

Computação II

MAB 225 - EPT/EP1

Interface Gráfica - Tkinter

Brunno Goldstein

bfgoldstein@cos.ufrj.br

www.cos.ufrj.br/~bfgoldstein

Ementa

- Programação Orientada a Objetos
- Tratamento de Exceções
- Módulos
- Manipulação de Arquivos
- Interface Gráfica (Tkinter)
- Biblioteca Numérica (Numpy)

Ementa

- Programação Orientada a Objetos
- Tratamento de Exceções
- Módulos
- Manipulação de Arquivos
- Interface Gráfica (Tkinter)
- Biblioteca Numérica (Numpy)

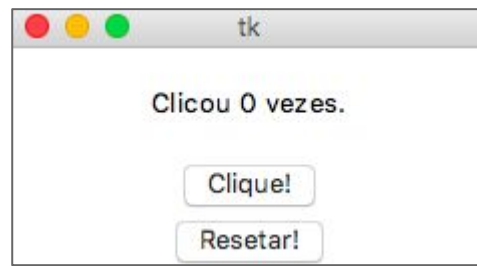
Conceitos Básicos

GUI - Graphical User Interface

- Interface gráfica;
- Programa que facilita a interação do usuário com outros programas;
- Fica em 'loop' infinito:
 - Até o usuário clicar em algum widget da interface;

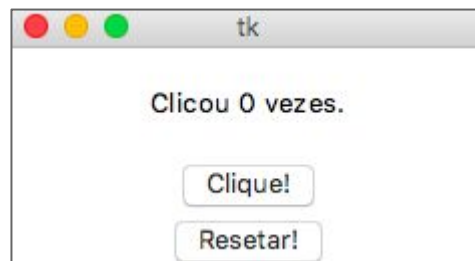
Exemplo de GUI

Exemplo de GUI



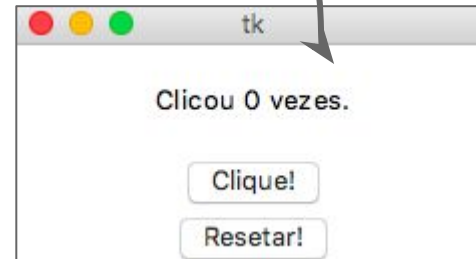
Tkinter

- Ferramenta em Python para desenvolvimento de GUIs;
- Termos que devemos conhecer:
 - Widget;
 - Event;
 - Event handler;
 - Binding;
 - Container;
 - Pack.



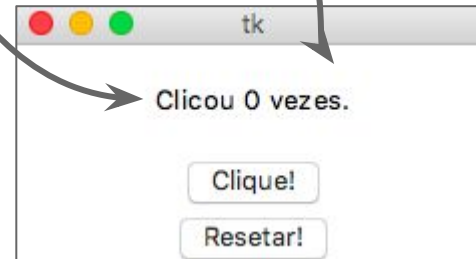
Widget

- Qualquer componente da nossa interface gráfica;
- No nosso exemplo nós temos 4 widgets:
 - Janela principal;
 - Campo do texto;
 - Botão "Clique!";
 - Botão "Resetar!".



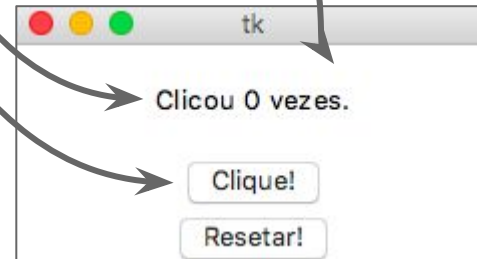
Widget

- Qualquer componente da nossa interface gráfica;
- No nosso exemplo nós temos 4 widgets:
 - Janela principal;
 - Campo do texto;
 - Botão "Clique!";
 - Botão "Resetar!".



