



# Interface Gráfica

UTILIZANDO TKINTER – PARTE 1

# 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, passar parâmetros, desenhar e exibir gráficos, etc.

# Interfaces Gráficas em Python

- ▶ Python possui camadas de portabilidades para várias bibliotecas de construção de interfaces. Exemplos:
  - ▶ PyQt
  - ▶ PyGtk
  - ▶ wxPython
  - ▶ Tkinter
- ▶ Multiplataforma (Windows, Linux, OSX)

# Usando Tkinter

- ▶ Importar o módulo Tkinter

```
from Tkinter import *
```

- ▶ A classe Tk também é importada. Ela contém os elementos de interfaces (widgets), necessários para montar a interface gráfica (GUI).
- ▶ Os widgets correspondem a objetos de diversas classes. Por exemplo:
  - ▶ Frame (área retangular)
  - ▶ Button (botão)
  - ▶ Label (rótulo)
  - ▶ Text (caixa de texto)
  - ▶ Canvas (caixa de desenho)

# Exemplo 1

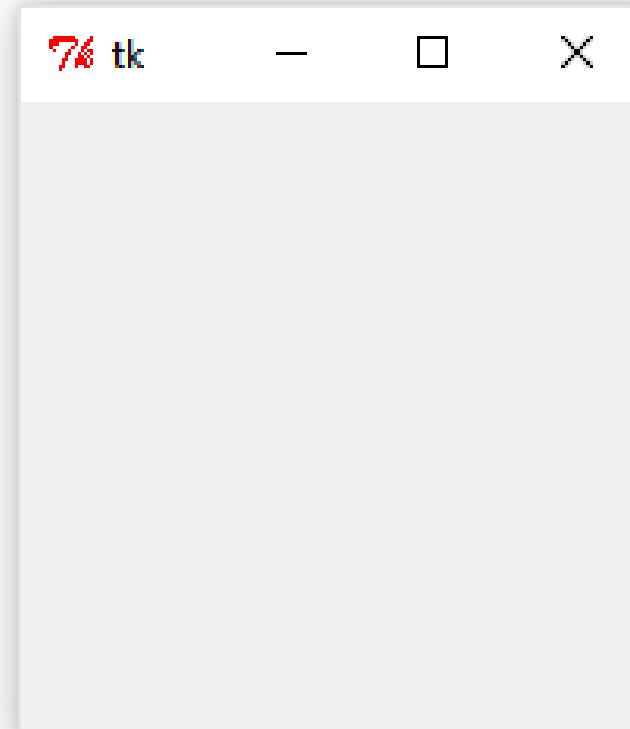
```
from Tkinter import *  
class Janela():  
    def __init__(self, instancia_de_Tk):  
        pass
```

```
raiz = Tk()  
Janela(raiz)  
raiz.mainloop()
```

A GUI fica aberta esperando por acontecimento de eventos. A janela fica aberta até que ocorra algum evento que a “destrua”, como clicar no X ou num botão “fechar”

# Exemplo 1

```
from Tkinter import *  
class Janela:  
    def __init__(self, objeto):  
        pass  
  
raiz = Tk()  
Janela(raiz)  
raiz.mainloop()
```



