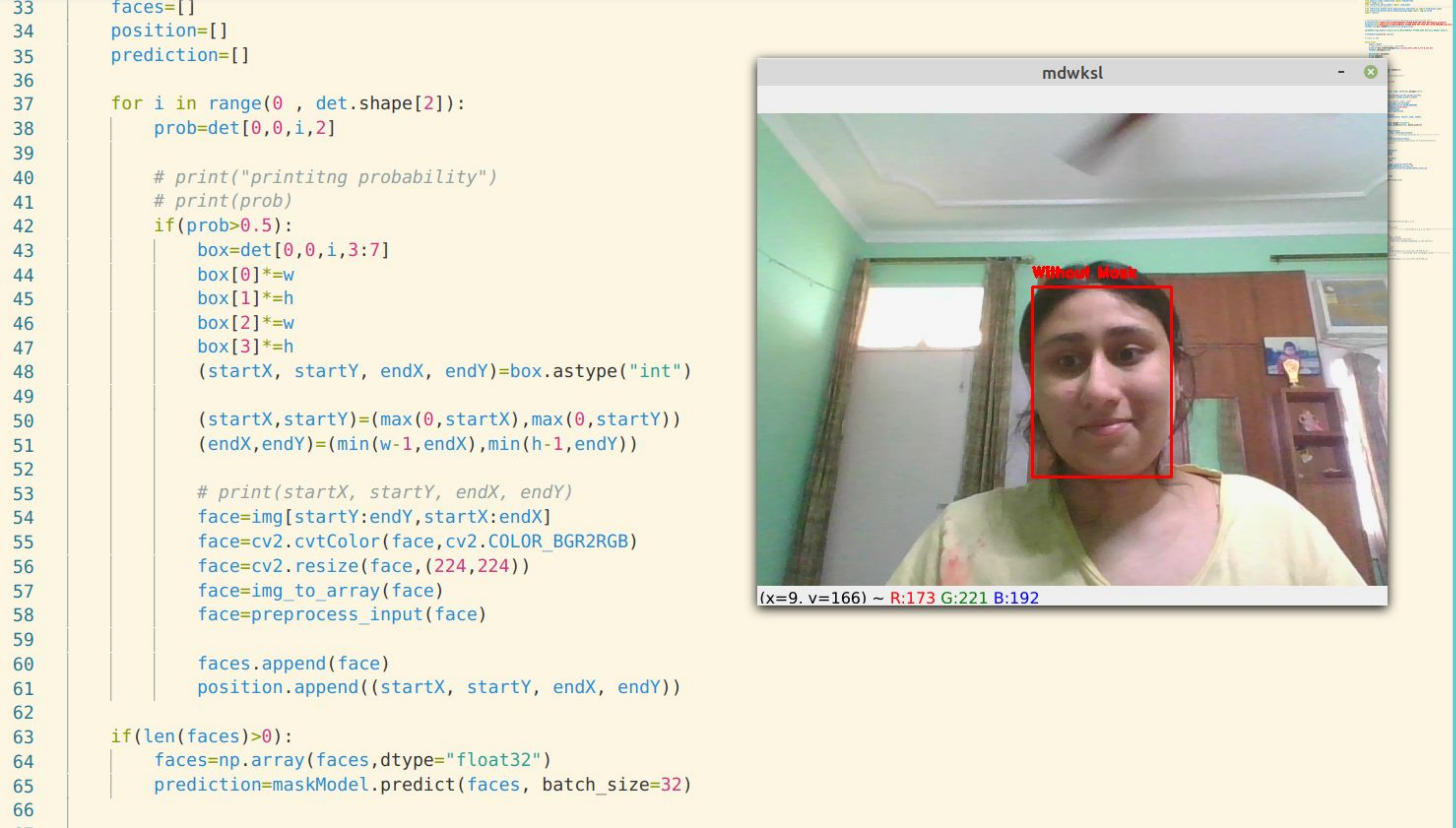
With the release of OpenCV 3.3 the **deep neural network** (dnn.) library has been substantially overhauled, allowing us to load pre-trained networks via the Caffe, TensorFlow, and Torch/PyTorch frameworks and then use them to classify input images. 21-Aug-2017



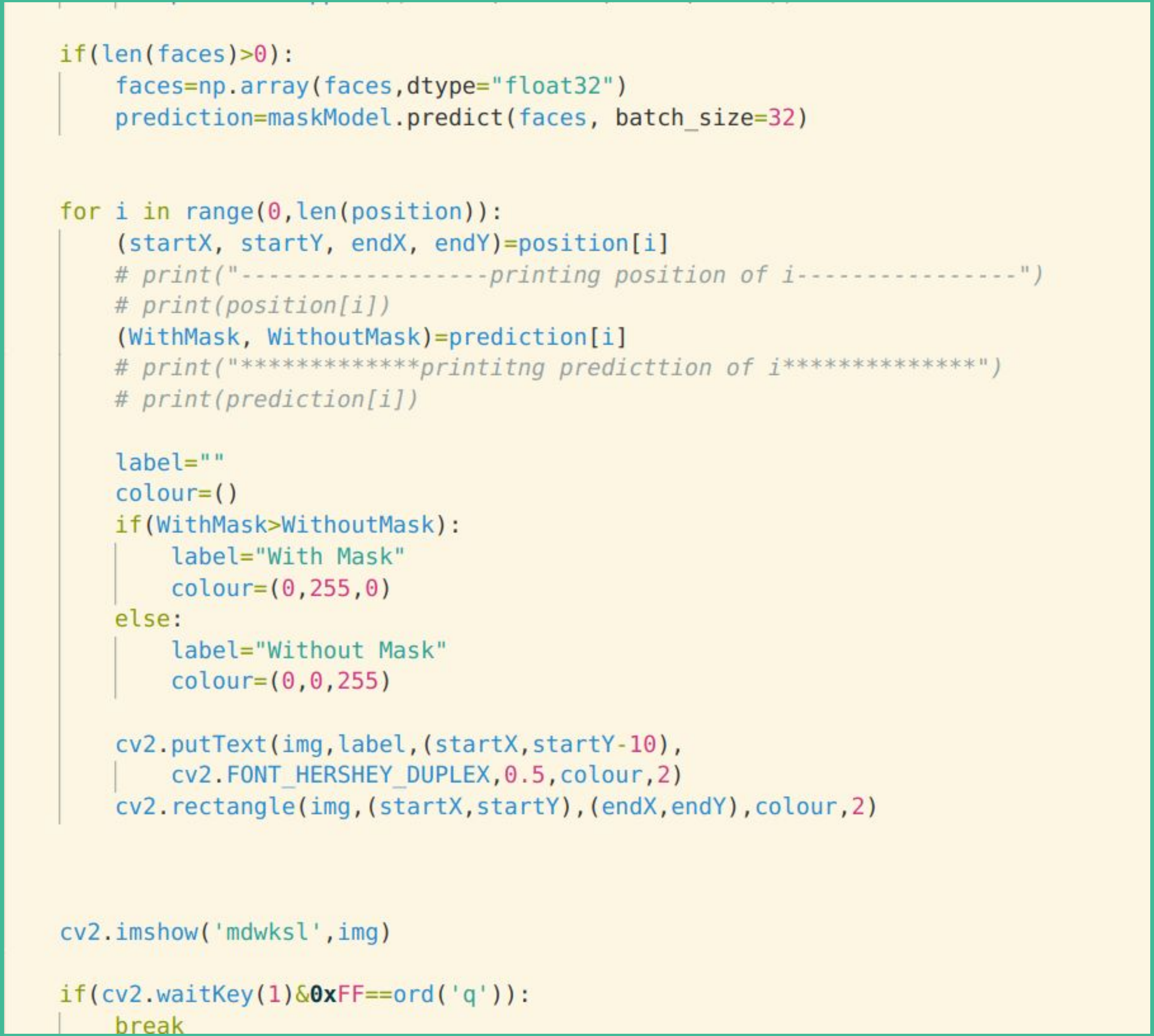
try.py > ...

```
import cv2
from imutils.video.videostream import VideoStream
import numpy as np
from tensorflow.keras.models import load_model
# from tensorflow import *
from tensorflow.python.keras.applications.mobilenet_v2 i
from tensorflow.python.keras.preprocessing.image import i
import imutils



# faceCascade=cv2.CascadeClassifier("Resouces/face_haarca
prototxtFile=r"/media/shirin/DATA/PROJECT PYTHON MASK DET
weightsFile=r"/media/shirin/DATA/PROJECT PYTHON MASK DET/
faceNet=cv2.dnn.readNet(prototxtFile,weightsFile)

maskModel=load_model("/media/shirin/DATA/PROJECT PYTHON M

vs=VideoStream(src=0).start()

# width as 500

while True:
    img=vs.read()
    # img=imutils.resize(img, width=500)
    blob=cv2.dnn.blobFromImage(img,1.0,(224,224),(104.0,177.0,123.0))
    faceNet.setInput(blob)

    det=faceNet.forward()
    h=img.shape[0]
    w=img.shape[1]

    faces=[]
    position=[]
    prediction=[]
```

(x=637, v=97) ~ R:164 G:178 B:173



```
    faces=[]
    position=[]
    prediction=[]

    for i in range(0 , det.shape[2]):
        prob=det[0,0,i,2]

        # print("printitng probability")
        # print(prob)
        if(prob>0.5):
            box=det[0,0,i,3:7]
            box[0]*=w
