

ALGORITMIA E ESTRUTURAS DE DADOS

A BIBLIOTECA TKINTER / CUSTOMTKINTER

TKINTER

CUSTOM TKINTER

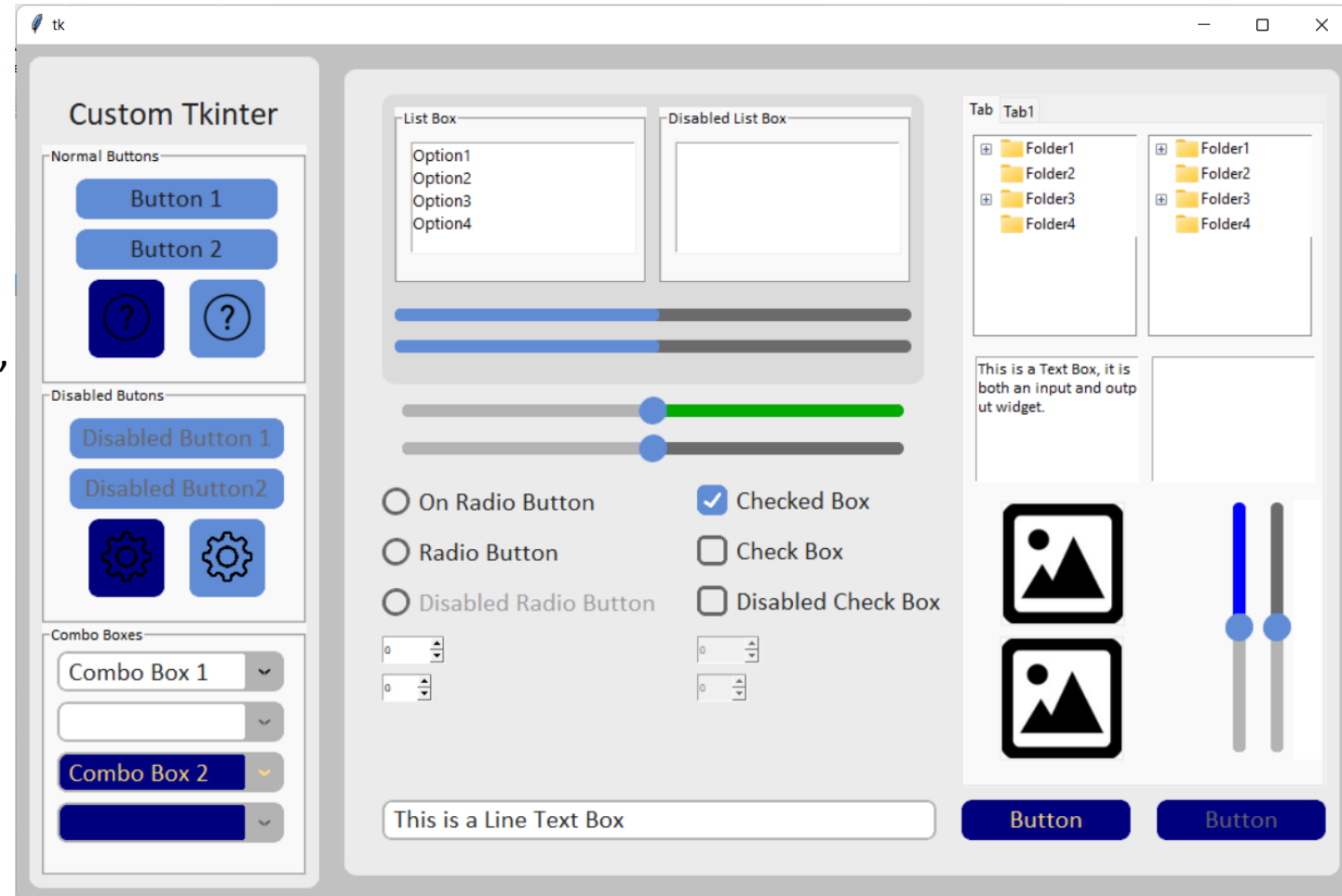
LICENCIATURA EM
TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB
#ESMAD #P.PORTO

- ❑ GUI - *Graphical User Interface*
- ❑ Bibliotecas para GUI em python
- ❑ A BIBLIOTECA TKINTER VS CUSTOMTKINTER
- ❑ Biblioteca CustomTkinter
 - ❑ Instalação
 - ❑ Windows
 - ❑ *Widgets*
 - ❑ *Containers*



❖ Graphical User Interface

- ❑ GUI - *Graphical User Interface* ou interface gráfica com o utilizador
- ❑ GUI são formadas por componentes visuais como janelas, menus, ícones, botões, seletores, caixas de texto, etc...
- ❑ Interação com interface gráfica através de teclado, rato ou *touchscreen*

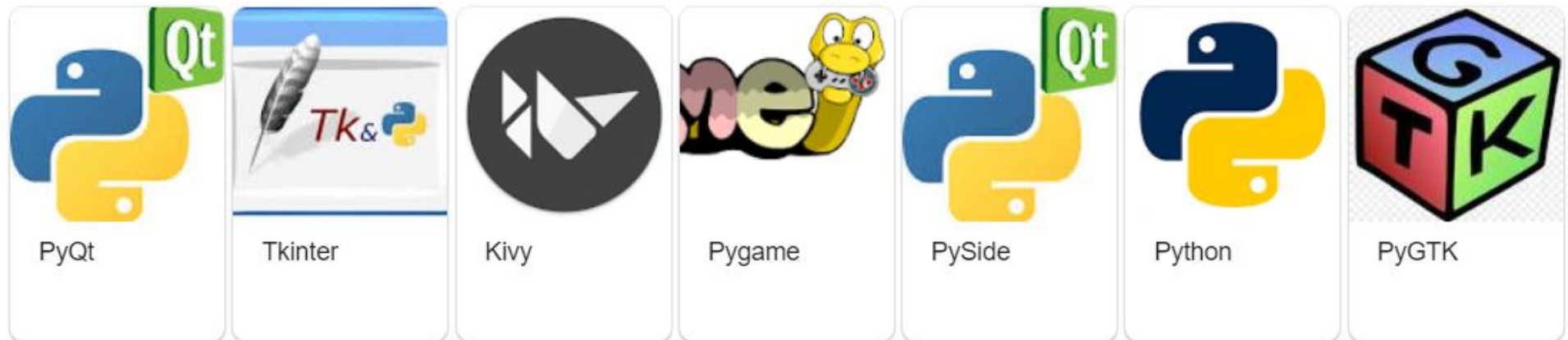


❖ Graphical User Interface

- ❑ Aplicações baseadas em GUI baseiam-se geralmente no paradigma de **programação guiada por eventos**
– *Event Driven Programming*
- ❑ Existe um ciclo que “observa” o ambiente ficando à espera da ocorrência de eventos (***event listening***)
- ❑ Sempre que ocorre um evento é despoletada uma função para gerir / responder a esse evento (***event handler***)
- ❑ Sempre que o utilizador interage com a interface gráfica é desencadeada uma ação (***callback***)

❖ Bibliotecas para GUI em python

- ❑ As interfaces gráfica são muito comuns na generalidade das aplicações
- ❑ Existem diversas bibliotecas / *frameworks* em python que suportam a criação de interfaces gráfica, tais como
 - ❑ PyGTK
 - ❑ WxPython
 - ❑ Kivy
 - ❑ PySide
 - ❑ PyQt
 - ❑ Tkinter / CustomTkinter
 - ❑



❖ Biblioteca Tkinter

❑ Vantagens

- ❑ Biblioteca Tkinter é distribuída com o pacote padrão do Python, pelo que não é necessário instalar nenhum *package* adicional
- ❑ É a biblioteca mais popular para construção de interfaces gráficas, em python
- ❑ Portabilidade: mesmo código funciona bem em diferentes SO como Linux, Unix, Windows e Mac
- ❑ Documentação: muita documentação, muitos tutoriais, vídeos, etc...
- ❑ Simplicidade na sua sintaxe

❖ Biblioteca CustomTkinter

- ☐ Biblioteca baseada na Tkinter
- ☐ Sintaxe semelhante à biblioteca Tkinter
- ☐ Maior capacidade de estilização e personalização
- ☐ Instalação:

```
Linha de comandos
C:\Users\mario>py -m pip install customtkinter
```

