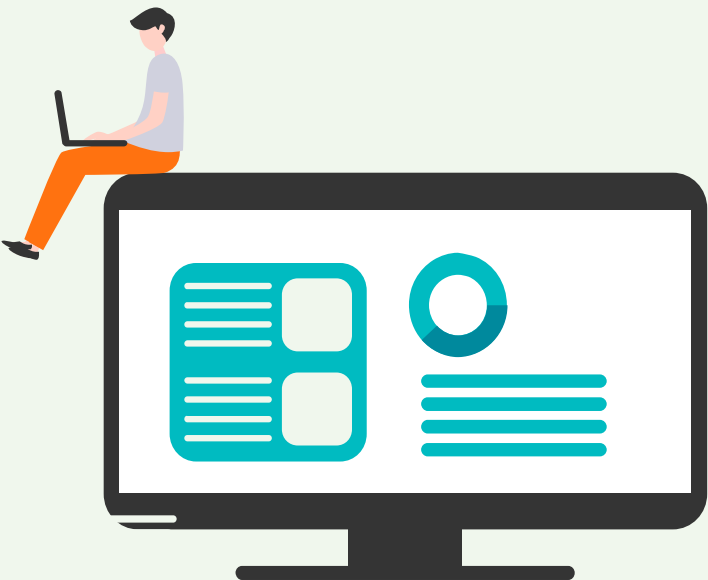


PYTHON PARA APLICATIVOS

Funções e Eventos

Profa. Esp. Sônia Gomes de Oliveira

PAULISTA, 2024



SOBRE MIM

Especialista em Educação Profissional e Tecnológica – UNIBF (2024)

Especialista em Banco de dados – UNIBF (2024)

MBA em Engenharia de Software – FAMEESP – 2023

Capacitada em Tecnologia da Informação e Comunicação – TIC UFG(2022)

Graduada em Ciência da Computação – UEPB (2020)





EXPERIÊNCIAS PROFISSIONAIS

SENAC – RECIFE (2024)

- Projeto Transforme-se

UNINASSAU – PAULISTA (2024)

- Back-End Frameworks

UNIBRA (2023 e 2024)

- Segurança da Informação
- Fundamentos de Banco de Dados

UNI SÃO MIGUEL (2023)

- Banco de Dados
- Lógica de Programação


SENAC – RECIFE (2023)

- Curso Técnico de Informática
- Programador de Sistemas
- Aprendizagem em Desenvolvimento de Software
- Ensino Médio Técnico

SENAC – RECIFE (2022)

- Programação em Python
- Recursos Tecnológicos (área de vendas)

UEPB – PARAÍBA – 2018/2019

- Tutora de Engenharia de Software
- 



01

FUNÇÕES E EVENTOS



Funções e Eventos

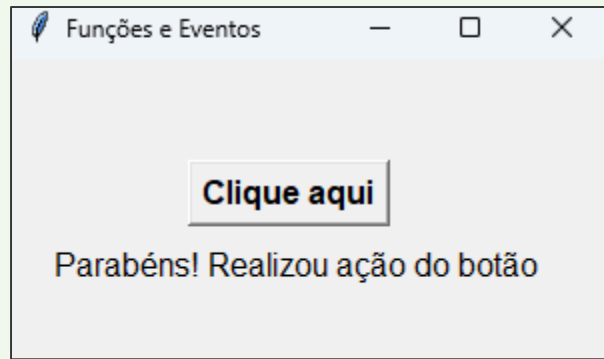
Para realizar um evento do botão deverá criar uma função que irá receber essa ação. E em seguida utiliza o **command** para receber essa função.

```
class Tela:
    def __init__(self, janela):
        self.janela = janela
        self.janela.title('Funções e Eventos')
        self.janela.geometry('300x200+550+100')

        self.botao = tk.Button(self.janela, text='Clique aqui',
                                font=('Arial', 12, 'bold'),
                                command= self.mostrar_texto)
        self.botao.place(x=90, y=50)

        self.mensagem = tk.Label(self.janela, font=('Arial', 12))
        self.mensagem.place(x=20, y=90)

    def mostrar_texto(self):
        self.mensagem['text'] = 'Parabéns! Realizou ação do botão'
```





