

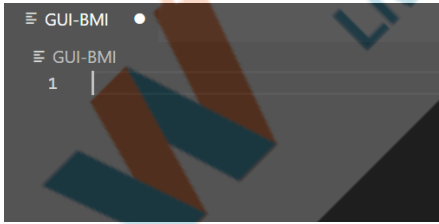
Topic	Introduction to Tkinter	
Class Description	Students will learn about Python Tkinter package and will make a graphical user Interface based on BMI Calculator	
Class	C-201	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> <li>• Introduction to Tkinter</li> <li>• Graphical user Interface</li> <li>• BMI Calculator</li> </ul>	
Resources Required	<ul style="list-style-type: none"> <li>• Teacher Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> <li>○ Visual Studio Code</li> </ul> </li> <li>• Student Resources:               <ul style="list-style-type: none"> <li>○ Laptop with internet connectivity</li> <li>○ Earphones with mic</li> <li>○ Notebook and pen</li> <li>○ Visual Studio Code</li> </ul> </li> </ul>	
Class structure	<b>Warm-Up</b> <b>Teacher - led Activity 1</b> <b>Student - led Activity 1</b> <b>Wrap-Up</b>	<b>10 mins</b> <b>10 mins</b> <b>20 mins</b> <b>5 mins</b>
<b>WARM UP SESSION - 10mins</b>		
<b>Teacher Action</b>		<b>Student Action</b>
<i>Hey &lt;student's name&gt;. How are you? It's great to see you!            Are you excited to learn something new today?</i>		<b>ESR:</b> Hi, thanks, Yes I am excited about it!

Q&A Session	
Question	Answer
Which of the following are not a socket property?  A. close () B. accept () C. bind () D. start ()	D
Which method is used to close the Socket?  A. socket.flush () B. Socket.close () C. socket.close () D. socket. destroy ()	B
TEACHER-LED ACTIVITY - 10mins	
Teacher Initiates Screen Share	
<b>ACTIVITY</b> <ul style="list-style-type: none"> <li>• Learning about Tkinter</li> <li>• Graphical user interface for BMI Calculator</li> </ul>	
Teacher Action	Student Action
In today's session we are going to do a very interesting thing which you have never done before!	<b>ESR:</b>
Are you excited?	<b>ESR:</b>
Did you ever try to make your own Graphical user interface?	Yes
You know what is GUI OR Graphical user interface?	<b>ESR:</b>

<p>A <b>graphical user interface or GUI</b> is an <b>application</b> that has buttons, windows, and lots of other widgets that you can use to interact with your application easily.</p> <p>Like your windows, calculator etc.</p>	<p>Yes</p>
<p>Do you think health is important?</p> <p>What do you do to keep yourself healthy?</p> <p>And how do you keep control of your weight?</p> <p>Is there any weight management system which tells us we need to gain or lose weight?</p> <p>Great!</p>	<p><b>ESR:</b> Yes! Walking, running, Cycling etc..</p> <p><b>ESR</b> Varied!</p>
<p>What if we make our own GUI or Graphical User Interface to keep check on our weight?</p> <p>Have you heard about the BMI Calculator?</p> <p>What is the purpose of using BMI Calculator?</p> <p>Perfect!</p> <p><b>BMI (Body Mass Index)</b> is a type of calculator which is generally used to measure body fat based on height and weight of a person.</p> <p>Let's make our own interface for the BMI.</p>	<p><b>ESR</b> Yes</p> <p>It will calculate our weight according to our height and weight measurements</p>
<p>But now the main question arises how we can make that happen.</p> <p>You know there is one module in python known as <b>Tkinter</b>.</p> <p>With the <b>Tkinter</b> module we can make our own graphical user interface.</p>	

<p>You can make your own things like calculators, games and much more</p>	
<p>Let's develop an application that will calculate your BMI. It will let you know in which category the user will fall in, if they are <b><i>underweight, overweight or obese</i></b>.</p>	
<p>With so much junk food available nowadays through countless food delivery apps, one might neglect our health and not know about it at all.</p> <p>Do you eat a lot of junk food?</p>	<p><b>ESR:</b> Varied!</p>
<p>Whenever we want to make GUI or graphical user interface you just need to keep in mind two steps:</p> <ul style="list-style-type: none"> <li>• The design of our GUI</li> <li>• What happens behind the scenes</li> </ul> <p>Now it's not like we have not built the GUI up until now. All the UI in our mobile apps and websites is also a GUI, but <b>Tkinter</b> helps us build GUI for the desktop.</p> <p>See how you open a folder and it's content appears in a box -</p> <p><i>Teacher opens any folder in their device</i></p> <p>This box in which the content appears is a graphical representation of the contents inside it. We can also check it's content from the command prompt/terminal by navigating into this directory and using the command</p> <p><b>Windows -</b> <b>dir</b></p> <p><b>Linux/MacOS -</b> <b>ls</b></p> <p><i>Teacher gives a demonstration to the student from the</i></p>	