- ☐ Visual Studio Code:

```
1 import customtkinter
2
3
```

CustomTkinter

A modern and customizable python UI-library based on Tkinter



Documentation

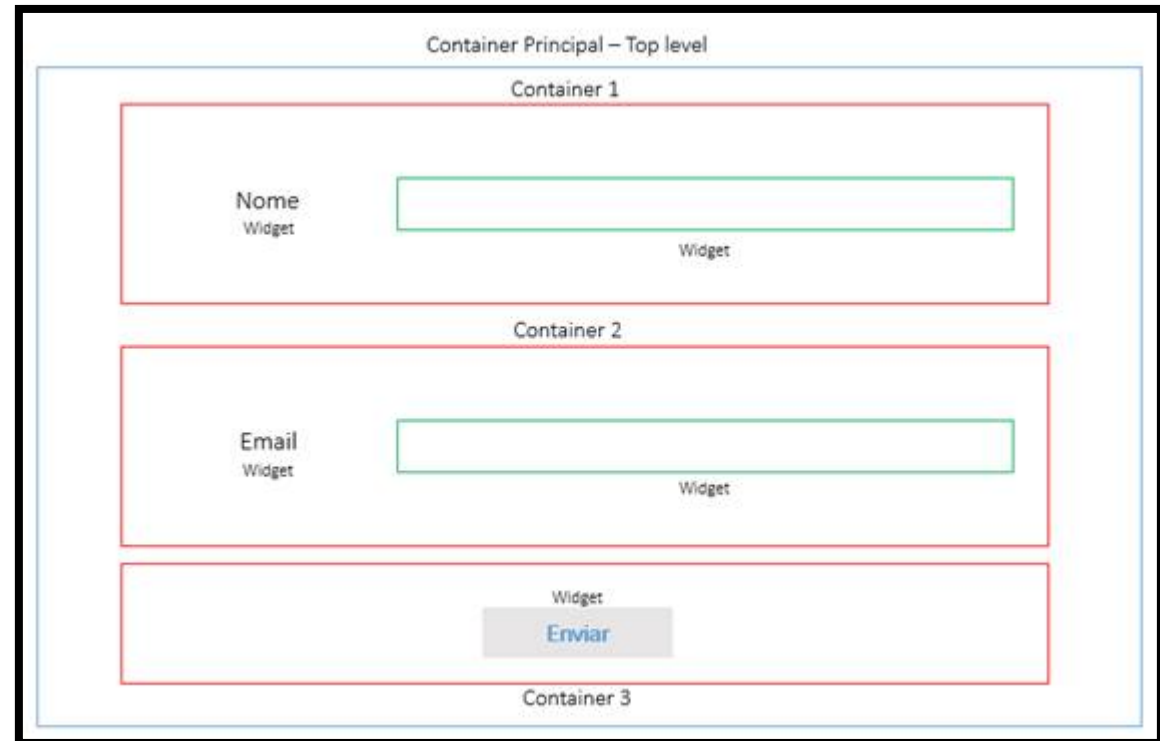
Tutorial

<https://customtkinter.tomschimansky.com/documentation/>

❖ Biblioteca CustomTkinter

❑ Conceitos base:

- ❑ **Containers:** objetos (componentes) onde podemos ancorar *widgets*. Todo o *widget* tem que estar dentro de um container
- ❑ **Widgets:** são componentes da interface gráfica: botões, labels, campos de texto, menus, comboboxs, progressbar, etc...



❖ Biblioteca CustomTkinter

☐ Containers & widgets: alguns exemplos

Containers

- Window
- Frame
- ScrollableFrame
- Canvas
- Tabview

widgets

- Button
- CheckBox
- ComboBox
- Label
- Entry
- Text
- RadioButton
- ProgressBar
- Textbox

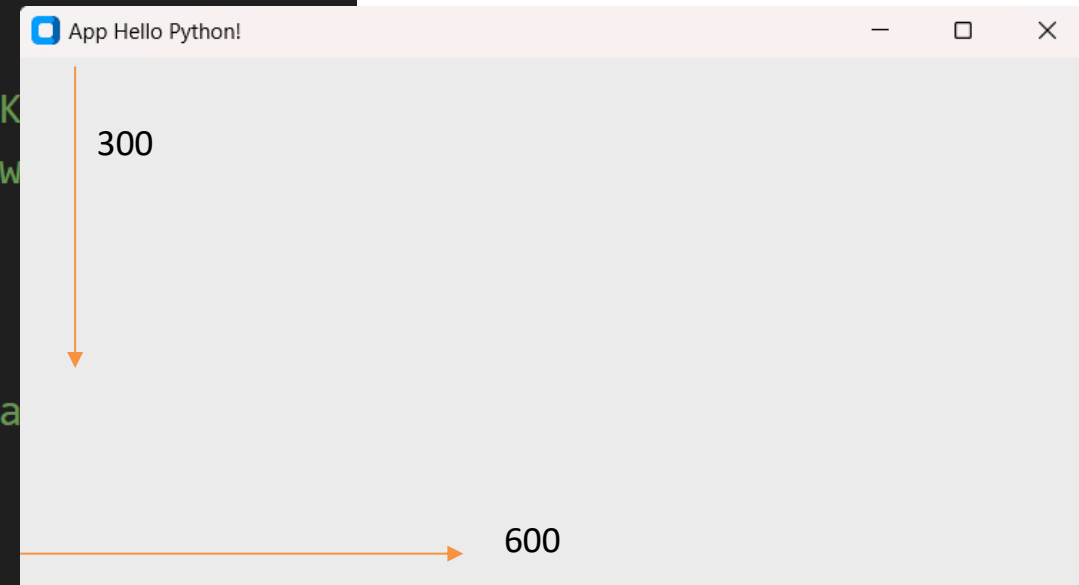
<https://customtkinter.tomschimansky.com/documentation/>

❖ Biblioteca CustomTkinter

❑ Window

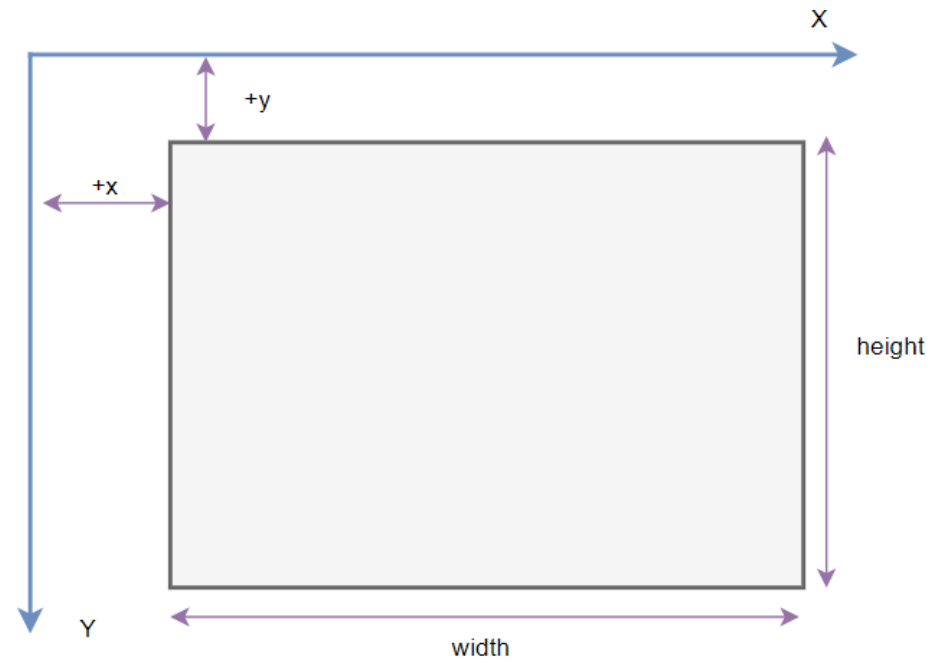
- ❑ A classe **CTk** é a base de qualquer programa CustomTkinter
- ❑ Cria a janela principal da aplicação. Deve haver apenas **uma** instância desta classe com uma única chamada do método **mainloop()**, que inicia a aplicação.
- ❑ As windows (janelas) adicionais são criadas usando a classe **CTkToplevel**

```
1  import customtkinter # Importa a biblioteca
2
3  app = customtkinter.CTk() # Invoca a classe CTk
4                               # cria a "main window"
5  app.geometry("600x300") # width, height
6  app.title("App Hello Python!")
7
8  # Aqui terá lugar a restante interface gráfica da
9
10 app.mainloop() # event Listening loop
11
```



