

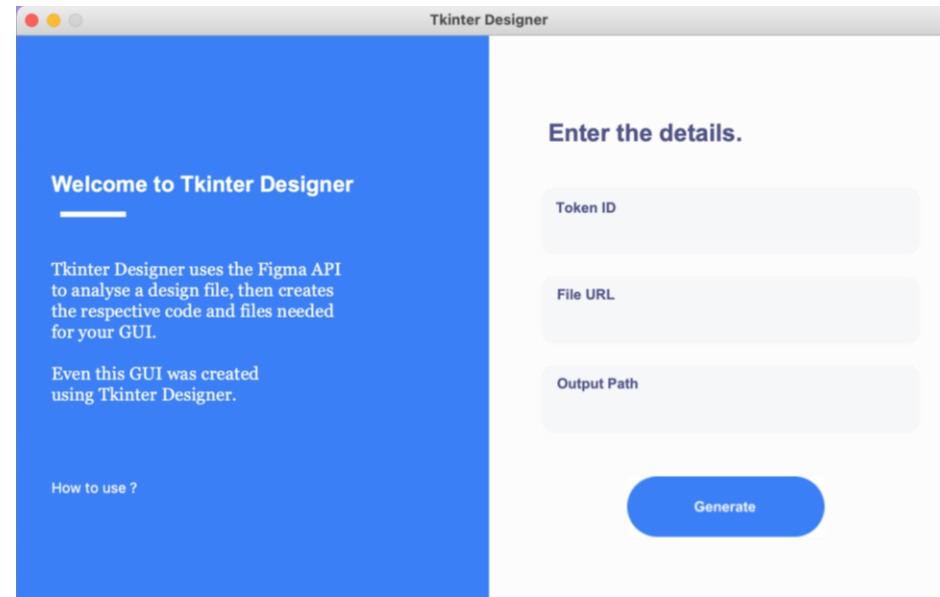
## ALGORITMIA E ESTRUTURAS DE DADOS

### MÓDULO V TKINTER

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

## A BIBLIOTECA TKINTER

- ❑ GUI - *Graphical User Interface*
- ❑ Bibliotecas para GUI em python
- ❑ Biblioteca tkinter
  - ❑ *Widgets*
  - ❑ *Containers*
- ❑ *Callbacks*
- ❑ Interação com Componentes



## ❖ 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*

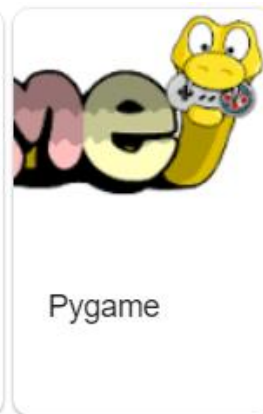
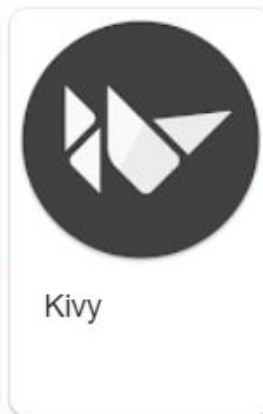


## ❖ Graphical User Interface

- ❑ 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 dispatch loop* - ***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
  - ❑ PyGTK
  - ❑ WxPython
  - ❑ Kivy
  - ❑ PySide
  - ❑ PyQt
  - ❑ Tkinter
  - ❑ .....



## ❖ 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 Tkinter

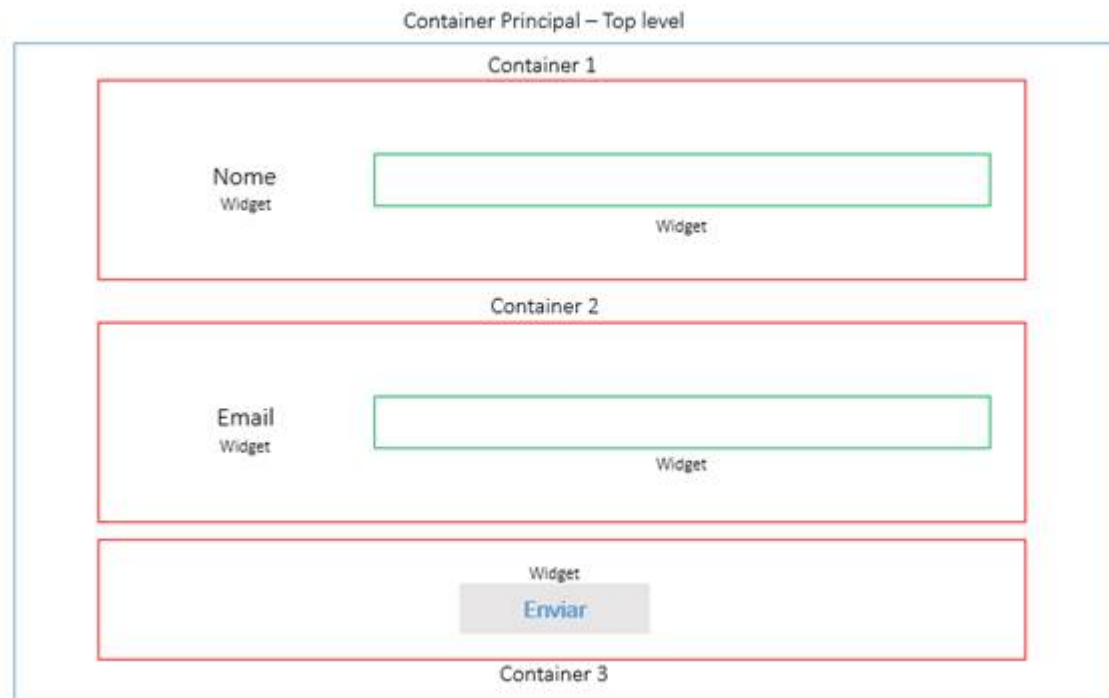
### ❑ Conceitos base:

- ❑ **Event loop:** Existe um ciclo que observa o ambiente ficando à espera da ocorrência de eventos (***event listening***)
- ❑ **Event handler:** Sempre que ocorre um evento é despoletada uma função para gerir / responder a esse evento. Essa função é designada de ***event handler***

## ❖ Biblioteca Tkinter

### ❑ 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, etc...





## ❖ Biblioteca Tkinter

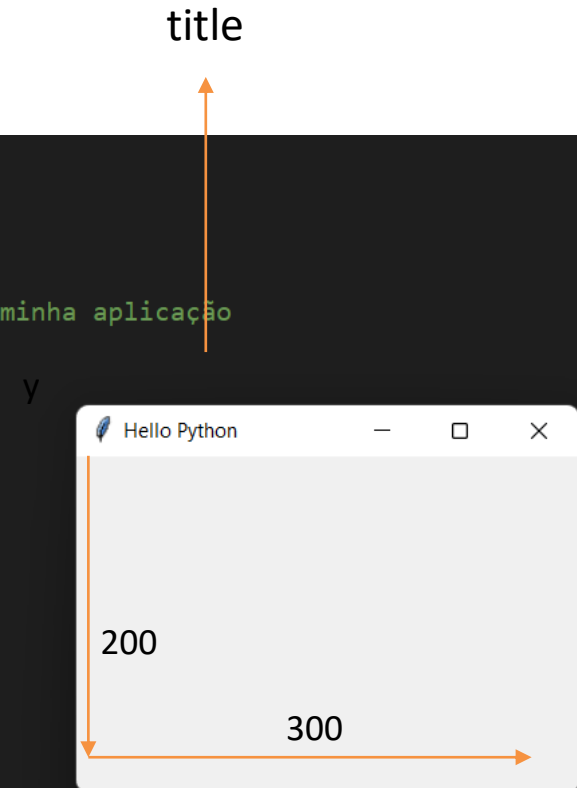
### ☐ Containers & widgets: alguns exemplos

Containers	Widgets: text	Widgets: buttons
<ul style="list-style-type: none"><li>• Window</li><li>• Panel</li><li>• Frame</li><li>• Canvas</li></ul>	<ul style="list-style-type: none"><li>• Label</li><li>• Entry</li><li>• Text</li><li>• Listbox</li><li>• Combobox</li><li>• Spinbox</li><li>• Scale</li><li>• ...</li></ul>	<ul style="list-style-type: none"><li>• Button</li><li>• Radiobutton</li><li>• Checkbutton</li><li>• ...</li></ul>

## ❖ Biblioteca Tkinter

### ❑ Window - A minha primeira Window:

```
1  # Biblioteca Tkinter: UI
2  from tkinter import *
3
4
5  # Criar classe Application, que vai conter as interfaces gráficas da minha aplicação
6  class Application:
7      def __init__(self, master=None):
8          pass
9
10
11
12  window=Tk()                # invoca classe Tk , cria a "main window"
13  Application(window)        # Cria objeto window na classe Application
14  window.geometry("300x200")
15  window.title('Hello Python')
16
17  window.mainloop()          # event listening loop
18
```



### Métodos:

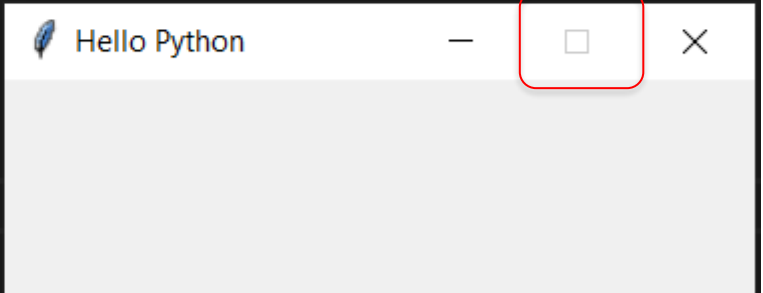
- ❑ Método **Geometry**(width x height) em pixels
- ❑ Método **mainloop()** : cria um *event listening loop*.

a aplicação fica à espera de um evento, que pode ser clicar no botão, inserir dados, escolher uma opção num menu, etc...

## ❖ Biblioteca Tkinter

❑ **Window** – Redimensionamento da janela / container âncora

```
1 # Biblioteca Tkinter: UI
2 from tkinter import *
3
4
5 window=Tk() # invoca classe tk , cria a "main window"
6 window.geometry("300x200")
7 window.title('Hello Python')
8 window.resizable(0,0)
9
10 window.mainloop()
```

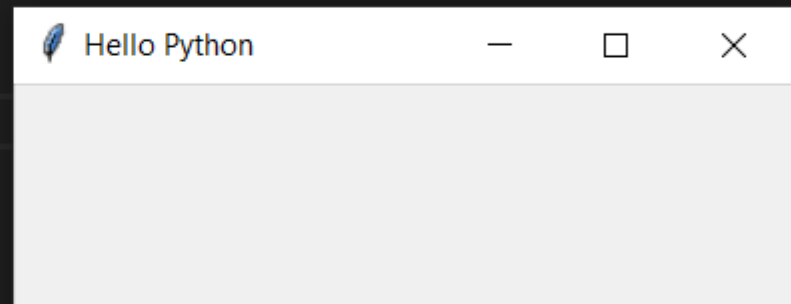
A screenshot of a Tkinter window titled "Hello Python". The window has a standard macOS-style title bar with a red box highlighting the maximize button (a square icon). The window content is a solid light gray.

Desativa redimensionamento da window, largura e altura

## ❖ Biblioteca Tkinter

❑ **Window** – Redimensionamento da janela / container âncora

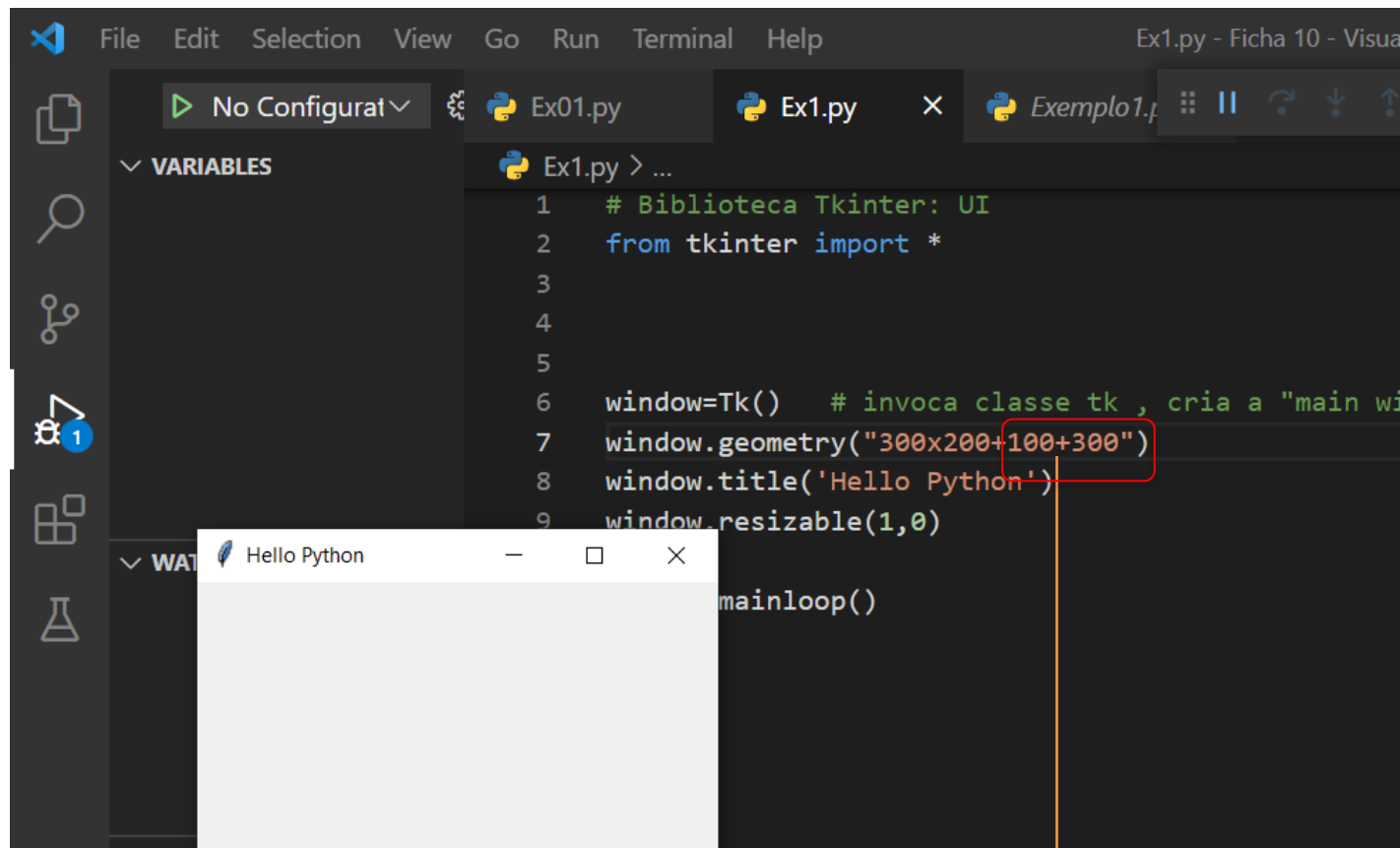
```
window=Tk()    # invoca classe tk , cria a "main window"  
window.geometry("300x200")  
window.title('Hello Python')  
window.resizable(1,0)  
  
window.mainloop()
```



