

**M.H. Saboo Siddik College of**

**Engineering**

(Department of Computer Engineering)

A Mini-Project Report on

**BMI CALCULATOR USING GUI**

Submitted in Partial Fulfilment of Requirements of the Course

**OPEN SOURCE TECH LAB**

By

**Arbaz Khan (CS3117019)**

**Moiz Sitabkhan(CS3117029)**

**Sreejith Nair (CS3117030)**

Under the Guidance of

**Prof. Asadullah Shaikh**

**PROBLEM DEFINITION:-**

To develop BMI Calculator using Python ( GUI & Database )

**BMI(BODY MASS INDEX)**

BMI stands for Body Mass Index is a measurement of a person’s weight with respect to his or her height. It is more of an indicator than a direct measurement of a person’s total body fat.

BMI, more often than not, correlates with total body fat. This means that as the BMI score increases, so does a person’s total body fat.

**BMI CALCULATIONS**

BMI in an individual is calculated by the use of a mathematical formula. It can also be estimated using tables in which one can match height in inches to weight in pounds to estimate BMI. There are convenient calculators available on internet sites that help calculate BMI as well.

The formula is - BMI = (Weight in kilograms) divided by (Height in metres squared)

A normal BMI score is one that falls between 18.5 and 24.9. This indicates that a person is within the normal weight range for his or her height. A BMI chart is used to categorize a person as underweight, normal, overweight, or obese.

|  |  |
| --- | --- |
| **Body Mass Index (BMI)** | **Weight Status** |
| Below 18.5 | Underweight |
| 18.5 - 24.9 | Normal |
| 25.0 - 29.9 | Overweight |
| 30 plus | Obese |

For example those with a high BMI are at risk of:-

* high blood cholesterol or other lipid disorders
* type 2 diabetes
* heart disease
* stroke
* high blood pressure

METHOD WE HAD APPLIED FOR CALCULATING

BMI USING PYTHON PROGRAMMGING.

**GUI(Graphic User Interface) in Python**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter:

* Importing the module – tkinter
* Create the main window (container)
* Add any number of widgets to the main window
* Apply the event Trigger on the widgets.

There are two main methods used you the user need to remember while creating the Python application with GUI.

**1.Tk(screenName=None, baseName=None, className=’Tk’, useTk=1):** To create a main window, tkinter offers a method ‘Tk(screenName=None, baseName=None, className=’Tk’, useTk=1)’. To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

Syntax :- m=tkinter.Tk()

where m is the name of the main window object.

**2. mainloop():** There is a method known by the name mainloop() is used when you are ready for the application to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

* There are a number of widgets which you can put in your tkinter application. Some of the major widgets we had used in our project are explained below:

**1. Button**:To add a button in your application, this widget is used.  
The general syntax is: **w=Button(master, option=value)**

master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

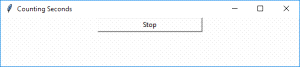
Syntax:- **import** tkinter as tk

r **=** tk.Tk()

r.title('Counting Seconds')

button **=** tk.Button(r, text**=**'Stop', width**=**25, command**=**r.destroy) button.pack()

r.mainloop()

Output:  
 

**2. Entry:**It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.  
The general syntax is: **w=Entry(master, option=value)**

master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

Syntax:- **from** tkinter **import** **\***

master **=** Tk()

Label(master, text**=**'First Name').grid(row**=**0)

Label(master, text**=**'Last Name').grid(row**=**1)

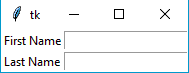
e1 **=** Entry(master)

e2 **=** Entry(master)

e1.grid(row**=**0, column**=**1)

e2.grid(row**=**1, column**=**1)

mainloop()

Output:  
 

**3. Frame:** It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.

The general syntax is: **w = Frame(master, option=value)**

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

**from** tkinter **import** **\***

root **=** Tk()

frame **=** Frame(root)

frame.pack()

bottomframe **=** Frame(root)

bottomframe.pack( side **=** BOTTOM )

redbutton **=** Button(frame, text **=** 'Red', fg **=**'red')

redbutton.pack( side **=** LEFT)

greenbutton **=** Button(frame, text **=** 'Brown', fg**=**'brown')

greenbutton.pack( side **=** LEFT )

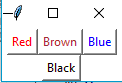
bluebutton **=** Button(frame, text **=**'Blue', fg **=**'blue')

bluebutton.pack( side **=** LEFT )

blackbutton **=** Button(bottomframe, text **=**'Black', fg **=**'black')

blackbutton.pack( side **=** BOTTOM)

root.mainloop()