❖ Biblioteca CustomTkinter

☐ Containers & widgets: alguns exemplos



`window.geometry('widthxheight±x±y')`

Posicionamento do canto superior esquerdo da window: X_{pos} , Y_{pos} , em pixels

❖ Biblioteca CustomTkinter

❑ Window

❑ Método **geometry** (width x height)
em pixéis

❑ Método **mainloop()**:
cria um *event listening loop*

```
1  import customtkinter # Importa a biblioteca
2
3  app = customtkinter.CTk() # Invoca a classe CTk,
4  app.title("App Hello Python!")
5
6  # Dimensões da interface da app
7  appwidth = 600
8  appHeight = 300
9
10 # Obter as dimensões do meu screen (em pixeis)
11 screenWidth = app.winfo_screenwidth()
12 screenHeight = app.winfo_screenheight()
13 # App centrada no screen, em função das suas dimensões# encontrar o
14 x = (screenWidth/2) - (appwidth/2)
15 y = (screenHeight/2) - (appHeight/2)
16 app.geometry(f'{appwidth}x{appHeight}+{int(x)}+{int(y)}')
17
18 # Aqui terá lugar a restante interface gráfica da app / window
19
20 app.mainloop() # event Listening loop
```

❖ Biblioteca CustomTkinter

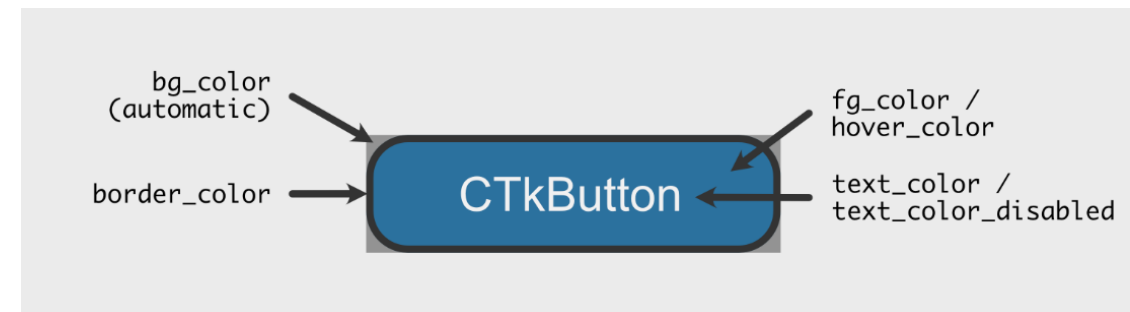
❑ Window: outros atributos

- ❑ minsize(width, height)
- ❑ maxsize (width, height)
- ❑ resizable(bool, bool)
- ❑ withdraw()
- ❑ configure (attribute = value....)

Desativa redimensionamento da window, largura e altura



```
1 app.configure(fg_color="gray")
2 app.resizable(False, False)
3
4 app.mainloop() # event Listening loop
5
```

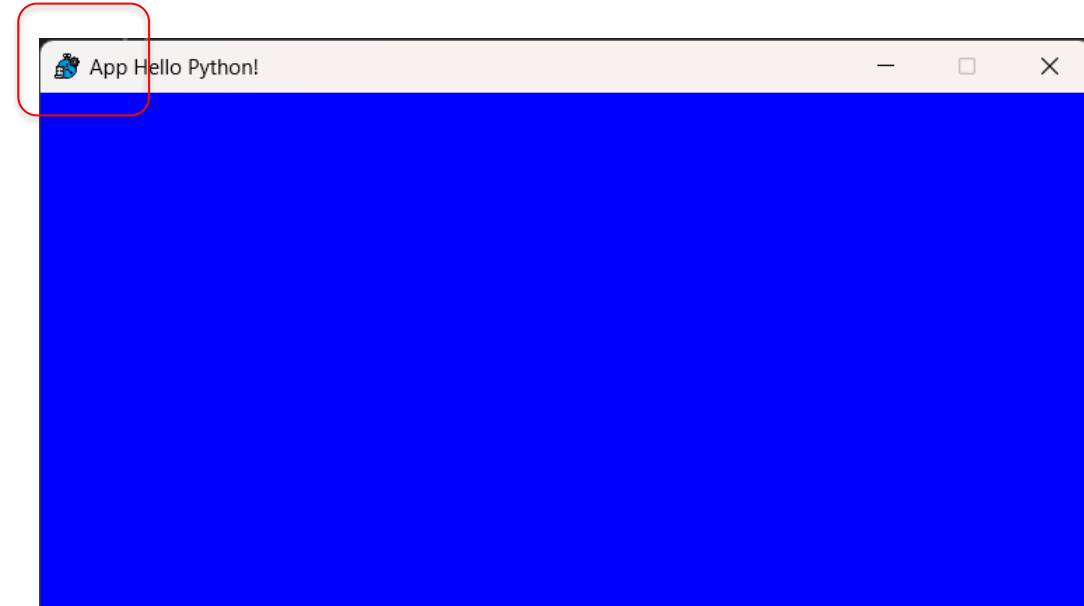


❖ Biblioteca CustomTkinter

❑ **Window** – método *iconbitmap()*



```
1 app.configure(fg_color = "blue")
2 app.resizable(False, False)
3
4 # ícone associado à app
5 app.iconbitmap(".\\ico\\logo.ico")
6 app.mainloop() # event Listening loop
7
8
```



Método *iconbitmap*:

- Icon com extensão .ico
- Converter png, jpg ou outro formato em .ico
- Colocar icon numa pasta do projeto
- Indicar path para a imagem de icon

<https://icon-icons.com/pt/>

❖ Biblioteca CustomTkinter

❑ Tkinter possui as seguintes classes para gerir o **posicionamento dos widgets** nos containers:

❑ Método **pack()**

Organiza os widgets em blocos antes de associa-los ao widget pai (window, p.e.)

❑ Método **place()**

Colocar os widgets numa determinada posição (coordenadas x e y, expressas em pixels) no widget pai (Windows, p.e.)

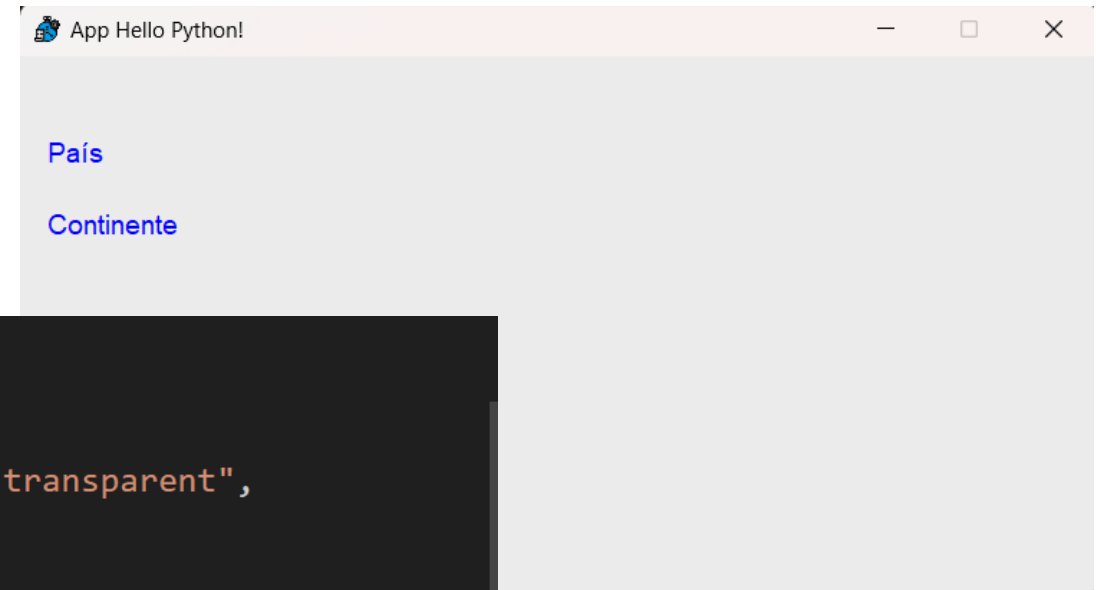
❑ Método **grid()**

Organiza os widgets em tabelas (linhas e colunas)