Utilizador pode redimensionar largura - *width*, mas não altura - *height*

## ❖ Biblioteca Tkinter

❑ **Window** – posicionamento do canto superior esquerdo (opcional)



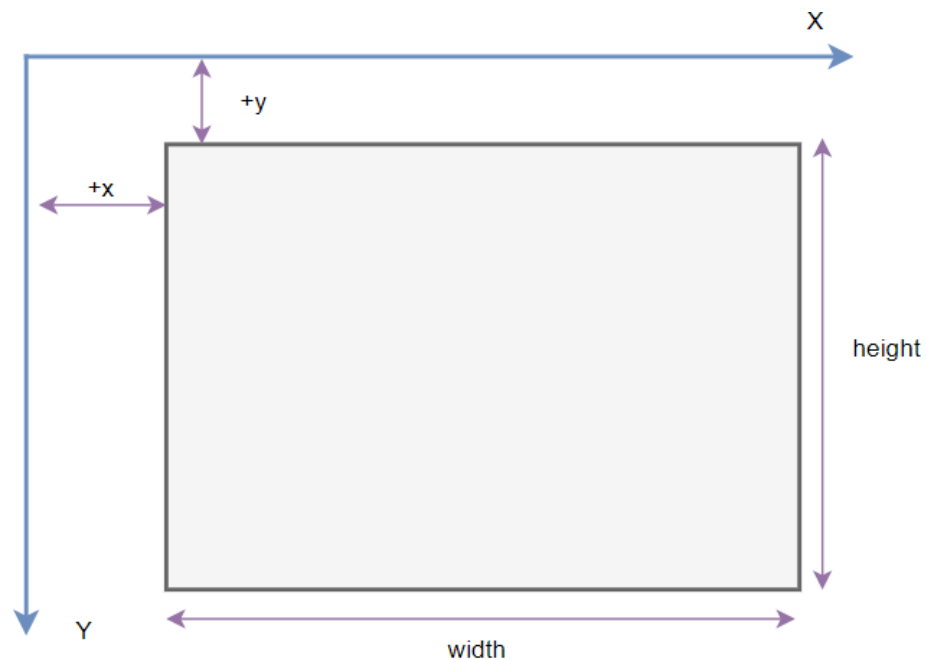
The screenshot shows a Python IDE with a dark theme. The main editor window displays a Python script for Tkinter. The script includes comments in Portuguese and code to create a window titled 'Hello Python'. The window is positioned at (100, 300) with a size of 300x200 pixels. A red box highlights the geometry string '300x200+100+300' in the code. An orange arrow points from this box to the text below. In the foreground, a small window titled 'Hello Python' is visible, representing the output of the script.

```
1 # Biblioteca Tkinter: UI
2 from tkinter import *
3
4
5
6 window=Tk() # invoca classe tk , cria a "main wi
7 window.geometry("300x200+100+300")
8 window.title('Hello Python')
9 window.resizable(1,0)
10
11 mainloop()
```

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

## ❖ Biblioteca Tkinter

❑ **Window** – posicionamento do canto superior esquerdo (opcional)

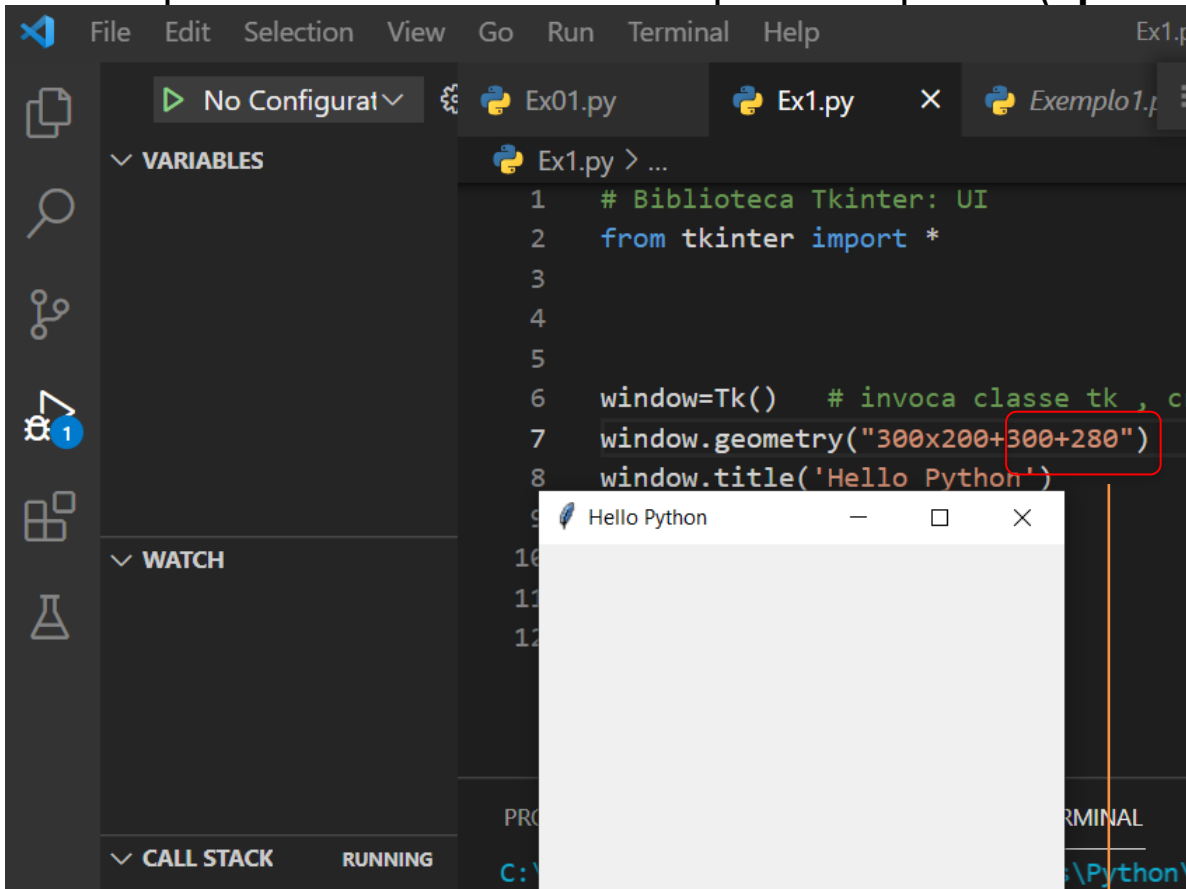


`window.geometry('widthxheight±x±y')`

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

## ❖ Biblioteca Tkinter

❑ **Window** – posicionamento do canto superior esquerdo (**opcional**)



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

## ❖ Biblioteca Tkinter

❑ **Window** – Centrar *main window* de acordo com tamanho do *screen*

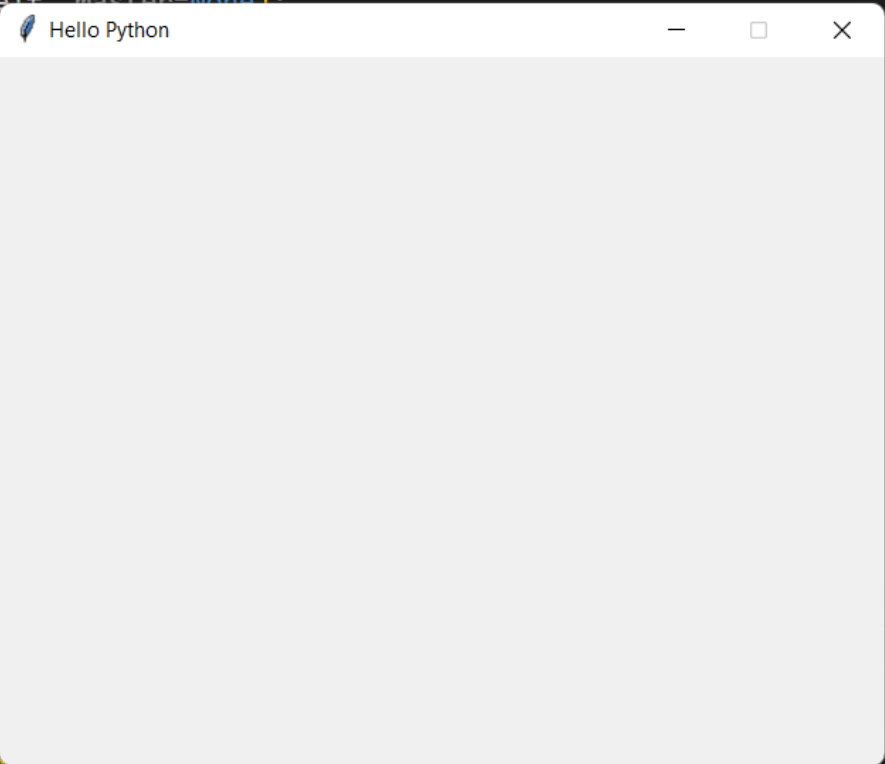
```
12  window=Tk()                # invoca classe Tk , cria a "main window"
13  Application(window)         # Cria objeto window na classe Application
14  window.title('Hello Python')
15  window.resizable(0,0)
16
17  # EXEMPLO para criar uma window CENTRADA no screen do meu dispositivo
18  #-----
19  # obter as dimensões /(em pixeis) do meu screen
20  screenWidth  = window.winfo_screenwidth()
21  screenHeight = window.winfo_screenheight()
22
23  appWidth = 500              # my app width
24  appHeight = 400             # my app height
25
26  x = (screenWidth/2) - (appWidth/2)
27  y = (screenHeight/2) - (appHeight/2)
28  window.geometry(f'{appWidth}x{appHeight}+{int(x)}+{int(y)}')
29
30  window.mainloop()           # event listening loop
31
```



## ❖ Biblioteca Tkinter

❑ **Window** – posicionamento do canto superior esquerdo (**opcional**)

```
5  # Criar classe Application, que vai conter as interfaces gráficas da minha aplicação
6  class Application:
7      def __init__(self, master=None):
8          pass
9
10
11
12  window=Tk()
13  Application(window)
14  window.title('Hello Python')
15  window.resizable(0,0)
16
17  # EXEMPLO para criar uma janela
18  #-----
19  # obter as dimensões da tela
20  screenWidth = window.winfo_screenwidth()
21  screenHeight = window.winfo_screenheight()
22
23  appWidth = 500
24  appHeight = 400
25
26  x = (screenWidth/2) - (appWidth/2)
27  y = (screenHeight/2) - (appHeight/2)
28  window.geometry(f'{appWidth}x{appHeight}+{int(x)}+{int(y)}')
```



## ❖ Biblioteca Tkinter

### ❑ Window – método *configure*

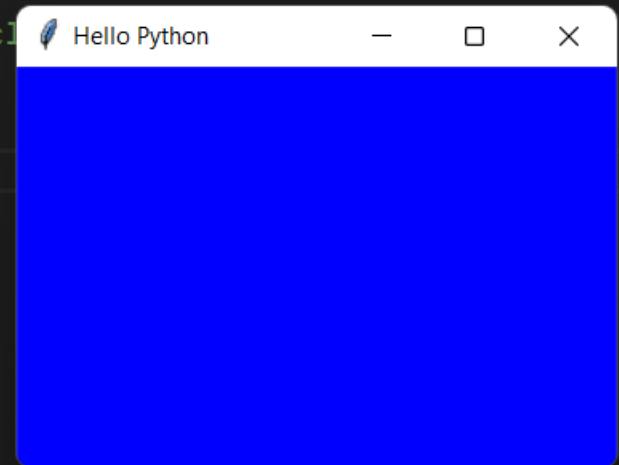
```
3 Application(window)          # Cria objeto window na classe Application
4 window.geometry("300x200+100+300")
5 window.title('Hello Python')
6 window.configure(bg = "white")
7
8 window.mainloop()
9
10 """
11 (cnf: dict[str, Any] | None = ..., *, background:
12   _Color = ..., bd: _ScreenUnits = ..., bg: _Color =
13   ..., border: _ScreenUnits = ..., borderwidth:
14   _ScreenUnits = ..., cursor: _Cursor = ..., height:
15   _ScreenUnits = ..., highlightbackground: _Color =
16   ..., highlightcolor: _Color = ...,
17   highlightthickness: _ScreenUnits = ..., menu: Menu =
18   ..., padx: _ScreenUnits = ..., pady: _ScreenUnits =
19   ..., relief: _Relief = ..., takefocus:
20   _TakeFocusValue = ..., width: _ScreenUnits = ...) ->
21   (dict[str, tuple[str, str, str, Any, Any]] | None)
22
23 panel1 = PanedWin
24 150, bd = "3", r
25
26 ^
27 1/2
28 v
29 Configure resources of a widget.
```

Método **Configure**: permite configurar um componente/widget.  
Neste exemplo, a *window*.

