back next

The Tkinter Pack Geometry Manager

The **Pack** geometry manager packs widgets in rows or columns. You can use options like **fill**, **expand**, and **side** to control this geometry manager.

The manager handles all widgets that are packed inside the same master widget. The packing algorithm is simple, but a bit tricky to describe in words; imagine a sheet of some elastic material, with a very small rectangular hole in the middle. For each widget, in the order they are packed, the geometry manager makes the hole large enough to hold the widget, and then place it against a given inner edge (default is the top edge). It then repeats the process for all widgets. Finally, when all widgets have been packed into the hole, the manager calculates the bounding box for all widgets, makes the master widget large enough to hold all widgets, and moves them all to the master.

When to use the Pack Manager

Compared to the **grid** manager, the pack manager is somewhat limited, but it's much easier to use in a few, but quite common situations:

1. Put a widget inside a frame (or any other container widget), and have it fill the entire frame
2. Place a number of widgets **on top of each other**
3. Place a number of widgets **side by side**

See the *Patterns* section for code examples.

If you need to create more complex layouts, you usually have to group the widgets using extra **Frame** widgets. You can also use the **grid** manager instead.

Note: Don't mix grid and pack in the same master window. Tkinter will happily spend the rest of your lifetime trying to negotiate a solution that both managers are happy with. Instead of waiting, kill the application, and take another look at your code. A common mistake is to use the wrong parent for some of the widgets.

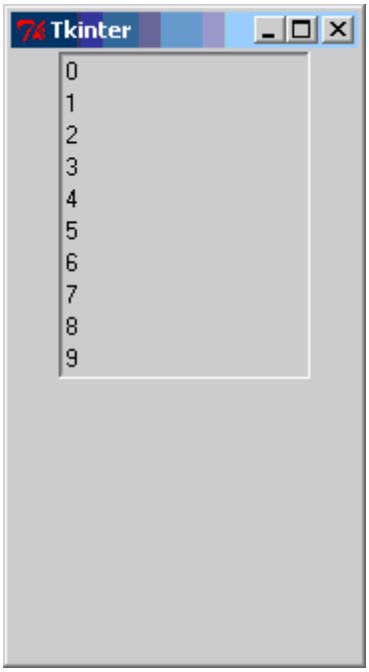
Patterns

Filling the entire parent widget

A common situation is when you want to place a widget inside a container widget, and have it fill the entire parent. Here's a simple example: a listbox placed in the root window:

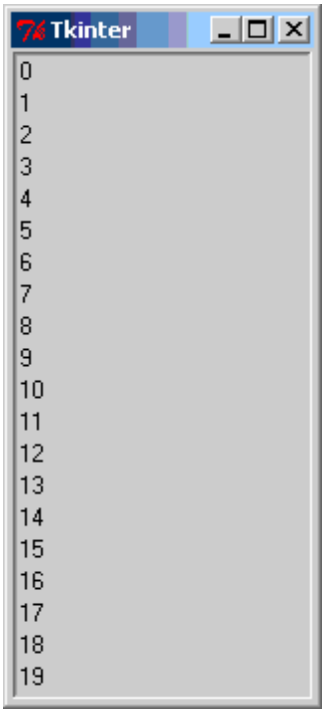
```
from Tkinter import *

root = Tk()

listbox = Listbox(root)
listbox.pack()

for i in range(20):
    listbox.insert(END, str(i))

mainloop()
```

By default, the listbox is made large enough to show 10 items. But this listbox contains twice as many. But if the user attempts to show them all by resizing the window, Tkinter will add padding around the listbox:



To make the widget fill the entire parent, also if the user resizes the window, add **fill** and **expand** options:

```
from Tkinter import *

root = Tk()

listbox = Listbox(root)
listbox.pack(fill=BOTH, expand=1)

for i in range(20):
    listbox.insert(END, str(i))

mainloop()
```



The **fill** option tells the manager that the widget wants fill the entire space assigned to it. The value controls how to fill the space; **BOTH** means that the widget should expand both horisontally and vertically, **X** means that it should expand only horisontally, and **Y** means that it should expand only vertically.

The **expand** option tells the manager to assign additional space to the widget box. If the parent widget is made larger than necessary to hold all packed widgets, any exceeding space will be distributed among all widgets that have the **expand** option set to a non-zero value.

Placing a number of widgets on top of each other

To put a number of widgets in a column, you can use the **pack** method without any options:

```
from Tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack()
w = Label(root, text="Green", bg="green", fg="black")
w.pack()
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack()

mainloop()
```

You can use the **fill=X** option to make all widgets as wide as the parent widget:

```
from Tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(fill=X)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(fill=X)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(fill=X)
```

```
mainloop()
```

Placing a number of widgets side by side

To pack widgets side by side, use the **side** option. If you wish to make the widgets as high as the parent, use the **fill=Y** option too:

```
from Tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(side=LEFT)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(side=LEFT)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(side=LEFT)

mainloop()
```

Reference

Pack (class) [#]

Pack geometry manager. This is an implementation class; all the methods described below are available on all widget classes.

pack(options)** [#]

Pack the widget as described by the options.

***options*
Geometry options.

anchor=
Where the widget is placed inside the packing box. Default is CENTER.

expand=
Specifies whether the widgets should be expanded to fill any extra space in the geometry master. If false (default), the widget is not expanded.

fill=
Specifies whether the widget should occupy all the space provided to it by the master. If **NONE** (default), keep the widget's original size. If **X** (fill horizontally), **Y** (fill vertically), or **BOTH**, fill the given space along that direction.
To make a widget fill the entire master widget, set **fill** to **BOTH** and **expand** to a non-zero value.

in=
Pack this widget inside the given widget. You can only pack a widget inside its parent, or in any decendant of its parent. This option should usually be left out, in which case the widget is packed inside its parent.
Note that **in** is a reserved word in Python. To use it as a keyword option, append an underscore (**in_**).

ipadx=
Internal padding. Default is 0.

ipady=
Internal padding. Default is 0.

padx=
External padding. Default is 0.

pady=

External padding. Default is 0.

side=

Specifies which side to pack the widget against. To pack widgets vertically, use **TOP** (default). To pack widgets horizontally, use **LEFT**.
You can also pack widgets along the **BOTTOM** and **RIGHT** edges. You can mix sides in a single geometry manager, but the results may not always be what you expect. While you can create pretty complicated layouts by nesting **Frame** widgets, you may prefer using the **grid** geometry manager for non-trivial layouts.

pack_configure(options)** [#]

Same as **pack**.

pack_forget() [#]

Removes the widget from its current manager. The widget is not destroyed, and can be displayed again by **pack** or any other manager.

pack_info() [#]

Returns a dictionary containing the current packer options.

Returns:
A dictionary mapping packer option names to option values.

pack_propagate(flag) [#]

(Manager method) Controls geometry propagation. If enabled, the manager widget will be resized if not large enough to hold all the child widgets.

Note that this method should be called on the master widget, not on an invidivual child widget.

pack_slaves() [#]

(Manager method) Returns a list of all child (“slave”) widgets managed by the packer for this widget.

Note that this method should be called on the master widget, not on an invidivual child widget.

Returns:
A list of child widgets.

back next