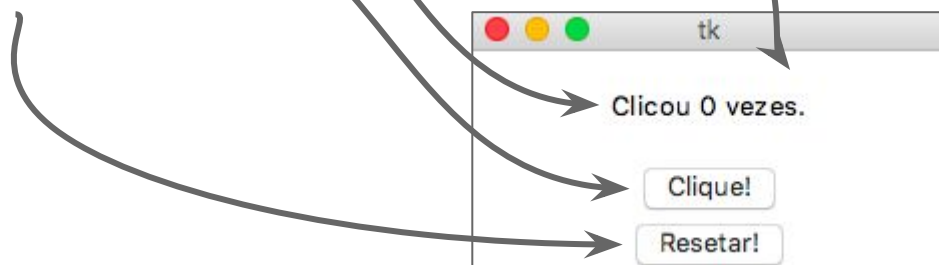
Widget

- Qualquer componente da nossa interface gráfica;
- No nosso exemplo nós temos 4 widgets:
 - Janela principal;
 - Campo do texto;
 - Botão "Clique!";
 - Botão "Resetar!".



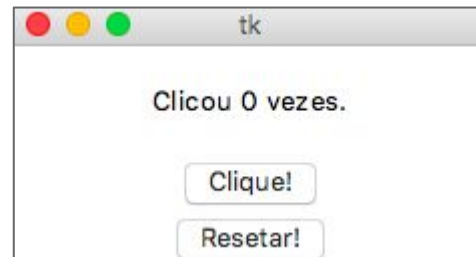
Widget

- Qualquer componente da nossa interface gráfica;
- No nosso exemplo nós temos 4 widgets:
 - Janela principal;
 - Campo do texto;
 - Botão "Clique!";
 - Botão "Resetar!".



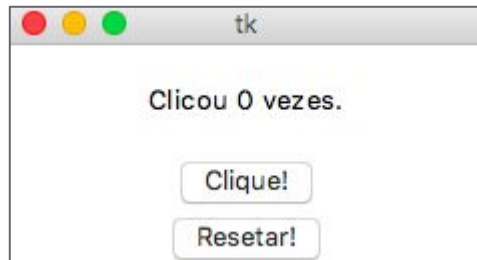
Event

- Interação do usuário com a interface;
- Eventos podem vir de:
 - Clique dos botões do mouse;
 - Pressionar uma ou várias tecla do teclado;
 - Clique em alguma área específica da interface:
 - Ex.: Clicar no botão "Clique!" ou "Resetar";



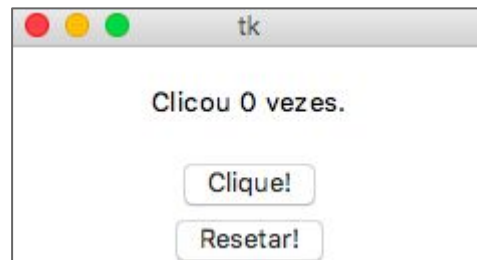
Event Handler

- Função/Método que são executados ao ocorrer um evento;
- No nosso exemplo:
 - Apertar o botão "Clique!" chama função que incrementa o contador;
 - Apertar o botão "Resetar!" chama função que zera o contador;



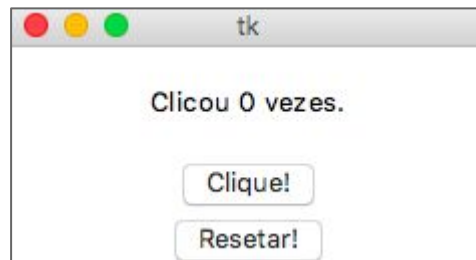
Binding

- Evento precisa saber qual seu event handling;
- Com isso, o evento sabe "o que fazer" quando ocorrer;
- Bind = Ligar/Associar um event a um event handling;