<i>command prompt/terminal</i>	
<p>Now let's discuss the design part first!</p> <p>To develop a GUI application in python, we use a package called Tkinter. It will provide you with many GUI components called widgets, which we need to add on our screen.</p> <p>Now you must be wondering what widgets are!</p>	
<p><b>Tkinter</b> provides us with a variety of common GUI elements which we can use to build our interface.</p> <p>Some of the few common widgets available are:</p> <ul style="list-style-type: none"> <li>• <b>Button</b>: When clicked, an operation can be triggered</li> <li>• <b>Label</b>: to display something on-screen</li> <li>• <b>Entry</b>: to get a single line user input</li> <li>• <b>List box</b>: provides a list of options</li> <li>• <b>Label frame</b>: Used for results to display in a frame.</li> </ul> <p>And there are many more available as per application design</p>	
<p>You will be surprised to know that to Develop a GUI application using tkinter we need steps and <b>These steps remain the same for all the GUI applications.</b></p> <ol style="list-style-type: none"> <li>1. Import Tkinter module</li> <li>2. Create your application window</li> <li>3. Add widgets to the window             <ol style="list-style-type: none"> <li>1. Create the widget</li> <li>2. Place it on screen</li> </ol> </li> <li>4. Call main loop</li> </ol>	
The second part was, what happens behind the scenes of the GUI?	<b>ESR:</b> Varied!

<p>We need to write a logic part for our application like what will happen when we click a button. All the user interactivity through buttons, etc. has a defined logic.</p> <p>Remember how we used to define logic for all the button clicks, etc. in JavaScript and also in React Native?</p> <p>Now how will it take user input?</p> <p>This part is the same as you were doing in all your programming parts. We can create input fields using tkinter.</p>	<p><b>ESR:</b> Varied</p>
<p><i>Teacher opens <a href="#">Teacher Activity 1</a> to refer to the <b>Tkinter</b> library's documentation</i></p> <p>Let's start our GUI</p>	<p><i>Student opens <a href="#">Student Activity 1</a> to refer to the <b>Tkinter</b> library's documentation</i></p>
<p>Let's Create Client side</p> <p><i>Teacher opens a new Visual Studio Code and creates a file called <b>GUI_BMI.py</b></i></p> 	
<p>As discussed, total 4 steps for design part, so let's start with first part</p> <p>First step to import tkinter,</p> <pre>from tkinter import *</pre> <p>Now in the second step and very main step we will create a parent window for the</p>	

graphical user interface. Here you need to create one variable which is used to represent the parent window.

create a main parent window, tkinter offers a method Tk.

```
window=Tk()
```

After creating the parent window, we will add all components inside our main parent window.

It's time to set the title and size for our parent window. Use **title()** method and for size, use **geometry()** property of tkinter..

```
window.title('BMI Calculator')  
window.geometry("400x400")
```

Now it's time to select color for your parent window so that we can use the **configure ()** method along with bg ().

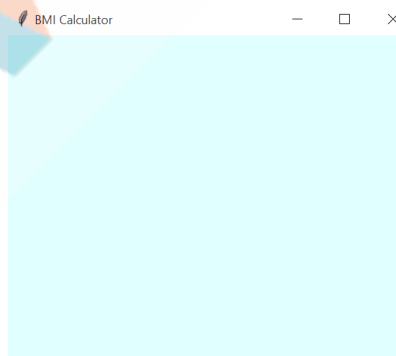
Here bg () is used for background color.

For coloring the window, we need to use **configure()**

```
window.configure(bg='lightcyan')
```

Look, our parent main window is prepared.

This is how it will look like this:



Next, in the third step add widgets on screen.

This third step can be done in two parts:

- Create Widget
- Place Widget

**Create Widget:** There are a number of widgets which you can put in your tkinter application. In our GUI-BMI calculator we need labels, entry box, button and label Frame.

**Place widget:** Set the geometric configuration of the widgets on the main window. We will use the place () method to organize the widgets by placing them on the main screen.

