



Mód. 5 - Interfaces gráficas / Tkinter

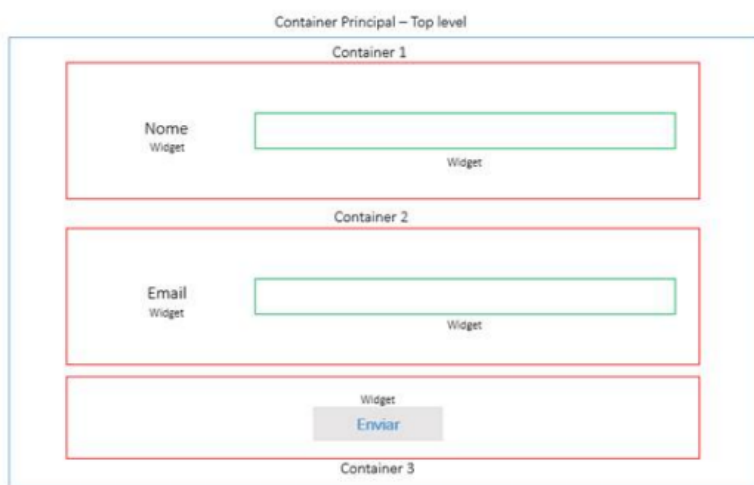
🕒 Created	@January 4, 2023 2:49 PM
📁 Class	Algoritmia e estruturas de dados
🔗 Materials	https://www.pythontutorial.net/tkinter/
✅ Reviewed	<input type="checkbox"/>
🔗 Link explicativo	(https://www.tutorialspoint.com/python/python_gui_programming.htm).

❑ 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...

<https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956>



`window.geometry = ("300x200+30+50")` → width x height + xpos + ypos

→ **Slide 16: centralizar a window de acordo com o tamanho da tela do usuário**

→ slide 19: `window.configure(bg = "blue")` ou `bg = "#d3d3d3"` → RGB(6 dígitos)
background color



Para sair de uma janela:

No command do botão: `command = nomeJanela.destroy()` (destroi a janela de todas as formas)

ou `nomeJanela.quit()` (sai da app)

ou `nomeJanela.withdraw()` esconde a janela dando a oportunidade de abrir ela de novo se for preciso

- **Para sair de uma janela, abrir outra e voltar pra essa:**

1. Cria a window normalmente
2. Cria uma função para abrir a nova janela
3. Dentro da nova função usa o método "`withdraw()`" para fechar a janela antiga mantendo os dados dela "em standby"
4. Cria um botão pra voltar pra página anterior com uma função que chama a página atual:

(* note: pra usar uma função com parametro tem q usar o lambda `command= lambda: goBack(windowMovimentos)`)

```
def goBack(currentWindow):  
    currentWindow.destroy()  
    oldWindow.deiconify() #restaura a janela anterior
```



Pra ver criar um ficheiro se ele nao existir:

```
ficheiro = "../ficha12//files//acessos.txt"

if not os.path.exists(ficheiro):    #cria o ficheiro se ele não existir
    os.mkdir(ficheiro)

# criar a pasta se ela nao existir
pasta = "..\\ficha12\\files"

if not os.path.exists(pasta):    #cria a pasta se ela não existir
    os.mkdir(pasta)
```