            box[1]*=h
            box[2]*=w
            box[3]*=h
            (startX, startY, endX, endY)=box.astype("int")

            (startX,startY)=(max(0,startX),max(0,startY))
            (endX,endY)=(min(w-1,endX),min(h-1,endY))

            # print(startX, startY, endX, endY)
            face=img[startY:endY,startX:endX]
            face=cv2.cvtColor(face,cv2.COLOR_BGR2RGB)
            face=cv2.resize(face,(224,224))
            face=img_to_array(face)
            face=preprocess_input(face)

            faces.append(face)
            position.append((startX, startY, endX, endY))

    if(len(faces)>0):
        faces=np.array(faces,dtype="float32")
        prediction=maskModel.predict(faces, batch_size=32)
```



```
if(len(faces)>0):
    faces=np.array(faces,dtype="float32")
    prediction=maskModel.predict(faces, batch_size=32)


for i in range(0,len(position)):
    (startX, startY, endX, endY)=position[i]
    # print("------------------printing position of i----------------")
    # print(position[i])
    (WithMask, WithoutMask)=prediction[i]
    # print("*************printitng predicttion of i**************")
    # print(prediction[i])

    label=""
    colour=()
    if(WithMask>WithoutMask):
        label="With Mask"
        colour=(0,255,0)
    else:
        label="Without Mask"
        colour=(0,0,255)

    cv2.putText(img,label,(startX,startY-10),
        cv2.FONT_HERSHEY_DUPLEX,0.5,colour,2)
    cv2.rectangle(img,(startX,startY),(endX,endY),colour,2)



cv2.imshow('mdwksl',img)

if(cv2.waitKey(1)&0xFF==ord('q')):
    break
```