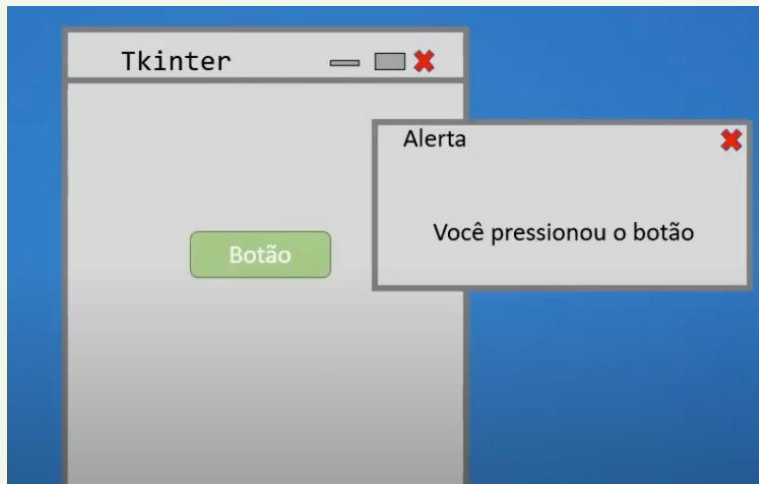
02

EVENTOS COM MESSAGEBOX E BIND



Eventos com o comando Bind

O comando **bind** é a ligação do evento com o tratador de eventos.



Quando usar uma função com o comando **bind** não é necessário chamar pelo **command**.

Chamando uma caixa de diálogo

```
from tkinter import messagebox
```

```
1 usage
```

```
class Tela:
```

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

```
        self.janela = janela
```

```
        self.janela.title('Funções e Eventos')
```

```
        self.janela.geometry('300x200+550+100')
```

```
        self.mensagem = tk.Label(self.janela,  
                                  text='Abrir caixa de diálogo',  
                                  relief='raised')
```

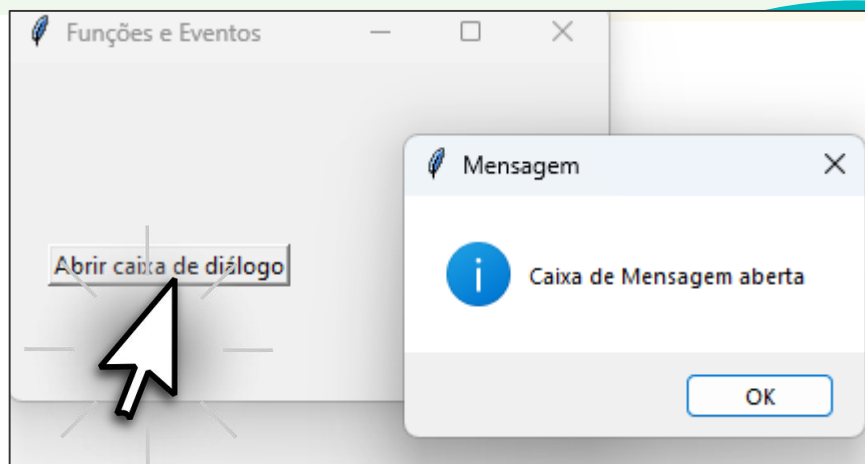
```
        self.mensagem.place(x=20, y=90)
```

```
        self.mensagem.bind('<Button-1>', self.resposta)
```

```
1 usage
```

```
def resposta(self, event):
```

```
    messagebox.showinfo('Mensagem', 'Caixa de Mensagem aberta')
```



Eventos sobre o botão do Mouse

O evento ocorre quando algum botão do mouse é apertado. Os botões são representados entre os **números 1 e 5**.

- < **Button-1** > : Botão esquerdo do mouse.
- < **Button-2** > : Botão de rolagem(Scroll) do mouse.
- < **Button-3** > : Botão direito do mouse.
- < **Button-4** > : Refere-se ao botão de rolagem para cima no mouse.
- < **Button-5** > : Refere-se ao botão de rolagem para baixo no mouse.