Output:  
 

**4.CheckButton:** To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is:

**w = CheckButton(master, option=value)**

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

**Syntax:- from** tkinter **import** **\***

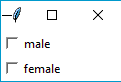
master **=** Tk()

var1 **=** IntVar()

Checkbutton(master, text**=**'male', variable**=**var1).grid(row**=**0, sticky**=**W)

var2 **=** IntVar()

Checkbutton(master, text**=**'female', variable**=**var2).grid(row**=**1, sticky**=**W) mainloop()

Output:  
 

**Database Programming in Python**

Pyton Database:-

Python Database.   
Data is retrieved from a database system using the SQL language.Data is everywhere and software applications use that. Data is either in memory, files or databases.

Python has bindings for many database systems including MySQL, Postregsql, Oracle, Microsoft SQL Server and Maria DB.  
  
One of these database management systems (DBMS) is called SQLite.  SQLite was created in the year 2000 and is one of the many management systems in the database zoo.

SQL is a special-purpose programming language designed for managing data held in a [databases](https://pythonspot.com/python-database/). The language has been around since 1986 and is worth learning. The [is an old funny video about SQL](https://www.youtube.com/watch?v=5ycx9hFGHog)

SQLite:-

SQLite, a relational database management system.SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain.

It is a self-contained, serverless, zero-configuration, transactional SQL database engine. The SQLite project is sponsored by Bloomberg and Mozilla.

SQL Create and Insert

The script below will store data into a new database called user.db

|  |
| --- |
| *#!/usr/bin/python*  *# -\*- coding: utf-8 -\*-*    **import** sqlite3 **as** lite  **import** sys    con = lite.connect('user.db')    **with** con:    cur = con.cursor()  cur.execute("CREATE TABLE Users(Id INT, Name TEXT)")  cur.execute("INSERT INTO Users VALUES(1,'Michelle')")  cur.execute("INSERT INTO Users VALUES(2,'Sonya')")  cur.execute("INSERT INTO Users VALUES(3,'Greg')") |

SQLite is a database management system that uses tables. These tables can have relations with other tables: it’s called relational database management system or RDBMS.  The table defines the structure of the data and can hold the data.  A database can hold many different tables. The table gets created using the command:

|  |
| --- |
| cur.execute("CREATE TABLE Users(Id INT, Name TEXT)") |

We add  records into the table with these commands:

|  |
| --- |
| cur.execute("INSERT INTO Users VALUES(2,'Sonya')")  cur.execute("INSERT INTO Users VALUES(3,'Greg')") |

**CODE**

from tkinter import \*

import sqlite3

import time

import datetime

conn = sqlite3.connect('bmidatabase.db')

c = conn.cursor()

class Welcome():

#This is the class defining the first welcoming window. This window is used to navigate between previous weights and the calculator.

def \_\_init\_\_(self,master):

#This is the GUI for the starting Menu area. Features three buttons for navigating towards the BMI Calculator, the Records where we have to store variables entered and the exit#

self.master=master

self.master.geometry('170x110+100+200')

self.master.title('Welcome!')

self.label1=Label(self.master,text='Welcome to the BMI Calculator',fg='red').grid(row=0,column=1)

self.button1=Button(self.master,text="BMI Calculator",fg='green',command=self.gotobmicalculator).grid(row=1,column=1)

self.button2=Button(self.master,text="Records",fg='blue',command=self.gotorecords).grid(row=2,column=1)

self.button3=Button(self.master,text="Exit",fg='red',command=self.exit).grid(row=3,column=1)

def exit(self):

#Exit protocol for the exit button. This part is completely done.#

self.master.destroy()

def gotobmicalculator(self):

#This is the BMI Calculator GUI#

root2=Toplevel(self.master)

myGUI=bmicalculator(root2)

def gotorecords(self):

#This is where the previous records of BMI will be kept, hasn't been put in yet#

root2=Toplevel(self.master)

mygui=records(root2)

class bmicalculator():