# Vocabulário

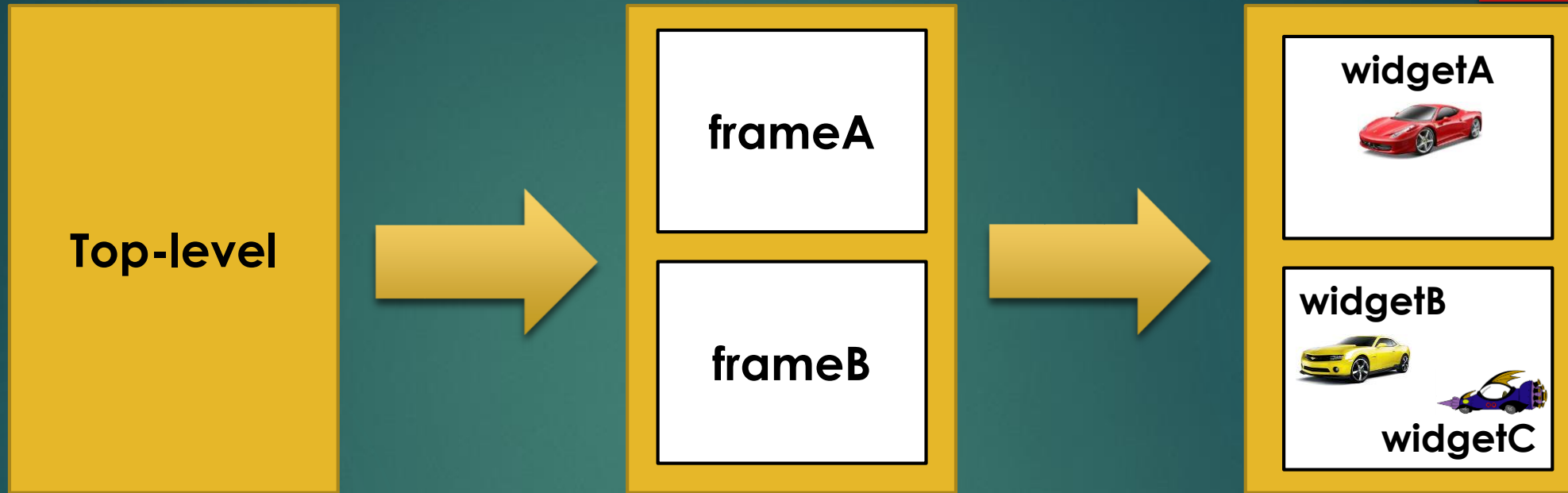
## ▶ Widgets

- ▶ Componente qualquer da interface gráfica: botão, menu, caixa de texto, etc..

## ▶ Containers

- ▶ Elementos da GUI onde os widgets são posicionados. O mais comum é o **Frame**

# Estrutura Hierárquica



## Código:

```
frameA (Top-level)  
frameB (Top-level)  
widgetA (frameA)  
widgetB (frameB)  
widgetC (frameB)
```

**Mestre**



# Gerenciadores de Geometria

- ▶ Indicam em que posição um container ou widget irá aparecer dentro do seu mestre
- ▶ Três opções: **Grid**; **Pack**; e **Place**



# Exemplo 2

```
from Tkinter import *  
class Janela():  
    def __init__(self, toplevel):  
        self.fr1 = Frame(toplevel)  
        self.fr1.pack()  
        self.botao = Button(self.fr1, text='Ola!', background='yellow')  
        self.botao.pack()
```

**Toplevel é o mestre de Frame**

**O pack() torna os componentes visíveis**

```
raiz = Tk()  
Janela(raiz)  
raiz.mainloop()
```

↑  
**texto**

↑  
**Cor de fundo**  
**Atributo equivalente: bg**

# Exemplo 2

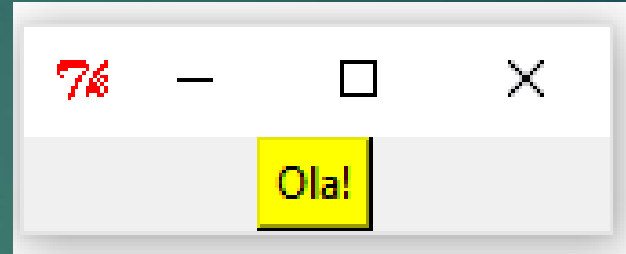
```
from Tkinter import *  
class Janela():
```

```
    def __init__(self, toplevel):  
        self.fr1 = Frame(toplevel)
```

```
        self.fr1.pack()  
        self.botao = Button(self.fr1, text='Ola!', background='yellow')  
        self.botao.pack()
```

```
raiz = Tk()  
Janela(raiz)  
raiz.mainloop()
```