* O posicionamento dos widgets será objeto de análise numa apresentação específica

❖ Biblioteca CustomTkinter

☐ Label



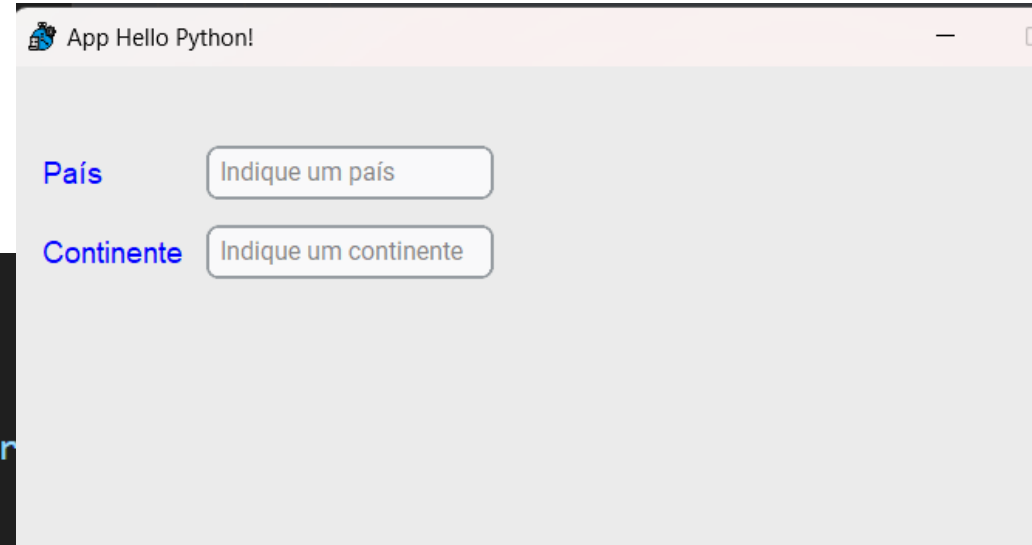
```
1  # INTERFACE -----
2  labelPais = customtkinter.CTkLabel(app, text="País", fg_color="transparent",
3                                     text_color="blue", font= ("Helvetica", 15))
4  labelPais.place(x=15, y=40)
5
6  labelContinente = customtkinter.CTkLabel(app, text="Continente", fg_color="transparent",
7                                             text_color="blue", font= ("Helvetica", 15))
8  labelContinente.place(x=15, y=80)
9
10
11 app.mainloop()    # event Listening loop
12
```

- **Cores padrão:** em hexadecimal ou paleta de cores: "white", "black", "red", "green", "blue", "cyan", "yellow", "magenta"

❖ Biblioteca CustomTkinter

❑ Entry

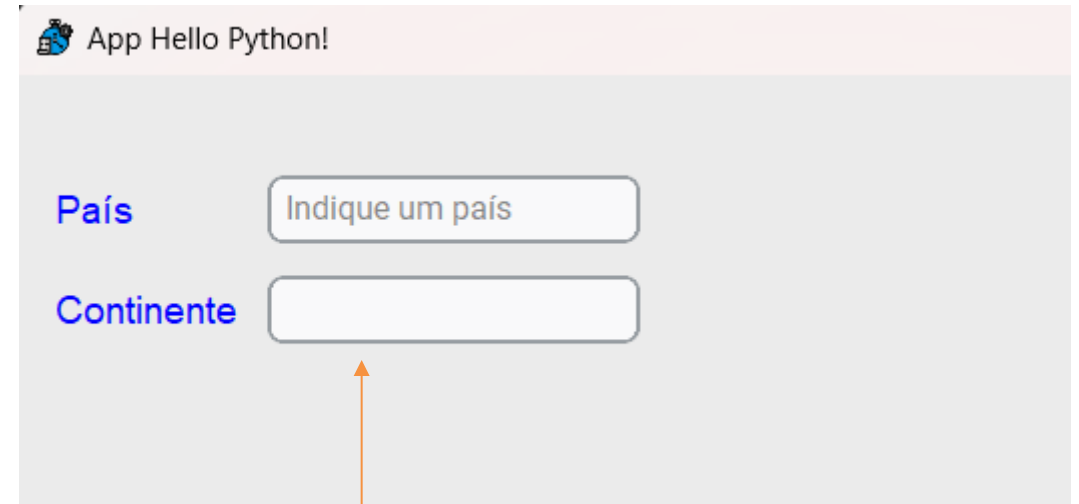
```
1  # INTERFACE -----
2  labelPais = customtkinter.CTkLabel(app, text="País", fg_color="transparent",
3                                     text_color="blue", font= ("Helvetica", 15))
4  labelPais.place(x=15, y=40)
5
6  labelContinente = customtkinter.CTkLabel(app, text="Continente", fg_color="transparent",
7                                             text_color="blue", font= ("Helvetica", 15))
8  labelContinente.place(x=15, y=80)
9
10 # Caixas de texto
11 entryPais = customtkinter.CTkEntry(app, placeholder_text="Indique um país",
12                                    width=150)
13 entryPais.place(x=100, y= 40)
14
15 entryContinente = customtkinter.CTkEntry(app, placeholder_text="Indique um continente",
16                                           width=150)
17 entryContinente.place(x=100, y= 80)
18
```



❖ Biblioteca CustomTkinter

❑ Entry

- ❑ width
- ❑ height
- ❑ textvariable
- ❑ fg_color
- ❑ bg_color
- ❑ font
- ❑ state ("normal", "disabled")

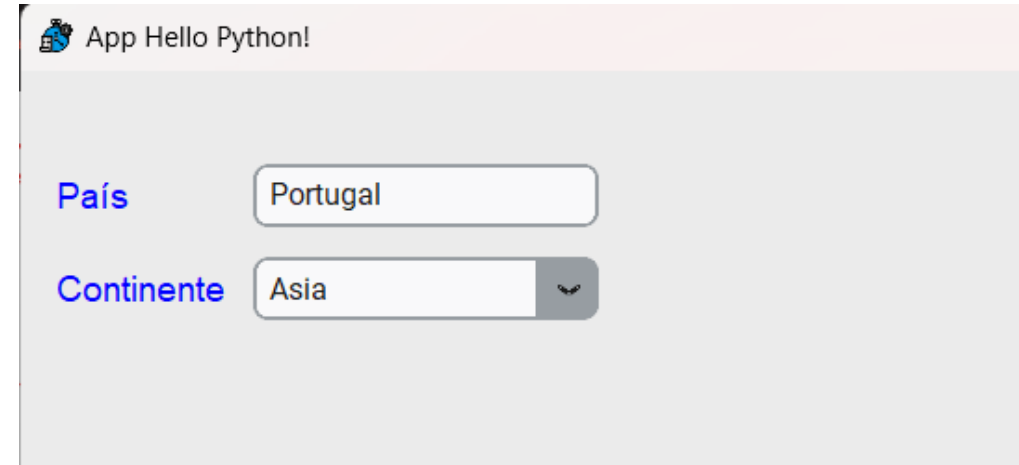


```
1 entryContinente = customtkinter.CTkEntry(app, placeholder_text="Indique um continente",
2                                     width=150, state = "disabled")
3 entryContinente.place(x=100, y= 80)
```

❖ Biblioteca CustomTkinter

❑ Entry

- ❑ show
- ❑ textvariable (variável associada a *textvariable* indica o conteúdo da Entry)