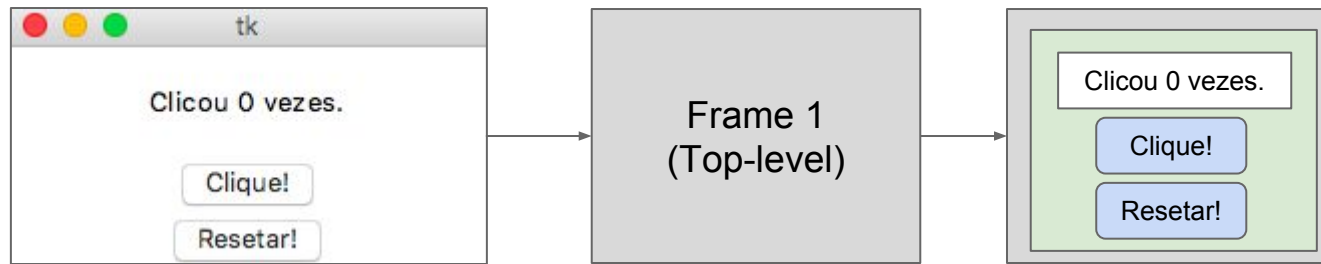
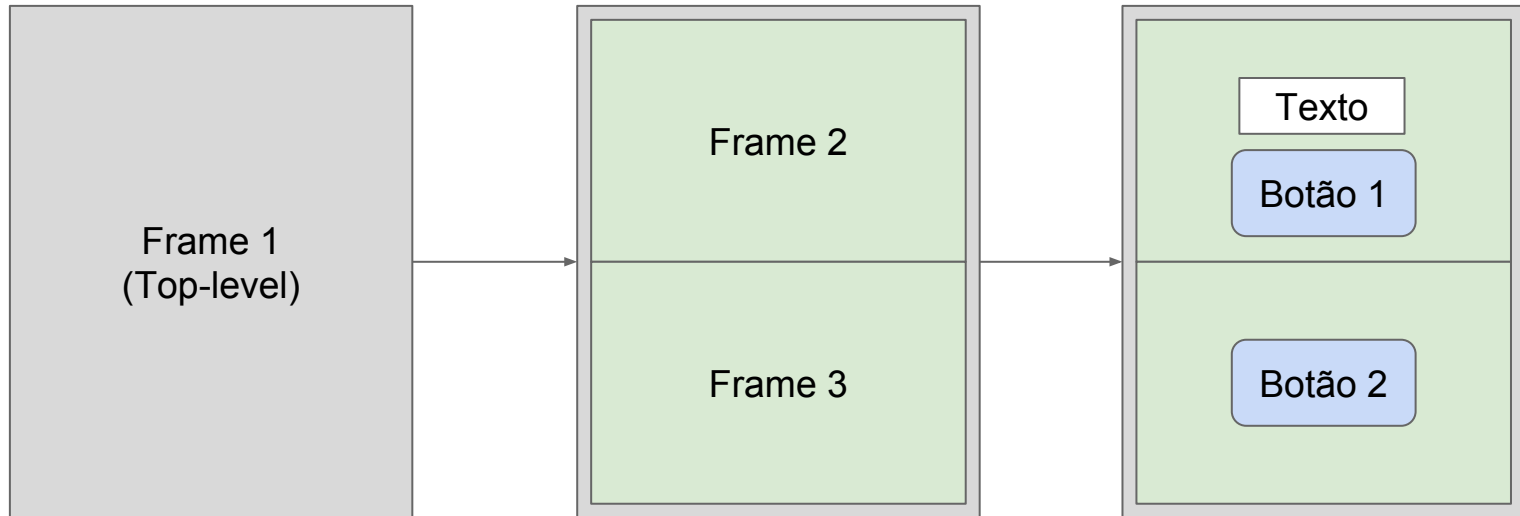


Container

- Elementos que dividem nossa interface ;
- Utilizado para dispor os widgets da melhor forma;
- Vamos utilizar o container da classe Frame;