**Toplevel é o mestre de Frame**



**O pack() torna os componentes visíveis**

**texto**

**Cor de fundo**  
**Atributo equivalente: bg**

# Sintaxe

- ▶ A sintaxe para criar um widget qualquer é:

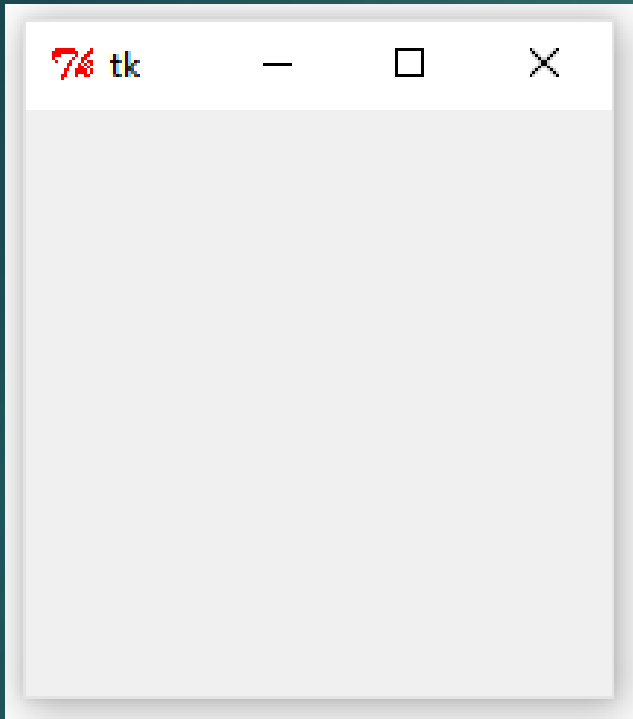
*nome\_do\_widget(mestre, opções de configuração)*

- ▶ Os widgets guardam as suas opções de configuração em forma de **dicionários**. Poderíamos ter configurado a opção do botão no exemplo anterior como:

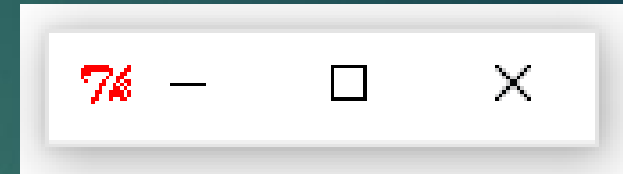
```
self.botao['bg'] = 'yellow'
```

```
self.botao['text'] = 'Ola!'
```

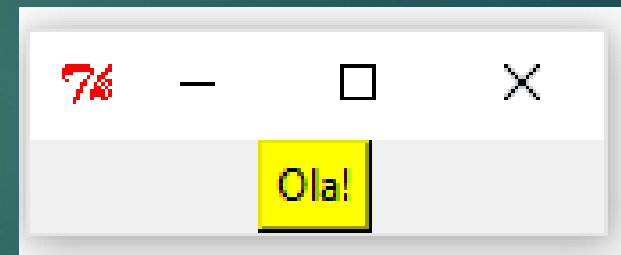
# Tamanho da janela



Tamanho padrão



Tamanho do frame



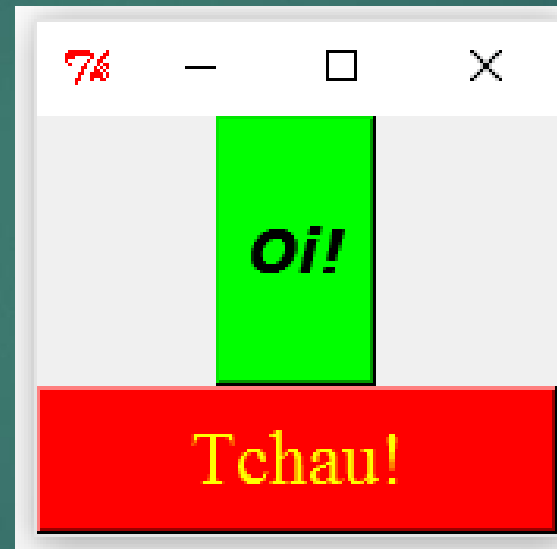
Tamanho necessário  
para comportar o botão