Valor por defeito

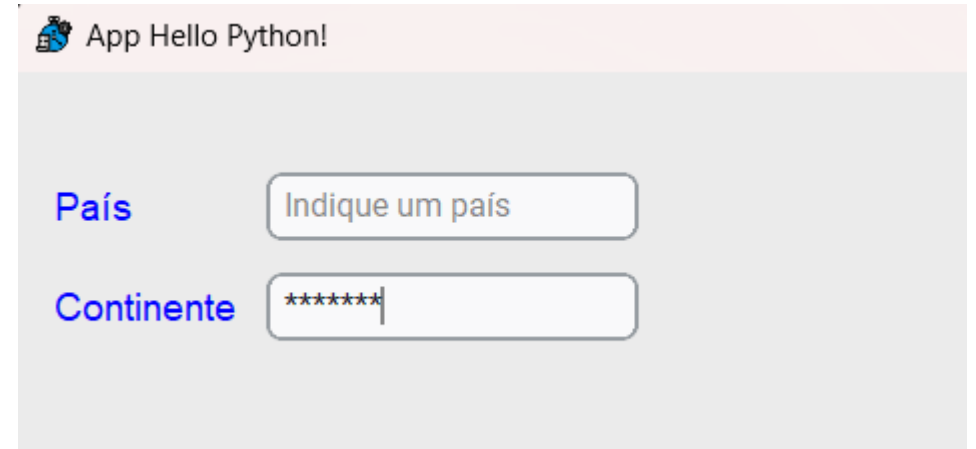
Conteúdo da Entry

```
1  # Caixas de texto
2  strPais = customtkinter.StringVar()
3  strPais.set("Portugal")
4  entryPais = customtkinter.CTkEntry(app, placeholder_text="Indique um país",
5                                     textvariable= strPais, width=150)
6  entryPais.place(x=100, y= 40)
7
8  print(strPais.get())
```

❖ Biblioteca CustomTkinter

❑ Entry

- ❑ show
- ❑ textvariable (variável associada a *textvariable* indica o conteúdo da Entry)

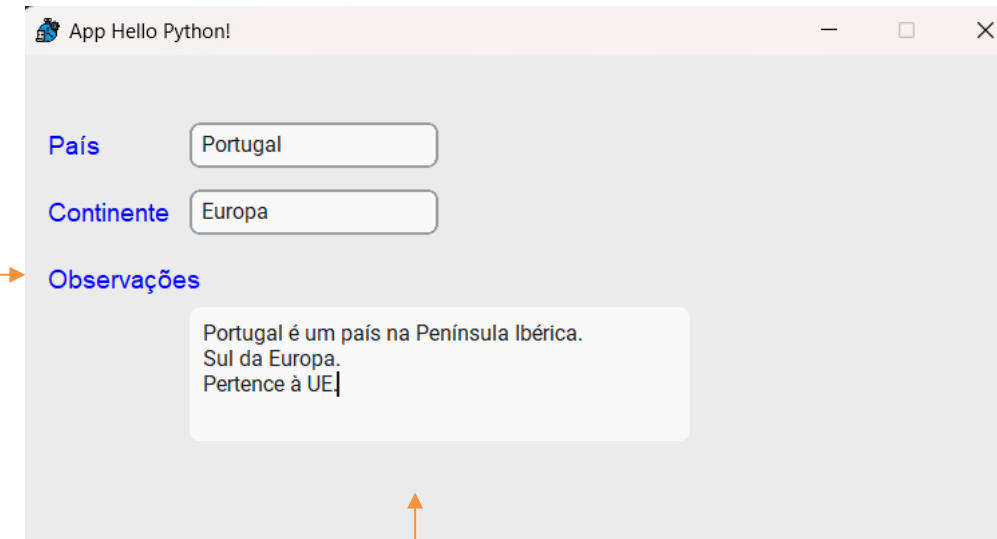


```
1 entryContinente = customtkinter.CTkEntry(app, placeholder_text="Indique um continente",  
2                                     width=150, show = "*")  
3 entryContinente.place(x=100, y= 80)
```

❖ Biblioteca CustomTkinter

❑ Textbox (mais do que uma linha de texto)

- ❑ width
- ❑ height
- ❑ font
- ❑ border_width
- ❑ activate_scrollbars (True/False)
- ❑ state ("normal", "disabled")



```
1 labelObs = customtkinter.CTkLabel(app, text="Observações", fg_color="transparent",  
2     text_color="blue", font= ("Helvetica", 15))  
3 labelObs.place(x=15, y=120)
```

```
1 # TextBox  
2 TxtObs= customtkinter.CTkTextbox(app, width=300, height= 80, border_color="gray")  
3 TxtObs.place (x= 100, y= 150)
```

❖ Biblioteca CustomTkinter

❑ Textbox (mais do que uma linha de texto)

- ❑ Pode ser vista como uma lista de linhas de texto
- ❑ métodos: *insert*, *delete*, *get*

```
1 lista = ["Africa", "Asia", "América", "Europa", "Oceania"]
2
3 textContinente = customtkinter.CTkTextbox(app, width=150, height=100)
4 for i in range(len(lista)):
5     textContinente.insert("end", lista[i]+'\\n')
6 textContinente.place(x=100, y= 80)
```



❖ Biblioteca CustomTkinter

❑ Textbox (mais do que uma linha de texto)

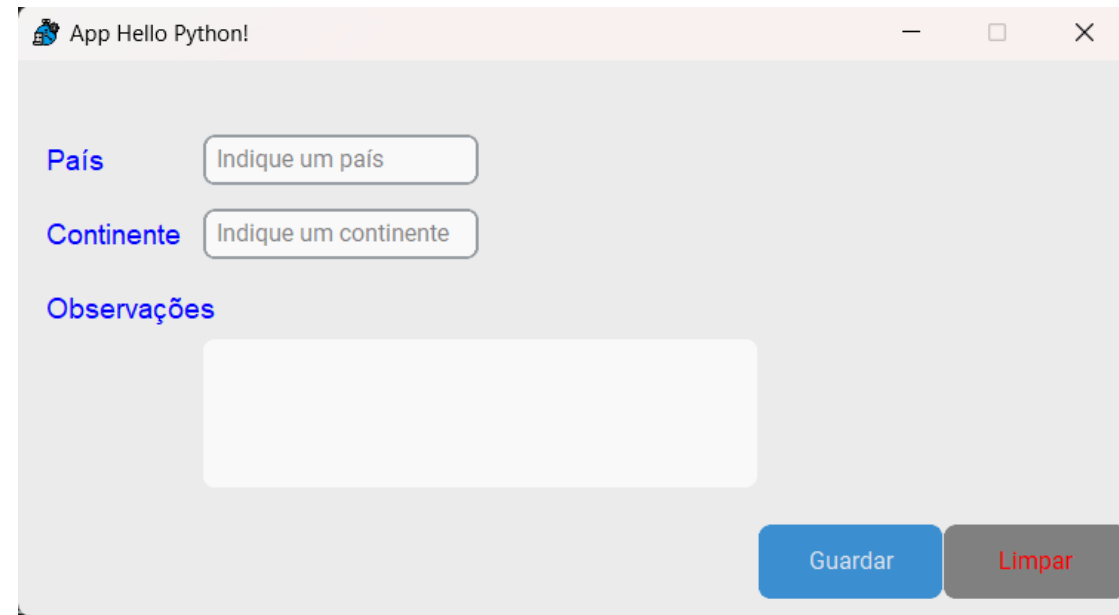
- ❑ Pode ser vista como uma lista de linhas de texto
- ❑ métodos: *insert*, *delete*, *get*

```
1  lista = ["Africa", "Asia", "América", "Europa", "Oceania"]
2
3  textContinente = customtkinter.CTkTextbox(app, width=150, height=100)
4  for i in range(len(lista)):
5      textContinente.insert("end", lista[i]+'\\n')
6  textContinente.place(x=100, y= 80)
7
8  textContinente.get("0.0", "end")      # Obtém todo o texto da TextBox
9  print(textContinente.get("1.0", "1.end"))    # Obtém 1ª linha de texto
10 textContinente.delete("0.0", "end")    # apaga todo o texto da TextBox
11
```

❖ Biblioteca CustomTkinter

❑ Button

- ❑ width
- ❑ height
- ❑ text_color
- ❑ fg_color
- ❑ text
- ❑ font
- ❑ state



```
1  # Button
2  btnGuardar = customtkinter.CTkButton(app, text="Guardar", command="",
3                                     width=100, height=40)
4  btnGuardar.place(x=400, y=250)
5
6  btnLimpar = customtkinter.CTkButton(app, text="Limpar", command="",
7                                     width=100, height=40, fg_color="gray", text_color="red")
8  btnLimpar.place(x=500, y=250)
```