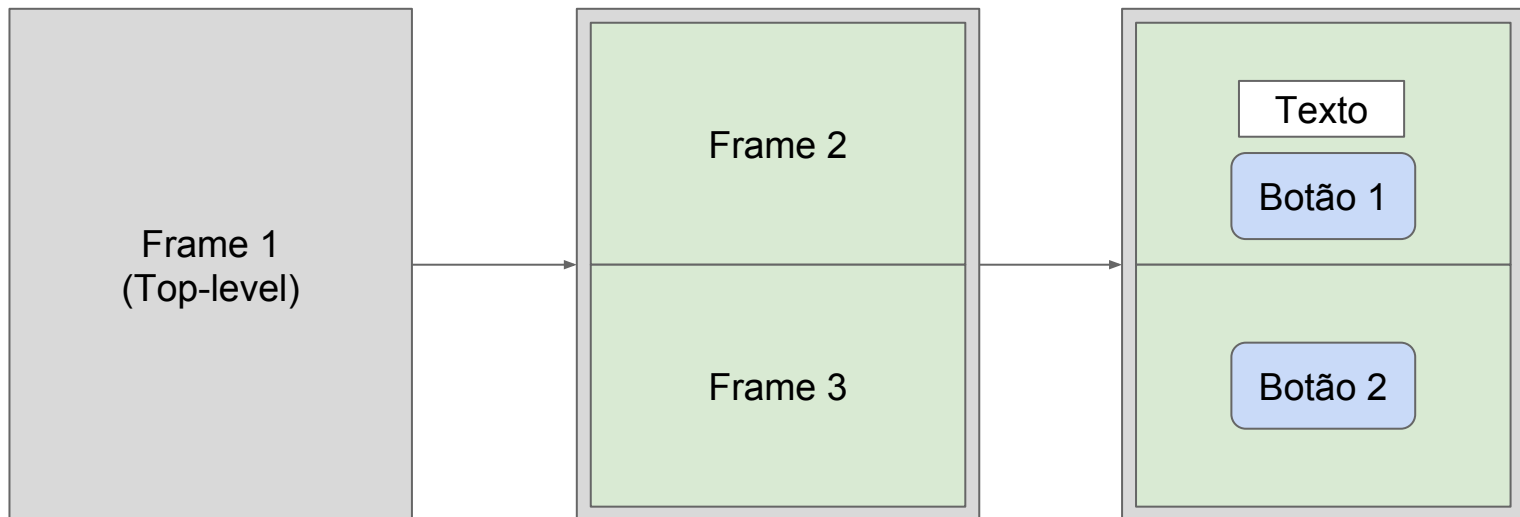


Container



Pack

- Método dos widgets;
- Indica em que posição o widget irá aparecer no container:
 - Sem parâmetro = TOP
 - Outras opções são LEFT, RIGHT, BOTTOM



Como programar?

- Primeiros passos:
 - Importar os módulos Tkinter
 - Instanciar um objeto Tk;
 - Executar método mainloop do objeto.

```
from Tkinter import *
```

```
janela = Tk()
```

```
janela.mainloop()
```

Como programar?

- Frames e widgets são representados por classes:
 - Janela/Frame - > Frame
 - Botão - > Button
 - Texto - > Label;
 - Lista -> List;
 - etc
- Parâmetros:
 - Objeto referente ao seu container (frame) pai;
 - Parâmetros específicos do widget:
 - Ex.: Button - > text, color, etc
 - E.: Label -> text, color, height, weight

Como programar?

- Criar a interface (basicamente):
 - Instanciar classes (widgets);
 - Setar seus atributos;
 - Criar métodos para eventos (ações dos widgets);
 - Usar o bind e pack;

Como programar?

- Vamos aos códigos de exemplo...

Widget - Parte 2

→ Vamos ver novos widgets para compor nosso trabalho.
São eles:

- ◆ Entry - Para input de dados;
- ◆ Listbox - Listar itens (tarefas ou projetos);
- ◆ ScrollBar - Adicionar opção de scroll na nossa Listbox;
- ◆ Treeview - Tabela/Lista com colunas
- ◆ tkMessageBox - Janelas/Pop-ups com avisos;
- ◆ Exceção no Tkinter - Verificar erros e tratá-los no Tkinter.

Widget - Entry

- Utilizado para fornecer ao usuário a possibilidade de inserir dados;
- Campo que captura a String digitada.



Widget - Entry

#Cria o widget Entry no frame1 | bd = largura da linha do box

```
e1 = Entry(self.frame1, bd =5)
```

```
e1.pack(side = TOP)
```

#Captura a String dentro do campo

```
tarefa = self.e1.get()
```

Widget - Listbox e Scrollbar

- Utilizado para criar uma lista de dados;
- Exibe de forma organizada uma lista de itens;
- Possibilita selecionar item para realizar alguma tarefa;
- Scrollbar utilizado para visualizar uma lista grande;
- Widgets trabalham junto para formar apenas um.



