Python:
Interfaces Gráficas com Tk

UFRJ

Interfaces Gráficas

- Também chamadas de Graphical User Interfaces (GUI)
- Usadas em aplicações modernas que requerem uma interação constante com o usuário
  - Maior usabilidade e naturalidade do que interfaces textuais
- Aplicação apresenta uma ou mais janelas com elementos gráficos que servem para comandar ações, especificar parâmetros, desenhar e exibir gráficos, etc
- Bibliotecas (*toolkits*) para construção de interfaces como
  - Qt
  - Gtk
  - wxWindows
  - Tk

Interfaces Gráficas em Python

- Python possui camadas de portabilidade (*bindings*) para várias bibliotecas de construção de interfaces. Ex.:
  - PyQt (Qt)
  - PyGtk (Gtk)
  - wxPython (wxWindows)
  - Tkinter (Tk)
- Multiplataforma (MS-Windows, Unix/Linux, OSX)

Tk

- Toolkit originalmente criado para utilização com a linguagem script Tcl
- Bastante leve, portátil e robusto
- Um tanto obsoleto frente a outros toolkits mais modernos como Qt ou Gtk
- Camada Tkinter normalmente distribuída com o Python
  - Inicia um processo Tcl que toma conta dos elementos de interface
  - Classes e funções do Tkinter se comunicam com o interpretador Tcl para especifcar aspecto e comportamento da interface

Usando Tkinter

- Importar o módulo Tkinter
  - from Tkinter import *
- Elementos de interface (*widgets*) correspondem a objetos de diversas classes. Por exemplo:
  - Frame (Área retangular)
  - Button (botão)
  - Label (rótulo)
  - Text (caixa de texto)
  - Canvas (caixa de desenho)
- Posição e tamanho dos elementos controlados por gerentes de geometria
  - Pack (mais comum), Place, Grid

Usando Tkinter (2)

- Para criar um widget, tem-se que informar o widget-pai (parâmetro *master*) onde geometricamente deverá ser encaixado e as opções de configuração para o widget. Ex.:

  w=Button(pai,text=”Cancelar”,command=cancelar)
- Tk já define por default uma janela principal
  - master=None (default) indica que o widget será filho da janela principal
  - Outras janelas podem ser criadas instanciando-se objetos da classe Toplevel
- A função mainloop tem que ser invocada para que a aplicação entre no modo de tratamento de eventos

Exemplo

```
from Tkinter import *

class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.msg = Label(self, text="Hello World”)
        self.msg.pack ()
        self.bye = Button (self, text="Bye", command=self.quit)
        self.bye.pack ()
        self.pack()

app = Application()
mainloop()
```

Exemplo


```
from Tkinter import *

class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.msg = Label(self, text="Hello World”)
        self.msg.pack ()
        self.bye = Button (self, text="Bye", command=self.quit)
        self.bye.pack ()
        self.pack()

app = Application()
mainloop()
```

Exemplo

```
from Tkinter import *

class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.msg = Label(self, text="Hello World”)
        self.msg.pack ()
        self.bye = Button (self, text="Bye", command=self.quit)
        self.bye.pack ()
        self.pack()

app = Application()
mainloop()
```

Elemento principal derivado de Frame

Construtor da classe base

Janela tem um rótulo e um botão

Interface é instanciada

Laço de tratamento de eventos é iniciado

Classes de componentes

- Button Um botão simples usado para executar um comando
- Canvas Provê facilidades de gráficos estruturados
- Checkbutton Representa uma variável que pode ter dois valores distintos (tipicamente um valor booleano). Clicando no botão alterna-se entre os valores
- Entry Um campo para entrada de uma linha de texto
- Frame Usado como agrupador de widgets
- Label Mostra um texto ou uma imagem
- Listbox Mostra uma lista de alternativas. Pode ser configurado para ter comportamento de checkbutton ou radiobutton

Classes de componentes (cont.)

- Menu Um painel de menu. Implementa menus de janela, pulldowns e popups
- Message Similar ao widget Label, mas tem mais facilidade para mostrar texto quebrado em linhas
- Radiobutton Representa um possível valor de uma variável que tem um de muitos valores. Clicando o botão, a variável assume aquele valor
- Scale Permite especificar um valor numérico através de um ponteiro em uma escala linear
- Scrollbar Barra de rolamento para widgets que têm superfície útil variável (Text, Canvas, Entry, Listbox)
- Text Exibe e permite editar texto formatado. Também suporta imagens e janelas embutidas
- Toplevel Uma janela separada

A Classe Tk

- É a que define uma janela principal e o interpretador Tcl
- Em geral, nunca precisa ser instanciada
  - É instanciada automaticamente quando um widget filho é criado
- Pode ser instanciada explicitamente
- Possui vários métodos, entre os quais
  - title (*string*) Especifica o título da janela
  - geometry(*string*) Especifica tamanho e posição da janela
    - String tem a forma *largura*x*altura*+*x*+*y*

Exemplo

```
from Tkinter import *

class Application(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.msg = Label(self, text="Hello World”)
        self.msg.pack ()
        self.bye = Button (self, text="Bye", command=self.quit)
        self.bye.pack ()
        self.pack()

app = Application()
app.master.title("Exemplo”)
app.master.geometry("200x200+100+100”)
mainloop()
```

Opções de *Widgets*

- *Widgets* (elementos de interface) têm opções com nomenclatura unificada. Ex.:
  - text Texto mostrado no elemento
  - background cor de fundo
  - foreground cor do texto
  - font fonte do texto
  - relief relevo da borda ('flat', 'raised', 'ridge', 'sunken', 'groove')
- Opções são especificadas
  - No construtor
  - Através do método configure

Exemplo

```
from Tkinter import *
top = Frame() ; top.pack()
rotulo = Label (top, text="Rótulo Exemplo", foreground="blue")
rotulo.pack ()
rotulo.configure(relief="ridge", font="Arial 24 bold", border=5,
    background="yellow”)
mainloop()
```

O método configure

- Usado com pares do tipo *opção=valor*, modifica os valores dos atributos
- Usado com uma string "*nomeopção*" retorna a configuração da opção com esse nome
  - A configuração é uma tupla com 5 valores
    - nome do atributo
    - nome do atributo no banco de dados (X11)
    - nome da classe no banco de dados (X11)
    - objeto que representa a opção
    - valor corrente da opção
- Se configure é usado sem argumentos, retorna um dicionário com todas as opções
- Pode-se obter diretamente o valor de uma opção usando o método cget

Exemplo

```
>>> rotulo.configure(relief="ridge”)
>>> rotulo.configure("relief”)
('relief', 'relief', 'Relief', <index object at 0x85f9530>, 'ridge’)
>>> rotulo.configure()["relief"]
('relief', 'relief', 'Relief', <index object at 0x85f9530>, 'ridge’)
>>> rotulo.configure("relief")[4]
'ridge'
>>> rotulo.cget("relief”)
'ridge'
```

Gerenciando geometrias

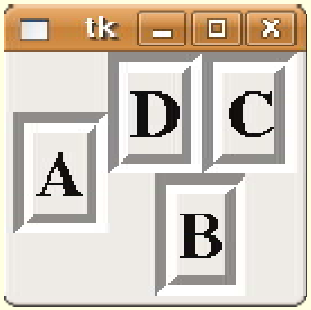
- Todos os elementos de interface ocupam uma área retangular na janela
- A posição e tamanho de cada elemento é determinada por um gerenciador de geometria
  - O elemento não “aparece” enquanto não for informado ao gerenciador
- A geometria resultante depende de
  - Propriedades dos elementos (tamanho mínimo, tamanho da moldura, etc)
  - Opções do gerenciador
  - Algoritmo usado pelo gerenciador
- O gerenciador mais usado em Tk é o **pack**

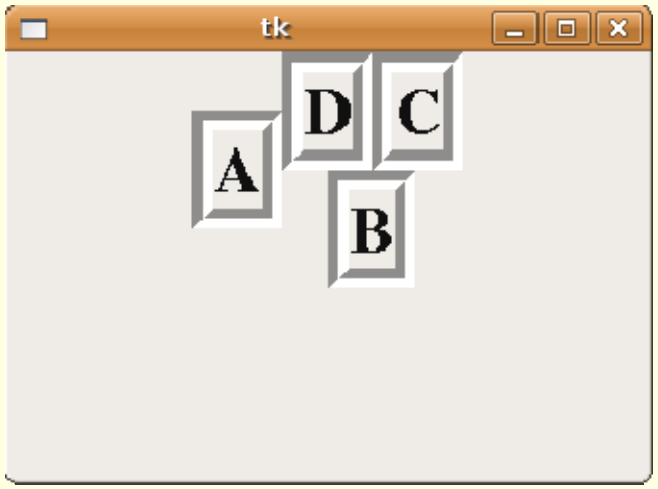
Usando o *pack*

- Para informar que um elemento deve ser gerenciado pelo pack, use o método pack (*opções*)
- O pack considera o espaço do elemento “pai” como uma cavidade a ser preenchida pelos elementos filhos
- O algoritmo usado pelo pack consiste em empacotar os filhos de um elemento “pai” segundo o lado (side) especificado
  - Os lados possíveis são 'top', 'left', 'right' e 'bottom'
  - Deve-se imaginar que sempre que um elemento filho escolhe um lado, a cavidade disponível fica restrita ao lado oposto

Exemplo

```
from Tkinter import *
top = Frame() ; top.pack()
a = Label (top, text="A") ; a.pack (side="left”)
b = Label (top, text="B") ; b.pack (side="bottom”)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", border=10,
                     font="Times 24 bold”)
top.mainloop()
```





Redimensionamento

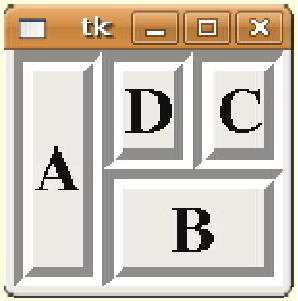
- Por default, o pack não redimensiona os filhos quando o pai é redimensionado
- Duas opções controlam o redimensionamento dos filhos
  - expand (booleano)
    - Se verdadeiro, indica que o filho deve tomar toda a cavidade disponível no pai
    - Caso contrário, toma apenas o espaço necessário (default)
  - fill ('none', 'x', 'y' ou 'both')
    - Indica como o desenho do elemento irá preencher o espaço alocado
    - 'x' / 'y' indica que irá preencher a largura / altura
    - 'both' indica preenchimento de todo o espaço
    - 'none' indica que apenas o espaço necessário será ocupado (default)

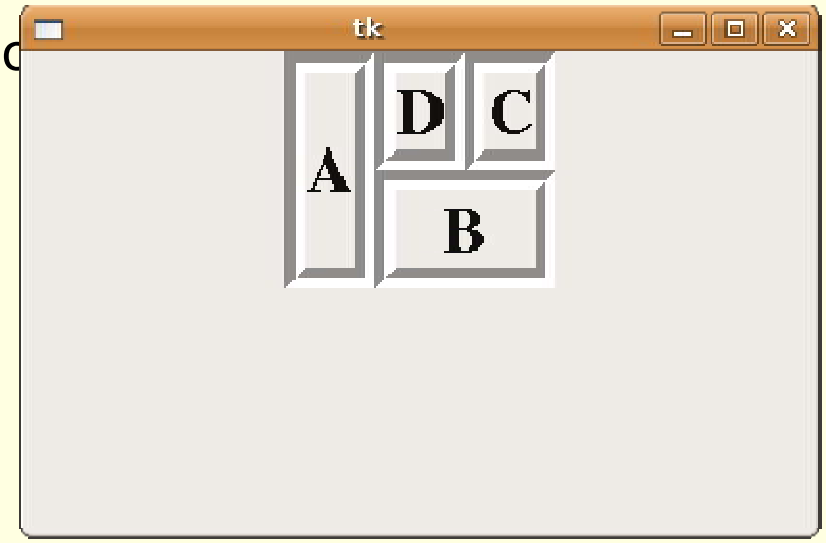
Exemplo

```
from Tkinter import *
top = Frame() ; top.pack()
a = Label (top, text="A") ; a.pack (side="left", fill="y“)
b = Label (top, text="B") ; b.pack (side="bottom", fill="x“)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", border=10, font="Times 24 bold”)
top.mainloop()
```

Exemplo

```
from Tkinter import *
top = Frame() ; top.pack()
a = Label (top, text="A") ; a.pack (side="left", fill="y”)
b = Label (top, text="B") ; b.pack (side="bottom", fill="x”)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", bc
top.mainloop()
```



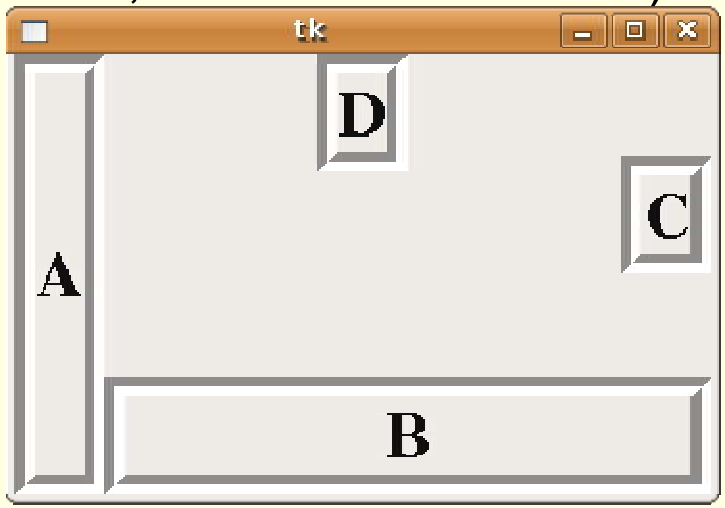


Exemplo

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
a = Label (top, text="A") ; a.pack (side="left",fill="y")
b = Label (top, text="B") ; b.pack (side="bottom",fill="x")
c = Label (top, text="C") ; c.pack (side="right")
d = Label (top, text="D") ; d.pack (side="top")
for widget in (a,b,c,d):
    widget.configure(relief="groove", border=10, font="Times 24 bold")
top.mainloop()
```

Exemplo

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
a = Label (top, text="A") ; a.pack (side="left",fill="y”)
b = Label (top, text="B") ; b.pack (side="bottom",fill="x”)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", border=10, font="Times 24 bold”)
top.mainloop()
```

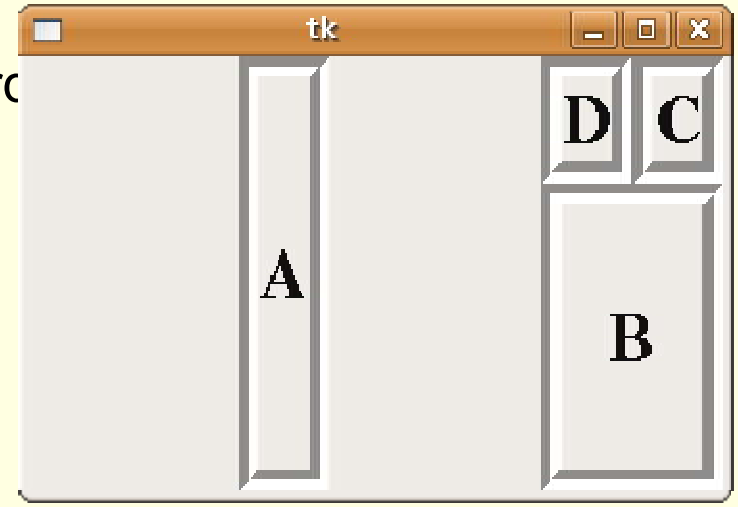


Exemplo

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
a = Label (top, text="A") ; a.pack (side="left",expand=True,fill="y”)
b = Label (top, text="B") ; b.pack
    (side="bottom",expand=True,fill="both”)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", border=10, font="Times 24 bold”)
top.mainloop()
```

Exemplo

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
a = Label (top, text="A") ; a.pack (side="left",expand=True,fill="y”)
b = Label (top, text="B") ; b.pack
    (side="bottom",expand=True,fill="both”)
c = Label (top, text="C") ; c.pack (side="right”)
d = Label (top, text="D") ; d.pack (side="top”)
for widget in (a,b,c,d):
    widget.configure(relief="groove", borc
top.mainloop()
```



Usando frames

- Frames podem ser usados para auxiliar no layout dos elementos com pack. Ex.:

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
f = Frame (top); f.pack (fill='x')
a = Label (f, text="A")
b = Label (f, text="B")
c = Label (f, text="C")
d = Label (top, text="D")
for w in (a,b,c,d):
    w.configure(relief="groove", border=10, font="Times 24 bold")
    w.pack(side="left", expand=True, fill="both")
top.mainloop()
```

Usando frames

- Frames podem ser usados para auxiliar no layout dos elementos com pack. Ex.:

```
from Tkinter import *
top = Frame() ; top.pack(fill='both', expand=True)
f = Frame (top); f.pack (fill='x')
a = Label (f, text="A")
b = Label (f, text="B")
c = Label (f, text="C")
d = Label (top, text="D")
for w in (a,b,c,d):
    w.configure(relief="groove", bo
    w.pack(side="
top.mainloop()
```