Eventos sobre o teclado

O evento ocorre quando alguma tecla é apertada. As teclas são representadas como:

<Return>: A tecla return / enter

<KeyPress> ou **<Key>** : Dispara qualquer tecla ao ser pressionada.

< KeyRelease> : Dispara quando a tecla é solta ou liberada.

< KeyPress-tecla> ou **>key-tecla>** : Quando uma tecla específica é pressionada.

<KeyRelease-*tecla*> : Quando a *tecla* específica é solta ou liberada

<prefixo-*tecla*> - Dispara quando você aperta a tecla específica *tecla* enquanto segura a *prefixo*. Esse prefixo pode ser Alt, Shift ou Control apenas.

Conhecendo os tipos de caixas de mensagens

Tem que importar o messagebox. O evento das caixas de mensagens:

showinfo	→ TEXTO DE INFORMAÇÃO
showwarning	→ AVISO
showerror	→ ERRO
askquestion	→ PERGUNTA
askyesnocancel	→ SIM - NÃO - CANCELAR





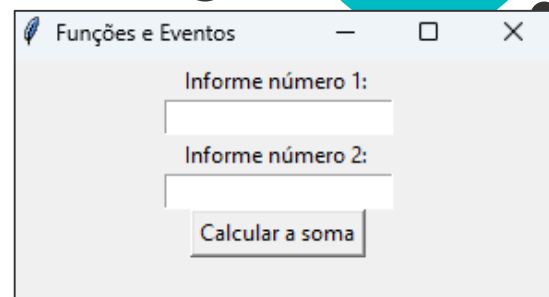
03

CALCULANDO TIPOS PRIMITIVOS DE DADOS



Criando os elementos

```
def __init__(self, janela):  
    self.janela = janela  
    self.janela.title('Funções e Eventos')  
    self.janela.geometry('300x200+550+100')  
  
    self.Label_n1 = tk.Label(self.janela, text='Informe número 1: ')  
    self.Label_n1.pack()  
    self.campo_n1 = tk.Entry(self.janela)  
    self.campo_n1.pack()  
    self.label_n2 = tk.Label(self.janela, text='Informe número 2: ')  
    self.label_n2.pack()  
    self.campo_n2 = tk.Entry(self.janela)  
    self.campo_n2.pack()  
    self.botao = tk.Button(self.janela, text='Calcular a soma', command=self.soma)  
    self.botao.pack()  
    self.mensagem = tk.Label(self.janela, font=('Arial', 14, 'bold'))  
    self.mensagem.pack()
```



Função Soma

1 usage

```
def soma(self):  
    try:  
        self.n1 = int(self.campo_n1.get())  
        self.n2 = int(self.campo_n2.get())  
        self.mensagem['text'] = f'A soma é {self.n1 + self.n2}'  
    except ValueError:  
        self.mensagem['text'] = 'Só aceitamos números!'
```

Funções e Eventos

Informe número 1:
2

Informe número 2:
3

Calcular a soma

A soma é 5

Funções e Eventos

Informe número 1:
aaaaaa

Informe número 2:
3

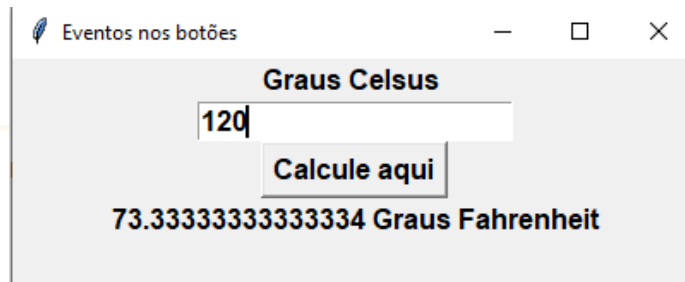
Calcular a soma

Só aceitamos números!

