tkinter All Versions

Released in Tkinter is Python's most popular GUI (Graphical User Interface) library. This topic explains proper usage of this library and its features.

Examples

A minimal tkinter Application

tkinter is a GUI toolkit that provides a wrapper around the Tk/Tcl GUI library and is included with Python. The following code creates a new window using tkinter and places some text in the window body.

> Note: In Python 2, the capitalization may be slightly different, see Remarks section below.

```
import tkinter as tk

# GUI window is a subclass of the basic tkinter Frame object
class HelloWorldFrame(tk.Frame):
    def __init__(self, master):
        # Call superclass constructor
        tk.Frame.__init__(self, master)
        # Place frame into main window
        self.grid()
        # Create text box with "Hello World" text
        hello = tk.Label(self, text="Hello World! This label can hold strings!")
        # Place text box into frame
        hello.grid(row=0, column=0)

# Spawn window
if __name__ == "__main__":
    # Create main window object
    root = tk.Tk()
    # Set title of window
    root.title("Hello World!")
    # Instantiate HelloWorldFrame object
    hello_frame = HelloWorldFrame(root)
    # Start GUI
    hello_frame.mainloop()
```

Geometry Managers

Tkinter has three mechanisms for geometry management: place , pack , and grid .

The place manager uses absolute pixel coordinates.

The pack manager places widgets into one of 4 sides. New widgets are placed next to existing widgets.

The grid manager places widgets into a grid similar to a dynamically resizing spreadsheet.

Place

The most common keyword arguments for widget.place are as follows:

- x , the absolute x-coordinate of the widget
- y , the absolute y-coordinate of the widget
- height , the absolute height of the widget
- width , the absolute width of the widget

A code example using place :

```
class PlaceExample(Frame):
    def __init__(self,master):
        Frame.__init__(self,master)
        self.grid()
        top_text=Label(master,text="This is on top at the origin")
        #top_text.pack()
        top_text.place(x=0,y=0,height=50,width=200)
        bottom_right_text=Label(master,text="This is at position 200,400")
        #top_text.pack()
        bottom_right_text.place(x=200,y=400,height=50,width=200)
# Spawn Window
if __name__=="__main__":
    root=Tk()
    place_frame=PlaceExample(root)
    place_frame.mainloop()
```

Pack

widget.pack can take the following keyword arguments:

widget.pack can take the following keyword arguments:

- expand , whether or not to fill space left by parent
- fill , whether to expand to fill all space (NONE (default), X, Y, or BOTH)
- side , the side to pack against (TOP (default), BOTTOM, LEFT, or RIGHT)

Grid

The most commonly used keyword arguments of widget.grid are as follows:

- row , the row of the widget (default smallest unoccupied)
- rowspan , the number of colums a widget spans (default 1)
- column , the column of the widget (default 0)
- columnspan , the number of columns a widget spans (default 1)
- sticky , where to place widget if the grid cell is larger than it (combination of N,NE,E,SE,S,SW,W,NW)

The rows and columns are zero indexed. Rows increase going down, and columns increase going right.

A code example using grid :

```
from tkinter import *

class GridExample(Frame):
    def __init__(self,master):
        Frame.__init__(self,master)
        self.grid()
        top_text=Label(self,text="This text appears on top left")
        top_text.grid() # Default position 0, 0
        bottom_text=Label(self,text="This text appears on bottom left")
        bottom_text.grid() # Default position 1, 0
        right_text=Label(self,text="This text appears on the right and spans both rows",
                         wraplength=100)
        # Position is 0,1
        # Rowspan means actual position is [0-1],1
        right_text.grid(row=0,column=1,rowspan=2)

# Spawn Window
if __name__=="__main__":
    root=Tk()
    grid_frame=GridExample(root)
    grid_frame.mainloop()
```

Never mix pack and grid within the same frame! Doing so will lead to application deadlock!

Syntax

Parameters

Remarks

The capitalization of the tkinter module is different between Python 2 and 3. For Python 2 use the following:

```
from Tkinter import *  # Capitalized
```

For Python 3 use the following:

```
from tkinter import *  # Lowercase
```

For code that works with both Python 2 and 3, you can either do

```
try:
    from Tkinter import *
except ImportError:
    from tkinter import *
```

or

```
from sys import version_info
if version_info.major == 2:
    from Tkinter import *
elif version_info.major == 3:
    from tkinter import *
```

See the tkinter Documentation for more details