Widget - Listbox

#Cria o scrollbar no frame2

```
scrollbar = Scrollbar(frame2)
```

```
scrollbar.pack(side = RIGHT, fill = Y)
```

#Cria o widget Listbox no frame2 setando o scrollbar como comando

de scroll vertical

```
lb1 = Listbox(frame2, yscrollcommand = scrollbar.set)
```

```
lb1.pack(fill=BOTH, expand=1)
```

#Configura o scrollbar como scroll vertical da Listbox

```
scrollbar.config(command = lb1.yview)
```

Widget - Treeview

- Formato mais elaborado de lista;
- Funciona como uma tabela;
- Possibilita adicionar colunas;
- Possui "diretórios" que armazenam entradas;
- Necessário importar o pacote ttk (import ttk);

	Data de Criação	Data Limite
Tarefa 1	01/04/2016	05/10/2018
▼ Projeto 2		
Tarefa 1	20/05/2016	10/09/2018
▼ Projeto 3		
Tarefa 1	25/01/2015	23/10/2016
Tarefa 2		
Tarefa 3		
Tarefa 4		

Widget - Treeview

```
tree = ttk.Treeview(toplevel)

tree["columns"] = ("one","two")
tree.column("one", width = 100 )
tree.column("two", width = 100)
tree.heading("one", text = "Data de Criação")
tree.heading("two", text = "Data Limite")

tree.insert("", 0, text = "Tarefa 1", values = ("01/04/2016","05/10/2018"))

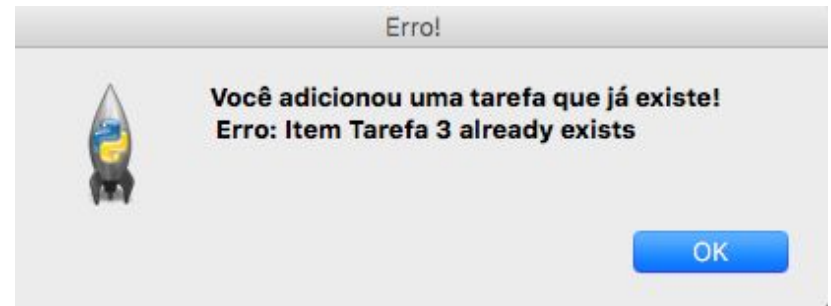
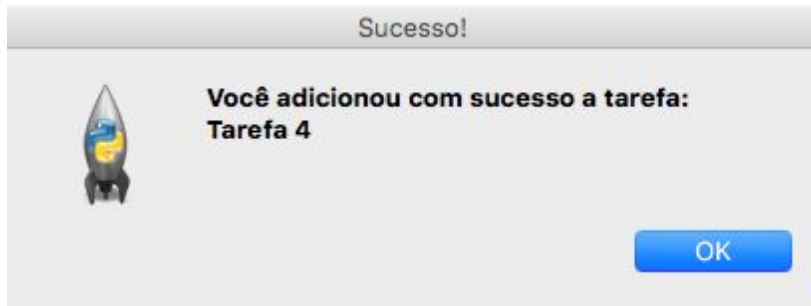
id2 = self.tree.insert("", 1, "dir2", text = "Projeto 2")
tree.insert(id2, "end", "dir 2", text = "Tarefa 1", values = ("20/05/2016","10/09/2018"))

tree.insert("", 3, "dir3", text = "Projeto 3")
tree.insert("dir3", 3, text = "Tarefa 1",values = ("25/01/2015"," 23/10/2016"))

tree.pack()
```

Widget - tkMessageBox

- Widget para exibir mensagens;
- Tkinter criar um pop-up exibindo a mensagem para o usuário;
- O padrão é um pop-up com:
 - Título da janela;
 - Mensagem;
 - Botão de confirmação.



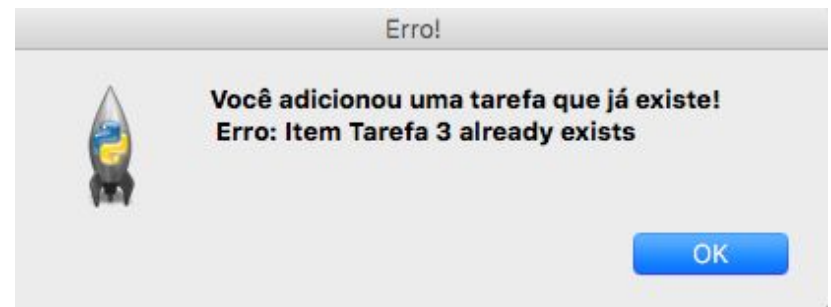
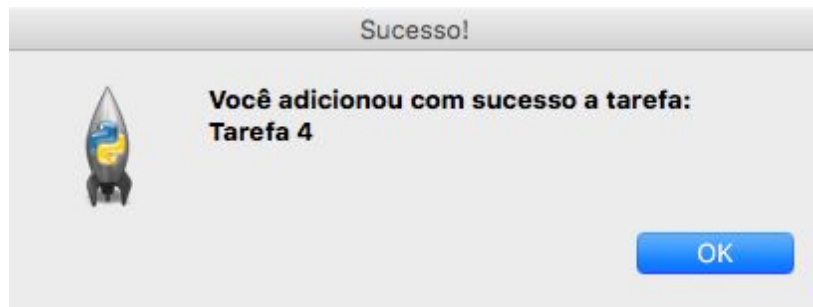
Widget - tkMessageBox

```
def showMsgBox(self, tarefa):
```

```
    tkMessageBox.showinfo("Sucesso!", "Você adicionou com sucesso a tarefa: " +  
tarefa)
```

```
def showErrorBox(self, err):
```

```
    tkMessageBox.showerror("Erro!", "Você adicionou uma tarefa que já existe! \n Erro:  
" + err.message)
```