# Outras configurações do Button

- ▶ **height**: altura do botão em número de linhas de texto
- ▶ **width**: largura do botão em quantidade de letras
- ▶ **font**: fonte do texto. Representada pela tupla:  
(“tipo da fonte”, “tamanho”, “negrito”, “itálico”,...)  
Exemplo: font = ('Arial', 15, 'bold', 'italic')
- ▶ **fg** ou **foreground**: cor do texto

# Exercício (3)

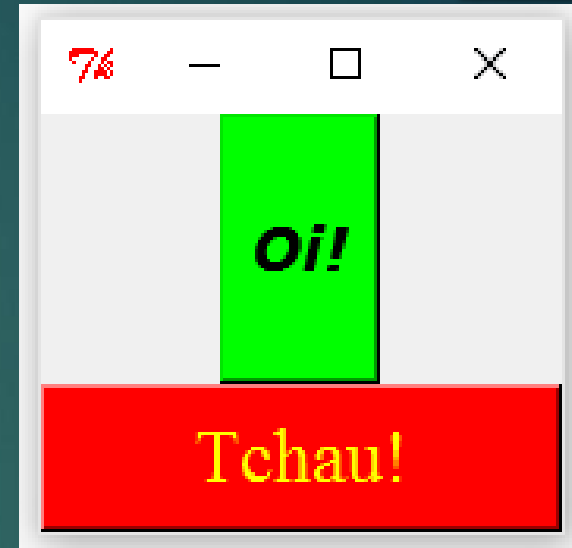
- ▶ Tente criar a GUI a seguir:



# Exercício (3)

```
from Tkinter import *
class Janela():
    def __init__(self, toplevel):
        self.fr1 = Frame(toplevel)
        self.fr1.pack()
        self.botao1 = Button(self.fr1, text='Oi!', bg='green', height=3)
        self.botao1['font']=('Verdana',14,'italic','bold')
        self.botao1.pack()
        self.botao2 = Button(self.fr1, text='Tchau!', bg='red', width=12, fg='yellow', font=('Times',18))
        self.botao2.pack()

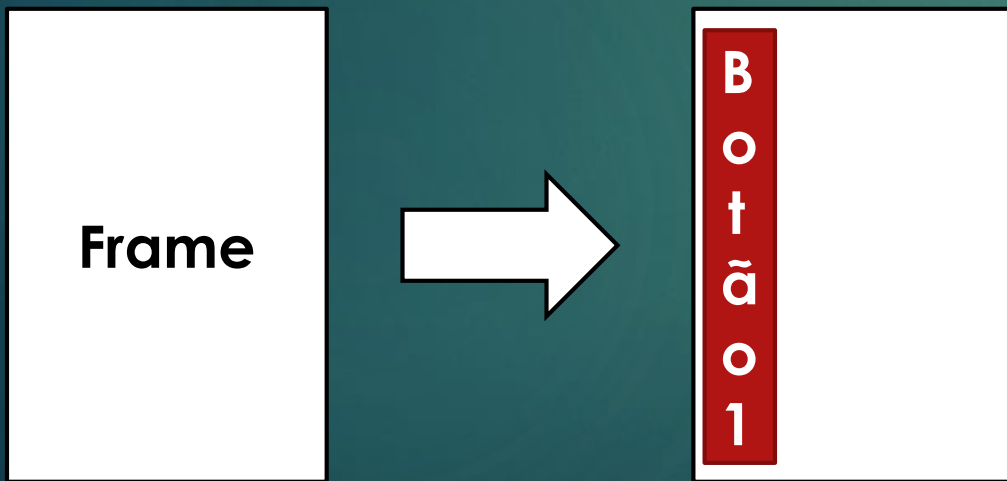
raiz = Tk()
Janela(raiz)
raiz.mainloop()
```