## ❖ Biblioteca Tkinter

### ❑ Window – método *configure()*

```
10
11
12 window=Tk()           # invoca classe Tk , cria a "main window"
13 Application(window)    # Cria objeto window na cl
14 window.geometry("300x200+100+300")
15 window.title('Hello Python')
16 window.configure(bg = "blue")
17
18
19 window.mainloop()      # event listening loop
20
21
22
```



Método *configure*:

- Uso da propriedade *bg* apenas a título de exemplo
- Mais tarde iremos ver diversas propriedades que podemos associar aos componentes

## ❖ Biblioteca Tkinter

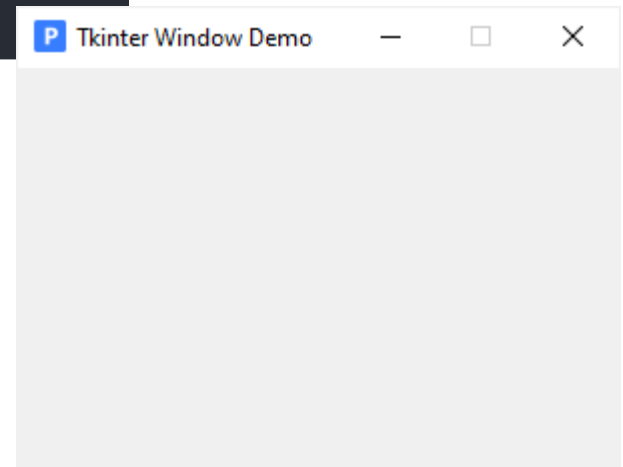
### ❑ Window – método *iconbitmap()*

```
root.title('Tkinter Window Demo')
root.geometry('300x200+50+50')
root.resizable(False, False)
root.iconbitmap('./assets/pythontutorial.ico')

root.mainloop()
```

Método *iconbitmap*:

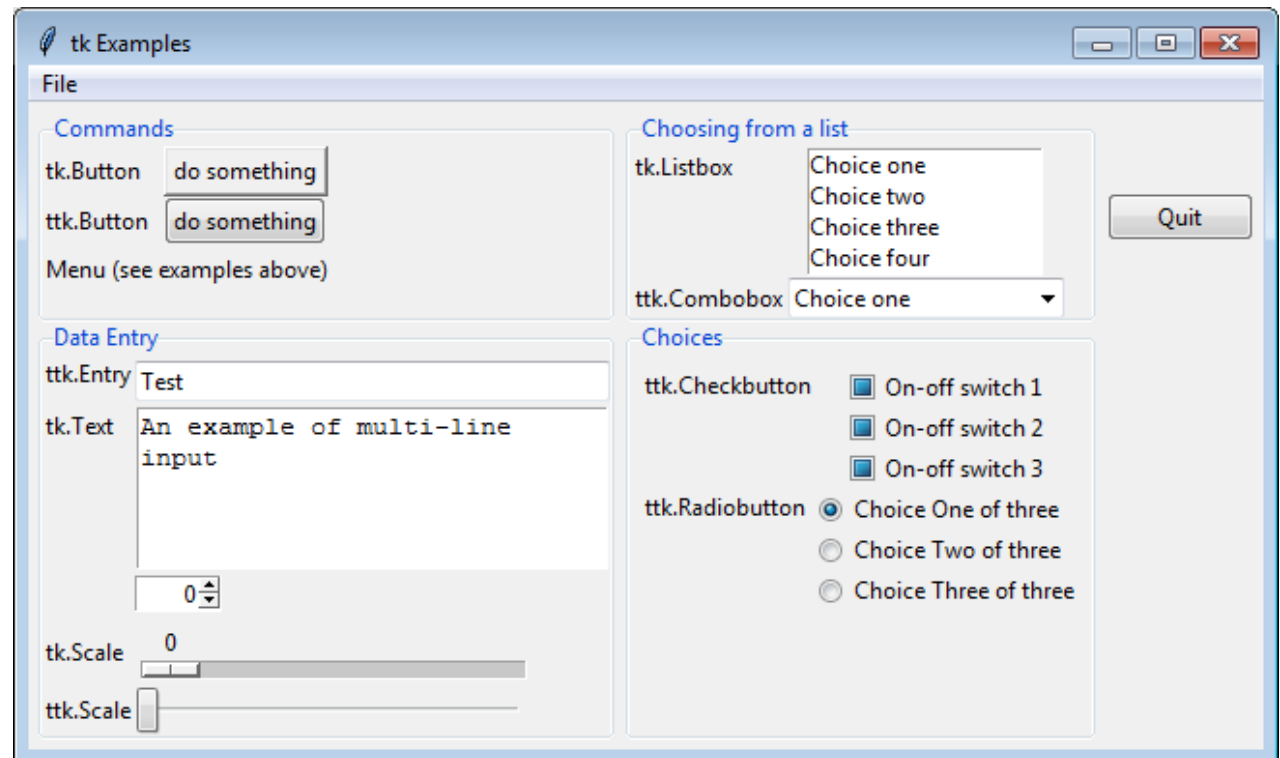
- Icon com extensão .ico
- Converter png, jpg ou outro formato em .ico
- Colocar icon numa pasta do projeto
- Indicar path para a imagem de icon



## ❖ Biblioteca Tkinter

❑ **Tkinter Widgets:** alguns componentes básicos da biblioteca TKinter:

- ❑ Button
- ❑ Label
- ❑ Entry
- ❑ Text
- ❑ Combobox
- ❑ Listbox
- ❑ Radiobutton
- ❑ Checkbutton
- ❑ Spinbox
- ❑ Scale
- ❑ LabelFrame
- ❑ PanedWindow



## ❖ Biblioteca Tkinter

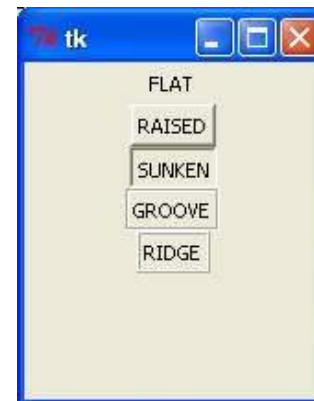
- ❑ Tkinter possui as seguintes classes para gerir o **posicionamento dos widgets** nos containers:
  - ❑ Método **pack()**  
Organiza os widgets em blocos antes de associa-los ao widget pai (window, p.e.)
  - ❑ Método **place()**  
Colocar os widgets numa determinada posição (coordenadas x e y, expressas 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 Tkinter

### ❑ Algumas propriedades comuns à generalidade dos widgets:

- height
- width
- borderwidth (espessura do border)
- padX e padY (espaçamento do widget nas direções X e Y)
- bg (background)
- fg (foreground)
- font
  - family
  - size
  - bold, normal, italic
- relief (estilo)
  - flat
  - raised
  - sunken
  - groove
  - ridge
- **Cores bg/fg:** em hexadecimal ou paleta de cores: "white", "black", "red", "green", "blue", "cyan", "yellow", "magenta"



## ❖ Biblioteca Tkinter

❑ **Algumas propriedades** comuns à generalidade dos widgets:

❑ **Fontes:** font

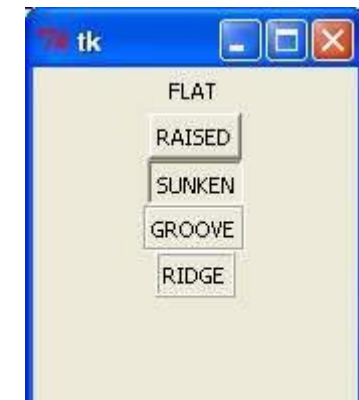
```
#Label
```

```
lbl_altura=Label(window, text="Altura em cm:", fg="blue", font=("Helvetica 9 bold") )
```

- ❑ family
- ❑ size
- ❑ bold, normal, italic

❑ **Estilo** do widget (similar estilo 3D): relief = “flat”, p.e.

- ❑ flat
- ❑ raised
- ❑ sunken
- ❑ groove
- ❑ ridge



```
# Text
```

```
txt_texto = Text(window, width = 55, height = 14, relief = "sunken", bd = 3)
```



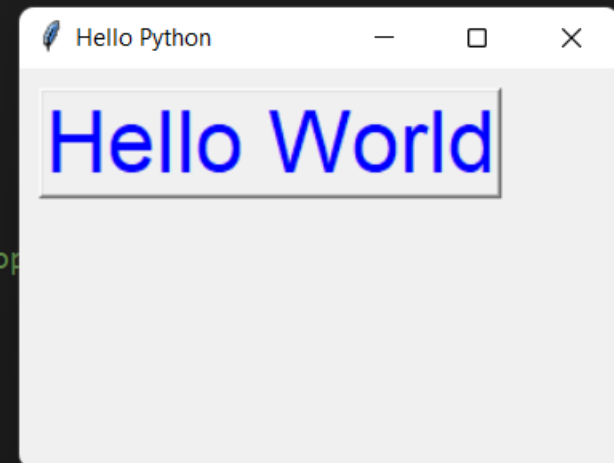
## ❖ Biblioteca Tkinter

### Label

- ❑ *text* : texto associado à label
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name, size;
- ❑ *image* : imagem associada a uma Label, em vez de texto
- ❑ *relief*: estilo associado ao componente

```
#Label
lblTexto= Label(window, text = "Hello World", fg = "blue", relief = "raised", font=(
lblTexto.place(x=10,y=10)

window.mainloop()           # event listening loop
```

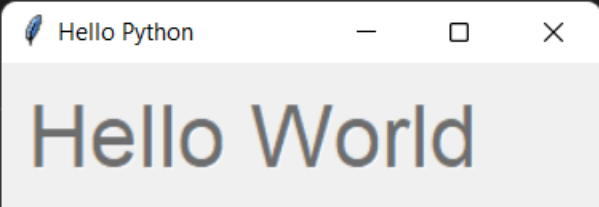


## ❖ Biblioteca Tkinter

### Label

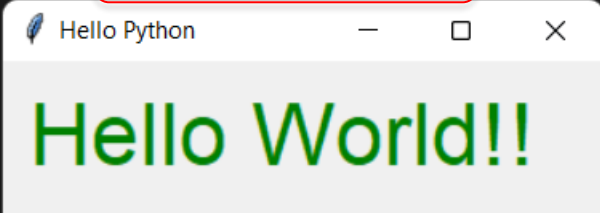
- ❑ *state*: active (por omissão), disabled
- ❑ *textvariable*: para associar o conteúdo de uma Label a uma variável

```
#Label
lblTexto= Label(window, text = "Hello World", fg = "blue", state = "disable", font=("Helvetica", 32))
lblTexto.place(x=10,y=10)
```



MS 6 OUTPUT DEBUG CONSOLE TERMINAL

```
16
17
18 #Label
19 texto = StringVar ()
20 lblTexto= Label(window, fg = "green", textvariable= texto, font=("Helvetica", 32))
21 lblTexto.place(x=10,y=10)
22
23 texto.set("Hello World!!")
24
25
```

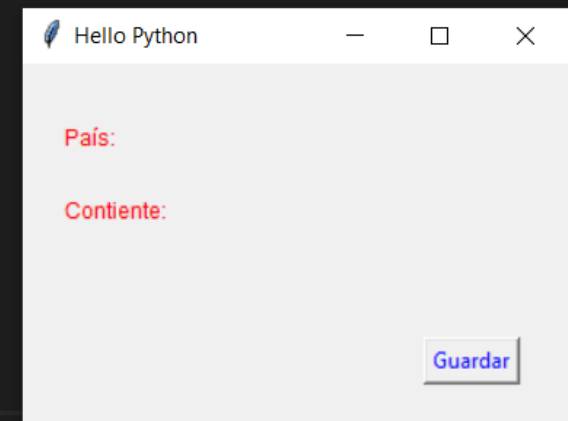


## ❖ Biblioteca Tkinter

### Label

- ❑ *text* : caption of the button; *state*
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name, size; *image* : to be displayed instead of text
- ❑ *command* : function to be called when clicked

```
Exemplo1.py > ...
1  # Biblioteca Tkinter: UI
2  from tkinter import *
3
4
5  window=Tk() # invoca classe tk , cria a "main window"
6  window.geometry("300x200")
7  window.title('Hello Python')
8
9  #Label
10 lbl_pais=Label(window, text="País:", fg='red', font=("Helvetica", 9))
11 lbl_pais.place(x=20, y=30)
12
13 lbl_continente=Label(window, text="Contiente:", fg='red', font=("Helvetica", 9))
14 lbl_continente.place(x=20, y=70)
15
16 # Button
17 btn=Button(window, text="Guardar", fg='blue')
18 btn.place(x=220, y=150)
19
20 window.mainloop() # event listening loop by calling the mainloop()
21
```

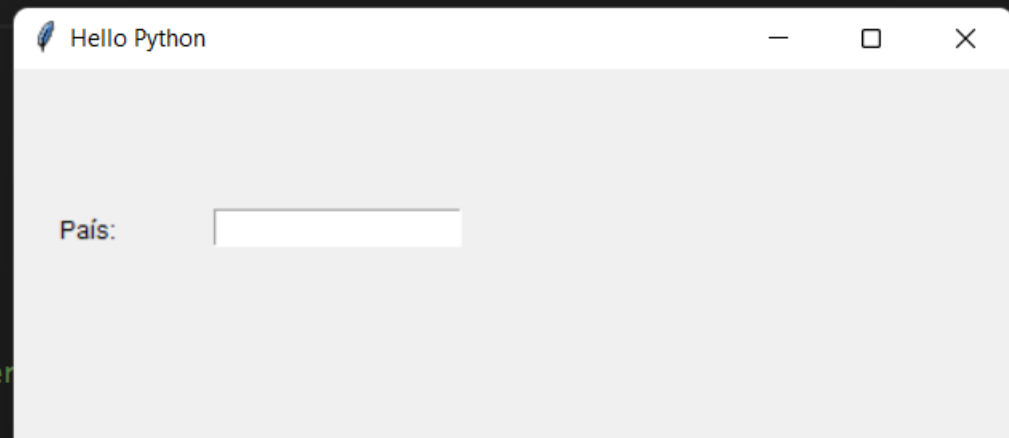


## ❖ Biblioteca Tkinter

### Entry

- ❑ *width*: comprimento do componente
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name , size
- ❑ *bd* : border (2 pixels por defeito).
- ❑ *show* : para converter a Entry num campo de password, *show* = "\*".

```
17
18 #Label
19 lblPais= Label(window, text = "País:", font=("Helvetica", 9))
20 lblPais.place(x=20,y=70)
21
22 #Entry
23 txtPais = Entry(window, width=20)
24 txtPais.place(x=100, y=70)
25
26
27
28 window.mainloop()           # event listener
29
```



\*Widget Entry limitado a uma única linha de texto

## ❖ Biblioteca Tkinter

### Entry

- ❑ *width*: comprimento do componente
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name , size
- ❑ *bd* : border (2 pixels por defeito).
- ❑ *show* : para converter a Entry num campo de password, show = "\*".

```
8 #Label
9 lblPais= Label(window, text = "País:", font=("Helvetica", 9))
10 lblPais.place(x=20,y=70)
11 lblCopntinente= Label(window, text = "Continente:", font=("Helvetica", 9))
12 lblCopntinente.place(x=20,y=120)
13
14 #Entry
15 txtPais = Entry(window, width=20)
16 txtPais.place(x=100, y=70)
17 txtContinente = Entry(window, width=20, state = "disabled" )
18 txtContinente.place(x=100, y=120)
19
20 window.mainloop()           # event listening loop
```

Hello Python

País:

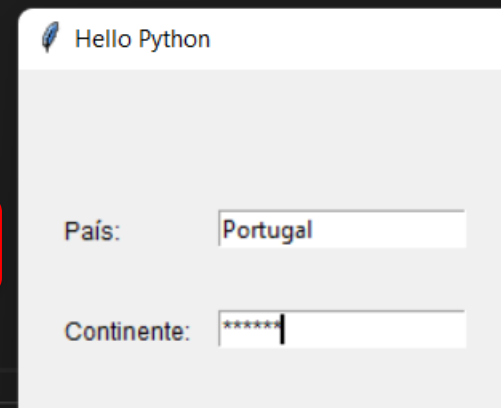
Continente:

## ❖ Biblioteca Tkinter

### Entry

- ❑ *width*: comprimento do componente
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name , size
- ❑ *bd* : border (2 pixels por defeito).
- ❑ *show* : para converter a Entry num campo de password, show = "\*".

```
18  #Label
19  lblPais= Label(window, text = "País:", font=("Helvetica", 9))
20  lblPais.place(x=20,y=70)
21  lblCopntinente= Label(window, text = "Continente:", font=("Helvetica", 9))
22  lblCopntinente.place(x=20,y=120)
23
24  #Entry
25  txtPais = Entry(window, width=20)
26  txtPais.place(x=100, y=70)
27  txtContinente = Entry(window, width=20, show="*" )
28  txtContinente.place(x=100, y=120)
29
30
```

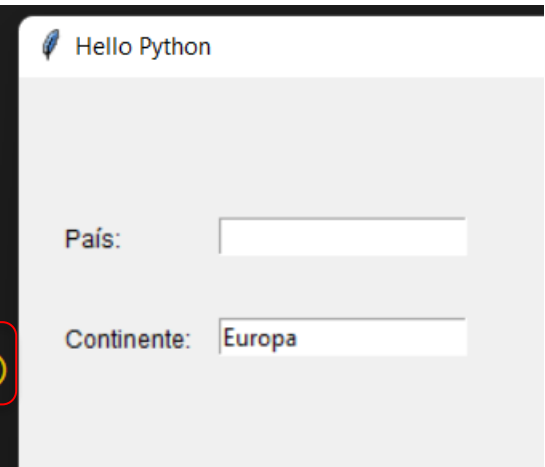


## ❖ Biblioteca Tkinter

### Entry

- ❑ *width*: comprimento do componente
- ❑ *bg* : background; *fg* : foreground
- ❑ *font* : font name , size
- ❑ *bd* : border (2 pixels por defeito).
- ❑ *show* : para converter a Entry num campo de password, show = "\*".

```
23
24 #Entry
25 pais = StringVar()
26 pais.set("")
27 continente = StringVar()
28 continente.set("Europa")
29 txtPais = Entry(window, width=20, textvariable=pais)
30 txtPais.place(x=100, y=70)
31 txtContinente = Entry(window, width=20, textvariable=continente)
32 txtContinente.place(x=100, y=120)
33
```



Hello Python

País:

Continente:

## ❖ Biblioteca Tkinter

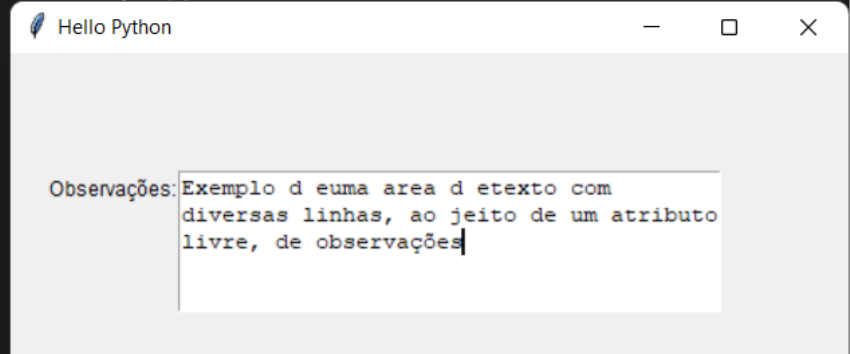
**Text** (mais do que 1 linha de texto)

TIP!

- ☐ *width*: comprimento do componente,
- ☐ *bg* : background; *fg* : foreground
- ☐ *font* : font name , size
- ☐ *bd* : border (2 pixels por defeito)
- ☐ *show* : para converter a Entry num campo de password, *show* = "\*".
- ☐ *state* = "disabled" para tornar a Text inativa
- ☐ *wrap* = "word" , "none", "char"

```
#Label
lblObserv= Label(window, text = "Observações:", font=("Helvetica", 9))
lblObserv.place(x=20,y=70)

txtObserv = Text(window, width=40, height=5, wrap="word")
txtObserv.place(x=100, y=70)
```





## ❖ Biblioteca Tkinter

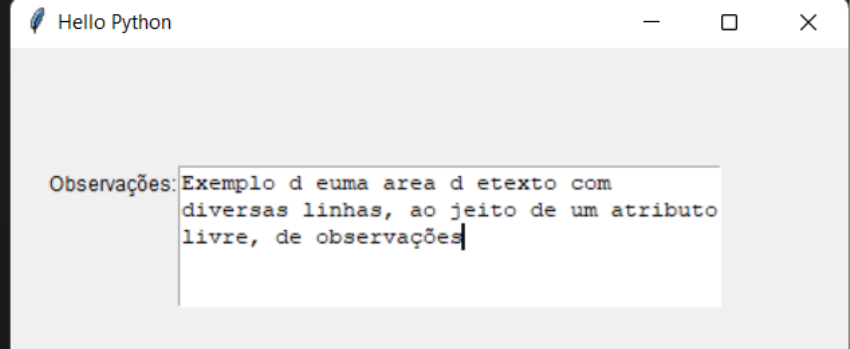
**Text** (mais do que 1 linha de texto)

TIP!

- ☐ *width*: comprimento do componente,
- ☐ *bg* : background; *fg* : foreground
- ☐ *font* : font name , size
- ☐ *bd* : border (2 pixels por defeito)
- ☐ *show* : para converter a Entry num campo de password, *show* = "\*".
- ☐ *state* = "disabled" para tornar a Text inativa
- ☐ *wrap* = "word" , "none", "char"

```
#Label
lblObserv= Label(window, text = "Observações:", font=("Helvetica", 9))
lblObserv.place(x=20,y=70)

txtObserv = Text(window, width=40, height=5, wrap="word")
txtObserv.place(x=100, y=70)
```



## ❖ Biblioteca Tkinter

**Text** (mais do que 1 linha de texto)

TIP!

- ☐ *width*: comprimento do componente,
- ☐ *bg* : background; *fg* : foreground
- ☐ *font* : font name , size
- ☐ *bd* : border (2 pixels por defeito)
- ☐ *show* : para converter a Entry num campo de password, *show* = "\*".
- ☐ *state* = "disabled" para tornar a Text inativa
- ☐ *wrap* = "word" , "none", "char"

```
#Label
lblObserv= Label(window, text = "Observações:", font=("Hel
lblObserv.place(x=20,y=70)

txtObserv = Text(window, width=40, height=5, wrap="none")
txtObserv.place(x=100, y=70)
```

Hello Python

Observações: texto livre para ilustrar uma text que |

## ❖ Biblioteca Tkinter

Button(window, attributes)

### Button

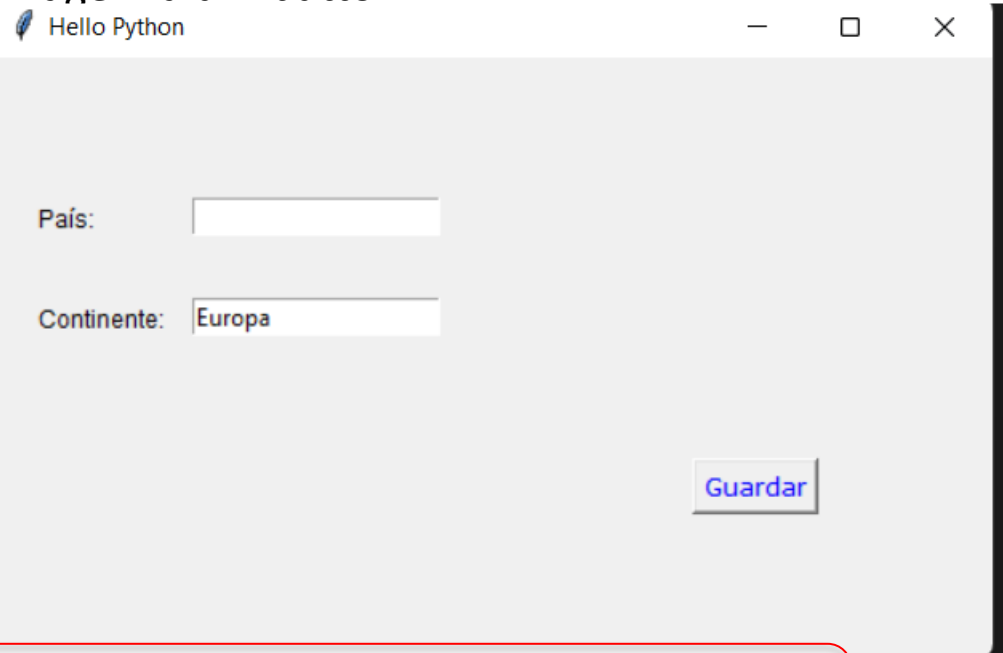
- ❑ *text* : texto associado ao button;
- ❑ *state*: active, disabled
- ❑ *bg* : background color; *fg* : foreground color
- ❑ *font* : font name, size; *bd* (border, default 2)
- ❑ *image* : para associar uma imagem a um button

```
lblPais.place(x=20,y=70)
lblCopntinente= Label(window, text =
lblCopntinente.place(x=20,y=120)

#Entry
pais = StringVar()
pais.set("")
continente = StringVar()
continente.set("Europa")
txtPais = Entry(window, width=20, tex
txtPais.place(x=100, y=70)
txtContinente = Entry(window, width=2
txtContinente.place(x=100, y=120)
```

### #Button

```
btnGuardar = Button(window, font = ("verdana", 10), text = "Guardar", fg="blue")
btnGuardar.place(x=350, y=200)
```



## ❖ Biblioteca Tkinter

Button(window, attributes)

### Button

- ❑ *text* : texto associado ao button;
- ❑ *bg* : background color; *fg* : foreground color
- ❑ *font* : font name, size; *bd* (border, default 2)
- ❑ width, height

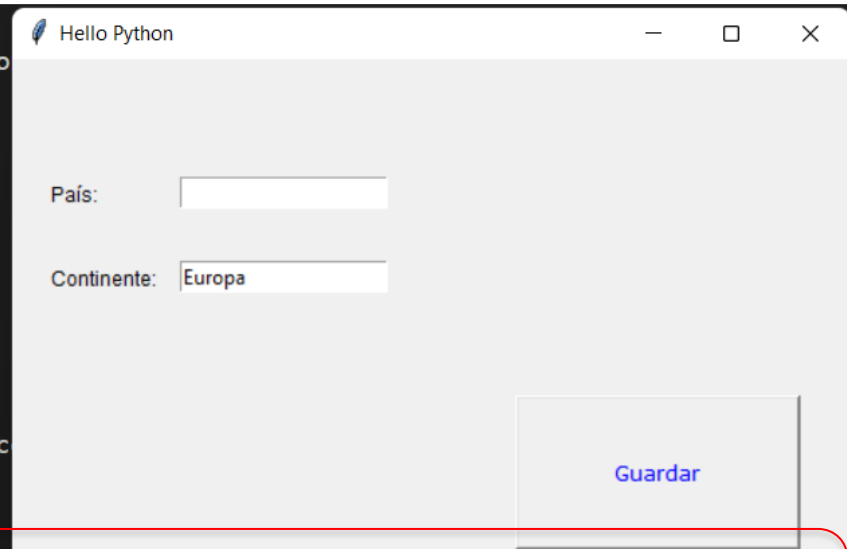
```
lblPais.place(x=20,y=70)
lblCopntinente= Label(window, text = "Continente:", fo
lblCopntinente.place(x=20,y=120)
```

#### #Entry

```
pais = StringVar()
pais.set("")
continente = StringVar()
continente.set("Europa")
txtPais = Entry(window, width=20, textvariable=pais)
txtPais.place(x=100, y=70)
txtContinente = Entry(window, width=20, textvariable=c
txtContinente.place(x=100, y=120)
```

#### #Button

```
btnGuardar = Button(window, width = 20, height = 5, font = ("verdana", 10), text = "Guardar", fg="blue")
btnGuardar.place(x=300, y=200)
```



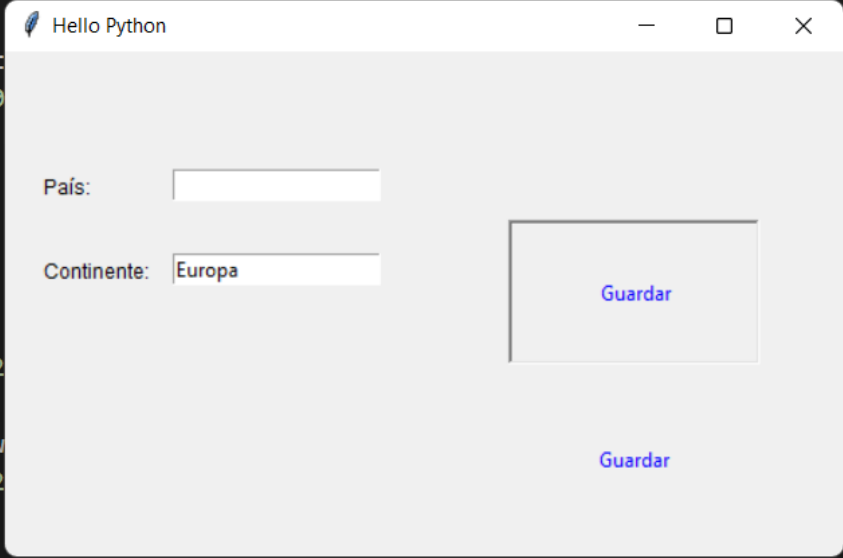
## ❖ Biblioteca Tkinter

Button(window, attributes)

### Button

☐ *relief* : raised, groove, sunken, flat

```
20 lblPais= Label(window, text = "País:", font=("Helvetica", 9))
21 lblPais.place(x=20,y=70)
22 lblCopntinente= Label(window, text = "Continente:", font=("Helvetica", 9))
23 lblCopntinente.place(x=20,y=120)
24
25 #Entry
26 pais = StringVar()
27 pais.set("")
28 continente = StringVar()
29 continente.set("Europa")
30 txtPais = Entry(window, width=20)
31 txtPais.place(x=100, y=70)
32 txtContinente = Entry(window, width=20)
33 txtContinente.place(x=100, y=120)
34
35 #Button
36 btnGuardar = Button(window, width =20, height=5, relief = "sunken", text = "Guardar", fg = "blue")
37 btnGuardar.place(x=300, y=100)
38
39 btnGuardar2 = Button(window, width =20, height=5, relief = "flat", text = "Guardar", fg = "blue")
40 btnGuardar2.place(x=300, y=200)
41
```



## ❖ Biblioteca Tkinter

Button(window, attributes)

### Button

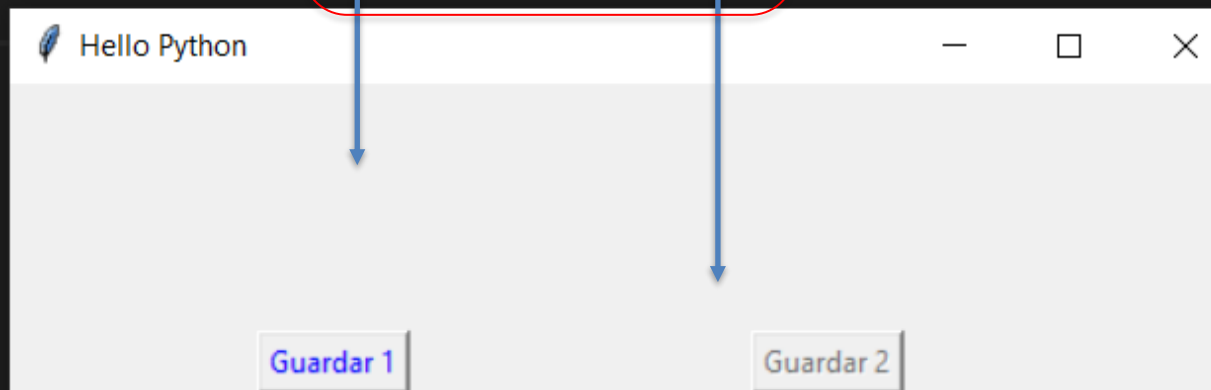
❑ *state: active, disabled*

```
window=Tk()    # invoca classe tk , cria a "main window"
window.geometry("500x300+300+280")
window.title('Hello Python')

# Button
btn1=Button(window, text = "Guardar 1" , state = "active", fg = "blue")
btn1.place(x=100, y=100)

btn2=Button(window, text = "Guardar 2" , state = "disable", fg = "blue")
btn2.place(x=300, y=100)

window.mainloop()
```



## ❖ Biblioteca Tkinter

Button(window, attributes)

### Button

❑ bitmaps em buttons: *error, hourglass, info, question, warning, questhead*

```
# Button
btn1=Button(window, relief = "raised", bitmap = "error")
btn2=Button(window, relief = "groove", bitmap = "hourglass")
btn3=Button(window, relief = "sunken", bitmap = "info")
btn4=Button(window, relief = "flat", bitmap = "question")
```

```
btn1.place(x=50, y=200)
btn2.place(x=100, y=200)
btn3.place(x=150, y=200)
btn4.place(x=200, y=200)
```

Hello Python



## ❖ Biblioteca Tkinter

### Button

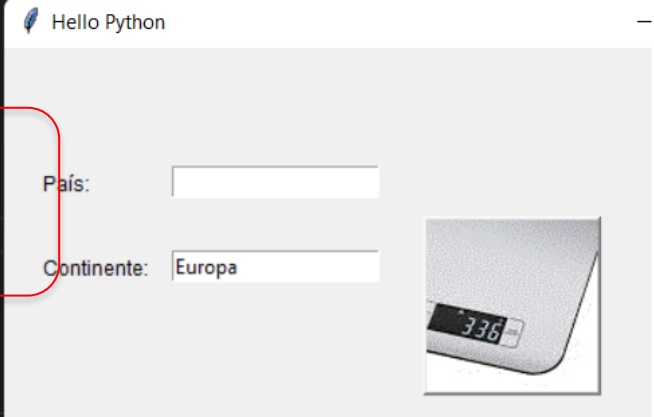
❏ *image*

Button(window, attributes)

```
txtContinente = Entry(window, width=20, textvariable=continente)
txtContinente.place(x=100, y=120)

#Button
imagem = PhotoImage(file = "./balanca.gif")
btnGuardar = Button(window, width =100, height=100, image = imagem)
btnGuardar.place(x=250, y=100)

window.mainloop()           # event listening loop
```





## ❖ Biblioteca Tkinter

### Combobox

❑ values

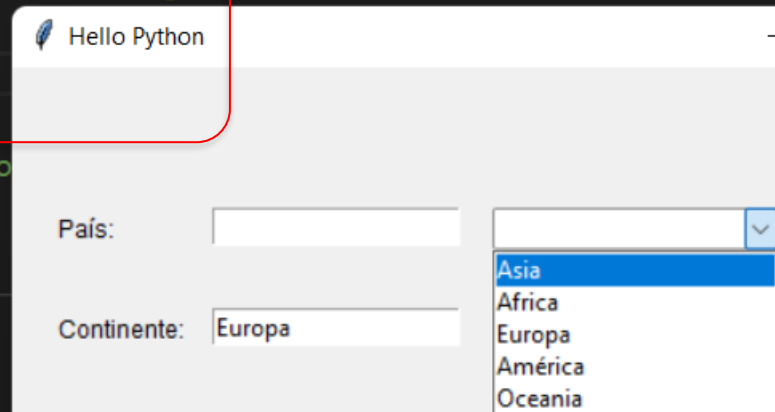
```
1 # Biblioteca Tkinter: UI
2 from tkinter import *
3 from tkinter.ttk import Combobox
4
5
```

```
lista = ['Asia', 'Africa', 'Europa', 'América', 'Oceania']
cbContinente = Combobox(window, values=lista)
cbContinente.place(x=240, y=70)
```

```
window.mainloop() # event listening loop
```

TERMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

ows PowerShell  
right (C) Microsoft Corporation. All rights reserved.



Combobox:

- ❑ Este widget está definido no modulo ttk!
- ❑ Preenche uma lista *dropdown* a partir de uma lista, ou uma coleção de dados
- ❑ Só é possível selecionar um item da Combobox



## ❖ Biblioteca Tkinter

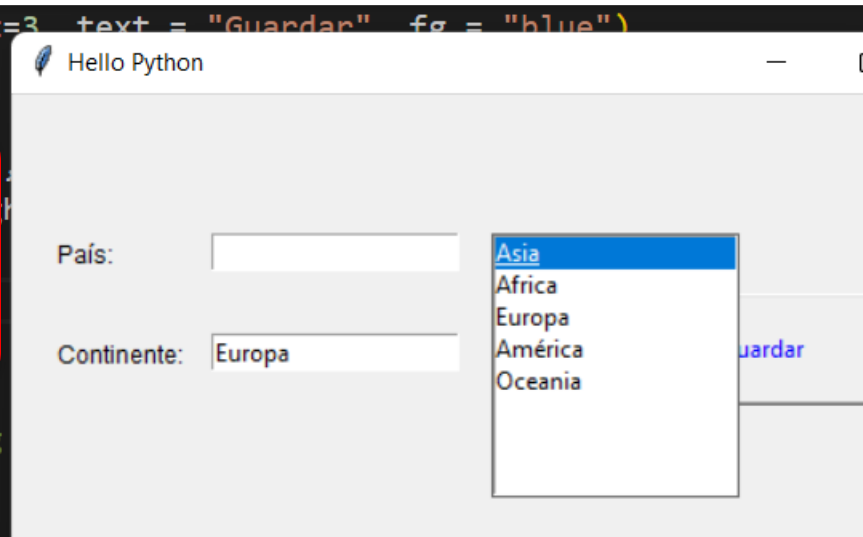
### Listbox

❑ *width* (default 20), *height* (default 10)

```
btnGuardar = Button(window, width =20, height=3, text = "Guardar" fg = "blue")  
btnGuardar.place(x=300, y=100)
```

```
lista = ['Asia', 'Africa', 'Europa', 'América']  
lbContinente = Listbox(window, width=20, height=10)  
for pais in lista:  
    lbContinente.insert(END, pais)  
lbContinente.place(x=240, y= 70)
```

```
window.mainloop() # event listening
```



Listbox:

- ❑ Preenche uma lista *dropdown* a partir de uma lista, ou uma coleção de dados
- ❑ É possível selecionar um ou mais items da Listbox

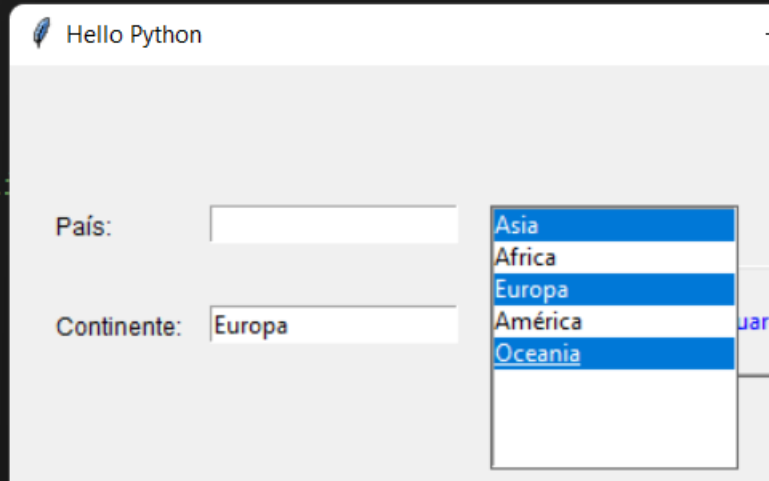
## ❖ Biblioteca Tkinter

### Listbox

- ❑ *bg*: background; *fg*: foreground (texto), *font*, *bd* (border, default 2)
- ❑ *Selectmode*: qts items podem ser selecionados.:
  - ❑ *Single* (pode selecionar 1 item, sem a possibilidade de alterar a seleção)
  - ❑ *multiple* (pode selecionar diversos items)
  - ❑ *browse* (**default**, pode selecionar apenas um item, mas pode alterar)

```
lista = ['Asia', 'Africa', 'Europa', 'América', 'Oceania']
lbContinente = Listbox(window, width=20, height=8, selectmode="multiple")
for pais in lista:
    lbContinente.insert(END, pais)
lbContinente.place(x=240, y=70)

window.mainloop() # event 1:
```

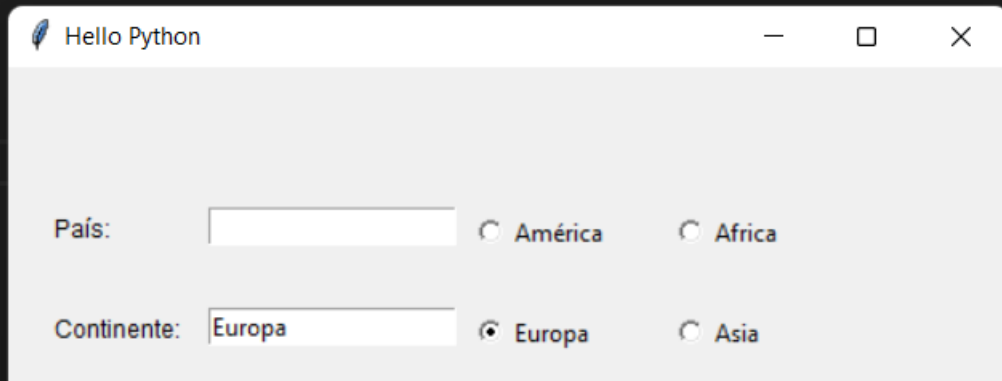


## ❖ Biblioteca Tkinter

**Radiobutton** - seleção exclusiva (apenas 1 opção pode estar selecionada)

- ❑ *fg, bg, bd* (border size, default 2)
- ❑ *font*; *state*, *variable*
- ❑ *variave* : variável da classe StringVar, deve estar associada a todos os radiobuttons de um conjunto de opções. Permite ativar uma das opções

```
47  #RadioButtuons
48  selected = StringVar()
49  selected.set("Europa")
50  rd1 = Radiobutton(window, text = "América", value = "América", variable = selected)
51  rd2 = Radiobutton(window, text = "Africa", value = "Africa", variable = selected)
52  rd3 = Radiobutton(window, text = "Europa", value = "Europa", variable = selected)
53  rd4 = Radiobutton(window, text = "Asia", value = "Asia", variable = selected)
54  rd1.place(x=230, y=70)
55  rd2.place(x=330, y=70)
56  rd3.place(x=230, y=120)
57  rd4.place(x=330, y=120)
58
59  window.mainloop()
60
61
```

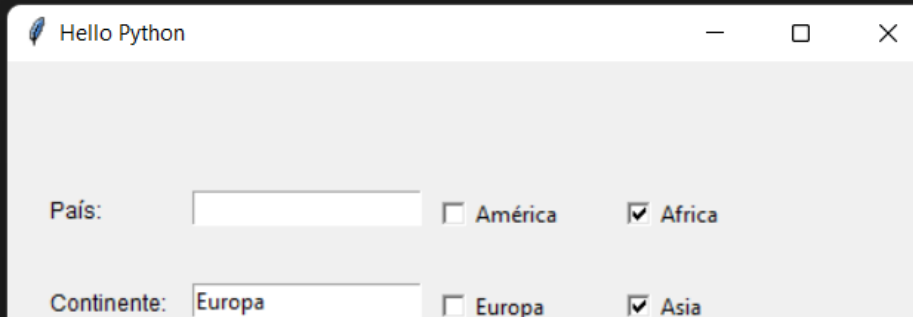


## ❖ Biblioteca Tkinter

**Checkbox** – podemos seleccionar mais do que uma opção

- ☐ *fg, bg*
- ☐ *bd* (border size, default 2)
- ☐ *font; state*

```
47 #CheckButtons
48 checkVar1 = IntVar()
49 checkVar2 = IntVar()
50 checkVar3 = IntVar()
51 checkVar4 = IntVar()
52 checkVar2.set(1)      # 1 - significa que o checkbox está ativo
53 checkVar4.set(1)      # Neste exemplo, os checkboxes 2 e 4 estão ativo, por predefinição
54 cb1 = Checkbutton(window, text = "América", variable = checkVar1)
55 cb2 = Checkbutton(window, text = "Africa", variable = checkVar2)
56 cb3 = Checkbutton(window, text = "Europa", variable = checkVar3)
57 cb4 = Checkbutton(window, text = "Asia", variable = checkVar4)
58 cb1.place(x=230, y=70)
59 cb2.place(x=330, y=70)
60 cb3.place(x=230, y=120)
61 cb4.place(x=330, y=120)
62
63 window.mainloop()
64
65
66
```



Hello Python

País:  ☐ América ☒ Africa

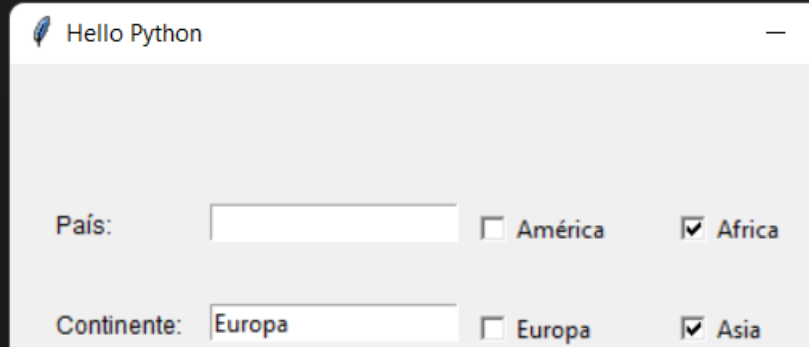
Continente:  Europa ☐ Europa ☒ Asia

## ❖ Biblioteca Tkinter

### Checkbox

```
#CheckButtons
checkVar1 = IntVar()
checkVar2 = IntVar()
checkVar3 = IntVar()
checkVar4 = IntVar()
checkVar2.set(1)      # 1 - significa que o checkbox está ativo
checkVar4.set(1)      # Neste exemplo, os checkboxes 2 e 4 estão
cb1 = Checkbutton(window, text = "América", variable = checkVar1)
cb2 = Checkbutton(window, text = "Africa", variable = checkVar2)
cb3 = Checkbutton(window, text = "Europa", variable = checkVar3)
cb4 = Checkbutton(window, text = "Asia", variable = checkVar4)
cb1.place(x=230, y=70)
cb2.place(x=330, y=70)
cb3.place(x=230, y=120)
cb4.place(x=330, y=120)

window.mainloop()
```



**variable** - variável de controlo do estado atual do checkbox. Normalmente, esta variável é da classe IntVar, em que **0** significa não selecionado e **1** significa selecionado

## ❖ Biblioteca Tkinter

### Spinbox

- ❑ increment (default 1)
- ❑ from\_ to
- ❑ values (associar uma lista ou dicionário)
- ❑ width, font, state, relief, bg, bd (border), ...

Limite inferior      Limite superior



```
#spinbutton
```

```
spin = Spinbox(window, width = 10, from_=0, to=100, increment = 10)  
spin.place(x=70, y=150)
```

```
lista_num = [1,3,5,7,9,11,13,15,17,19]  
spin1 = Spinbox(window, width = 10, values = lista_num)  
spin1.place(x=70, y=200)
```

Hello Python

País:

Contiente:

0

1

## ❖ Biblioteca Tkinter

### Spinbox

- ❑ textvariable : conteúdo do widget,  
atribuir valor por defeito

Valor por defeito

```
#spinbutton  
  
valor = IntVar()  
valor.set(30)  
spin = Spinbox(window, width = 10, from_=0, to=100, textvariable = valor)  
spin.place(x=70, y=150)
```





## ❖ Biblioteca Tkinter

### Scale

- ❑ increment (default 1)
- ❑ from\_to
- ❑ width, font, state, relief, bg, bd (border), ...
- ❑ orient: vertical, horizontal
- ❑ label

```
58  
59  
60 # Scale  
61  
62  
63 scale1 = Scale(window, width = 20, from_=0, to=100)  
64 scale1.place(x=70, y=150)  
65  
66
```



Limite inferior

Limite superior

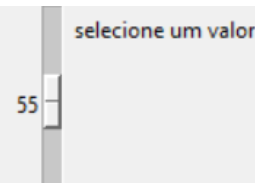
## ❖ Biblioteca Tkinter

### ❑ scale

- ❑ orient (default vertical)
- ❑ variable (variável de controlo para a escala)
- ❑ label

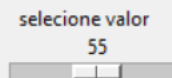
```
Application(window)          # Cria objeto window
window.geometry("600x400+100+300")
window.title('Hello Python')
```

```
valor = IntVar()
valor.set(55)
sc1 = Scale(window, width=10, from_=1, to=100, label = "selecione um valor", variable=valor)
sc1.place(x=120, y=250)
```



```
Application(window)          # Cria objeto window
window.geometry("600x400+100+300")
window.title('Hello Python')
```

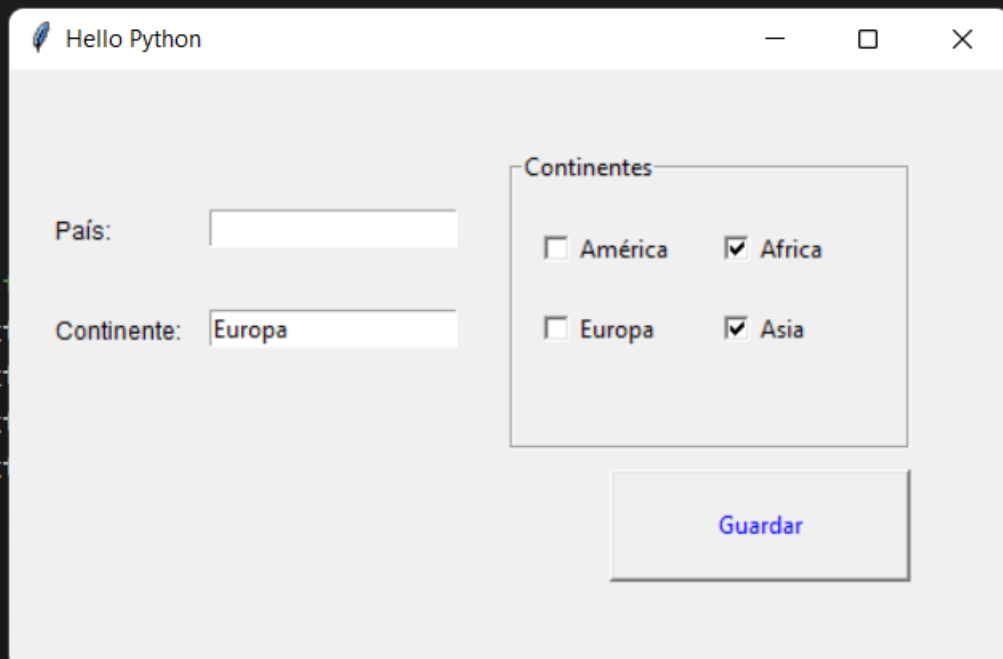
```
valor = IntVar()
valor.set(55)
sc1 = Scale(window, width=10, from_=1, to=100, orient = "horizontal", label = "selecione valor", variable=valor)
sc1.place(x=120, y=250)
```



## ❖ Biblioteca Tkinter

- ❑ **LabelFrame** - é um **container**, cujo objetivo é agrupar componentes em layouts mais complexos

```
47 #LabelFrame
48 frame1 = LabelFrame(window, text = "Continentes", width=200, height=150)
49 frame1.place(x=250, y=40)
50
51 #CheckButtons
52 checkVar1 = IntVar()
53 checkVar2 = IntVar()
54 checkVar3 = IntVar()
55 checkVar4 = IntVar()
56 checkVar2.set(1) # 1 -
57 checkVar4.set(1) # Nest
58 cb1 = Checkbutton(frame1, text
59 cb2 = Checkbutton(frame1, text
60 cb3 = Checkbutton(frame1, text
61 cb4 = Checkbutton(frame1, text
62 cb1.place(x=10, y=20)
63 cb2.place(x=100, y=20)
64 cb3.place(x=10, y=60)
65 cb4.place(x=100, y=60)
```



TIP!

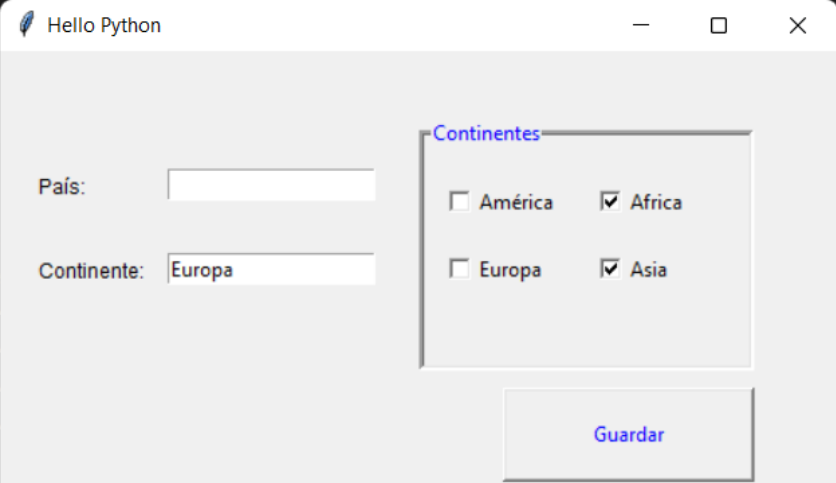
Os widgets posicionados no container baseiam-se nas coordenadas (x, y) do container e não da window!

## ❖ Biblioteca Tkinter

❑ **LabelFrame** - é um container, cujo objetivo é agrupar componentes em layouts mais complexos

❑ *bg, fg, bd, width, height, font, relief, text*

```
46
47 #LabelFrame
48 frame1 = LabelFrame(window, text = "Continentes", width=200, height=150, relief="sunken", bd = 3, fg = "blue")
49 frame1.place(x=250, y=40)
50
51 #CheckButtons
52 checkVar1 = IntVar()
53 checkVar2 = IntVar()
54 checkVar3 = IntVar()
55 checkVar4 = IntVar()
56 checkVar2.set(1)      # 1 - significa que o c
57 checkVar4.set(1)      # Neste exemplo, os che
58 cb1 = Checkbutton(frame1, text = "América", var
59 cb2 = Checkbutton(frame1, text = "Africa", var
60 cb3 = Checkbutton(frame1, text = "Europa", var
61 cb4 = Checkbutton(frame1, text = "Asia", var
62 cb1.place(x=10, y=20)
```



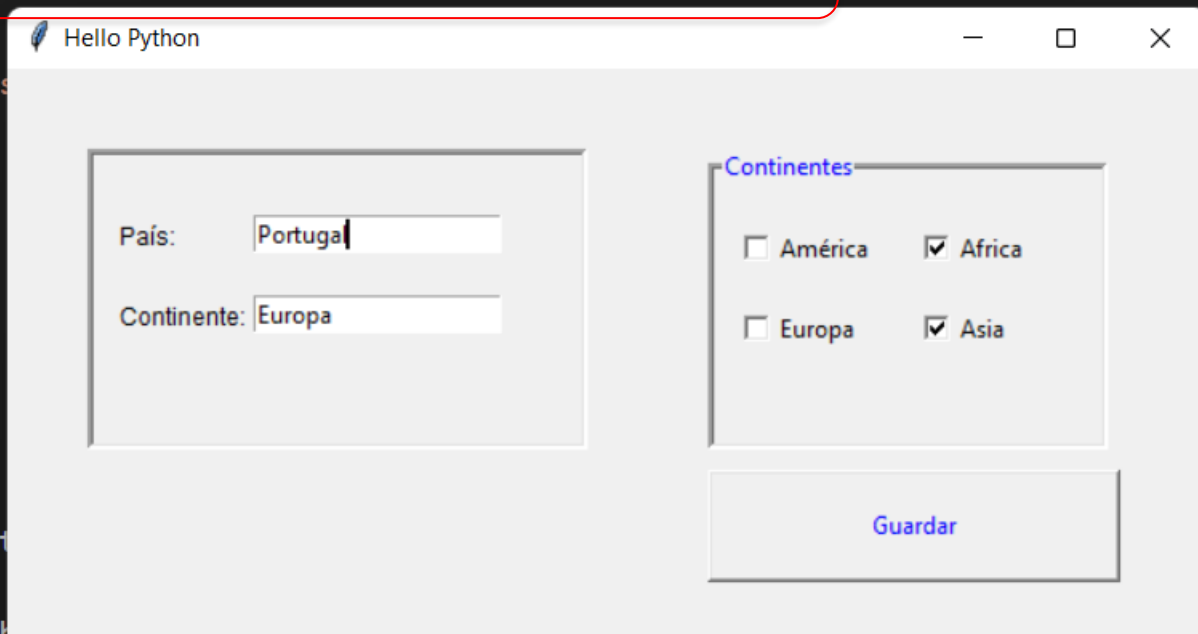
## ❖ Biblioteca Tkinter

- ❑ **PaneledWindow** - é um container que permite organizar o layout da aplicação de uma forma lógica
  - ❑ *bg, fg, bd, width, height, font, relief, text*
  - ❑ *orient* (default horizontal)

```
#Panel
panel1 = PanedWindow(window, width=250, height=150, bd = 3, relief = "sunken")
panel1.place(x=40, y=40)
```

```
#Label
lblPais= Label(panel1, text = "País")
lblPais.place(x=10,y=30)
lblCopcontinente= Label(panel1, text = "Continente")
lblCopcontinente.place(x=10,y=70)
```

```
#Entry
pais = StringVar()
pais.set("")
continente = StringVar()
continente.set("Europa")
txtPais = Entry(panel1, width=20, textvariable=pais)
txtPais.place(x=80, y=30)
txtContinente = Entry(panel1, width=20, textvariable=continente)
txtContinente.place(x=80, y=70)
```



Os widgets posicionados no container baseiam-se nas coordenadas (x, y) do container e não da window!

## ❖ Biblioteca Tkinter

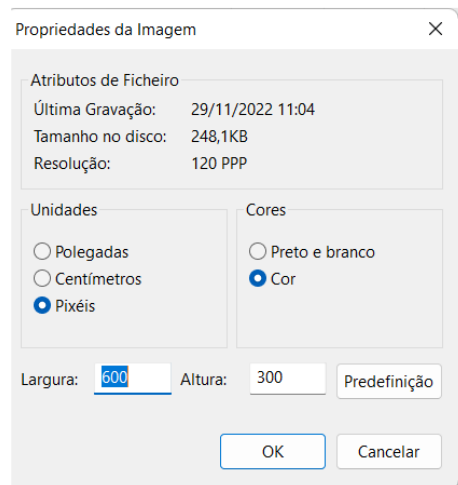
❑ **Canvas** - é um container especial, que suporta imagens ou desenhos geométricos

❑ bg, fg, bd, width, height, relief

Imagens:

❑ .png

❑ .gif



```
#Canvas
```

```
ctnImage= Canvas(window, width=600, height=300, bd = 2, relief = "sunken")  
ctnImage.place (x=10, y=10)
```

```
img= PhotoImage(file = "continentes.png")  
ctnImage.create_image(300,150, image = img)
```

centro da imagem  
(imagem centrada no controlo canvas)

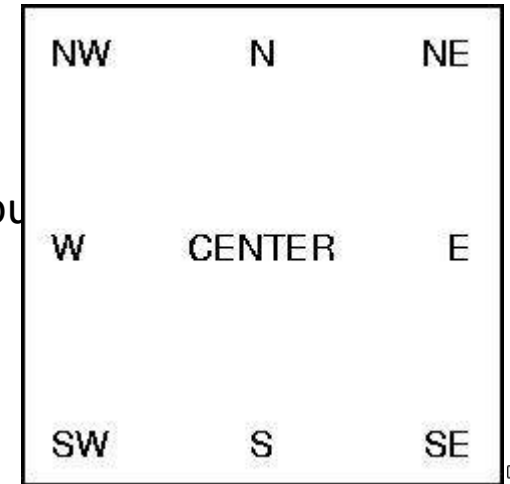
## ❖ Biblioteca Tkinter

❑ **Canvas** - é um container especial, que suporta imagens ou

❑ bg, fg, bd, width, height, relief

anchor:

❑ nw , n , ne , w, center, e, sw, s, se ...



Hello Python



```
25
26 #Canvas
27 ctnImage= Canvas(window, width=600, height=300, bd = 2, relief = "sunken")
28 ctnImage.place (x=10, y=10)
29
30 img= PhotoImage(file = "balanca2.png")
31
32 ctnImage.create_image(0,0, anchor ="nw" , image = img)
33
34
```

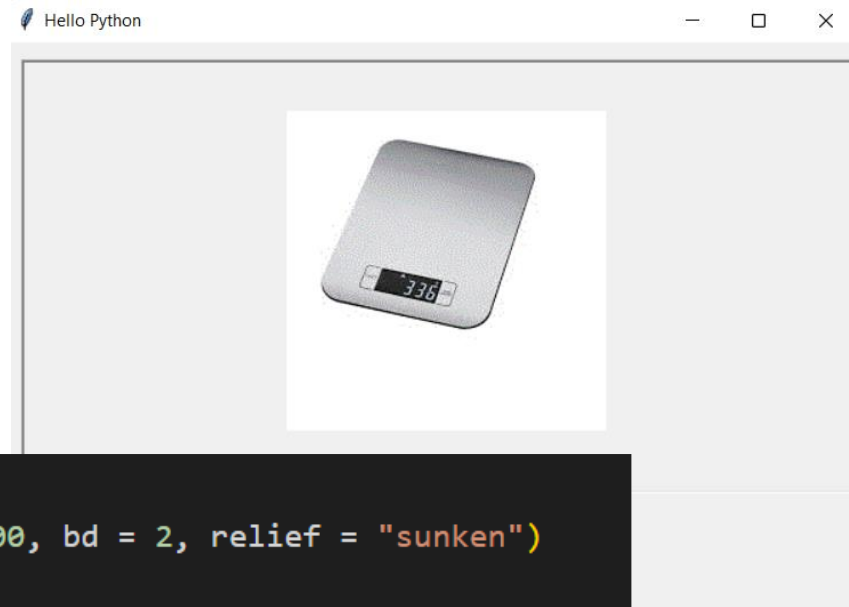
## ❖ Biblioteca Tkinter

❑ **Canvas** - é um container especial, que suporta imagens ou desenhos geométricos

❑ bg, fg, bd, width, height, relief

anchor:

❑ nw , n , ne , w, center, e, sw, s, se ...



```
#Canvas
ctnImage= Canvas(window, width=600, height=300, bd = 2, relief = "sunken")
ctnImage.place (x=10, y=10)

img= PhotoImage(file = "balanca2.png")

ctnImage.create_image(300,150, anchor ="center" , image = img)
```



## ❖ Callbacks

❑ Widgets e containers permitem construir a interface gráfica de uma aplicação

❑ Método *mainloop()* fica à espera de um evento - **event listening**

```
# container Canvas, usado para aplicações de desenho: imagens e formas geométricas
ctn_canvas = Canvas(window, width = 200, height = 200, bd = 2, relief = "sunken")
ctn_canvas.place(x=10, y=10)

img = ImageTk.PhotoImage(Image.open("balanca1.JPG"))
ctn_canvas.create_image(100,100, image = img)

|
window.mainloop() # event listening loop by calling the mainloop()
```

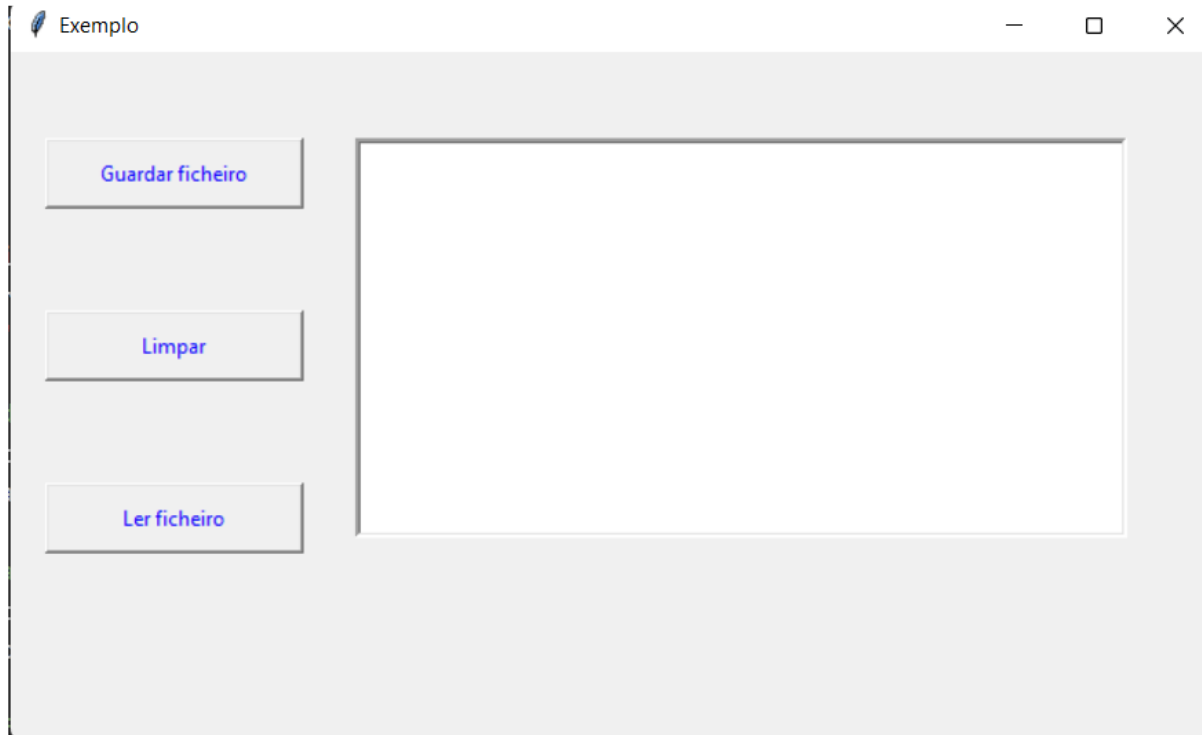
❑ Quando o utilizador interage com a interface gráfica, é desencadeado um **evento**: **event handler**

❑ A resposta a esse evento faz-se com uma **ação (callback)**, que consiste na chamada de uma função para executar determinado código

- A chamada de uma função faz-se com a instrução **command =** , associada a um componente

## ❖ Callbacks

❑ Um exemplo:



## ❖ Callbacks

```
10 # Variavel global com designacao do ficheiro
11 ficheiro = ".\\ficheiros\\texto.txt"
12
13
14 > def guarda_ficheiro():...
21
22
23 # Limpa o conteúdo da Text
24 > def limpar():...
26
27
28
29 # Guarda o conteudo da Text em ficheiro
30 > def ler_ficheiro():...
37
```

```
44
45 # Button Guardar
46 btnGuardar=Button(window, text = "Guardar ficheiro" , width = 20, height = 2, fg = "blue", command = guarda_ficheiro)
47 btnGuardar.place(x=20, y=50)
48
49 # Button Limpar
50 btnLimpar=Button(window, text = "Limpar",width = 20, height = 2, fg = "blue", command = limpar)
51 btnLimpar.place(x=20, y=150)
52
53 # Button Ler ficheiro
54 btnLer=Button(window, text = "Ler ficheiro", width = 20, height = 2, fg = "blue", command = ler_ficheiro)
55 btnLer.place(x=20, y=250)
56
57 txtTexto = Text(window, width = 55, height = 14, relief = "sunken", bd = 3)
58 txtTexto.place(x = 200, y= 50)
59
60 window.mainloop() # event listening loop by calling the mainloop()
```

## ❖ Métodos associados a widgets/componentes

### ❑ Text

❑ Métodos *insert*, *delete*, *get*

❑ `insert(index, string)`

❑ `delete(index1, index2)`

❑ `get(index1, index2)`

"1.0" - linha 1, character 0

"end" – final da Text

"end-1c" – final da Text

sem \n no final da Text

```
# Guarda o conteúdo da Text em ficheiro
def ler_ficheiro():
    linha1 = "linha 1\n"
    linha2 = "linha 2\n"
    linha3 = "linha 3\n"
    linha4 = "linha 4\n"

    txtTexto.insert("insert", linha1)
    txtTexto.insert("1.0", linha2)
    txtTexto.insert("0.0", linha3)
    txtTexto.insert("end", linha4)

    #txtTexto.delete("1.0", "1.5")
```

Exemplo

Guardar ficheiro

Limpar

Ler ficheiro

linha 3  
linha 2  
linha 1  
linha 4

## ❖ Métodos associados a widgets/componentes

### ❑ Text

❑ Métodos *insert*, *delete*, *get*

❑ `insert(index, string)`

❑ `delete(index1, index2)`

❑ `get(index1, index2)`

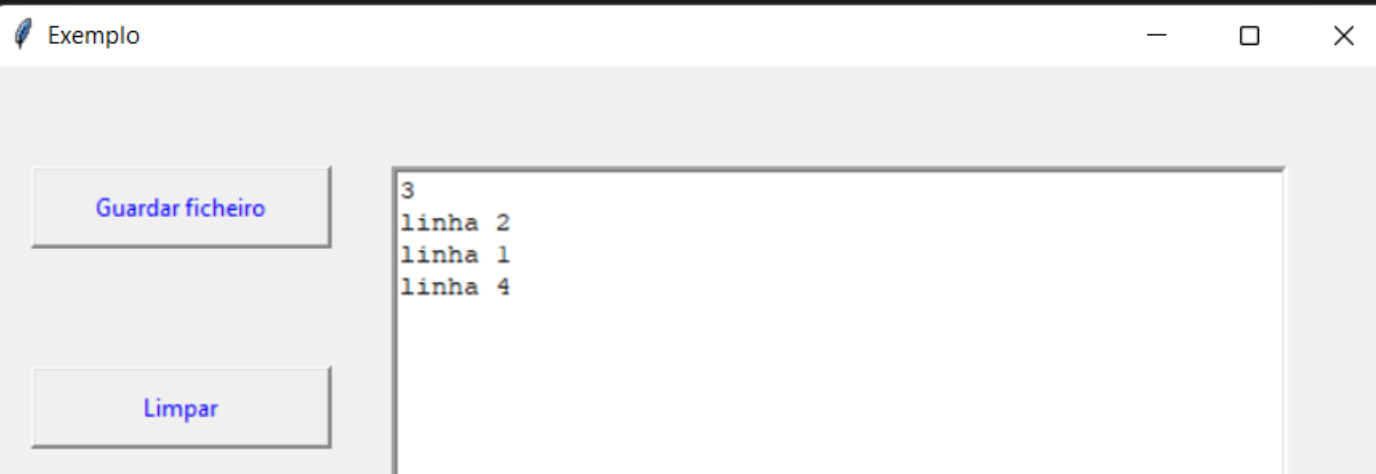
"1.0" - linha 1, character 0

"end" – final da Text

"end-1c" – final da Text

sem \n no final da Text

```
txtTexto.delete("1.0", "1.6")    # remove do primeiro ao último caracter (1ª LINHA)  
#txtTexto.delete("1.0", "end")  # remove do 1º ao último caracter da Text
```



## ❖ Métodos associados a widgets/componentes

### ❑ Text

❑ Métodos *insert*, *delete*, *get*

❑ **insert**(*index*, *string*)

❑ **delete**(*index1*, *index2*)

❑ **get**(*index1*, *index2*)

"1.0" - linha 1, character 0

"end" – final da Text

"end-1c" – final da Text

sem \n no final da Text

```
texto1 = StringVar()  
texto1 = txtTexto.get("0.0", "end")  
print(texto1)
```

Exemplo

Guardar ficheiro

Limpar

Ler ficheiro

```
este  
é um exercicio da ficha 10  
biblioteca TKinter  
aaa  
bbb
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
ter/../../debugpy\launcher' '52203' '--'  
este  
é um exercicio da ficha 10  
biblioteca TKinter  
aaa  
bbb
```

## ❖ Métodos associados a widgets/componentes

### ❑ Entry

- ❑ **get()** – obter
- ❑ **set()** – atribuir

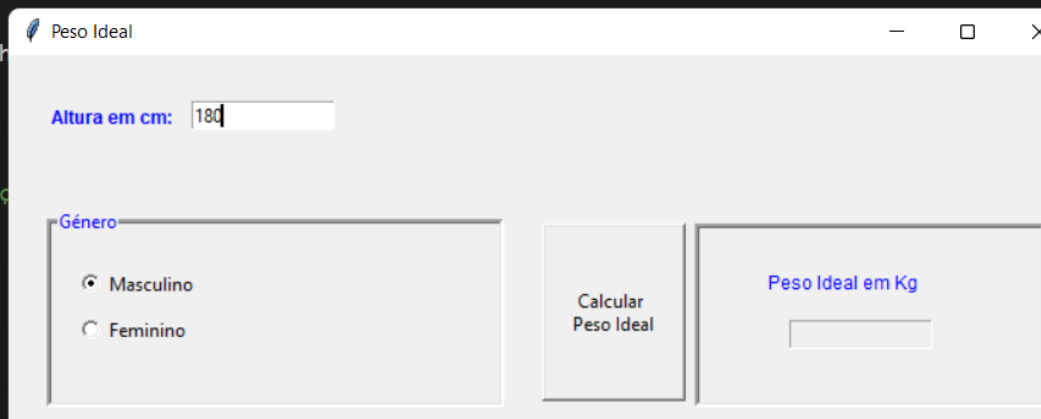
```
def PesoIdeal():  
    txt_PesoIdeal.config( fg = "red", width = 10, font = ("Helvetica", 11))  
    k=4  
    if selected.get() == "Masculino":  
        k=4  
    else:  
        k=2  
  
    # Obter valor da Entry altura: altura.get()  
    peso = (altura.get() - 100) - (altura.get() - 150)/k  
    peso_ideal.set(str(peso))
```

```
#Entry  
altura = IntVar()  
txt_altura=Entry(window, width = 15, textvariable = altura)  
txt_altura.place(x=120, y=30)
```

```
#Radiobutton  
lframe = LabelFrame(window, width=150, height=100)  
lframe.place(x=25, y=100)
```

```
selected = StringVar()  
selected.set("Masculino") # Opção inicial  
rd1 = Radiobutton(lframe, text = "Masculino", variable=selected, value="Masculino")  
rd1.place(x=15, y=20)  
rd2 = Radiobutton(lframe, text = "Feminino", variable=selected, value="Feminino")  
rd2.place(x=15, y=50)
```

```
#Button
```



## ❖ Interação com componentes

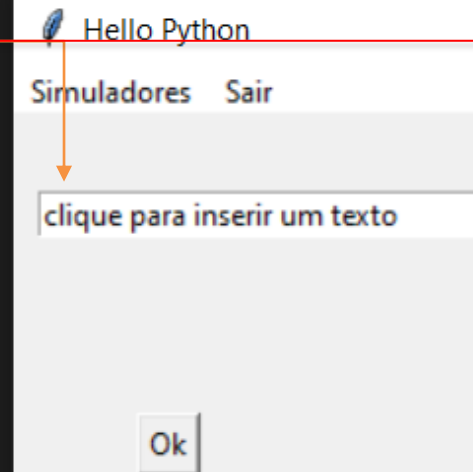
### ❑ Entry

- ❑ Associar conteúdo de um componente Entry a uma variável

```
#Entry
texto_intro = StringVar()
texto_intro.set("clique para inserir um texto")
txt_texto = Entry(window, width = 30, relief = "sunken", textvariable = texto_intro )
txt_texto.place(x=10, y=30)
```

```
def mensagem():
    msg = Message(window, text = texto_intro.get())
    msg.place(x=100, y=200)

btn = Button(window, text = "Ok" , command = mensagem)
btn.place(x=50, y=120)
```





## ❖ Métodos associados a widgets/componentes

### ❑ Radiobutton

- ❑ **get()** – obter
- ❑ **set()** - atribuir

```
def PesoIdeal():  
    txt_PesoIdeal.config( fg = "red", width = 10,  
                          k=4  
    if selected.get() == "Masculino":  
        k=4  
    else:  
        k=2
```

```
31 selected = StringVar()  
32 selected.set("Masculino") # Opção selecionada por defeito  
33 rd1 = Radiobutton(lframe, text = "Masculino", value = "Masculino", variable = selected)  
34 rd1.place(x=15, y=20)  
35 rd2 = Radiobutton(lframe, text = "Feminino", value = "Feminino", variable = selected)  
36 rd2.place(x=15, y=50)  
37  
38 #Button  
39 btn_PesoIdeal=Button(window, text="Calcular", command=PesoIdeal)  
40 btn_PesoIdeal.place(x=350, y=110)  
41  
42  
43 # Panel  
44 panel1 = PanedWindow(window, width=300, height=100)  
45 panel1.place(x=450, y=110)  
46  
47 lbl_PesoIdeal=Label(panel1, text="Peso Ideal em Kg")  
48 lbl_PesoIdeal.place(x=42, y=25)
```

Peso Ideal

Altura em cm: 0

Género

☒ Masculino  
☐ Feminino

Calcular  
Peso Ideal

Peso Ideal em Kg

## ❖ Interação com componentes

### ☐ Checkbutton

```
#Checkbutton
cb1 =IntVar()
cb1.set(1)      # 1 => selected.
cb2 =IntVar()
cb3 =IntVar()
cb4 =IntVar()

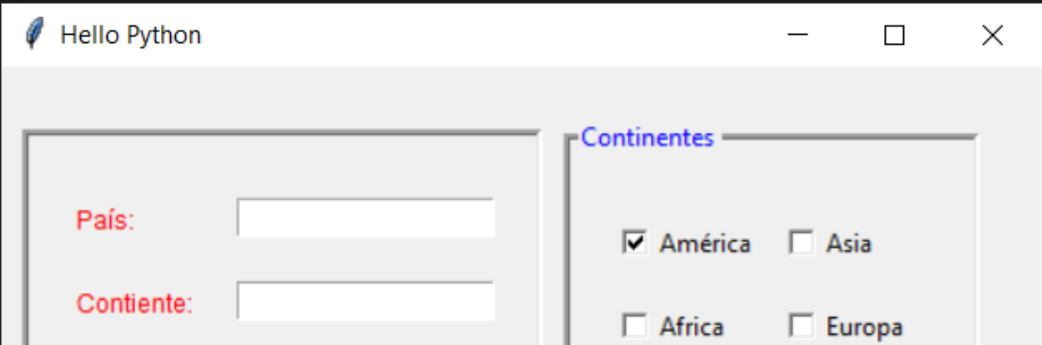
frame1 = LabelFrame(window, text="Continentes ", width =200, height=150, relief = "sunken", bd = "3" , fg = "blue")
frame1.place(x=270, y=25)

cb1_continente = Checkbutton(frame1, text = "América", variable = cb1, command = escolha)
cb2_continente = Checkbutton(frame1, text = "Asia",    variable = cb2, command = escolha)
cb3_continente = Checkbutton(frame1, text = "Africa",  variable = cb3, command= escolha)
cb4_continente = Checkbutton(frame1, text = "Europa",  variable = cb4, command= escolha)

cb1_continente.place(x=20, y=30)
cb2_continente.place(x=100, y=30)
cb3_continente.place(x=20, y=70)
cb4_continente.place(x=100, y=70)

# Button
btn=Button(window, text = "Guardar")
btn.place(x=400, y=250)
```

Variável associada a cada Checkbutton  
1 significa selecionado, 0 desselecionado



## ❖ Interação com componentes

### ☐ Checkbutton

Se variável associada a cada Checkbutton == 1:  
significa que seleccionei o objeto  
coloco o seu conteúdo na variável associada à Entry

```
def escolha():  
    if cb1.get() == 1: # 1 => selected, 0 => not selected  
        continente.set("América")  
    if cb2.get() == 1:  
        continente.set("Asia")  
    if cb3.get() == 1:  
        continente.set("Africa")  
    if cb4.get() == 1:  
        continente.set("Europa")
```

#Checkbutton

```
cb1 = IntVar()  
cb1.set(1) # 1 => selected.  
cb2 = IntVar()  
cb3 = IntVar()
```

Hello Python

País:

Contiente:

Continentes

- ☐ América ☒ Asia  
☐ Africa ☐ Europa

## ❖ Interação com componentes

### ❑ Listbox

- ❑ get, insert, delete
- ❑ curselection
- ❑ size

IPC

```
def num_tarefas():  
    txt_count_tarefas.delete(0, "end")  
    cont = lbox_tarefas.size()  
    txt_count_tarefas.insert(0, str(cont))
```

→ Nº de linhas da ListBox

```
def adicionar():  
    tarefa = txt_tarefa.get()  
    lbox_tarefas.insert("end", tarefa)  
    txt_tarefa.delete(0, "end")  
    num_tarefas()
```

→ Adicionar à ListBox: (posição, conteúdo)

```
def limpar():  
    lbox_tarefas.delete(0, "end")  
    num_tarefas()
```

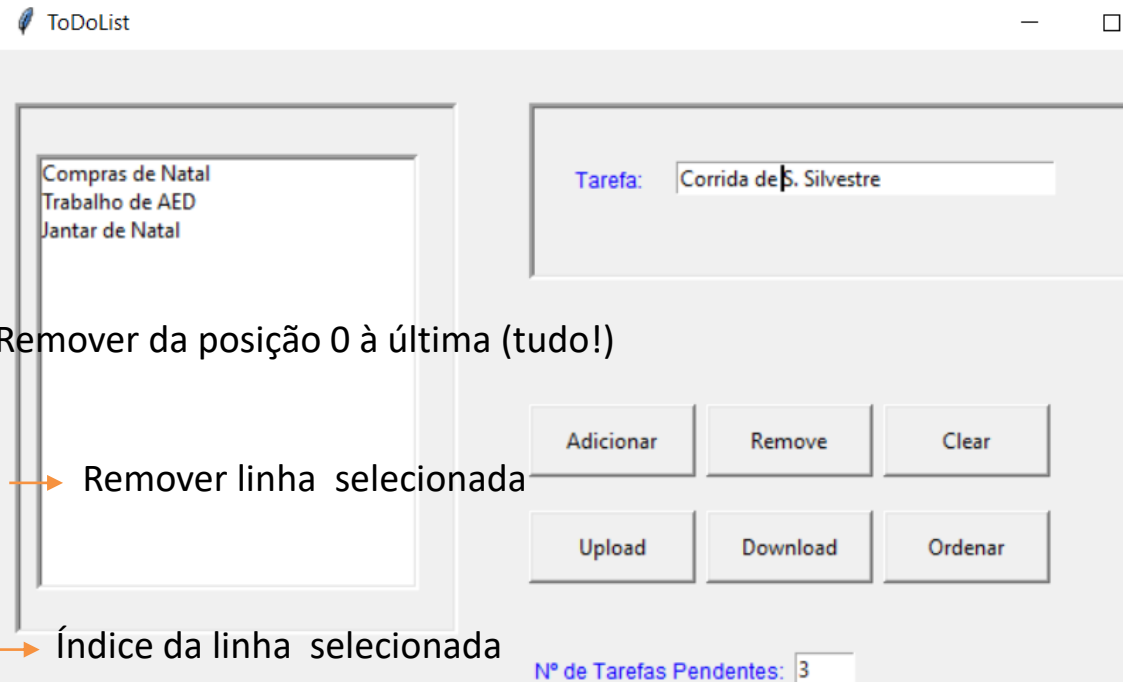
→ Remover da posição 0 à última (tudo!)

```
def remover():  
    lbox_tarefas.delete(lbox_tarefas.curselection())  
    txt_tarefa.delete(0, "end")  
    num_tarefas()
```

→ Remover linha selecionada

```
def selecao_item(event):  
    indice = lbox_tarefas.curselection()  
    texto = lbox_tarefas.get(indice)  
    txt_tarefa.insert(0, texto)
```

→ Índice da linha selecionada



## ❖ Interação com componentes

### ❑ Listbox

- ❑ get, insert, delete
- ❑ curselection
- ❑ size

```
#Listbox
lista=["América", "Asia", "Africa", "Antartida", "Europa", "Oceania"]
lbox_continente=Listbox(window, height=8, selectmode = "single")
for pais in lista:
    lbox_continente.insert(END,pais)
lbox_continente.place(x=300, y=30)
```