Função - Números com casas decimais

```
self.Label_n1 = tk.Label(self.janela, text='Graus Celsius ')
self.Label_n1.pack()
self.campo_n1 = tk.Entry(self.janela)
self.campo_n1.pack()
self.botao2 = tk.Button(self.janela, text='Calcule aqui', command=self.numeros_decimais)
self.botao2.pack()
self.mensagem = tk.Label(self.janela, font=('Arial', 14, 'bold'))
self.mensagem.pack()

def numeros_decimais(self):
    try:
        self.grausF = float(self.campo_n1.get())
        self.grausC = (self.grausF - 32) * (5/6)
        self.mensagem['text'] = str(self.grausC) + ' Graus Fahrenheit'
    except ValueError:
        self.mensagem['text'] = 'Apenas números!'
```





**Adicionando
Imagens**

Adicionando Imagens PNG

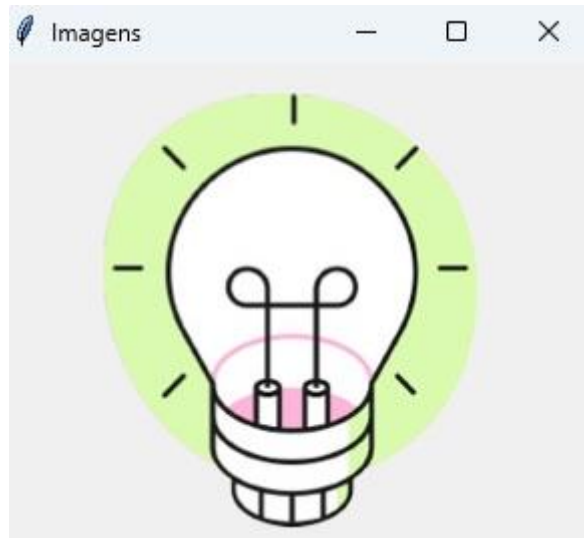
```
import tkinter as tk
from tkinter import *
from tkinter import messagebox

1 usage

class Tela:
    def __init__(self, janela):
        self.janela = janela
        self.janela.title('Imagens')
        self.janela.geometry('300x200+550+100')

        self.imagem = tk.PhotoImage(file='image.png')
        self.label = Label(self.janela, image=self.imagem)
        self.label.pack()

janela = tk.Tk()
objeto = Tela(janela)
janela.mainloop()
```





Criando Frames

Criando Frames

Um Frame é uma região retangular na tela. Serve para organizar componentes e facilitar na organização de itens dentro da janela.

```
class Tela:
    def __init__(self, janela):
        self.janela = janela
        self.janela.geometry('320x250')

        self.f_1 = tk.Frame(self.janela, width=150, height=100, bg="blue")
        self.f_1.grid(row=0, column=0, padx=5, pady=5)

        self.f_2 = tk.Frame(self.janela, width=150, height=100, bg="yellow")
        self.f_2.grid(row=0, column=1, padx=5, pady=5)

        self.f_3 = tk.Frame(self.janela, width=150, height=100, bg="red")
        self.f_3.grid(row=1, column=0, padx=5, pady=5)

        self.f_4 = tk.Frame(self.janela, width=150, height=100, bg="green")
        self.f_4.grid(row=1, column=1, padx=5, pady=5)

janela = tk.Tk()
objeto = Tela(janela)
janela.mainloop()
```



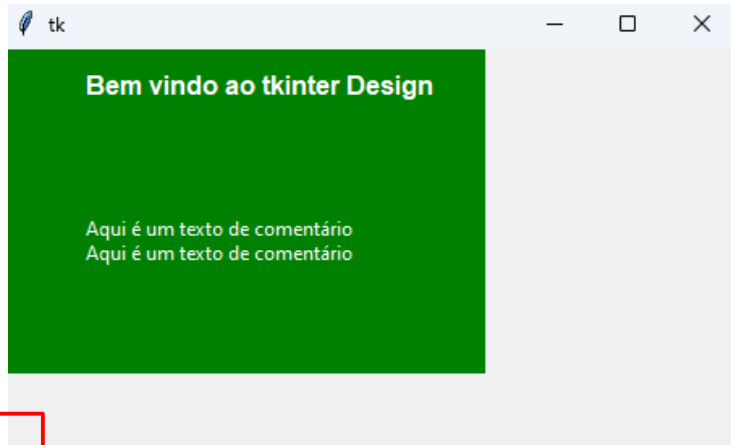
Colocando elementos no Frame

```
class Tela:
    def __init__(self, janela):
        self.janela = janela
        self.janela.geometry('320x250')

        self.frame = tk.Frame(self.janela, bg='green', width=300, height=200)
        self.frame.grid(columns=1, row=0)
        self.texto = tk.Label(self.frame, text='Bem vindo ao tkinter Design',
                               font='Arial 12 bold', bg='green', fg='white')
        self.texto.place(x=50, y=10)

        self.texto1 = tk.Label(self.frame, bg='green', fg='white')
        self.texto1['text'] = 'Aqui é um texto de comentário' \
                               '\nAqui é um texto de comentário'
        self.texto1.place(x=50, y=100)

janela = tk.Tk()
objeto = Tela(janela)
janela.mainloop()
```