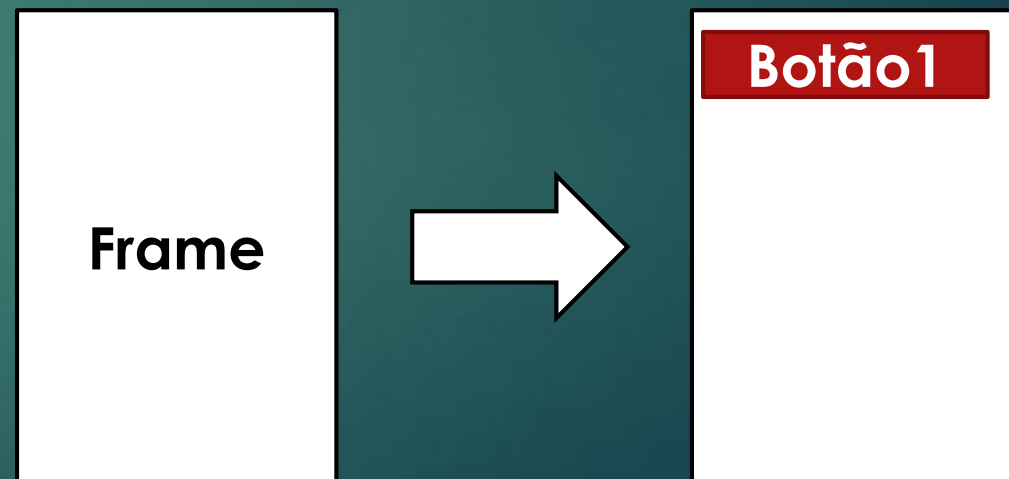


# Posição dos Widgets

- ▶ O método **pack** pode receber o argumento **side**, capaz de assumir quatro valores: LEFT (esquerda), RIGHT (direita), TOP (em cima), BOTTOM (embaixo)
- ▶ Estes valores determinam em que lugar o widget deve se posicionar. O valor padrão é TOP



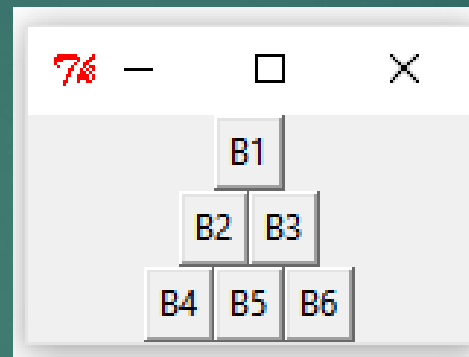
`Botao1.pack(side=LEFT)`



`Botao1.pack(side=TOP)`

# Exemplo 4

- ▶ Vamos criar a GUI abaixo:



# Exemplo 4

```
from Tkinter import *
```

```
class Janela():
```

```
    def __init__(self, toplevel):
```

```
        self.container1 = Frame(toplevel)
```

```
        self.container2 = Frame(toplevel)
```

```
        self.container1.pack()
```

```
        self.container2.pack()
```

```
        self.b1 = Button(self.container1, text='B1')
```

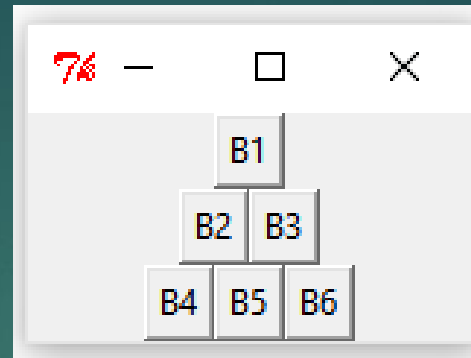
```
        self.b1.pack()
```

```
        self.b2 = Button(self.container1, text='B2')
```

```
        self.b2.pack(side=LEFT)
```

```
        self.b3 = Button(self.container1, text='B3')
```

```
        self.b3.pack(side=LEFT)
```



```
        self.b4 = Button(self.container2, text='B4')
```

```
        self.b5 = Button(self.container2, text='B5')
```

```
        self.b6 = Button(self.container2, text='B6')
```

```
        self.b4.pack(side=LEFT)
```

```
        self.b5.pack(side=LEFT)
```

```
        self.b6.pack(side=LEFT)
```

