Please READ the following to get a better understanding of the code.

Temperature Identity Code Documentation

By Ayush Singh

For explanation purposes, I will include a step-by-step explanation of the code.

Step 1: Accessing the Python Code
To view the code, please visit https://github.com/ayushcv/TemperatureIdentity

Once arrived, please click on the folder "Code(Temp Identity)"

Please ignore "_pycache_" (This is an automated file)

The final code is called "Final_code.py"

If interested, all extra codes and test cases are present in "Extra Code for testing cases"

Step 2: Opening the Python Code

You should now be looking at "Final_code.py" Click on it, and the code will be present. There should be a total of 203 lines.

Step 3: Understanding the Python Code

a. Libraries (Lines 1- 18)
   i. **Sys** and **os** libraries help set system-specific parameters and allow the use of the operating system.
   ii. **Cv2** and **NumPy** are OpenCV libraries that allow the use of computer vision in the code.
   iii. **Pyzbar.pyzbar** is used for understanding the sequences of the bar code.
   iv. **Time** is used to import the current time using the device's location.
   v. **RPI.GPIO** is used to get the servo readings
   vi. **Googleapiclient.HTTP** and **Google** are used to create the google drive service.
   vii. **SMBus** and **mlx90414** are used to get the infrared temperature sensor readings. This library allows certain methods and functions to be used. SMBUS allows the signal to be readable.

b. SetVariables (Lines 20-50)
    i. **Google Client Services**, **JSON files**, and **API Version** are all provided to allow the use of the Google Drive API. The JSON File acts as the encryption code for the access of the Google servers.
    ii. **Service** = The google drive services are created with the information above. Then the designation of the database file is provided with IDs.
    iii. **GPIO.setmode** = tells the Raspberry Pi to enable the output pins, and provides the location of the servos. (pin 11,13)
    iv. **Objects**, **facecascade**, and **cap** are set to enable the functionality of the computer vision. This creates the frame for the live video.
c. Barcode Database (Lines 52-56)
    i. The barcode database file is opened from the directory, and the data is stored in the **myDataList** variable.
d. Restart Function (Lines 59-61)
    i. **restart_program()** is used to restart the program once the while loop of each user ends. This allows a seamless behavior from a user flow perspective.
e. Main While Loop (Lines 64 - 199)
    i. Video capture is stored and displayed on the screen.
    ii. **For Loop** (barcode)
        1. Data is read from the database, and stored into **myData**.
    iii. **If Statement** checks if the barcode is authorized.
        1. IF TRUE
            a. If the barcode is authorized...OpenCV will run and convert the video stream into grayscale to create the face outlines.
            b. As the face is detected, a **For Loop** will create a rectangular box around the face and the forehead. These x and y values will also be used to create text on the screen.
            c. An Algorithm for x-axis and y-axis with ranges is created. These ranges are evenly divided to cover the whole range of movements. The ranges are calculated with averages and the degrees which are evenly divided.
                i. As the person moves from one range to the other, the servo movement occurs accordingly. The servo successfully follows the person's forehead.
            d. An **SMBus** is opened to allow the temperature readings to come at the exact moment. The bus is open with

binary digit 1 and the temperature is stored into **ter**. Then the bus is closed.

e. The database document is open with the var **r.**
f. **If Statement** creates a threshold for the temperature readings.
    1. If True, the Name/temp/time is stored in the **database.**
    2. If False, the Name/temp/time and a WARNING is stored into the database as the student's temperature may be harmful. A DANGER TEXT is displayed on the screen.
g. The obj becomes True and the loop is broken. The stream is broken and the students are good to go/ may be rejected as the DANGER sign is displayed if the temp is greater than the threshold.

2. ELSE:

a. A message is displayed that the user is un-authorized to enter the system. The error message is displayed on the screen using the **cv2.putText.**
b. The live video continues until the correct barcode is presented.

iv. **If statement** checks whether the above loop has been completed.

1. IF TRUE:
    a. The type of file is detected from the directory.
    b. Media = the media type file being uploaded into the cloud.
    c. A service is created and the files are updated using the google drive service API.
    d. Then the **database is** updated.

f. Restart_program() Line (203)

i. The program restarts to create seamless behavior for the next user.

Please READ the following to get a better understanding of the code.

Step 4: Accessing the Website Code

To view the code, please visit
https://github.com/ayushcv/ayushcv.github.io/tree/master/TEJ4MO

The final code is called "index.html"

If interested, all assets and images are present in "Assets, Images" The cascading style sheets are also present in these folders.

Step 5: Opening the Website Code

You should now be looking at "index.html" Click on it, and the code will be present. There should be a total of 189 lines.

Step 6: Understanding the Website Code

a. **Head** (lines 4 -11)
   i. This allows for the title of the web page to be saved and the CSS and JS files are imported for use.
b. **Body** (llnes 12-187)
   i. The first header of the website is present. The text is classified into columns and rows.
   ii. A Button is added to enable a scroll down functionality.
   iii. A container is used to set the placement of the text and features. This section includes the databases and the name of the classroom. It is set in a container provided by the CSS.
   iv. AN **IFRAME** IS CREATED TO MAKE A PORTAL BETWEEN THE DATABASE and the website. Updated at every input.
   v. A Scroll Button to go down the page is added again.
   vi. Section 3. This includes a couple of pictures and buttons which take you to the desired web pages for more information. The styling of the buttons is done in the CSS files and they are called with styles and set with HREF links for redirection to the various webpages.
   vii. THIS IS ANOTHER SCROLL BUTTON TO GO TO THE CONTACT ME SECTION.
   viii. The footer includes social media, and the links are there to go to the contact information.
      1. The class of the icons is brought from the CSS and the links are provided with the HREF.

ix. My Javascript files are called to use their functionalities to allow browser and scrolling capabilities.

Optional

Step 7: Javascript Code Overview

Please Access "main.js"

1. The Javascript file is used to create the breakpoints, and add the animations on the page load. The scrollable button is used to navigate through the webpage.

Step 8: CSS Code Overview

Please Access "main.css"

1. The CSS is made to set the methods, class, container, and boxes. Alongside that, the button styling all occurs here.
2. The file is very long as you have to configure each dimension, color-code, animation, and text.
3. There is a bird animation trails code in there, it is extra and is not used in the finalized HTML files as it is not necessary.

Thank you for your time. This is the end of the documentation.