#Import tkinter modules

import tkinter as tk

from tkinter import \*

from tkinter import ttk

import sqlite3

from matplotlib import pyplot as plt

Impact\_Detected\_Int\_list = []

Impact\_Detected\_Entry\_list = []

Impact\_Detected\_DateTime\_list = []

No\_Impact\_Detected\_Int\_list = []

No\_Impact\_Detected\_Entry\_list = []

No\_Impact\_Detected\_DateTime\_list = []

Temp\_Int\_list = []

Temp\_Entry\_list = []

Temp\_Times = []

Invalid\_Temp\_Int\_list = []

Invalid\_Temp\_Entry\_list = []

Invalid\_Temp\_DateTime\_list = []

# creating file path to connect to the dbfile, you will need to download sqlite pip in order to use this file and use your own file path where the DB bis located

dbfile = 'c:/Users/yossa/Downloads/Valid\_Temp\_Range\_Final\_Project\_1 (1).db'

# Create a SQL connection to our SQLite database

con = sqlite3.connect(dbfile)

# creating cursor

cur = con.cursor()

# reading all table names with the select sql command to copy all the records over from that table

table\_view\_Impact1 = [a for a in cur.execute("SELECT \*  FROM 'Impact\_Detected'")]

table\_view\_Impact2 = [a for a in cur.execute("SELECT \*  FROM 'No\_Impact\_Detected'")]

table\_view\_Temperature1 = [a for a in cur.execute("SELECT \* FROM 'Normal\_Temp\_Ranges'")]

table\_view\_Temperature2 = [a for a in cur.execute("SELECT \* FROM 'Invalid\_Temp\_Ranges'")]

table\_view\_Mositure1 = [a for a in cur.execute("SELECT \* FROM 'No\_Moisture\_Detected'")]

table\_view\_Moisture2 = [a for a in cur.execute("SELECT \* FROM 'Moisture\_Detected'")]

#This series of for loops will populate the different lists that will be used in the methods below for various reasons including generating graphs

for row in table\_view\_Mositure1 :

   Impact\_Detected\_Int\_list.append(row[1])

   Impact\_Detected\_Entry\_list.append(row[0])

   Impact\_Detected\_DateTime\_list.append(row[2])

for row in table\_view\_Moisture2 :

   Impact\_Detected\_Int\_list.append(row[1])

   Impact\_Detected\_Entry\_list.append(row[0])

   Impact\_Detected\_DateTime\_list.append(row[2])

for row in table\_view\_Impact1:

   Impact\_Detected\_Int\_list.append(row[1])

   Impact\_Detected\_Entry\_list.append(row[0])

   Impact\_Detected\_DateTime\_list.append(row[2])

for row in table\_view\_Impact2:

   No\_Impact\_Detected\_Int\_list.append(row[1])

   No\_Impact\_Detected\_Entry\_list.append(row[0])

   No\_Impact\_Detected\_DateTime\_list.append(row[2])

for row in table\_view\_Temperature1:

   Temp\_Int\_list.append(row[1])

   Temp\_Entry\_list.append(row[0])

   Temp\_Times.append(row[2])

for row in table\_view\_Temperature2:

   Invalid\_Temp\_Int\_list.append((row[1]))

   Invalid\_Temp\_Entry\_list.append(row[0])

   Invalid\_Temp\_DateTime\_list.append(row[2])

#---Initializes Main Window---#

adminDash = Tk()

adminDash.title("Admin Dashboard")

adminDash.geometry("500x500")

#---Constants---#

maintList = ("Battery #5679", "Battery #3565", "Battery #4890", "Battery #6780")

maint\_var = tk.StringVar(*value*=maintList)

#----Runs Maintenance Check---#

*def* runMaintCheck(*i*):

  #counter variables used to trigger the various windows: impact, moisture and temperature

    heat\_check = 0

    cold\_check = 0

    impact\_check = 0

    water\_check =  0

    if *i* == 1:

          #checks the given test table to see if the cold or hot window is activated

            for row in table\_view\_Temperature1:

                if row[1] > 80:

                    print ("This battery is too hot and requires maintenance")

                    heat\_check += 1

                elif row[1] < 30:

                    print ("This battery is too cold and requires maintenance")

                    cold\_check += 1

                else:

                    print("No maintance")

            for row in  table\_view\_Moisture2:

                if row[1] >= 1:

                    print("Moisture Detected")

                    water\_check += 1

            if water\_check > 0:

                waterWindow(*i*)

            if heat\_check > 0:

                heatWindow(*i*)

            elif heat\_check <= 0:

                    plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                    plot1.bar(Temp\_Entry\_list, Temp\_Int\_list)

                    plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                    plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                    plt.tight\_layout()

                    plt.savefig("Environment1.jpg")

                    plt.show()

            if cold\_check > 0:

                coldWindow(*i*)

