Aula 16

Layout

A biblioteca tkinter do Python oferece várias formas de organizar o layout de seus widgets em um container, como janelas ou frames. Algumas das principais formas incluem:

- pack() : este método organiza os widgets em uma pilha, um em cima do outro. Ele é fácil de usar e geralmente é a primeira escolha para organizar widgets simples. O método pack() aceita várias opções, como side, fill, padx e pady, que podem ser usadas para controlar a posição e o tamanho dos widgets;

- grid() : este método organiza os widgets em uma grade de linhas e colunas. Ele é mais flexível do que o método pack() e é útil para organizar widgets mais complexos. O método grid() aceita várias opções, como row, column, rowspan, e columnspan, que podem ser usadas para controlar a posição e o tamanho dos widgets na grade;

- place() : este método permite que você posicione widgets em qualquer lugar dentro do container, especificando as coordenadas x e y. Ele é útil para criar layouts personalizados e para posicionar widgets com precisão. O método place() aceita várias opções, como x, y, width, e height, que podem ser usadas para controlar a posição e o tamanho dos widgets;

pack()

O método pack() é usado na biblioteca tkinter do Python para organizar widgets dentro de um container (como uma janela ou frame). Ele coloca o widget em questão em um lugar específico dentro do container, baseado em suas opções de configuração. Por padrão, o widget é colocado no topo do container, com largura e altura ajustadas para se adequar ao tamanho do widget. Você pode especificar várias opções ao chamar o método pack() para controlar a posição e o tamanho do widget, como side, fill, padx, pady, ipadx e ipady.

Veja um exemplo simples de uso:

```
from tkinter import Button, Tk

janela = Tk()
janela.geometry('400x400')

# cria um botão
botao = Button(janela, text="Clique aqui")

# adiciona o botão à janela usando o método pack
botao.pack()

janela.mainloop()
```

Neste exemplo, o botão é adicionado à janela e é exibido no topo da janela. O método pack() é chamado sem qualquer argumento adicional, então o botão é posicionado no topo da janela e seu tamanho é ajustado para se adequar ao tamanho do texto contido nele.

Vamos cada uma das opções.

side

A opção side do método pack() é usada para especificar em qual lado do container o widget deve ser posicionado. Ela pode ser usada para controlar a disposição dos widgets em um container.

A opção side aceita um dos seguintes valores :

- TOP (ou top) : posiciona o widget na margem superior da janela (`valor padrão`);
- BOTTOM (ou bottom) : posiciona o widget na margem inferior da janela;
- LEFT (ou left) : posiciona o widget na margem esquerda da janela;
- RIGHT (ou right) : posiciona o widget na margem direita da janela;

Por exemplo, se você quiser adicionar um botão à esquerda de uma janela, você pode usar o seguinte código :

```
from tkinter import Button, LEFT, Tk

janela = Tk()
janela.geometry('400x400')

botao = Button(janela, text="Clique aqui")
botao.pack(side=LEFT)

janela.mainloop()
```

Se quiser adicionar múltiplos widgets lado a lado, você pode usar a opção side para especificar a posição de cada widget :

```
from tkinter import Button, LEFT, Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")

btn_1.pack(side=LEFT)
btn_2.pack(side=LEFT)

janela.mainloop()
```

No exemplo abaixo, os dois botões são adicionados à janela e são posicionados um cima do outro, mas na parte de baixo da janela. Repare que a ordem dos botões corresponde à ordem de execução.

```
from tkinter import BOTTOM, Button, Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")

btn_1.pack(side=BOTTOM)
btn_2.pack(side=BOTTOM)

janela.mainloop()
```

fill

A opção fill do método pack() é usada para especificar se o widget deve ocupar todo o espaço disponível no container. Ela pode ser usada para controlar o tamanho dos widgets em um container.

A opção fill aceita um dos seguintes valores :

- X (ou x) : O widget ocupa toda a largura disponível no container;
- Y (ou y) : O widget ocupa toda a altura disponível no container;
- BOTH (ou both) : O widget ocupa toda a largura e altura disponíveis no container;
- NONE (ou none) : O widget não é redimensionado (valor padrão);

Por exemplo, se você quiser adicionar um botão que ocupe toda a largura disponível em uma janela, você pode usar o seguinte código :

```
from tkinter import Button, Tk, X

janela = Tk()
janela.geometry('400x400')

botao = Button(janela, text="Clique aqui")
botao.pack(fill=X)

janela.mainloop()
```

Se você quiser adicionar múltiplos widgets que ocupem toda a largura e altura disponíveis no container, você pode usar a opção fill para especificar isso para cada widget:

```
from tkinter import BOTH, Button, LEFT,Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")

btn_1.pack(side=LEFT,fill=BOTH)
btn_2.pack(fill=BOTH)

janela.mainloop()
```

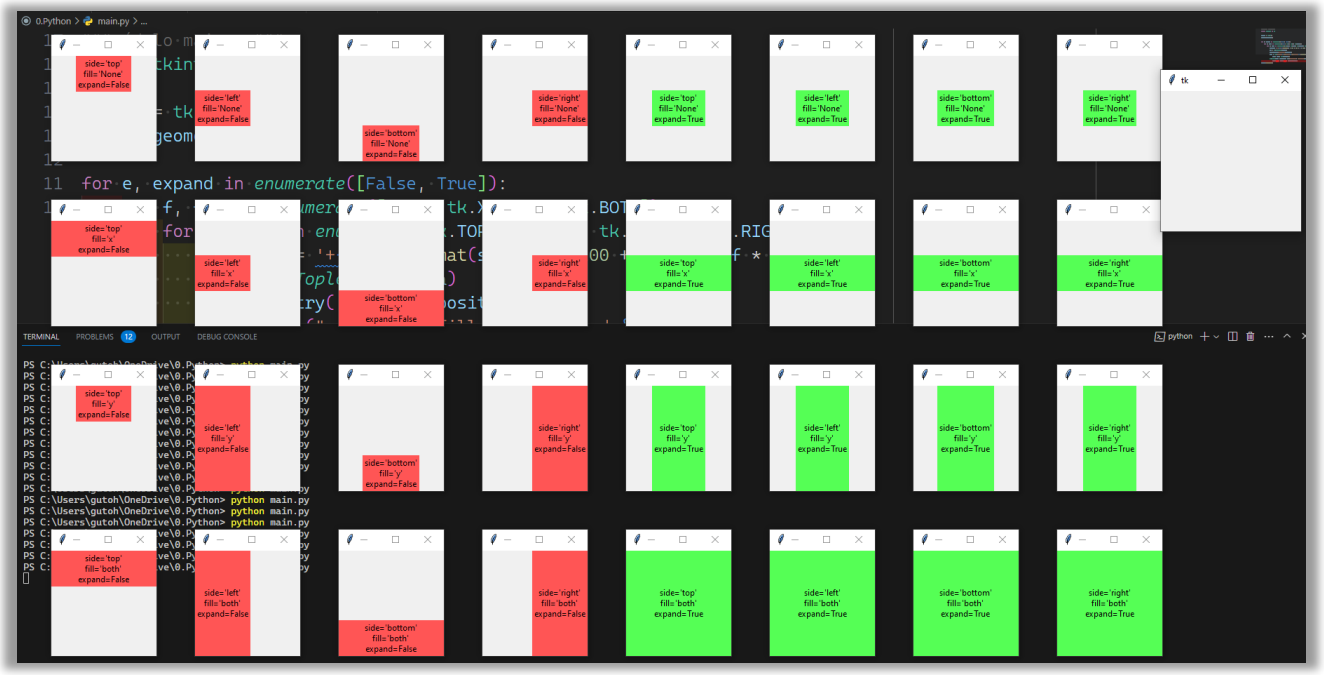
O script abaixo mostra diversas opções de side, fill e expand.

```
import tkinter as tk

janela = tk.Tk()
janela.geometry()

for e, expand in enumerate([False, True]):
    for f, fill in enumerate([None, tk.X, tk.Y, tk.BOTH]):
        for s, side in enumerate([tk.TOP, tk.LEFT, tk.BOTTOM, tk.RIGHT]):
            position = '+{}+{}'.format(s * 205 + 100 + e * 820, f * 235 + 100)
            win = tk.Toplevel(janela)
            win.geometry('150x150'+position)
            text = str("side='{}'\nfill='{}'\nexpand={}".format(
                side, fill, str(expand)))
            tk.Label(win, text=text, bg=['#FF5555', '#55FF55'][e]).pack(
                side=side, fill=fill, expand=expand)
janela.mainloop()
```

O resultado é a imagem abaixo:



Fonte : stackoverflow? https://stackoverflow.com/questions/28089942/difference-between-fill-and-expand-options-for-tkinter-pack-method

padx & pady

As opções padx e pady do método pack() são usadas para adicionar espaçamento entre um widget e os limites do container (janela ou frame) em que ele está sendo exibido. Elas são especificadas em pixels e podem ser usadas para controlar a aparência do layout.

padx é usado para adicionar espaçamento à esquerda e à direita do widget. Por exemplo, se você quiser adicionar 120 pixels de espaçamento à esquerda e à direita de um botão, você pode usar o seguinte código :

```
from tkinter import Button, Tk

janela = Tk()

botao = Button(janela, text="Clique aqui")
botao.pack(padx=120)

janela.mainloop()
```

pady é usado para adicionar espaçamento acima e abaixo do widget. Por exemplo, se você quiser adicionar 100 pixels de espaçamento acima e abaixo de um botão, você pode usar o seguinte código :

```
from tkinter import Button, Tk

janela = Tk()

botao = Button(janela, text="Clique aqui")
botao.pack(pady=100)

janela.mainloop()
```

Você também pode usar ambos padx e pady simultaneamente :

```
from tkinter import Button, Tk

janela = Tk()

botao = Button(janela, text="Clique aqui")
botao.pack(padx=120, pady=100)

janela.mainloop()
```

Ainda há uma variação do padx e pady, que são os ipadx e ipady. O comportamento é muito similar, com a diferença que o espaçamento será entre o conteúdo do widget e sua borda.

grid()

O método grid() é uma das formas principais de organizar o layout de widgets no tkinter. Ele organiza os widgets em uma grade de linhas e colunas, permitindo a criação de layouts mais complexos.

Para usar o método grid(), você primeiro precisa criar um container, como uma janela ou frame, e adicionar widgets a ele. Em seguida, você pode chamar o método grid() para posicionar os widgets na grade.

Por exemplo, você pode adicionar três botões a uma grade com duas linhas e duas colunas :

```
from tkinter import Button, Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")
btn_3 = Button(janela, text="Botão 3")

btn_1.grid(row=0, column=0)
btn_2.grid(row=0, column=1)
btn_3.grid(row=1, column=0)

janela.mainloop()
```

Neste exemplo, o botão 1 é posicionado na linha 0, coluna 0, o botão 2 é posicionado na linha 0, coluna 1, e o botão 3 é posicionado na linha 1, coluna 0.

Você também pode especificar a largura e altura de uma célula usando as opções rowspan e columnspan respectivamente. Por exemplo, se você quiser que o botão 3 ocupe duas células na coluna, você pode usar o seguinte código :

```
from tkinter import Button, Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")
btn_3 = Button(janela, text="Botão 3")

btn_1.grid(row=0, column=0)
btn_2.grid(row=0, column=1)
btn_3.grid(row=1, column=0, columnspan=2)

janela.mainloop()
```

Além disso, você pode controlar o espaçamento entre os widgets e o container, usando as opções padx e pady como no método pack().

place()

O método place() é outra forma de organizar o layout de widgets no tkinter. Ele permite posicionar widgets em qualquer lugar dentro do container, especificando as coordenadas x e y. Ele é útil para criar layouts personalizados e para posicionar widgets com precisão.

Para usar o método place(), você primeiro precisa criar um container, como uma janela ou frame, e adicionar widgets a ele. Em seguida, você pode chamar o método place() para posicionar os widgets dentro do container.

Por exemplo, você pode adicionar dois botões a uma janela e posicioná-los em coordenadas específicas :

```
from tkinter import Button, Tk

janela = Tk()
janela.geometry('400x400')

btn_1 = Button(janela, text="Botão 1")
btn_2 = Button(janela, text="Botão 2")

btn_1.place(x=50, y=50)
btn_2.place(x=100, y=100)

janela.mainloop()
```

Neste exemplo, o botão 1 é posicionado na coordenada x=50, y=50 e o botão 2 é posicionado na coordenada x=100, y=100.

Você também pode especificar o tamanho do widget usando as opções width e height. Por exemplo, se você quiser que o botão 1 tenha 100 pixels de largura e 50 pixels de altura, você pode usar o seguinte código :

```
from tkinter import Button, Tk

janela = Tk()
janela.geometry('400x400')

botao = Button(janela, text="Botão 1")

botao.place(x=50, y=50, width=100, height=50)

janela.mainloop()
```

É importante notar que quando você usa o método place() para posicionar widgets, deve especificar as coordenadas x e y em pixels. Além disso, é importante que tenha definido o tamanho do container (janela ou frame) usando o método geometry() ou o método config(), caso contrário, ele não terá nenhum efeito.