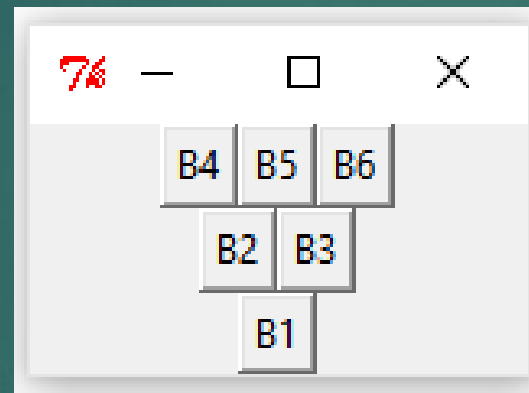
```
raiz = Tk()
```

```
Janela(raiz)
```

```
raiz.mainloop()
```

# Exercício (5)

► Tente criar:

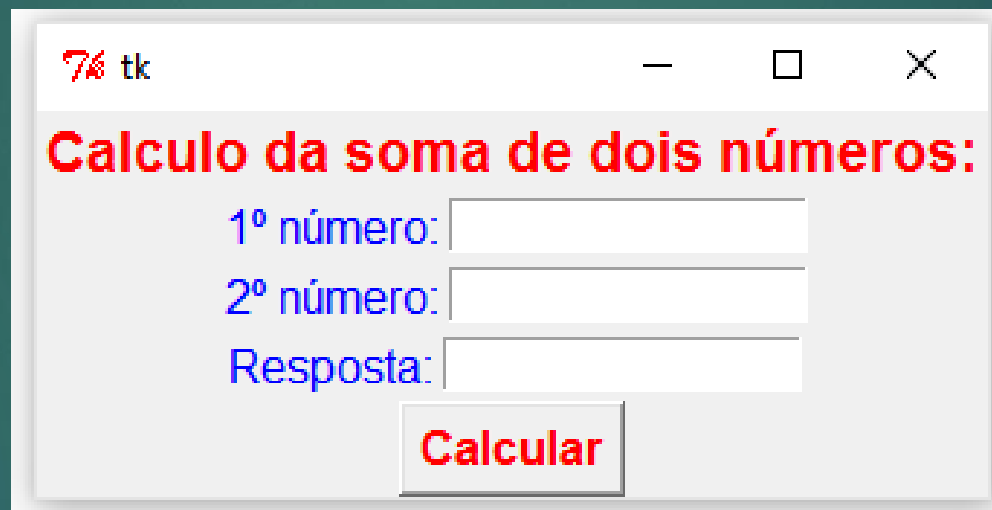


# Widget Label e Entry

- ▶ **Label** serve para textos na tela
- ▶ O **Label** possui todas as opções de configuração apresentadas para o Button
- ▶ O **Entry** é um meio de recolher entradas de dados do usuário.
- ▶ O **Entry** aceita todas as opções de configuração do Label e Button, menos o *height*, porque a altura dos campos de dados é sempre de uma linha

# Exemplo 6

- Como fazer a tela abaixo?



A screenshot of a Tkinter window titled "7 tk". The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a green window control button. The main content area has a light gray background. At the top, the text "Calculo da soma de dois números:" is displayed in bold red font. Below this, there are three labels in blue font: "1º número:", "2º número:", and "Resposta:". Each label is followed by a white text input field. At the bottom center, there is a button with a gray border and a red "Calcular" label.