**Cr iando Janelas
Secundár ias**

Janelas Secundárias com TopLevel()

Existe duas maneira de criar interfaces secundárias com: **TopLevel** e por **Instância**.

```
class Tela:
```

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

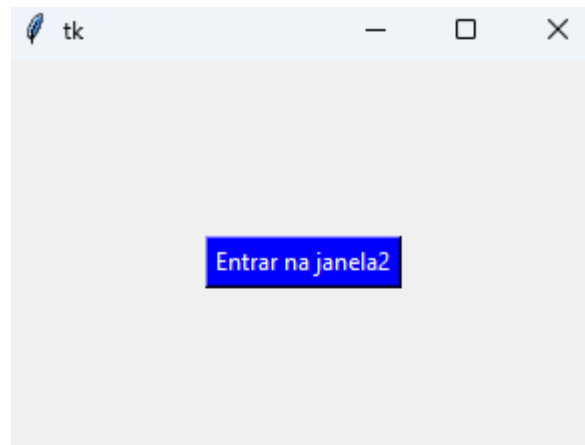
```
        self.janela1 = janela
```

```
        self.janela1.geometry('300x200+550+100')
```

```
        self.botao1 = tk.Button(self.janela1,  
                                text='Entrar na janela2',  
                                bg='blue', fg='white',  
                                command=self.segunda_janela)
```

```
        self.botao1.pack(expand=True)
```

Criando a primeira janela e colocando um botão



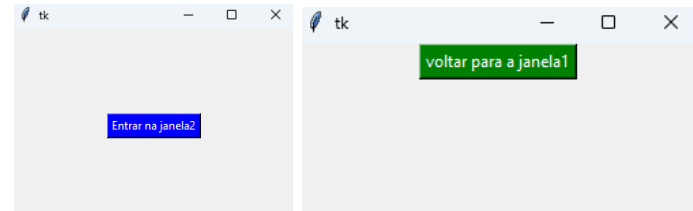
Janelas Secundárias com Toplevel()

Agora precisamos criar os métodos

```
def segunda_janela(self):  
    self.janela1.withdraw() #esconde a janela1  
    self.janela2 = Toplevel()  
    self.janela2.geometry('300x200+550+100')  
    self.botao2 = tk.Button(self.janela2, text='voltar para a janela1',  
                             bg='green', fg='white', command=self.voltar_janela1)  
    self.botao2.pack()
```

1 usage

```
def voltar_janela1(self):  
    self.janela2.withdraw() #esconde a janela2  
    self.janela1.deiconify() #mostra a janela1  
    #self.janela1.destroy() #destrói a janela
```



O primeiro método se refere a 2ª Janela

O segundo método volta para a janela principal

Janelas Secundárias com Instância

Agora precisamos criar os métodos

```
def instancia_janela(self):  
    self.janela1.withdraw()  
    self.janela2 = Tk()  
    self.janela2.geometry('300x200+550+100')  
    self.botao2 = tk.Button(self.janela2, text='voltar para a janela1',  
                             bg='green', fg='white', command=self.voltar_janela1)  
    self.botao2.pack()
```

```
def voltar_janela1(self):  
    self.janela2.destroy()  
    self.janela1.deiconify()
```