Considerações Finais

O método place() é mais flexível do que o método pack() e grid() pois ele permite posicionar widgets com precisão, mas ele é mais trabalhoso de se usar, pois você precisa calcular as coordenadas dos widgets manualmente. Ele é útil para criar layouts personalizados e para posicionar widgets com precisão, mas é menos indicado para organizar grande quantidade de widgets.

Exercícios para Praticar

1. Crie uma janela com um botão e o posicione na borda superior da janela usando o pack e o side.
2. Crie uma janela com um botão e o posicione na borda inferior da janela usando o pack e o side.
3. Crie uma janela com um botão e o posicione na borda esquerda da janela usando o pack e o side.
4. Crie uma janela com um botão e o posicione na borda direita da janela usando o pack e o side.
5. Crie uma janela com dois botões e os posicione na borda superior e inferior da janela usando o pack e o side.
6. Crie uma janela com dois botões e os posicione na borda esquerda e direita da janela usando o pack e o side.
7. Crie uma janela com quatro botões e os posicione nas bordas superior, inferior, esquerda e direita da janela usando o pack e o side.
8. Crie uma janela com um botão que ocupe toda a largura da janela usando o pack e o fill.
9. Crie uma janela com um botão que ocupe toda a altura da janela usando o pack e o fill.
10. Crie uma janela com um botão que ocupe toda a largura e altura da janela usando o pack e o fill.
    a. Repita os 3 últimos exercícios usando dois e depois 3 botões.
11. Crie uma janela com um botão que ocupe toda a largura da janela usando o pack e o fill e que tenha um espaçamento de 10 pixels em todas as direções.
12. Crie uma janela com um botão que ocupe toda a largura da janela usando o pack e o fill e que tenha um espaçamento de 10 pixels na horizontal e 20 pixels na vertical.
13. Crie uma janela com um botão que ocupe um tamanho de 50x50 e tenha um espaço interno de 10 pixels em todas as direções.
14. Crie uma janela que tenha 3 botões lado a lado usando o método grid.
15. Crie uma janela que tenha 3 botões lado a lado usando o método grid e que cada um ocupe 1/3 da largura da janela.
16. Crie uma janela que tenha 3 botões um embaixo do outro usando o método grid.
17. Crie uma janela que tenha 3 botões um embaixo do outro usando o método grid e que cada um ocupe 1/3 da altura da janela.
18. Crie uma janela que tenha 3 botões um embaixo do outro usando o método grid e que cada um ocupe 1/3 da altura da janela e 1/3 da largura da janela.
    a. Adicione um espaçamento de 10 pixels entre os botões do exercício anterior.
19. Crie uma janela com dois botões lado a lado e um botão abaixo deles usando o método grid.
20. Crie uma janela com dois botões lado a lado e um botão abaixo deles usando o método grid e que o botão de baixo ocupe o mesmo tamanho dos botões de cima.
21. Crie uma janela com dois botões um em cima do outro, outro botão ao lado deles e que ocupe a mesma altura dos botões do lado usando o método grid.
22. Crie uma janela e posicione o botão a uma distância de 10 pixels da borda superior da janela usando o método place.
23. Crie uma janela e posicione o botão a uma distância de 10 pixels da borda inferior da janela usando o método place.
24. Crie uma janela e posicione o botão a uma distância de 10 pixels da borda esquerda da janela usando o método place.
25. Crie uma janela e posicione o botão a uma distância de 10 pixels da borda direita da janela usando o método place.
26. Crie uma função para o exercício anterios que, ao ser chamada, posicione o botão a uma distância aleatória entre 5 e 100 pixels da borda e em uma borda aleatória da janela usando o método place.
27. Crie uma janela com um botão que, ao ser clicado, o posicione na borda oposta à que ele está usando o pack e o side.
28. Crie uma janela com um botão que, sempre que for clicado, o posicione em uma borda aleatória da janela usando o pack e o side.
29. Crie uma janela com um botão que, sempre que for clicado, o posicione em uma célula aleatória da janela usando o método grid.
30. Crie o layout de uma calculadora.