#class created for the bmi calculator GUI and processing the numbers (pain in the ass to make)#

def \_\_init\_\_(self,master):

c.execute('CREATE TABLE IF NOT EXISTS BMIStorageNew(name TEXT,timestamp TEXT,bodymassindex REAL,weightclass TEXT)')

self.heightcm=DoubleVar()

self.weightkg=DoubleVar()

self.nametxt=StringVar()

self.master=master

self.master.geometry('250x200+100+200')

self.master.title('BMI Calculator')

self.label2=Label(self.master,text='Welcome to the BMI Calculator',fg='red').grid(row=0,column=0)

self.label3=Label(self.master,text='Please enter your Name',fg='black').grid(row=3,column=0)

self.label3=Label(self.master,text='Please enter your height in centimetres',fg='black').grid(row=4,column=0)

self.label4=Label(self.master,text='Please enter your weight in kilograms',fg='black').grid(row=5,column=0)

self.myName=Entry(self.master,textvariable=self.nametxt).grid(row=3,column=1)

self.myheight=Entry(self.master,textvariable=self.heightcm).grid(row=4,column=1)

self.myweight=Entry(self.master,textvariable=self.weightkg).grid(row=5,column=1)

self.button4=Button(self.master,text="Calculate BMI",fg='red',command=self.bmicalculation).grid(row=7,column=0)

self.button5=Button(self.master,text="Exit",fg='red',command=self.exit).grid(row=9,column=0)

def bmicalculation(self):

bmiName=self.nametxt.get()

print(bmiName)

bmiheight=self.heightcm.get()

print (bmiheight)

bmiweight=self.weightkg.get()

bmi= float((bmiweight)/((bmiheight / 100)\*\*2))

self.bmi = bmi

print (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='This places you in the underweight group.',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='This places you 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='This places you in the overweight group.',fg='orange').grid(row=6,column=0)

totalindex = 'overweight'

self.totalindex = totalindex

elif bmi >=30:

self.label5=Label(self.master,text='This places you in the obese group.',fg='red').grid(row=6,column=0)

totalindex = 'obese'

self.totalindex = totalindex

if bmi >0 and bmi <999999999999999999999:

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

#this is what adds the data to the database. Bmi has to be changed to the value of bmi and weightclass has to be change to the weightclass

Name = str(self.nametxt.get())

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

bodymassindex = self.bmi

weightclass = self.totalindex

c.execute("INSERT INTO BMIStorageNew (Name, timestamp, bodymassindex, weightclass) VALUES (?, ?, ?, ?)",(Name, timestamp, bodymassindex, weightclass))

conn.commit()

self.writetodatabase()

def writetodatabase(self):

for i in range(1):

time.sleep(1)

c.close()

conn.close()

def exit(self):

#Exit protocol for the exit button. This part is completely done.#

self.master.destroy()

class records():

#class created to see records that have been previously inputted#

def \_\_init\_\_(self,master):

self.master=master

self.master.geometry('250x200+100+200')

self.master.title('Records')

self.connection = sqlite3.connect('bmidatabase.db')

self.cur = self.connection.cursor()

self.NameLabel = Label(self.master, text="Name", width=15)

self.NameLabel.grid(row=0, column=0)

self.dateLabel = Label(self.master, text="Date", width=10)

self.dateLabel.grid(row=0, column=1)

self.BMILabel = Label(self.master, text="BMI", width=10)

self.BMILabel.grid(row=0, column=2)

self.stateLabel = Label(self.master, text="Status", width=10)

self.stateLabel.grid(row=0, column=3)

self.showallrecords()

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)

Label(self.master, text=dat[3]).grid(row=index+1, column=3)

def readfromdatabase(self):

self.cur.execute("SELECT \* FROM BMIStorageNew")

return self.cur.fetchall()

def main():

root=Tk()

myGUIWelcome=Welcome(root)

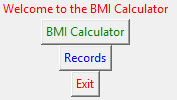
root.mainloop()

if \_\_name\_\_ == '\_\_main\_\_':

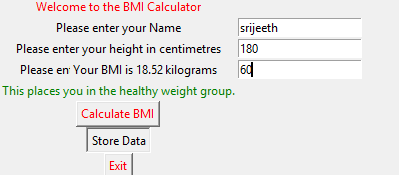
main()

**OUTPUT:-**

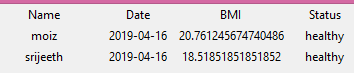
-->Welcome page



-->BMI Calculation



-->Records



**CONCLUSION**

In this project we have calculated bmi index of a person.

We have used GUI and Database programming in python to create a bmi calculator and keep records of all the bmi indexes of all the people whose bmi has been calculated and also show them in which weight category they are and also keep record of that.

Database programming becomes very easy in python because it is arguably one of the most effcient languages also platfrom independent and supports sql servers.

Graphic user interface in python is very easy to create and is very user-friendly and makes it very easy to create a attractive and colourful GUI.

**Refrences:-**

* pythonspot.com
* opensourceforu.com
* google.com