❖ Biblioteca CustomTkinter

❑ Button

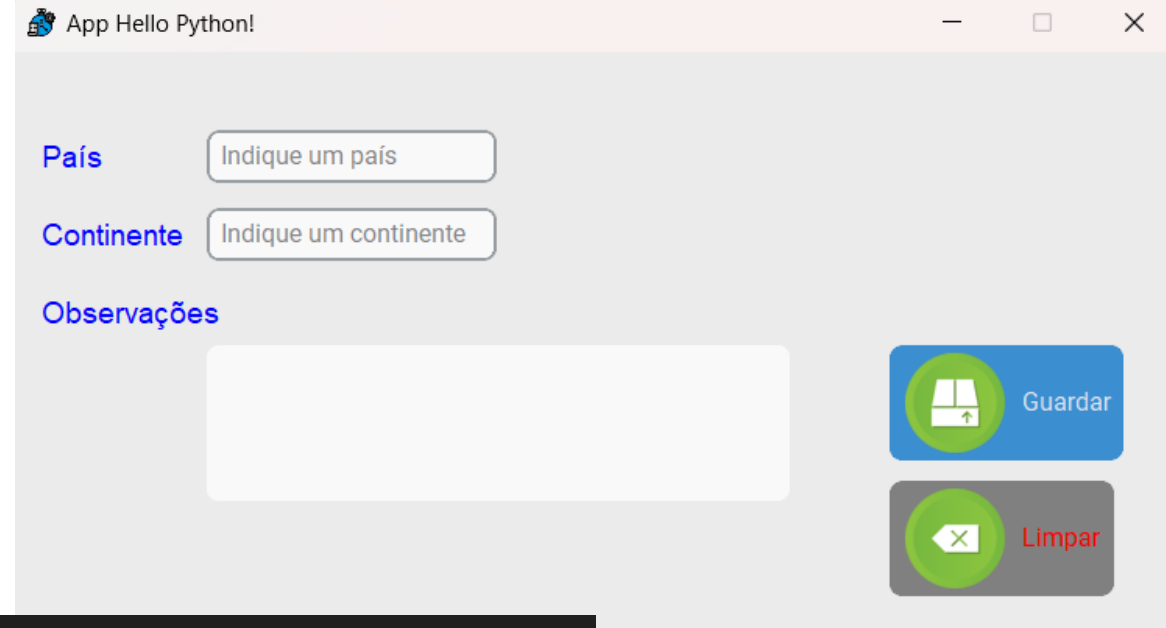
❑ image (.png, jpeg, etc.)



```
1 import customtkinter # Importa a biblioteca
2 from PIL import ImageTk
3
```



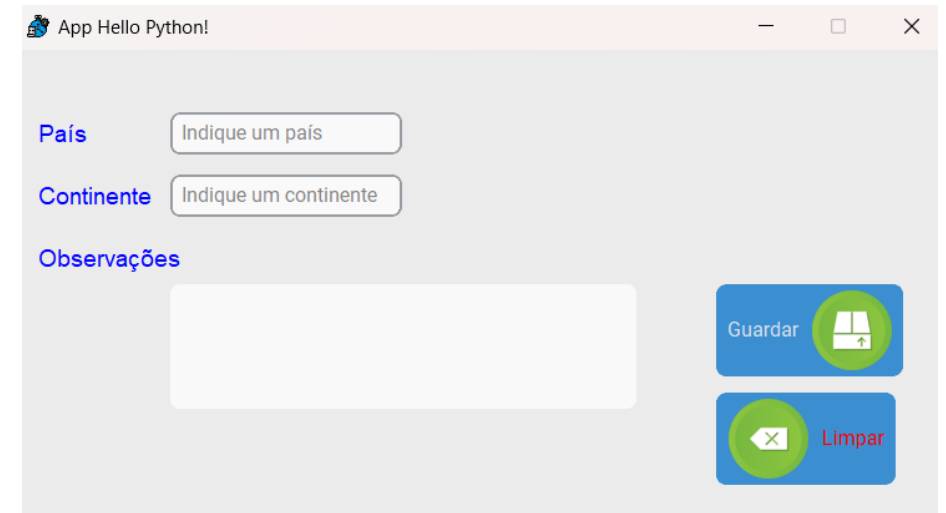
```
1 # Buttons
2 img1Button = ImageTk.PhotoImage(file = ".\\ico\\guardar.png")
3 btnGuardar = customtkinter.CTkButton(app, image= img1Button, text = "Guardar", command="",
4                                     width=100, height=40)
5 btnGuardar.place(x=450, y=150)
6
7 img2Button = ImageTk.PhotoImage(file = ".\\ico\\limpar.png")
8 btnLimpar = customtkinter.CTkButton(app, image = img2Button, text="Limpar", command="",
9                                     width=100, height=40, fg_color="gray", text_color="red")
10 btnLimpar.place(x=450, y=220)
```



❖ Biblioteca CustomTkinter

❑ Button

- ❑ image (.png, jpeg, etc.)
- ❑ compound: "top", "left", "bottom", "right"



```
1  # Buttons
2  img1Button = ImageTk.PhotoImage(file = ".\\ico\\guardar.png")
3  btnGuardar = customtkinter.CTkButton(app, image= img1Button, text = "Guardar", command="",
4      width=100, height=40, compound="right")
5  btnGuardar.place(x=450, y=150)
6
7  img2Button = ImageTk.PhotoImage(file = ".\\ico\\limpar.png")
8  btnLimpar = customtkinter.CTkButton(app, image = img2Button, text="Limpar", command="",
9      width=100, height=40, compound="left", text_color="red")
10 btnLimpar.place(x=450, y=220)
11
```

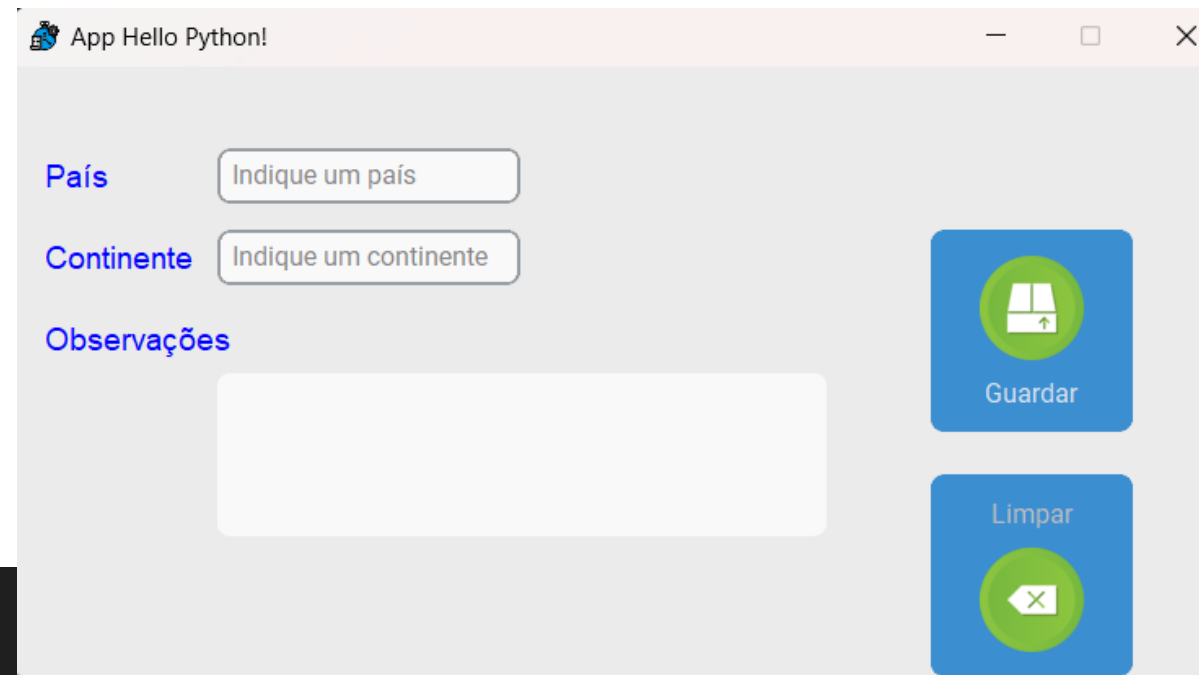
❖ Biblioteca CustomTkinter

❑ Button

- ❑ image (.png, jpeg, etc.)
- ❑ compound: "top", "left", "bottom", "right"



```
1  # Buttons
2  img1Button = ImageTk.PhotoImage(file = ".\\ico\\guardar.png")
3  btnGuardar = customtkinter.CTkButton(app, image= img1Button, text = "Guardar", command="",
4                                     width=100, height=100, compound="top")
5  btnGuardar.place(x=450, y=80)
6
7
8  img2Button = ImageTk.PhotoImage(file = ".\\ico\\limpar.png")
9  btnLimpar = customtkinter.CTkButton(app, image = img2Button, text="Limpar", command="",
10                                     width=100, height=100, compound="bottom", state = "disabled", text_color="red")
11 btnLimpar.place(x=450, y=200)
```