To add a label, we need to create one variable which is equal to **Label ()** widget. Now set your label inside the parent window, that's why we are using the window inside the label and all other widget attributes inside the label. Now set different attributes for labels for your parent window.

We are using label attributes like text which we want to show on your window, foreground color (fg), background color(bg), font, font size, border(bd).

```
heading_label=Label(window, text="BMI CALCULATOR", fg = "black", bg = "lightcyan", font=("Calibri", 20),bd=5)
```

Here app\_label is variable = Label is widget () which we are creating for parent window, i.e. window inside Label () widget and then attributes foreground color (fg), background color(bg), font , font size, border(bd).

Now it's time to place your widget on the screen that will use the **place()** method. Here x, y are coordinates according to your parent window.

```
heading_label.place(x=50, y=20)
```

Now create the name label using the same method and place it on the parent window.

```
name_label=Label(window, text="Your Name", fg = "black", bg = "lightcyan", font=("Calibri", 12),bd=1)  
name_label.place(x=20, y=90)
```

*widgets like Button, Label, Entry box and Label frame always take first letter upper case*



Now in the same way will create an entry box: Entry box is used to take input from the screen.

Create variable equal to Entry () widget, window inside the widget and then set attributes for entry box

```
username=Entry(window, text="", bd=2, width=22)
username.place(x=150, y=92)
```

To show output, we need to create label frame,

A **LabelFrame ()** is different from a normal Label (),

LabelFrame's primary purpose is to act as a spacer or container for output layouts.

This **LabelFrame ()** widget has the features of a frame, plus the ability to display a label result.

So first we will create a result frame container.

```
result_frame = LabelFrame(window,text="Result", bg = "lightcyan", font=("Calibri", 12))
result_frame.pack(padx=20, pady=20)
result_frame.place(x=20,y=300)
```

In the Label Frame () we are using **pack ()** to pack the text on the window

Now create Label () to display label inside your result frame label frame

```
result_label=Label(result_frame,text=" ", bg = "lightcyan", font=("Calibri", 12), width=33)
result_label.place(x=20,y=20)
result_label.pack()
```

*Here instead of a window, we will use the result frame as we are showing our result in this label frame container.*

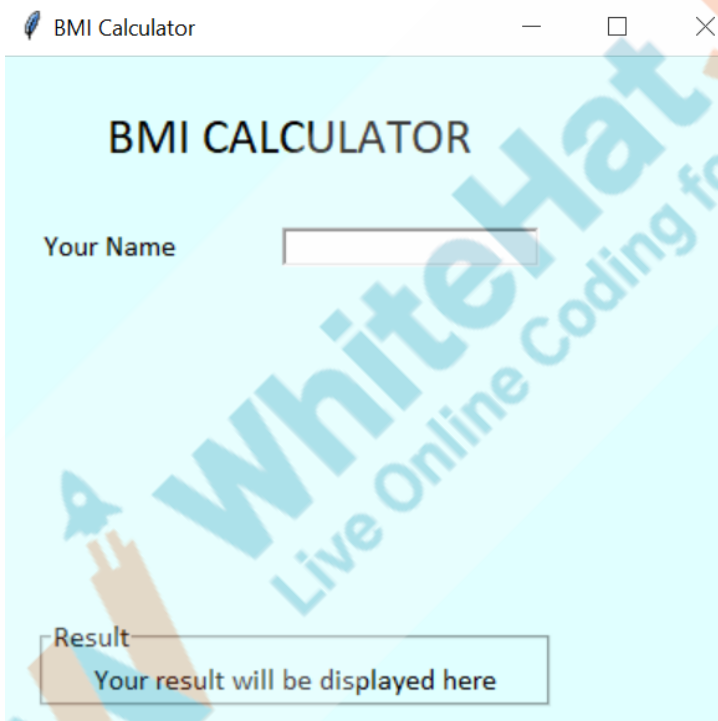
Fourth step is to call the main loop ()

