Python Tkinter

Python provides the standard library Tkinter for creating the graphical user interface for desktop-based applications.

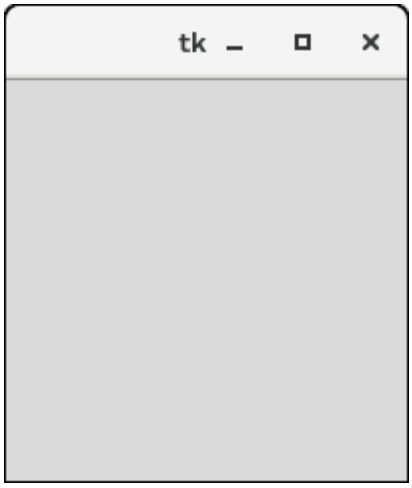
Developing desktop-based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Example

```
1. # !/usr/bin/python3
2. from tkinter import *
3. #creating the application main window.
4. top = Tk()
5. #Entering the event main loop
6. top.mainloop()
```

Output:



Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python application.
2	Canvas	The canvas widget is used to draw the canvas on the window.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.
7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.
9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.

11	Radiobutton	The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them.
12	Scale	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	Text	It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it.
14	Toplevel	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container
18	MessageBox	This module is used to display the message-box in the desktop based applications.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Let's discuss each one of them in detail.

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

The syntax to use the pack() is given below.

syntax

1. widget.pack(options)

A list of possible options that can be passed in pack() is given below.

- **expand:** If the expand is set to true, the widget expands to fill any space.
- **Fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
- **side:** it represents the side of the parent to which the widget is to be placed on the window.

Example

```
1. # !/usr/bin/python3
2. from tkinter import *
3. parent = Tk()
4. redbutton = Button(parent, text = "Red", fg = "red")
5. redbutton.pack( side = LEFT)
6. greenbutton = Button(parent, text = "Black", fg = "black")
7. greenbutton.pack( side = RIGHT )
8. bluebutton = Button(parent, text = "Blue", fg = "blue")
9. bluebutton.pack( side = TOP )
10. blackbutton = Button(parent, text = "Green", fg = "red")
11. blackbutton.pack( side = BOTTOM)
12. parent.mainloop()
```

Output:

Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax

1. widget.grid(options)

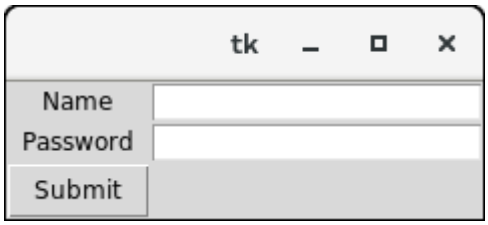
A list of possible options that can be passed inside the grid() method is given below.

- **Column**
  The column number in which the widget is to be placed. The leftmost column is represented by 0.
- **Columnspan**
  The width of the widget. It represents the number of columns up to which, the column is expanded.
- **ipadx,ipady**
  It represents the number of pixels to pad the widget inside the widget's border.
- **padx,pady**
  It represents the number of pixels to pad the widget outside the widget's border.
- **row**
  The row number in which the widget is to be placed. The topmost row is represented by 0.
- **rowspan**
  The height of the widget, i.e. the number of the row up to which the widget is expanded.
- **Sticky**
  If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

Example

```
# !/usr/bin/python3
from tkinter import *
parent = Tk()
name = Label(parent,text = "Name").grid(row = 0, column = 0)
e1 = Entry(parent).grid(row = 0, column = 1)
password = Label(parent,text = "Password").grid(row = 1, column = 0)
e2 = Entry(parent).grid(row = 1, column = 1)
submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
parent.mainloop()
```

Output:



Python Tkinter place() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

Syntax

```
widget.place(options)
```

A list of possible options is given below.

- **Anchor:** It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)
- **bordermode:** The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.
- **height, width:** It refers to the height and width in pixels.
- **relheight, relwidth:** It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.
- **relx, rely:** It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Example

```
1. # !/usr/bin/python3
2. from tkinter import *
3. top = Tk()
4. top.geometry("400x250")
5. name = Label(top, text = "Name").place(x = 30,y = 50)
6. email = Label(top, text = "Email").place(x = 30, y = 90)
7. password = Label(top, text = "Password").place(x = 30, y = 130)
8. e1 = Entry(top).place(x = 80, y = 50)
9. e2 = Entry(top).place(x = 80, y = 90)
10. e3 = Entry(top).place(x = 95, y = 130)
11. top.mainloop()
```

Output: