Developing a GUI-based BMI Calculator using Python & Tkinter

Table of Contents:

1- Introduction:

The objective of this page is to create a GUI-based BMI caculator using python3 and tkinter. We will start off by importing the necessary packages and establishing a SQL database for our application. Next, we will create several classes, each with their respective functions in which we outline the size of our application, the buttons we wish to have, and the commands and functionality we would like to utilize. We will then pull our file together and create a standalone executable file (bmi-calc.exe).

In order to visualize the calculater we plan to build, please see the following image:

title

In order to install tkinter, run the following code:

pip install tkinter

In order to install sqlite3, run the following code:

pip install sqlite3

2- Import Packages of Interest:

In [1]:
```
# Import packages:

# Import tkinter to build the GUI interface of this app:
from tkinter import *

# Import sqlite to store the data we will generate:
import sqlite3

# Import time to manage the time of our entries:
import time
import datetime

# import image to add BU logo:
from PIL import Image
from PIL import ImageTk, Image
```

3- Create and Connect to the SQL Database:

In [2]:
```
# Connect to the SQL database and create a shortcut for easy access:

BMIDB = sqlite3.connect("bmidatabase.db")
c = BMIDB.cursor()
```

4- Create the Welcome Page:

In [3]:
```
class Welcome():

    def __init__ (self, master):

        self.master = master
        self.master.geometry('260x260+100+200')
        self.master.title('Welcome')
        self.master.configure(background='white')

        import random
        for x in range(10):
            happycalc = random.randint(1,4)

        happy = ['Amazing','Wonderful','Great','Awesome', 'Perfect']
        happyword = happy[happycalc]

        finalHappyText = "The " + happyword + " Boston University BMI Calculat
or"

        #print(happyword)

        self.label1 = Label(self.master, text = finalHappyText, fg='red', back
ground = 'white').grid(row=1, column=1)


#         img = ImageTk.PhotoImage(Image.open("bu1.png"))
#         self.label2 = Label(self.master, image = img).grid(row=0, column=1)

        today = datetime.datetime.now()

        #print(today)

        repr(today)

        self.label3 = Label(self.master, text = today, fg='black', background
= 'white').grid(row=8, column=1)



        self.button1 = Button(self.master, text= "BMI Calculator", fg = 'blue'
, command = self.gotobmicalculator).grid(row=3, column=1)
        self.button2 = Button(self.master, text = "Records", fg = 'blue', comm
and = self.gotorecords).grid(row=5, column=1)
#         self.button3 = Button(self.master, text = "Settings",fg='blue',comman
d=self.gotosettings).grid(row=3,column=3)
        self.button4 = Button(self.master, text = "Exit",fg='blue',command=sel
f.exit).grid(row=7,column=1)

    def exit(self):
        # Create an exit
        self.master.destroy()

    def gotobmicalculator(self):
        # The actual calcualtor
        root2 = Toplevel(self.master)
        myGUIO = bmicalculator(root2)
    def gotorecords(self):
        #This is where the previous records are stored
        root2=Toplevel(self.master)
        mygui=records(root2)
```