❖ Biblioteca CustomTkinter

❑ ComboBox

- ❑ values = *lista*
- ❑ width, height
- ❑ font, textcolor
- ❑ state
- ❑ variable

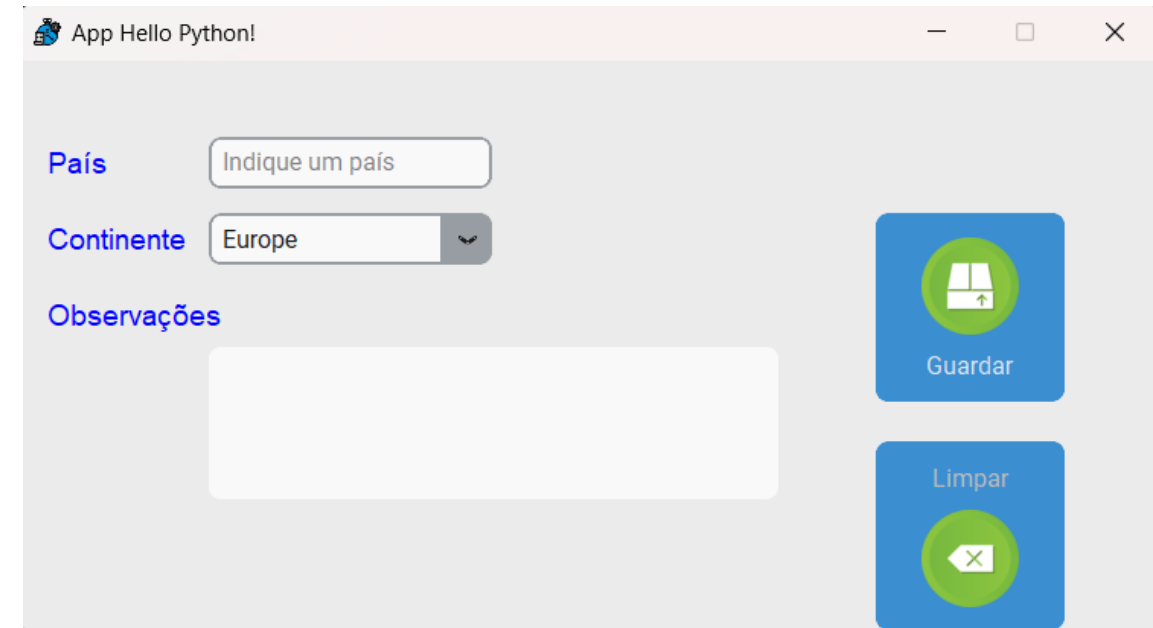


```
1 #ComboBox
2 lista = ["Africa", "Asia", "América", "Europa", "Oceania"]
3 combContinente = customtkinter.CTkComboBox(app, values=lista, width=150, command="")
4 combContinente.place(x=100, y= 80)
```

❖ Biblioteca CustomTkinter

❑ ComboBox

❑ Método set()



```
1 #ComboBox
2 lista = ["Africa", "Asia", "América","Europa", "Oceania"]
3 combContinente = customtkinter.CTkComboBox(app, values=lista, width=150, command="")
4 combContinente.place(x=100, y= 80)
5 combContinente.set("Europe") # Valor dor defeito
```

❖ Biblioteca CustomTkinter

❑ ComboBox

- ❑ variable
- ❑ Método set()
- ❑ Método get()

```
1 #ComboBox
2 strContinente= customtkinter.StringVar()
3
4 lista = ["Africa", "Asia", "América","Europa", "Oceania"]
5 strContinente = customtkinter.StringVar()
6 strContinente.set("Asia")
7 combContinente = customtkinter.CTkComboBox(app, variable = strContinente, values=lista,
8                                           width=150, command="")
9 combContinente.place(x=100, y= 80)
```

Atribuir valor por defeito



```
1
2 print(strContinente.get())
3
```

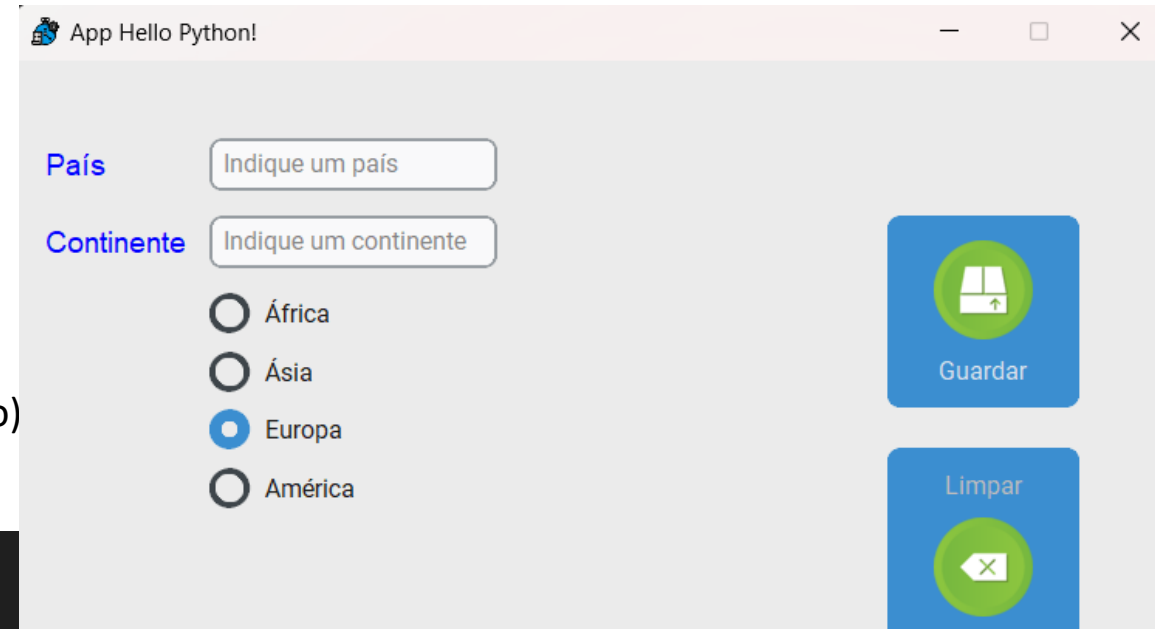
Valor selecionado na ComboBox

❖ Biblioteca CustomTkinter

❑ **RadioButton** - Apenas 1 opção pode estar seleccionada!