03

CAPTANDO DADOS DOS CAMPOS DE TEXTOS



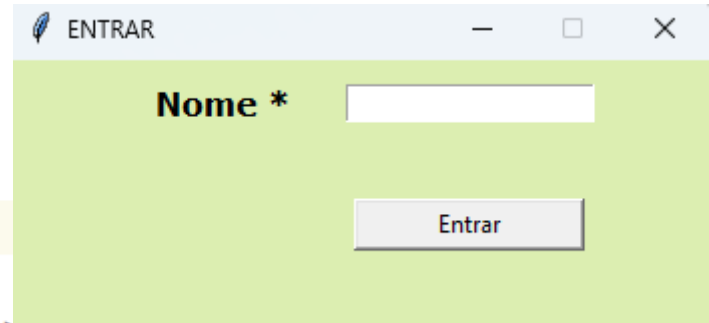
Criando os elementos

Coloca um ícone na janela

```
from tkinter import messagebox

def __init__(self, janela):
    self.janela = janela
    self.janela.title('ENTRAR')
    #self.janela.iconbitmap('icone.ico')
    self.janela['background'] = ('#dceeb1')
    self.janela.geometry('350x150+450+100')
    self.janela.resizable(width=False, height=False)
```

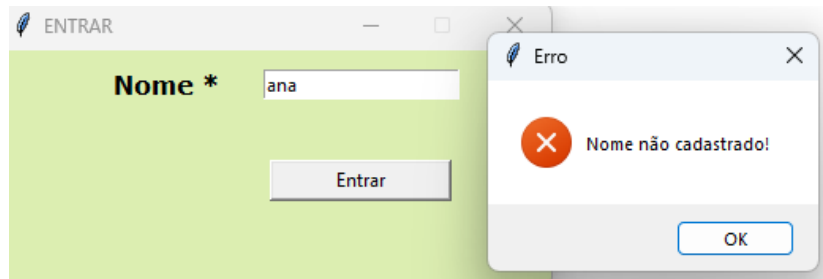
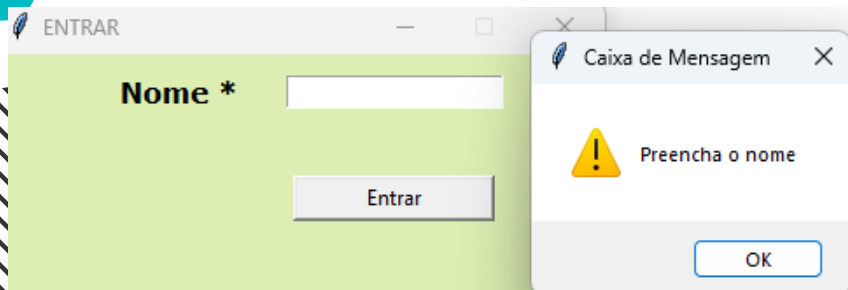
```
self.login=tk.Label(self.janela, text='Nome * ', font='verdana 12 bold', bg='#dceeb1')
self.login.grid(column=0, row=0, padx=(70,20), pady=(10))
self.campo1 = tk.Entry(self.janela)
self.campo1.grid(column=1, row=0)
self.botao1 = tk.Button(self.janela, text='Entrar', width=15, command=self.verifica)
self.botao1.grid(column=1, row=2, pady=25)
```



Método de Captação dos dados

```
def verifica(self):  
    nome = self.campo1.get()  
    if nome == 'joao':  
        self.iniciar() #Método de criação de outra janela  
    elif nome == '':  
        messagebox.showwarning('Caixa de Mensagem', 'Preencha o nome')  
    else:  
        messagebox.showerror('Erro', 'Nome não cadastrado!')
```

Criando o método de verificação se o nome é João. Só irá permitir ir para outra tela caso o nome seja válido.



Método de mostrar janela

O comando `focus`, dá foco a janela secundária. Já o `destroy()` apaga permanentemente a janela principal

```
def iniciar(self):  
    self.janela1 = Tk()  
    self.janela1 = self.janela1  
    self.janela1['background'] = ('#008F8C')  
    self.janela1.geometry('800x500+320+100')  
    self.janela1.resizable(width=False, height=False)  
    self.janela1.focus_force() # dando foco na segunda tela  
    self.janela.destroy()  
    self.janela1.mainloop()
```