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Temp\_Entry\_list, Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment1.jpg")

                plt.show()

            for row in table\_view\_Impact1:

                    if row[1] >= 140:

                        print("This battery suffered a serious impact")

                        impact\_check += 1

            if impact\_check > 0:

                impactWindow(*i*)

            elif impact\_check <= 0:

                plot3 = plt.subplot2grid((3, 3), (1, 0), *rowspan*=2)

                plot3.plot(Impact\_Detected\_Entry\_list, Impact\_Detected\_Int\_list)

                plot3.set\_title('Impact Graph')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment2.jpg")

                plt.show()

    elif *i* == 0:

            for row in  table\_view\_Mositure1:

                if row[1] >= 1:

                    print("Moisture Detected")

                    water\_check += 1

            for row in table\_view\_Temperature2:

                if row[1] > 80:

                    print ("This battery is too hot and requires maintenance")

                    print(heat\_check)

                    heat\_check += 1

                elif row[1] < 30:

                    print ("This battery is too cold and requires maintenance")

                    print(cold\_check)

                    cold\_check += 1

                else:

                    print("No maintance")

            if water\_check > 0:

                waterWindow(*i*)

            if heat\_check > 0:

                heatWindow(*i*)

            elif heat\_check <= 0:

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Temp\_Entry\_list, Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment.jpg")

                plt.show()

            if cold\_check > 0:

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Invalid\_Temp\_Entry\_list, Invalid\_Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment.jpg")

                plt.show()

                coldWindow(*i*)

            for row in table\_view\_Impact2:

                if row[1] >= 140:

                    print("This battery suffered a serious impact")

                    impact\_check += 1

                else:

                    print("No maintenance required")

            if impact\_check > 0:

                impactWindow(*i*)

            elif impact\_check <= 0:

                plot3 = plt.subplot2grid((3, 3), (1, 0), *rowspan*=2)

                plot3.plot(Impact\_Detected\_Entry\_list, Impact\_Detected\_Int\_list)

                plot3.set\_title('Impact Graph')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment2.jpg")

                plt.show()

    elif *i* == 2:

        for row in  table\_view\_Mositure1:

                if row[1] >= 1:

                    print("Moisture Detected")

                    water\_check += 1

        for row in table\_view\_Temperature2:

                if row[1] > 80:

                    print ("This battery is too hot and requires maintenance")

                    print(heat\_check)

                    heat\_check += 1

                elif row[1] < 30:

                    print ("This battery is too cold and requires maintenance")

                    print(cold\_check)

                    cold\_check += 1

                else:

                    print("No maintance")

        if water\_check > 0:

                waterWindow(*i*)

        if heat\_check > 0:

                heatWindow(*i*)

        elif heat\_check <= 0:

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Temp\_Entry\_list, Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment.jpg")

                plt.show()

        if cold\_check > 0:

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Invalid\_Temp\_Entry\_list, Invalid\_Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment.jpg")

                plt.show()

                coldWindow(*i*)

        for row in table\_view\_Impact2:

                if row[1] >= 140:

                    print("This battery suffered a serious impact")

                    impact\_check += 1

                else:

                    print("No maintenance required")

        if impact\_check > 0:

                impactWindow(*i*)

        elif impact\_check <= 0:

                plot3 = plt.subplot2grid((3, 3), (1, 0), *rowspan*=2)

                plot3.plot(Impact\_Detected\_Entry\_list, Impact\_Detected\_Int\_list)

                plot3.set\_title('Impact Graph')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment2.jpg")

                plt.show()

    elif *i* == 3:

        for row in  table\_view\_Mositure1:

                if row[1] >= 1:

                    print("Moisture Detected")

                    water\_check += 1

        for row in table\_view\_Temperature1:

                if row[1] > 80:

                    print ("This battery is too hot and requires maintenance")

                    print(heat\_check)

                    heat\_check += 1

                else:

                    print("No maintance")

        for row in table\_view\_Temperature1:

                    if row[1] < 75:

                        print ("This battery is too cold and requires maintenance")

                        print(cold\_check)

                        cold\_check += 1

                    else:

                        print("No maintance")

        if water\_check > 0:

                waterWindow(*i*)

        if heat\_check > 0:

                heatWindow(*i*)

        elif heat\_check <= 0:

                plot1 = plt.subplot2grid((3, 3), (0, 0), *colspan*=2)

                plot1.bar(Temp\_Entry\_list, Temp\_Int\_list)

                plot1.set\_title('Temperature Graph (Temps(F) over time(hr))')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment.jpg")

                plt.show()

        if cold\_check > 0:

                coldWindow(*i*)

        for row in table\_view\_Impact2:

                if row[1] >= 140:

                    print("This battery suffered a serious impact")

                    impact\_check += 1

                else:

                    print("No maintenance required")

        if impact\_check > 0:

                impactWindow(*i*)

        elif impact\_check <= 0:

                plot3 = plt.subplot2grid((3, 3), (1, 0), *rowspan*=2)

                plot3.plot(Impact\_Detected\_Entry\_list, Impact\_Detected\_Int\_list)

                plot3.set\_title('Impact Graph')

                plt.xlabel('Close this window when finished viewing', *fontsize*=10)

                plt.tight\_layout()

                plt.savefig("Environment2.jpg")

                plt.show()