- ❑ width, height, fg_color, texto_color
- ❑ text, state
- ❑ value (valor devolvido quando clico numa opção: string ou int)
- ❑ variable (variável que controla o estado dos buttons – botão ativo)

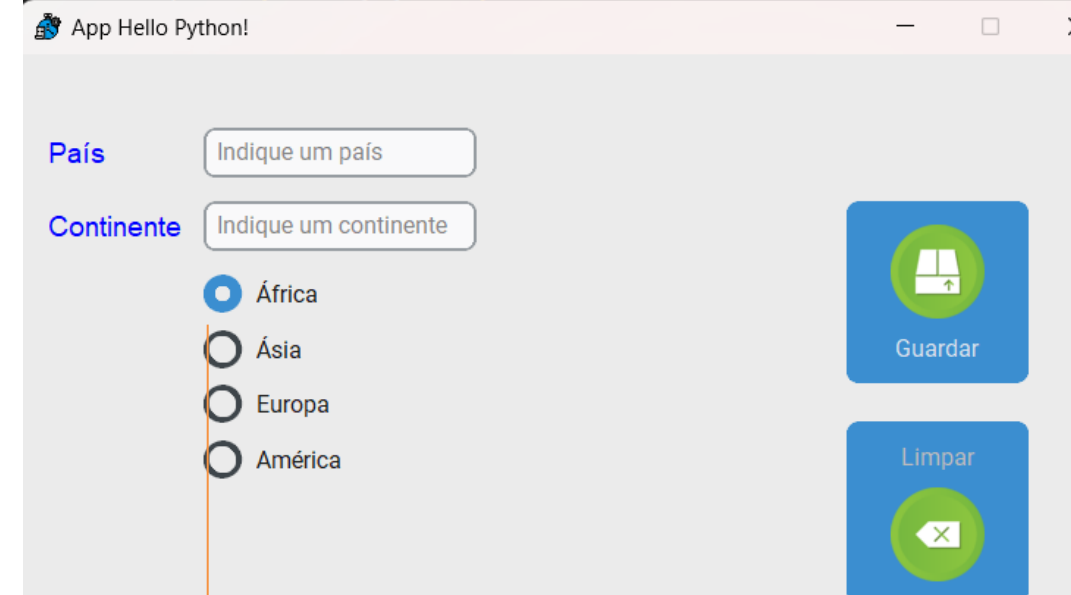
```
1 # RadioButtons
2 radioVariable = customtkinter.StringVar(value="Europa")
3 radiobutton1 = customtkinter.CTkRadioButton(app, text="África",
4                                             variable= radioVariable, value="África")
5 radiobutton1.place(x=100, y=120)
6 radiobutton2 = customtkinter.CTkRadioButton(app, text="Ásia",
7                                             variable= radioVariable, value="Ásia")
8 radiobutton2.place(x=100, y=150)
9 radiobutton3 = customtkinter.CTkRadioButton(app, text="Europa",
10                                             variable= radioVariable, value="Europa")
11 radiobutton3.place(x=100, y=180)
12 radiobutton4 = customtkinter.CTkRadioButton(app, text="América",
13                                             variable= radioVariable, value="América")
14 radiobutton4.place(x=100, y=210)
15
```



❖ Biblioteca CustomTkinter

❑ **RadioButton** - Apenas 1 opção pode estar seleccionada!

- ❑ width, height, fg_color, texto_color
- ❑ text, state
- ❑ value (valor devolvido quando clico numa opção: string ou int)
- ❑ variable (variável que controla o estado dos buttons – botão ativo)
É a mesma variável para **TODOS** os RadioButtons



↓ método get()

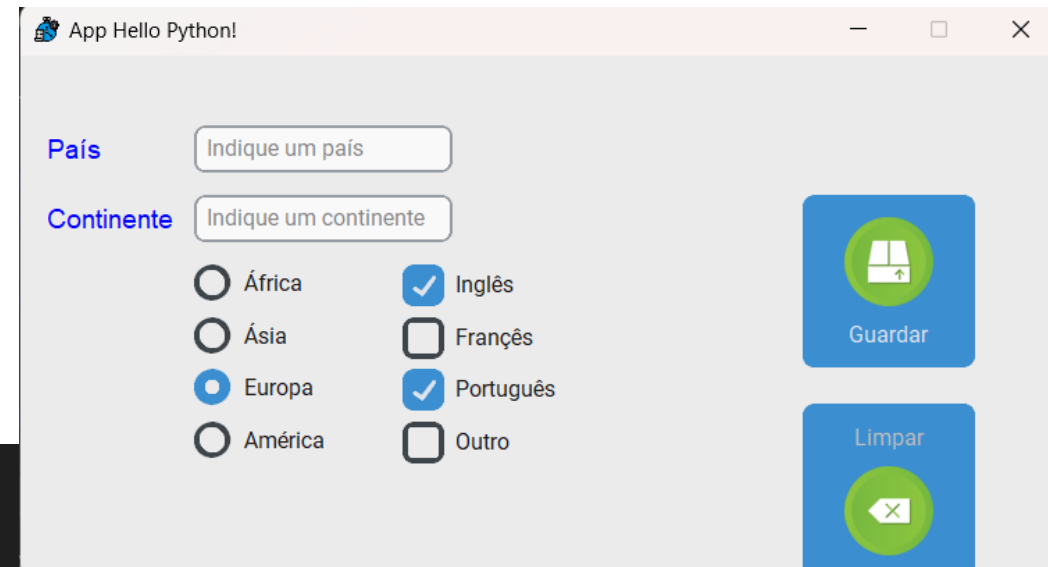
```
1 # RadioButtons
2 radioVariable = customtkinter.StringVar(value="Europa")
3 radiobutton1 = customtkinter.CTkRadioButton(app, text="África",
4                                             variable= radioVariable, value="África")
5 radiobutton1.place(x=100, y=120)
6 radiobutton2 = customtkinter.CTkRadioButton(app, text="Ásia",
7                                             variable= radioVariable, value="Ásia")
8 radiobutton2.place(x=100, y=150)
9 radiobutton3 = customtkinter.CTkRadioButton(app, text="Europa",
10                                            variable= radioVariable, value="Europa")
11 radiobutton3.place(x=100, y=180)
12 radiobutton4 = customtkinter.CTkRadioButton(app, text="América",
13                                            variable= radioVariable, value="América")
14 radiobutton4.place(x=100, y=210)
15
```

```
1
2 continenteSlt = radioVariable.get()
```


❖ Biblioteca CustomTkinter

- ❑ **CheckBox** – permite seleccionar mais do que uma opção
 - ❑ width, height, fg_color, texto_color, font
 - ❑ text, state
 - ❑ onvalue ou offvalue (“on”, “off”)
 - ❑ variable (variável que verifica o estado de cada CheckBox)

```
1 # CheckBox - Idiomas
2 checkVar1 = customtkinter.StringVar(value="on")
3 checkVar2 = customtkinter.StringVar(value="off")
4 checkVar3 = customtkinter.StringVar(value="on")
5 checkVar4 = customtkinter.StringVar(value="off")
6
7 checkboxEN = customtkinter.CTkCheckBox(app, text="Inglês", variable=checkVar1, onvalue="on", offvalue="off")
8 checkboxFR = customtkinter.CTkCheckBox(app, text="Françês", variable=checkVar2, onvalue="on", offvalue="off")
9 checkboxPT = customtkinter.CTkCheckBox(app, text="Português", variable=checkVar3, onvalue="on", offvalue="off")
10 checkboxOT = customtkinter.CTkCheckBox(app, text="Outro", variable=checkVar4, onvalue="on", offvalue="off")
11
12 checkboxEN.place(x=220, y= 120)
13 checkboxFR.place(x=220, y= 150)
14 checkboxPT.place(x=220, y= 180)
15 checkboxOT.place(x=220, y= 210)
```



❖ Biblioteca CustomTkinter

❑ Button

❑ Método get()

```
1  if checkVar1.get() == "on":
2      checkboxEN.configure( fg_color= "red")
3  else:
4      checkboxEN.configure( fg_color= "gray")
5
6  if checkVar2.get() == "on":
7      checkboxFR.configure( fg_color= "red")
8  else:
9      checkboxFR.configure( fg_color= "gray")
10
11 if checkVar3.get() == "on":
12     checkboxPT.configure( fg_color= "red")
13 else:
14     checkboxPT.configure( fg_color= "gray")
15
```

App Hello Python!

País

Continente


☐ África ☒ Inglês

☐ Ásia ☐ Francês

☒ Europa ☒ Português

☐ América ☐ Outro

 Guardar

 Limpar

❖ Biblioteca CustomTkinter

❑ Atividade Prática

Desenvolver apenas a interface gráfica:

- ❑ App 600x300
- ❑ Interface centrado no *screen*
- ❑ Janela não é *resizable*
- ❑ Título: Países do Mundo!
- ❑ Labels com font (Helvetica, 14)
- ❑ ComboBox com continentes
- ❑ Entry (países) e Combobox (continentes) com width de 150 px
- ❑ Idiomas: por defeito selecionado *Português*
- ❑ Hemisfério: por defeito, selecionado *Norte*
- ❑ Botão Limpar: state *disabled*