# Exemplo 6

```
from Tkinter import *
```

```
class Janela():
```

```
    def __init__(self, toplevel):
```

```
        self.fr1 = Frame(toplevel)
```

```
        self.fr2 = Frame(toplevel)
```

```
        self.fr3 = Frame(toplevel)
```

```
        self.fr4 = Frame(toplevel)
```

```
        self.fr5 = Frame(toplevel)
```

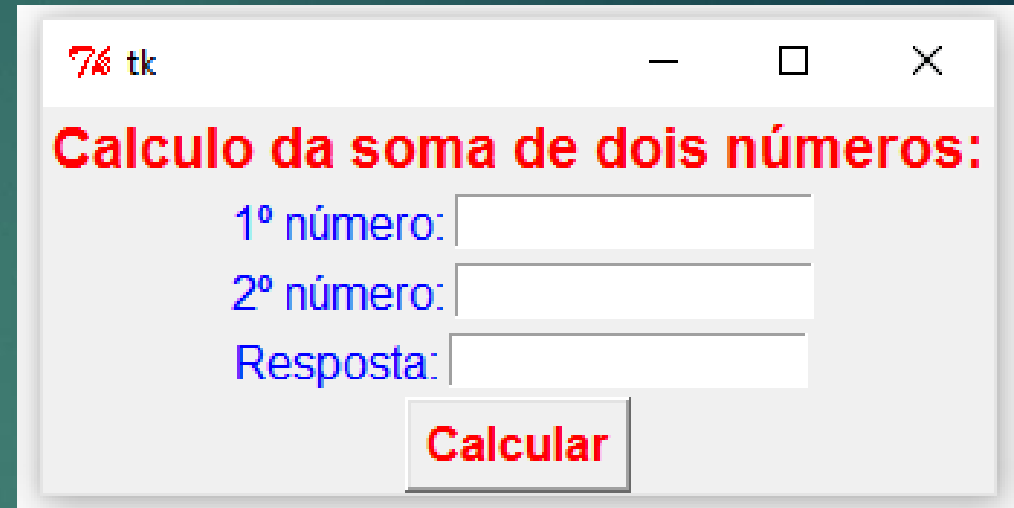
```
        self.fr1.pack()
```

```
        self.fr2.pack()
```

```
        self.fr3.pack()
```

```
        self.fr4.pack()
```

```
        self.fr5.pack()
```



# Exemplo 6

```
Label(self.fr1,text='Calculo da soma de dois números:', fg='red',
font=('Arial',14,'bold')).pack()

Label(self.fr2,text='1º número:',fg='blue',font=('Arial',12)).pack(side=LEFT)
self.e_n1 = Entry(self.fr2)
self.e_n1.pack(side=LEFT)

Label(self.fr3,text='2º número:',fg='blue',font=('Arial',12)).pack(side=LEFT)
self.e_n2 = Entry(self.fr3)
self.e_n2.pack(side=LEFT)

Label(self.fr4,text='Resposta:',fg='blue',font=('Arial',12)).pack(side=LEFT)
self.e_resposta = Entry(self.fr4)
self.e_resposta.pack(side=LEFT)

self.b_calcular = Button(self.fr5,text='Calcular',fg='red',font=('Arial',12,'bold'))
self.b_calcular.pack()

raiz = Tk()

...
```



# Eventos

- ▶ **Event handlers** (tratadores de eventos): métodos programados para serem executados quando um evento específico acontece
- ▶ Exemplo: Ao clicar no botão “Calcular”, o método que calcula a soma é o **event handler** correspondente a clicar neste botão
- ▶ Para que o método seja executado no clique do botão, é preciso fazer um **binding** (acomplamento) do botão com o método.