#---Grabs Battery Selection and determines which menu to pull up---#

*def* batterySelect():

    print ("Button Pressed")

    maintLog = maintLB.curselection()

    print (maintLog)

    #find value of selected item

    if maintLog == (0,):

        print("Battery #5679")

        i =0

        runMaintCheck(i)

    elif maintLog == (1,):

        i=1

        print("Battery #3565")

        runMaintCheck(i)

    elif maintLog ==(2,):

        print("Battery #4890")

        i = 2

        runMaintCheck(i)

    elif maintLog == (3, ):

        print("Battery #6780")

        i = 3

        runMaintCheck(i)

    else:

        print("Invalid Entry")

#----Removes the item from Maintenance List and from battListTemp---#

*def* maintenanceComplete(*i*):

    print("Maintenance Complete")

    maintLB.delete(*i*, )

#----Create new GUI windows  for maintenance protocols---#

*def* heatWindow(*i*):

    mc = "false"

    protocolWindow = Toplevel(adminDash)

    protocolWindow.title("Maintenance Protocol")

    protocolWindow.geometry("500x500")

    maintLbl = Label(protocolWindow, *text*="OverHeating Protocol")

    maintLbl.pack(*pady*=10)

    protocolText = Label(protocolWindow, *height*=10, *width*=50, *text* = "Instruction for overheating maintenance protocol:\n1, Step 1: Seal the case. \nStep 2: Evacuate people from the area \nStep 3: Contact local fire department .")

    protocolText.pack(*pady*=10)

    maintComplete = Button(protocolWindow, *text*="Complete Maintenance", *command*=maintenanceComplete(*i*))

    maintComplete.pack(*pady*=10)

*def* coldWindow(*i*):

    protocolWindow = Toplevel(adminDash)

    protocolWindow.title("Maintenance Protocol")

    protocolWindow.geometry("500x500")

    maintLbl = Label(protocolWindow, *text*="Freezing Protocol")

    maintLbl.pack(*pady*=5)

    protocolText = Label(protocolWindow, *height*=10, *width*=50, *text* = "Instruction for freezing maintenance protocol:\nStep 1. Seal the case usign sealing feature\nStep 2. Activate teh artifical heating\nStep3. Repeat previous stages until temperature in optimal range.")

    protocolText.pack(*pady*=10)

    maintComplete = Button(protocolWindow, *text*="Complete Maintenance", *command*=maintenanceComplete(*i*))

    maintComplete.pack(*pady*=10)

*def* impactWindow(*i*):

    protocolWindow = Toplevel(adminDash)

    protocolWindow.title("Maintenance Protocol")

    protocolWindow.geometry("500x500")

    maintLbl = Label(protocolWindow, *text*="Impact Damage Protocol")

    maintLbl.pack(*pady*=5)

    protocolText = Label(protocolWindow, *height*=10, *width*=50, *text* = "Instruction for impact damage protocol: \nStep 1. Seal case using seal features\nStep 2. If the battery is leaking contact emergency authorities\nStep 3. If not leaking, return to fasility for inspection.")

    protocolText.pack(*pady*=10)

    maintComplete = Button(protocolWindow, *text*="Complete Maintenance", *command*=maintenanceComplete(*i*))

    maintComplete.pack(*pady*=10)

*def* waterWindow(*i*):

    protocolWindow = Toplevel(adminDash)

    protocolWindow.title("Maintenance Protocol")

    protocolWindow.geometry("500x500")

    maintLbl = Label(protocolWindow, *text*="Water Damage Protocol")

    maintLbl.pack(*pady*=5)

    protocolText = Label(protocolWindow, *height*=10, *width*=50, *text* = "Instruction for water damage protocol: \nStep 1. Activate case sealing feature\nStep 2. Activate water pumps to remove water\nStep 3. Recheck status and repeat steps if necissary.")

    protocolText.pack(*pady*=10)

    maintComplete = Button(protocolWindow, *text*="Complete Maintenance", *command*=maintenanceComplete(*i*))

    maintComplete.pack(*pady*=10)

maintenanceLbl = Label(adminDash, *text*="Maintenance Logs")

maintenanceLbl.pack()

#create Listbox for Maintenance Logs

maintLB= tk.Listbox(adminDash, *listvariable*=maint\_var, *height*=5, *selectmode*='single')

maintLB.pack(*pady*=5)

maintBtn = tk.Button(adminDash, *text*="View Maintenance Log", *command* = batterySelect)

maintBtn.pack(*pady*=5)

adminDash.mainloop()

con.close()