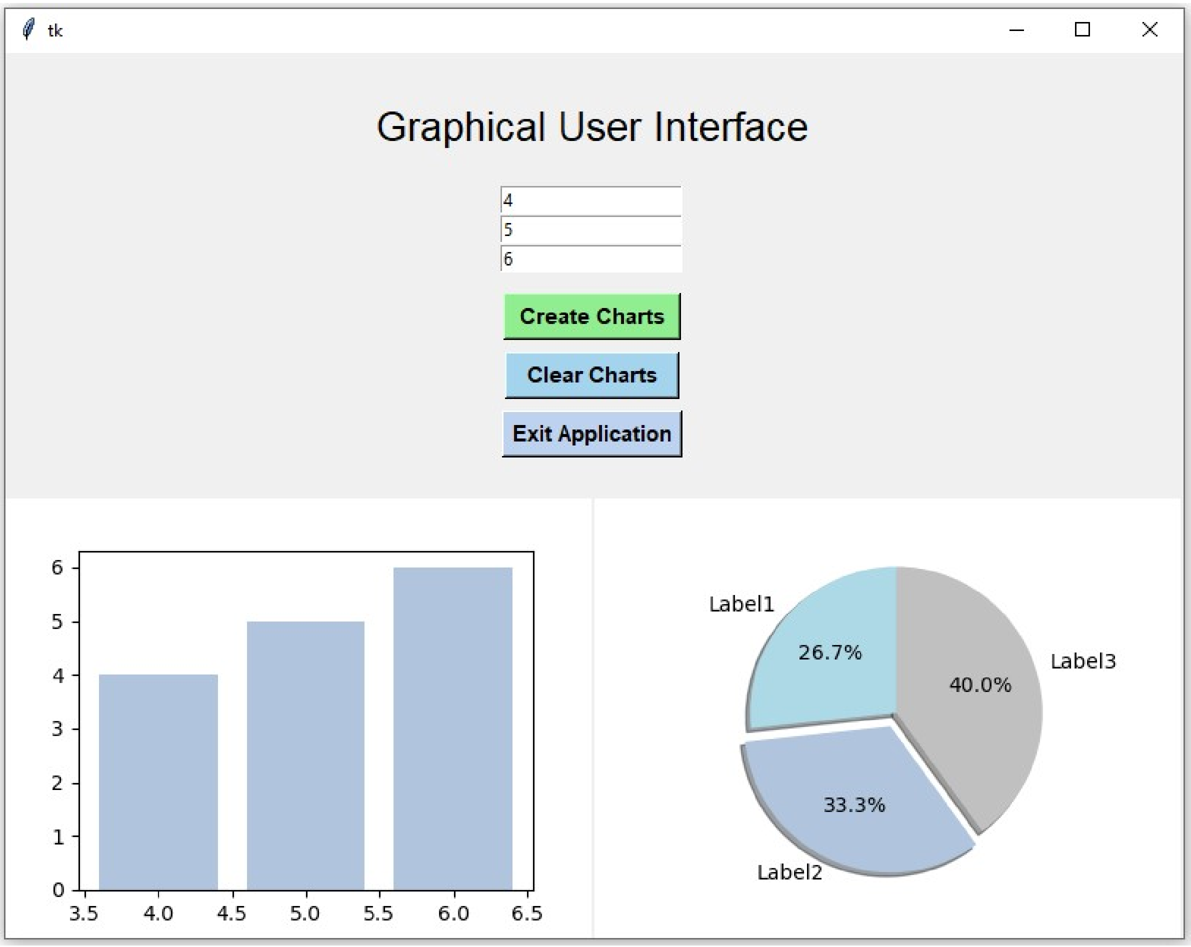
How to Create a GUI in Python using Tkinter

If so, in this tutorial, I'll show you how to create a tkinter GUI with the following components:

- Canvas screen, where you can place items, such as labels and buttons
- Labels to display text on top of the Canvas
- Entry Boxes to allow users type values
- Functions to display Bar and Pie charts
- Buttons to trigger the functions and exit the application
-

By the end of this tutorial, you'll be able to create the following GUI:



Create a GUI in Python using tkinter

You'll need to make sure that the *matplotlib* package is installed in python. This package is used to display the charts (i.e., bar and pie charts).

```
import tkinter as tk

from matplotlib.backends.backend_tkagg import
FigureCanvasTkAgg

from matplotlib.figure import Figure



root= tk.Tk()



canvas1 = tk.Canvas(root, width = 800, height = 300)

canvas1.pack()



label1 = tk.Label(root, text='Graphical User Interface')

label1.config(font=('Arial', 20))

canvas1.create_window(400, 50, window=label1)



entry1 = tk.Entry (root)

canvas1.create_window(400, 100, window=entry1)



entry2 = tk.Entry (root)

canvas1.create_window(400, 120, window=entry2)



entry3 = tk.Entry (root)
```

```
canvas1.create_window(400, 140, window=entry3)



def create_charts():

    global x1

    global x2

    global x3

    global bar1

    global pie2

    x1 = float(entry1.get())

    x2 = float(entry2.get())

    x3 = float(entry3.get())



    figure1 = Figure(figsize=(4,3), dpi=100)

    subplot1 = figure1.add_subplot(111)

    xAxis = [float(x1),float(x2),float(x3)]

    yAxis = [float(x1),float(x2),float(x3)]

    subplot1.bar(xAxis,yAxis, color = 'lightsteelblue')

    bar1 = FigureCanvasTkAgg(figure1, root)

    bar1.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH,
expand=0)
```

```
    figure2 = Figure(figsize=(4,3), dpi=100)

    subplot2 = figure2.add_subplot(111)

    labels2 = 'Label1', 'Label2', 'Label3'

    pieSizes = [float(x1),float(x2),float(x3)]

    my_colors2 = ['lightblue','lightsteelblue','silver']

    explode2 = (0, 0.1, 0)

    subplot2.pie(pieSizes, colors=my_colors2, 
explode=explode2, labels=labels2, autopct='%1.1f%%', 
shadow=True, startangle=90)

    subplot2.axis('equal')

    pie2 = FigureCanvasTkAgg(figure2, root)

    pie2.get_tk_widget().pack()



def clear_charts():

    bar1.get_tk_widget().pack_forget()

    pie2.get_tk_widget().pack_forget()



button1 = tk.Button (root, text=' Create Charts 
',command=create_charts, bg='palegreen2', font=('Arial', 
11, 'bold'))

canvas1.create_window(400, 180, window=button1)
```

```
button2 = tk.Button (root, text='  Clear Charts  ', 
command=clear_charts, bg='lightskyblue2', font=('Arial', 
11, 'bold'))

canvas1.create_window(400, 220, window=button2)



button3 = tk.Button (root, text='Exit Application', 
command=root.destroy, bg='lightsteelblue2', 
font=('Arial', 11, 'bold'))

canvas1.create_window(400, 260, window=button3)



root.mainloop()
```

Let’s now dive into the main components of the Python code:

The Canvas

The Canvas is your GUI screen in which you can place items, such as buttons, labels, entry boxes and more. You can control the dimensions of the Canvas by changing the width and height values.

```
canvas1 = tk.Canvas(root, width = 800, height = 300)

canvas1.pack()
```

The Label

Labels can be used to print text on top of the Canvas. Here, a label was added to display the following text:

‘Graphical User Interface’

You may specify a different font-family and font-size for your label. In our case, the font family is *‘Arial’* and the font size is *’20.’*

Finally, you can control the position of the label by modifying the coordinates on the last row below (for our example, the coordinates are 400 and 50):

```
label1 = tk.Label(root, text='Graphical User Interface')

label1.config(font=('Arial', 20))

canvas1.create_window(400, 50, window=label1)
```

The Entry Boxes

The 3 entry boxes are used to collect information from the user. That information will then be used to create the matplotlib charts.

As before, you can control the position of the entry boxes by specifying the coordinates.

```
entry1 = tk.Entry (root)

canvas1.create_window(400, 100, window=entry1)



entry2 = tk.Entry (root)

canvas1.create_window(400, 120, window=entry2)



entry3 = tk.Entry (root)

canvas1.create_window(400, 140, window=entry3)
```

The Functions

The '**create_charts**' function will be called when the user clicks on the first button (i.e., 'button1') to draw the charts.

The information collected in the entry boxes, will then be used to depict the bar and pie charts.

```
def create_charts():

    global x1

    global x2

    global x3

    global bar1

    global pie2

    x1 = float(entry1.get())

    x2 = float(entry2.get())

    x3 = float(entry3.get())



    figure1 = Figure(figsize=(4,3), dpi=100)

    subplot1 = figure1.add_subplot(111)

    xAxis = [float(x1),float(x2),float(x3)]

    yAxis = [float(x1),float(x2),float(x3)]

    subplot1.bar(xAxis,yAxis, color = 'lightsteelblue')

    bar1 = FigureCanvasTkAgg(figure1, root)

    bar1.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH,
expand=0)
```

```
    figure2 = Figure(figsize=(4,3), dpi=100)

    subplot2 = figure2.add_subplot(111)

    labels2 = 'Label1', 'Label2', 'Label3'

    pieSizes = [float(x1),float(x2),float(x3)]

    my_colors2 = ['lightblue','lightsteelblue','silver']

    explode2 = (0, 0.1, 0)

    subplot2.pie(pieSizes, colors=my_colors2,
explode=explode2, labels=labels2, autopct='%1.1f%%',
shadow=True, startangle=90)

    subplot2.axis('equal')

    pie2 = FigureCanvasTkAgg(figure2, root)

    pie2.get_tk_widget().pack()
```

The ‘**clear_charts**‘ function will be called when the user clicks on the second button (i.e., ‘button2’) to clear the charts from the GUI:

```
def clear_charts():

    bar1.get_tk_widget().pack_forget()

    pie2.get_tk_widget().pack_forget()
```

The Buttons

The first button (i.e., ‘button1’) can be used to trigger the ‘**create_charts**‘ function in order to draw the charts.

You can position the button on the Canvas by specifying the coordinates (in our case, the coordinates are 400 and 180):

```
button1 = tk.Button (root, text=' Create Charts
',command=create_charts, bg='palegreen2', font=('Arial',
11, 'bold'))

canvas1.create_window(400, 180, window=button1)
```

The second button (i.e., ‘button2’) triggers the ‘**clear_charts**‘ function to remove the previous charts from the GUI:

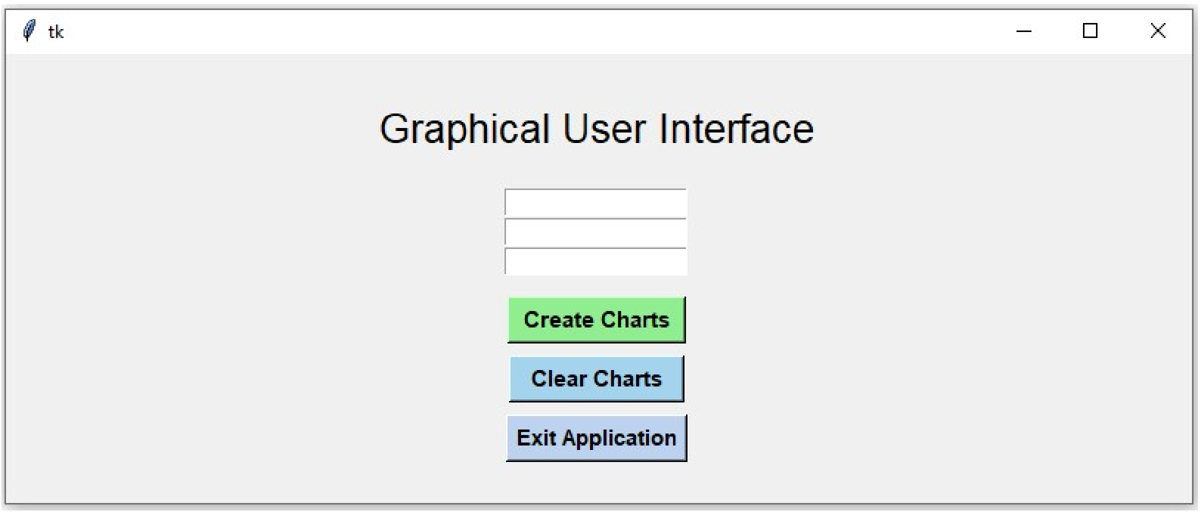
```
button2 = tk.Button (root, text='  Clear Charts   ',
command=clear_charts, bg='lightskyblue2', font=('Arial',
11, 'bold'))

canvas1.create_window(400, 220, window=button2)
```

The exit application button (i.e., ‘button3’) triggers the ‘destroy’ command to close the tkinter GUI upon a click:

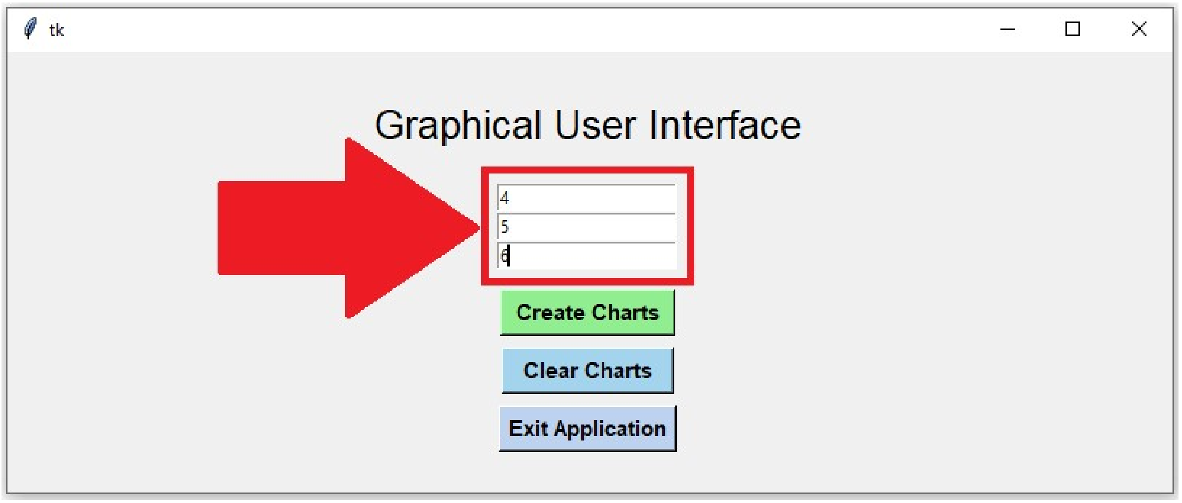
```
button3 = tk.Button (root, text='Exit Application',
command=root.destroy, bg='lightsteelblue2',
font=('Arial', 11, 'bold'))

canvas1.create_window(400, 260, window=button3)
```

Launch the tkinter GUI

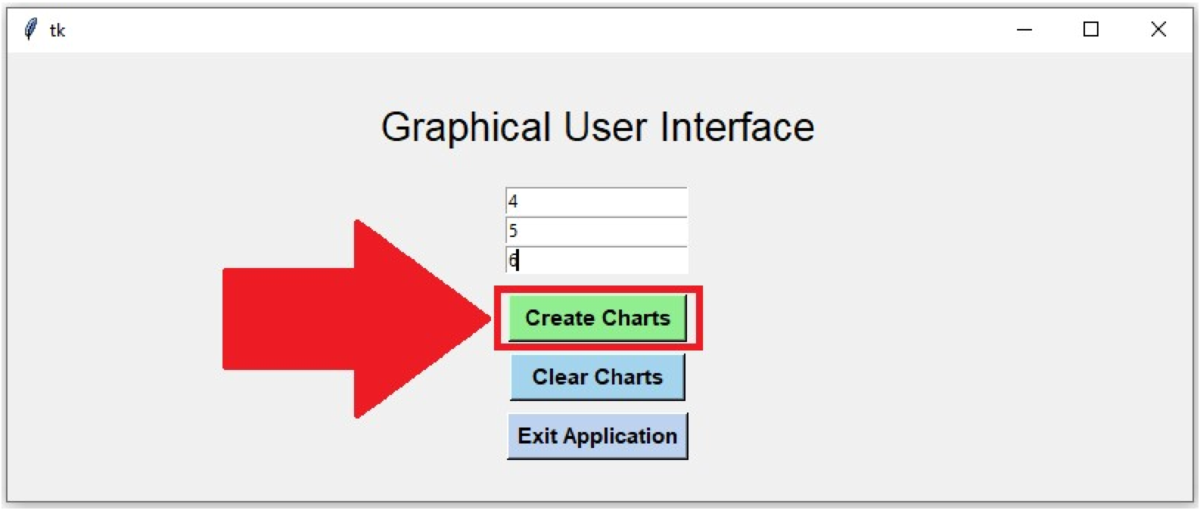
Once you’re ready, run the complete code in Python, and you’ll see this initial screen:



Type a value within each of the input boxes. For example, type the values of 4, 5 and 6 in the input boxes:

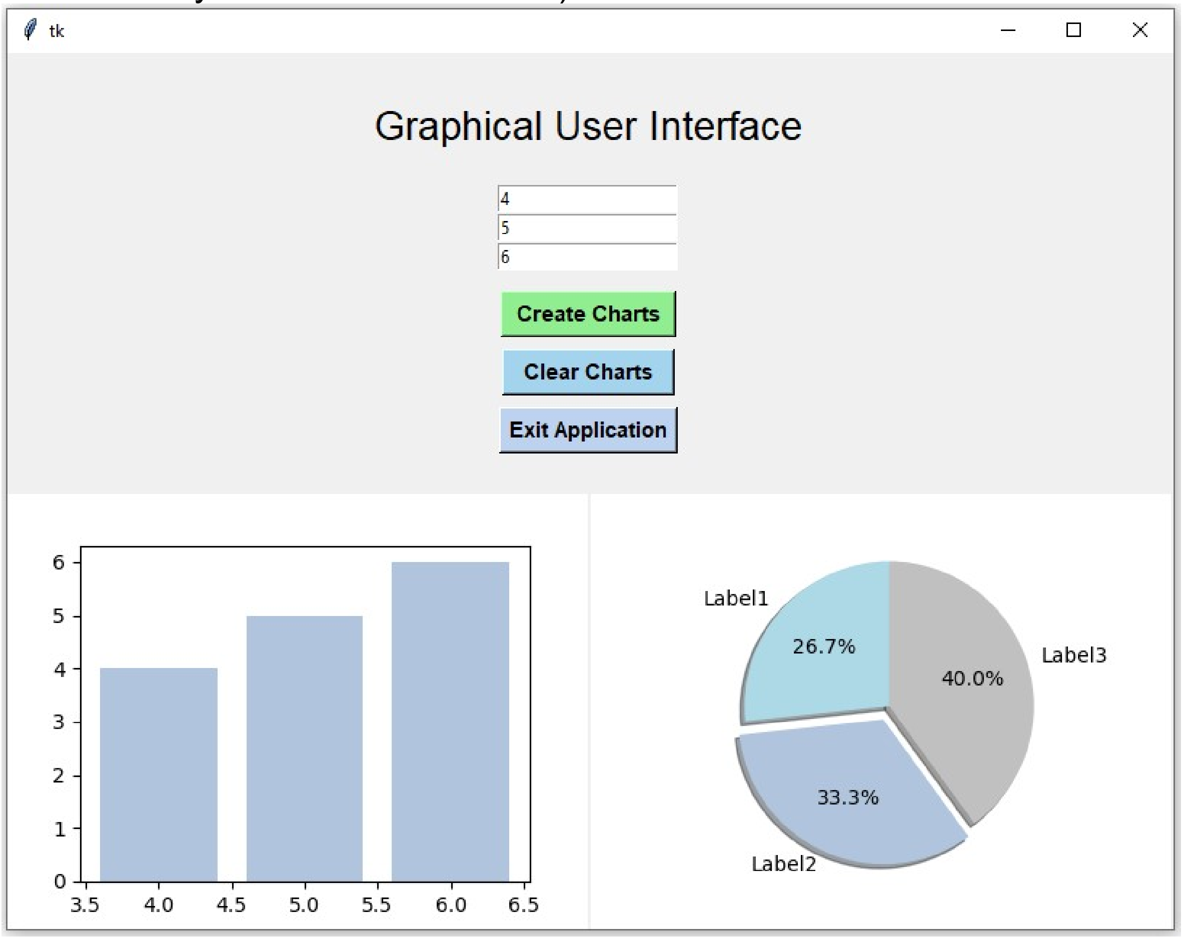


Then, click on the “**Create Charts**” button:



You’ll now see the two charts at the bottom section of your screen (based on the 3 values that you typed):

- On the left side, you’ll see the bar chart
- On the right side, you’ll observe the pie chart (representing your values in % terms)


tk
Graphical User Interface
4
5
6
Create Charts
Clear Charts
Exit Application
6
5
4
3
2
1
0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
Label1
26.7%
Label3
40.0%
33.3%
Label2