5- Create the Main > BMI Calculator Page

In [4]:
```
class bmicalculator():
    #class created for the bmi calculator GUI
    def __init__(self,master):

        c.execute('CREATE TABLE IF NOT EXISTS BMIStorage(timestamp TEXT,bodyma
ssindex REAL,weightclass TEXT)')

        self.heightcm=DoubleVar()
        self.weightkg=DoubleVar()

        self.master=master
        self.master.geometry('260x250+100+200')
        self.master.title('BMI Calculator')
        self.master.configure(background='white')

        self.label2=Label(self.master,text='Welcome to the BMI Calculator',fg=
'red').grid(row=0,column=0)
        self.label2=Label(self.master,text='Please enter your height in centim
etres',fg='black').grid(row=3,column=0)
        self.label2=Label(self.master,text='Please enter your weight in kilogr
ams',fg='black').grid(row=4,column=0)

        self.myheight=Entry(self.master,textvariable=self.heightcm).grid(row=3
,column=1)
        self.myweight=Entry(self.master,textvariable=self.weightkg).grid(row=4
,column=1)
        self.button4=Button(self.master,text="Calculate BMI",fg='blue',command
=self.bmicalculation).grid(row=7,column=0)
        self.button5=Button(self.master,text="Exit",fg='blue',command=self.exi
t).grid(row=9,column=0)


    def bmicalculation(self):
        bmiheight=self.heightcm.get()
        bmiweight=self.weightkg.get()
        bmi= float((bmiweight)/((bmiheight / 100)**2))
        bmi = round(bmi, 4)
        self.bmi = bmi
        self.label1=Label(self.master,text='Your BMI is %.2f' % bmi).grid(row=
5,column=0)

        if bmi <= 18.5:
            self.label2=Label(self.master,text='You are slightly underweight.'
,fg='blue').grid(row=6,column=0)
            totalindex = 'underweight'
            self.totalindex = totalindex
        elif bmi >18.5 and bmi <25:
            self.label3=Label(self.master,text='You are in the healthy weight
 group.',fg='green').grid(row=6,column=0)
            totalindex = 'healthy'
            self.totalindex = totalindex
        elif bmi >= 25 and bmi < 30:
            self.label4=Label(self.master,text='You are slightly overweight.',
fg='orange').grid(row=6,column=0)
            totalindex = 'overweight'
            self.totalindex = totalindex
        elif bmi >=30:
            self.label5=Label(self.master,text='You are in the obese weight gr
oup.',fg='red').grid(row=6,column=0)
            totalindex = 'obese'
            self.totalindex = totalindex

        if bmi >0 and bmi <9999999999999999999:
            self.button6=Button(self.master,text="Store Data",fg='red',command
=self.dynamic_data_entry).grid(row=8,column=0)

    def dynamic_data_entry(self):
        global dynamic_data_entry

        timestamp = str(datetime.datetime.now().date())

        bodymassindex = self.bmi
        weightclass = self.totalindex

        c.execute("INSERT INTO BMIStorage (timestamp, bodymassindex, weightcla
ss) VALUES (?, ?, ?)",(timestamp, bodymassindex, weightclass))
        BMIDB.commit()

        self.writetodatabase()

    def writetodatabase(self):
        for i in range(1):
            time.sleep(1)


    def exit(self):

        self.master.destroy()
```

6- Create the Main > Records Page:

In [5]:
```
class records():

    def __init__(self,master):
        self.master=master
        self.master.geometry('500x500+100+200')
        self.master.configure(background='white')

        self.master.title('Records')
        self.connection = sqlite3.connect('bmidatabase.db')
        self.cur = self.connection.cursor()
        self.dateLabel = Label(self.master, text="Date", width=10)
        self.dateLabel.grid(row=0, column=0)
        self.BMILabel = Label(self.master, text="BMI", width=10)
        self.BMILabel.grid(row=0, column=1)
        self.stateLabel = Label(self.master, text="Status", width=10)
        self.stateLabel.grid(row=0, column=2)
        self.showallrecords()
        self.button4=Button(self.master,text="Return",fg='red',command=self.ex
it).grid(row=7,column=0)

    def showallrecords(self):
        data = self.readfromdatabase()
        for index, dat in enumerate(data):
            Label(self.master, text=dat[0]).grid(row=index+1, column=0)
            Label(self.master, text=dat[1]).grid(row=index+1, column=1)
            Label(self.master, text=dat[2]).grid(row=index+1, column=2)

    def readfromdatabase(self):
        self.cur.execute("SELECT * FROM BMIStorage")
        return self.cur.fetchall()

    def exit(self):
        self.master.destroy()
```

7- Initiate the Application:

In [6]:
```
def main():
     root = Tk()
     myGUIWelcome = Welcome(root)
     root.mainloop()

if __name__ == '__main__':
     main()
```

8- Export to Create a Standalone Executable:

8.1 Navigate to your terminal command and install pyinstaller:

```
>> pip install pyinstaller
```

8.2 Save your file with a .py extension by going to File > Download As > Python File

8.3 Create the executable in your terminal command line:

```
>>> pyinstaller.exe --onefile --icon=bmi_icon.ico .\bmi_calculator_gui.py
```

...where:

```
>>> pyinstaller.exe
```

... is the package we are using to create the executable

```
>>> --onefile
```

... will produce one output file instead of multiple

```
>>> --icon=bmi_icon.ico
```

... will create a icon for your application assuming you have an icon file to use

```
>>> .\bmi_calculator_gui.py
```

... is the .py file you created in the previous step.

10- Summary:

In summary, we were able to create several classes each pertaining to a "view" or "window" in our application. Each class contained a series of defined functions that outlined a command of some sort for the program to execute. We then initiated the application to produce a GUI-based BMI Calculator. We then used pyinstaller to create a standalone executable file.