Importing Libraries

In [1]:
```
import os
import tkinter as tk
import cv2
import numpy as np

from tkinter import *
from PIL import Image, ImageTk
from keras.models import model_from_json
from keras.preprocessing import image
```

Loading Trained Model

In [2]:
```
emotion_model = model_from_json(open("Model/model.json", "r").read())
emotion_model.load_weights('Model/model.h5')
```

Emotion Dictionary Mapping

In [3]:
```
emotion_dict = {0: "   Angry   ", 1: "Disgusted", 2: "  Fearful  ", 3: "   Happy


emoji_dist={0:"./emojis/angry.png",2:"./emojis/disgusted.png",2:"./emojis/fearfu
```

Using OpenCV to check model on Live WebCam

In [*]:
```python
global last_frame1
last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
global cap1
show_text=[0]
def show_vid():
    cap1 = cv2.VideoCapture(0)
    if not cap1.isOpened():
        print("cant open the camera1")
    flag1, frame1 = cap1.read()
    frame1 = cv2.resize(frame1,(600,500))

    bounding_box = cv2.CascadeClassifier('cv_file/haarcascade_frontalface_defaul
    gray_frame = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    num_faces = bounding_box.detectMultiScale(gray_frame,scaleFactor=1.3, minNei

    for (x, y, w, h) in num_faces:
        cv2.rectangle(frame1, (x, y-50), (x+w, y+h+10), (255, 0, 0), 2)
        roi_gray_frame = gray_frame[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (
        prediction = emotion_model.predict(cropped_img)

        maxindex = int(np.argmax(prediction))
        cv2.putText(frame1, emotion_dict[maxindex], (x+20, y-50), cv2.FONT_HERSH
        show_text[0]=maxindex

    if flag1 is None:
        print ("Major error!")
    elif flag1:
        global last_frame1
        last_frame1 = frame1.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(pic)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
#         lmain.after(10, show_vid)

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


def show_vid2():
    frame2=cv2.imread(emoji_dist[show_text[0]])
    pic2=cv2.cvtColor(frame2,cv2.COLOR_BGR2RGB)
    img2=Image.fromarray(frame2)
    imgtk2=ImageTk.PhotoImage(image=img2)
    lmain2.imgtk2=imgtk2
    lmain3.configure(text=emotion_dict[show_text[0]],font=('arial',45,'bold'))
    lmain2.configure(image=imgtk2)

def run():
    while True:
        root.after(10)
        show_vid()
        show_vid2()
        root.update()

if __name__ == '__main__':
    root=tk.Tk()
    heading=Label(root,text="Photo to Emoji",pady=10, font=('Georgia',40,'bold')

    heading.pack()
    lmain = tk.Label(master=root,padx=50,bd=10)
    lmain2 = tk.Label(master=root,bd=10)

    lmain3=tk.Label(master=root,bd=10,fg="#CDCDCD",bg='black')
    lmain.pack(side=LEFT)
    lmain.place(x=50,y=100)
    lmain3.pack()
    lmain3.place(x=750,y=80)
    lmain2.pack(side=RIGHT)
    lmain2.place(x=700,y=180)


    root.title("Photo To Emoji")
    root.geometry("1200x690+50+10")
    root['bg']='black'

    exitbutton = Button(root, text='Quit',fg="red",command=root.destroy,font=('a
    startbutton = Button(root, text='Start',fg="green",command=run, font=('arial

    startbutton.place(x = 550, y = 620)
    exitbutton.place(x = 700, y = 620)

    root.mainloop()
```