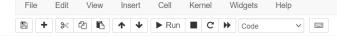


Trusted



```
In [2]: ▶ from tkinter import*
                     from tkinter import ttk
                     \textbf{from PIL import Image,} \textbf{ImageT} k
                     from tkinter import messagebox
                     import mysal.connector
                     from time import strftime
                     from datetime import datetime
                     import cv2
                     import os
                     import numpy as np
                     class FaceDetector:
                            def __init__(self,root):
                                   self.root=root
                                   self.root.geometry("1530x790+0+0")
                                   self.root.title("face recognition system")
                                   title_lbl=Label(self.root,text="FACE DETECTOR ", font=("times new roman",30,"bold"),bg="white",fg="red")
                                   title lbl.place(x=0,y=0,width=1350,height=45)
                                   img\_t = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial\ internship\internship\costacloud\face\ recog\college and the property of the property o
                                   img_t=img_t.resize((700,610),Image.ANTIALIAS)
                                   self.photoimg_t=ImageTk.PhotoImage(img_t)
                                   \verb|f_lbl=Label(self.root,image=self.photoimg_t)|\\
                                   f_lbl.place(x=0,y=43,width=650,height=610)
                                   img_b = Image.open(r"C:\Users\HP\OneDrive\Desktop\Adrija\industrial internship\internship\costacloud\face recog\colle
                                   img_b=img_b.resize((950,610),Image.ANTIALIAS)
                                   self.photoimg_b=ImageTk.PhotoImage(img_b)
                                   f_lbl=Label(self.root,image=self.photoimg_b)
                                   f\_lbl.place(x=500,y=43,width=950,height=610)
                                   b1_1=Button(f_lbl,text="FACE DETECTOR ",command=self.face_recog,cursor="hand2",font=("times new roman",18,"bold"),bg=
                                   b1_1.place(x=350,y=540,width=240,height=40)
                                   ###attendance####
                            def mark_attendance(self,i,n,r,d):
                                   with open("attendance.csv","r+",newline="\n") as f:
    myDataList=f.readlines()
                                          name_list=[]
                                          for line in myDataList:
                                                 entry=line.split((","))
                                                  name list.append(entry[0])
                                          if((i not in name_list) and (n not in name_list) and(r not in name_list) and(d not in name_list)):
                                                 now=datetime.now()
                                                 d1=now.strftime("%d/%m/%Y")
                                                 dtString=now.strftime("%H:%M:%S")
                                                 f.writelines(f"\n{i},{n},{r},{d},{dtString},{d1},Present")
                                   ###face detector####
                            def face_recog(self):
                                   {\tt def\ draw\_boundray} ({\tt img,classifier,scaleFactor,minNeighbors,color,text,clf}) :
                                          gray_image=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                                          features=classifier.detectMultiScale(gray_image,scaleFactor,minNeighbors)
                                          coord=[]
                                          for (x,y,w,h) in features:
                                                  cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
                                                 \verb|id,predict=clf.predict(gray_image[y:y+h,x:x+w|])|\\
                                                 confidence=int((100*(1-predict/300)))
                                                 conn=mysql.connector.connect(host="localhost".username="root".password="Dimpy#1609*".database="face_recognize
                                                 my cursor=conn.cursor()
                                                 my_cursor.execute("select Name from person where Person_ID="+str(id))
                                                 n=my_cursor.fetchone()
                                                 n="+".join(n)
                                                 my_cursor.execute("select Gender from person where Person_ID="+str(id))
                                                 r=my_cursor.fetchone()
r="+".join(r)
                                                 my_cursor.execute("select Department from person where Person_ID="+str(id))
                                                  d=my_cursor.fetchone()
                                                 d="+".join(d)
                                                 my cursor.execute("select Person ID from person where Person ID="+str(id))
                                                  i=my_cursor.fetchone()
                                                 i="+".join(i)
                                                 if confidence>77:
```

```
cv2.putText(img,f"Person_ID:{i}",(x,y-75),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                         cv2.putText(img,f"Name:{n}",(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
cv2.putText(img,f"Gender:{r}",(x,y-30),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
cv2.putText(img,f"Department:{d}",(x,y-5),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                         self.mark\_attendance(i,n,r,d)
                    else:
                         cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),3)
cv2.putText(img,"Unknown Face",(x,y-55),cv2.FONT_HERSHEY_COMPLEX,0.8,(255,255,255),3)
                    coord=[x,y,w,y]
               return coord
          def recognize(img,clf,faceCascade):
               coord=draw_boundray(img,faceCascade,1.1,10,(255,25,255),"Face",clf)
          faceCascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
          clf=cv2.face.LBPHFaceRecognizer_create()
          clf.read("classifier.xml")
          video_cap=cv2.VideoCapture(0)
          while True :
               ret,img=video_cap.read()
               img=recognize(img,clf,faceCascade)
               cv2.imshow("Welcome to face Recognition",img)
               if cv2.waitKey(1)==13:
                   break
          video_cap.release()
          cv2.destroyAllWindows()
if __name__=="__main__":
    root= Tk()
     obj=FaceDetector(root)
     root.mainloop()
```

In []: ▶