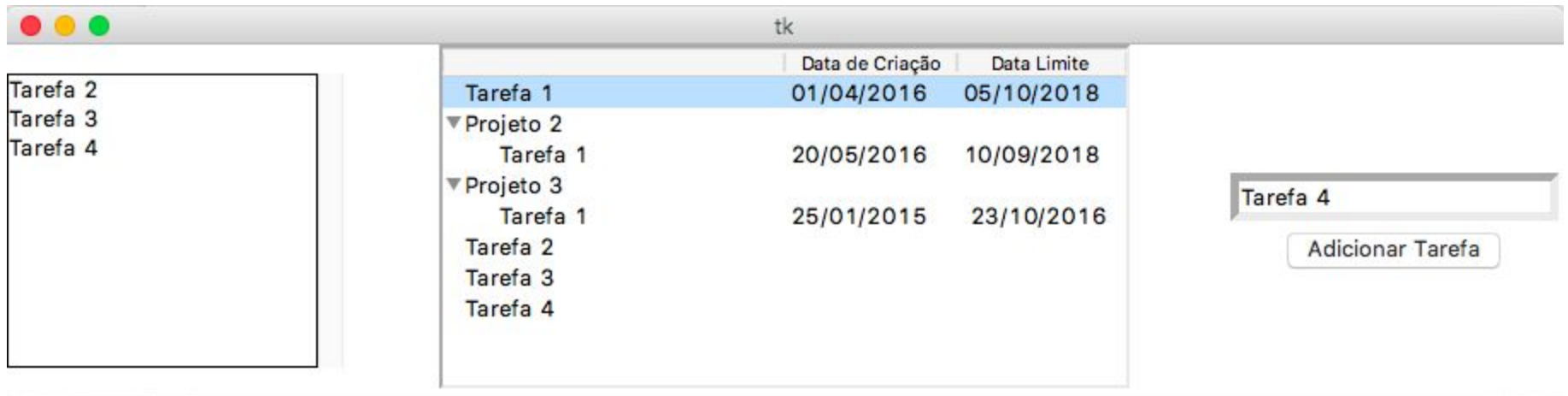
Widget - Exceção

- Utilizar os conceitos ensinados em aula;
- Utilizar cláusulas try/catch;
- Tratar possíveis erros do usuário;
- Erros de interface:
 - Capturar tkinter.TclError

Widget - Exceção

```
def addList(self, event):  
    try:  
        tarefa = self.e1.get()  
        self.lb1.insert(END, tarefa)  
        self.addTree()  
    except tkinter.TclError as err:  
        self.lb1.delete(END)  
        self.showErrorBox(err)  
    else:  
        self.showMsgBox(tarefa)
```

Juntando tudo...



The screenshot shows a Tkinter application window titled "tk". On the left, there is a list box containing "Tarefa 2", "Tarefa 3", and "Tarefa 4". The main area is a table with columns "Data de Criação" and "Data Limite". The table contains the following data:

	Data de Criação	Data Limite
Tarefa 1	01/04/2016	05/10/2018
▼ Projeto 2		
Tarefa 1	20/05/2016	10/09/2018
▼ Projeto 3		
Tarefa 1	25/01/2015	23/10/2016
Tarefa 2		
Tarefa 3		
Tarefa 4		

On the right side of the window, there is a text entry field containing "Tarefa 4" and a button labeled "Adicionar Tarefa".