# Binding

- ▶ Para associar o clique do mouse sobre um widget a um **event handler**, usamos o comando *command*

```
widget(mestre,...,command = self.event_handler)
```

- ▶ Voltando ao exemplo anterior... Acrescentaremos ao botão “Calcular” a chamada à função desejada

```
self.b_calcular = Button(self.fr5,text='Calcular',...,command=self.calcularSoma)
```

```
def calcularSoma(self):
```

```
    print “o botão foi clicado para calcular a soma”
```

# Entry

- ▶ Os dados informados pelo usuário neste campo são captados em forma de string, assim como acontece quando utilizamos a função *raw\_input*
- ▶ Através do método **get**, podemos trabalhar com os dados de entrada dentro do código

# Exemplo 7

- ▶ Ainda utilizando o exemplo da soma, iremos agora implementar o método *calcularSoma* e desabilitar o **Entry** de resposta para que o campo não possa ser digitado

```
self.e_resposta = Entry(self.fr4, state=DISABLED)
```

```
def calcularSoma(self):  
    num1 = float(self.e_n1.get())  
    num2 = float(self.e_n2.get())  
    print num1 + num2
```

- ▶ Se quiser habilitar o Entry, basta utilizar: `state = NORMAL`

# Exemplo 7

- ▶ Para mostrar a informação em um Entry é preciso vinculá-lo a uma variável de controle

```
self.e_resposta = Entry(self.fr4, state=DISABLED, textvar=tv)
```

- ▶ Variáveis de controle possuem dois métodos:

.get(): retorna o valor corrente de uma variável

.set(value): altera o valor corrente de uma variável

```
def calcularSoma(self):  
    num1 = float(self.e_n1.get())  
    num2 = float(self.e_n2.get())  
    tv.set(num1 + num2)
```

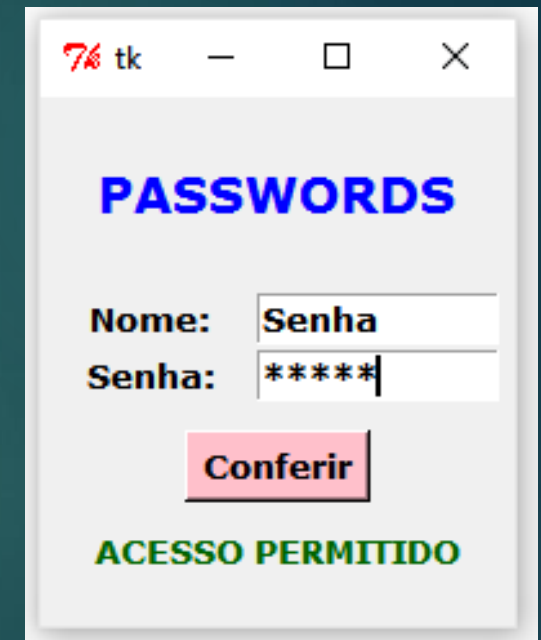
# Variáveis de Controle

- ▶ Além das modificações realizadas até aqui, ainda se faz necessário que as variáveis de controle estejam definidas globalmente, junto com seu tipo desejado. Os tipos disponíveis são:
  - ▶ DoubleVar(): guarda um float, e o valor default é 0.0
  - ▶ IntVar(): guarda um inteiro, e o valor default é 0
  - ▶ StringVar(): guarda uma String, e o valor default é ' '

```
raiz = Tk()  
tv = StringVar(raiz)  
Janela(raiz)  
raiz.mainloop()
```

# Exercício (8)

- ▶ Faça uma tela para verificação de senhas. Caso a senha digitada esteja correta, deve-se mostrar “ACESSO PERMITIDO”, caso contrário, “ACESSO NEGADO”.
- ▶ Enquanto não clicar no Conferir, escreva AGUARDANDO...". Assuma que a senha correta é o que for digitado no campo Nome.
- ▶ Para aparecer \* no lugar do que estiver sendo digitado, utilize a opção show='\*' no Entry da senha.



The screenshot shows a Tkinter window titled "PASSWORDS" with a standard macOS-style title bar (red, yellow, green buttons). Inside the window, the text "PASSWORDS" is displayed in large blue font. Below it, there are two labels: "Nome:" and "Senha:". The "Nome:" label is followed by a text entry field containing the word "Senha". The "Senha:" label is followed by a password entry field containing five asterisks "\*\*\*\*\*". Below these fields is a pink button labeled "Conferir". At the bottom of the window, the text "ACESSO PERMITIDO" is displayed in green font.