mainloop () that is used when your application is ready to run. mainloop () is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
window.mainloop()
```

*Note: window.mainloop () will always come at the end of the program all widgets and functions will come before that. When a student will add other widgets it will come before window.mainloop*

Output will look like this:



**Teacher Stops Screen Share**

**STUDENT-LED ACTIVITY - 20 mins**

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Fullscreen.

**ACTIVITY**

- Create Height & Weight Label
- Create Entry Box
- Create function for Button
- Create Output \_Message

Teacher Action	Student Action
<p><i>Guide the student to get the boilerplate code from Student Activity 2</i></p> <p><i>Teacher helps the student in writing the code</i></p>	<p><i>Student clones the code from Student Activity 2</i></p> <p><i>Student writes the code</i></p>
<p>Create height label using <b>Label ()</b> widget and the place it on parent window () using <b>place ()</b></p> <pre>height_label=Label(window, text="Enter Height (cm)", fg = "black", bg = "lightcyan", font=("Calibri", 12)) height_label.place(x=20, y=140)</pre> <p>Create entry box using <b>Entry ()</b> widget and the place it on parent window () using <b>place()</b></p> <pre>height_entry=Entry(window, text="", bd=2, width=15) height_entry.place(x=150, y=142)</pre>	
<p>Create weight label using <b>Label ()</b> widget and weight entry () using Entry () widget and the place it on parent window () using <b>place ()</b></p> <pre>weight_label=Label(window, text="Enter Weight (Kg)", fg = "black", bg = "lightcyan", font=("Calibri", 12)) weight_label.place(x=20, y=185) weight_entry=Entry(window, text="", bd=2, width=15) weight_entry.place(x=150, y=187)</pre>	
<p>Now it's time to create event i.e., Button widget ()</p>	

An event is an action, which when occurs, can trigger another action.  
Example: clicking a button (an action) will calculate your BMI.

Create button () widget along then write the function for Button

Let's create function first:

We need to calculate Body mass index as per weight and height of the user.

*Note: Write function after window.config ()*

```
def calculate_bmi():  
    weight = int(weight_entry.get())  
    height = int(height_entry.get())/100  
    bmi = weight/(height*height)  
    bmi = round(bmi, 1)  
  
    name = username.get()
```

Now get the user `_weight`, convert it to integers, and then store the value in the variable `kg`.

Get the user height, convert it into integers, divide the result with 100 so that centimeters become meters, and then store it in a variable `height`.

Now create one variable `bmi` to store output which we will get using the BMI formula. i.e., **`weight/height*height`**.

We need to round off the result using `round ()` function, because it appears in multiple decimal values, with round function it looks simplified & easy to read.

<p>Get the name from the user using the get () command. <i>Teacher helps the student in writing the code</i></p>	<p><i>Student writes the code</i></p>
<p>To show your message_ output we need to destroy the result_label.</p> <pre>result_label.destroy()</pre>	<p><i>Student writes the code</i></p>
<p>Now it's time to display the final result depending upon the BMI value &amp; BMI category it belongs to.</p> <pre>msg=""</pre> <p>But before that we need to create variable</p> <p>We will write If-else conditions for checking BMI category:</p> <p>If bmi &lt;= 18.5 then display message "you are underweight"          Elseif bmi between 18.5–24.9 then display message "you are in normal range"          Elseif bmi between 25–29.9 then display message "you are overweight"          Elseif bmi between 25–29.9 then display message "you are obese"          Else display message "something went wrong"</p> <pre>msg="" if bmi &lt; 18.5:     msg="you areUnderweight" elif bmi &gt; 18.5 and bmi &lt;=24.9:     msg="is in Normal Range" elif bmi &gt; 25 and bmi &lt;=29.9:     msg="you are Overweight" elif bmi &gt; 30:     msg="you are Obese" else:     msg="Something Went Wrong"</pre>	
<p>Now It's time to display the output_message in the result frame container.</p>	

```
output_message=Label(result_frame,text=name+" , your BMI is "+str(bmi)+" and "+msg, bg = "lightcyan", font=("Calibri", 12), width=42)
output_message.place(x=20,y=40)
output_message.pack()
```

Create output message variable for Label widget (), inside Label widget use Result frame, now in place of text use your name variable, use string formatting to display your bmi value and message

Create Button () Widget and place the widget on the screen.

```
calculate_button=Button(window,text="CALCULATE",fg = "black", bg = "cyan",bd=4,command=calculate_bmi)
calculate_button.place(x=20,y=250)
```

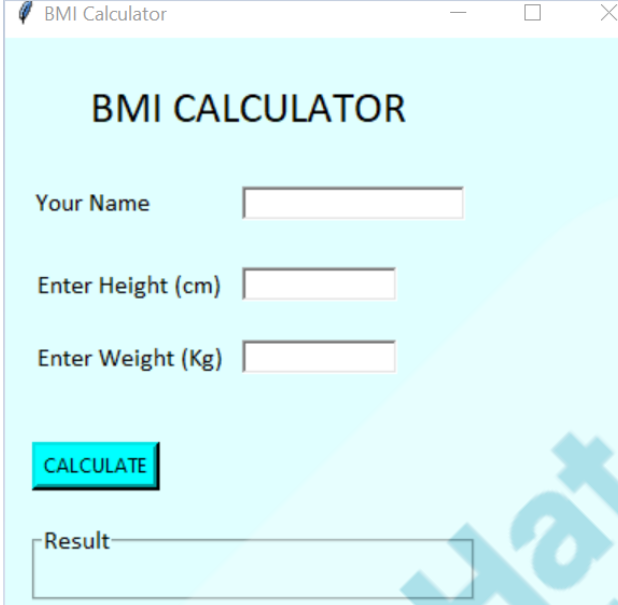
*In the button () widget we need to call the function **calculate\_bmi()** function to compare the BMI value with the BMI categories. Button () widget will use command attributes to call your function*

Remember at the end we need to call mainloop.

```
window.mainloop()
```

**If the teacher has already called the mainloop(), then there is no need to do the step again. All widgets will come before this loop.**

Your final output will look like this :






**Teacher Guides Student to Stop Screen Share**

**WRAP UP SESSION - 5 Mins**

**Quiz time - Click on in-class quiz**

Question	Answer
<p>What is the use of config () in Python tkinter?</p> <p>A. Make new widget B. Place the widget C. Destroy the widget D. Configure the widget</p>	<b>D</b>
<p>What does fg () in the tkinter widget stand for?</p> <p>A. For gap B. foreground C. background D. None of the above</p>	<b>B</b>

<p>For user input, which widget do we use in tkinter?</p> <p>A. Entry B. Text C. Input D. None of the above</p>	<p><b>A</b></p>
<p><b>End the quiz panel</b></p>	
<p style="text-align: center;"><b><u>FEEDBACK</u></b></p> <ul style="list-style-type: none"> <li>• Appreciate the students for their efforts in the class.</li> <li>• Ask the student to make notes for the reflection journal along with the code they wrote in today's class.</li> </ul>	
<p style="text-align: center;"><b>Teacher Action</b></p>	<p style="text-align: center;"><b>Student Action</b></p>
<p>You get Hats off for your excellent work!</p> <p>In the next class</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div style="display: flex; flex-direction: column; align-items: center;"> <div style="display: flex; align-items: center; margin-bottom: 10px;"> <div style="background-color: #0072bc; color: white; padding: 5px; text-align: center;">Creatively Solved Activities</div> <div style="margin-left: 10px; text-align: center;">   <b>+10</b> </div> </div> <div style="display: flex; align-items: center; margin-bottom: 10px;"> <div style="background-color: #0072bc; color: white; padding: 5px; text-align: center;">Great Question</div> <div style="margin-left: 10px; text-align: center;">   <b>+10</b> </div> </div> <div style="display: flex; align-items: center;"> <div style="background-color: #0072bc; color: white; padding: 5px; text-align: center;">Strong Concentration</div> <div style="margin-left: 10px; text-align: center;">   <b>+10</b> </div> </div> </div>
<p><b>Project Discussion</b></p>	
<div style="display: flex; justify-content: space-between; align-items: center;"> <div>Teacher Clicks</div> <div style="background-color: red; color: white; padding: 10px 20px; border-radius: 15px; font-weight: bold;">✕ End Class</div> </div>	



## ADDITIONAL ACTIVITIES

### Additional Activities

*Encourage the student to write reflection notes in their reflection journal using markdown.*

Use these as guiding questions:

- What happened today?
  - Describe what happened.
  - The code I wrote.
- How did I feel after the class?
- What have I learned about programming and developing games?
- What aspects of the class helped me? What did I find difficult?

*The student uses the markdown editor to write her/his reflections in the reflection journal.*

## ACTIVITY LINKS

Activity Name	Description	Link
Teacher Activity 1	Tkinter Library	<a href="https://docs.python.org/3/library/tkinter.html">https://docs.python.org/3/library/tkinter.html</a>
Teacher Activity 2	Reference Code	<a href="https://github.com/pro-whitehatjr/PRO-201_ReferenceCode">https://github.com/pro-whitehatjr/PRO-201_ReferenceCode</a>
Student Activity 1	Tkinter Library	<a href="https://docs.python.org/3/library/tkinter.html">https://docs.python.org/3/library/tkinter.html</a>
Student Activity 2	Boilerplate Code	<a href="https://github.com/pro-whitehatjr/PRO-201_Boilerplate">https://github.com/pro-whitehatjr/PRO-201_Boilerplate</a>