Icons

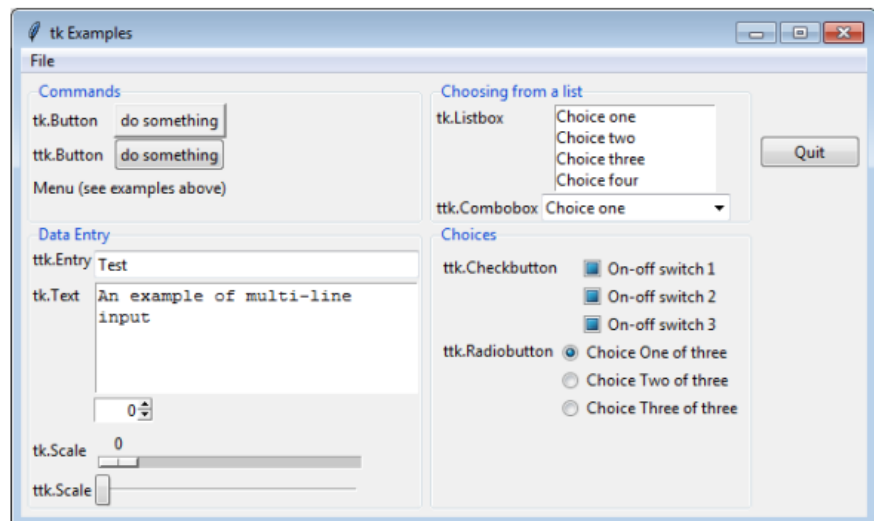
Link de sugestão de site: icon-icons.com

```
window.iconbitmap("link/pathDoIcône.ico")
```

Widgets

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

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



❏ Containers & widgets: alguns exemplos

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

- Posição **dos widgets em um container**:
 - 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)
- Label (slide 26)



Imagem só aceita png

Base para o código:

```
from tkinter import *

window = Tk()
window.geometry("800x500") #(width x height) em px
window.title("Ex 01")
window.resizable(0,0) #colocar 1 se quiser a opcao de redimensionar

screenWidth = window.winfo_screenwidth()
screenHeight = window.winfo_screenheight()

appWidth = 800 #colocar a width e height do app
appHeight = 500

x = (screenWidth/2) - (appWidth/2)
y = (screenHeight/2) - (appHeight/2)
window.geometry(f'{appWidth}x{appHeight}+{int(x)}+{int(y)}')

window.mainloop()
```

com o 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

Para definir onde a janela vai abrir na tela:

window.geometry("width x height + Xpos + Ypos") em px

- Para centralizar a janela na tela:

```
from tkinter import *

window = Tk()
window.geometry("800x500") #(width x height) em px
window.title("Ex 01")
window.resizable(0,0) #colocar 1 se quiser a opcao de redimensionar

screenWidth = window.winfo_screenwidth()
screenHeight = window.winfo_screenheight()

appWidth = 800 #colocar a width e height do app
appHeight = 500
```

```
x = (screenWidth/2) - (appWidth/2)
y = (screenHeight/2) - (appHeight/2)
window.geometry(f'{appWidth}x{appHeight}+{int(x)}+{int(y)}')

window.mainloop()
```

- **Método configure:**

Selecionar cor de fundo (background)

```
window.configure(bg = #f1f1f1)
```

Widgets

- **Label**

state: active (por omissão) ou disabled

textvariable: para associar o conteúdo de uma Label a uma variável

text : legenda do botão

bg : background; fg : cor do texto

font : font name, size;

image : to be displayed instead of text

command : function to be called when clicked

```
#Label
lbl_pais=Label(window, text="País:", fg='red', font=("Helvetica", 9))
lbl_pais.place(x=20, y=30)

lbl_continente=Label(window, text="Contiente:", fg='red', font=("Helvetica", 9))
lbl_continente.place(x=20, y=70)
```

- **Entry**

width: comprimento do componente

bg : background; fg : foreground

font : font name , size

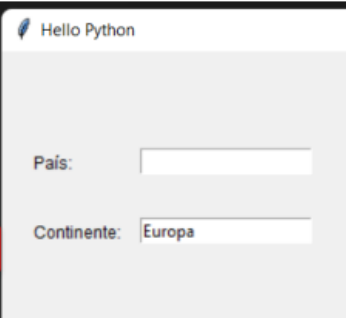
bd : border (2 pixels por defeito).

show : para converter a Entry num campo de password, show = "*".
textvariable: para associar o conteúdo a uma variável



Widget Entry limitado a uma única linha de texto

```
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
```



`widgetEntry.set("textoTextoTexto")` → texto para ficar no lugar da entry enquanto o usuário não adiciona nada. Ex.: “Adicione um texto aqui”

→ **Pra salvar o valor de uma entry ou um widget dentro de uma variável:**
definir o nome da variável antes (`StringVar` ou `IntVar`), colocar nos parâmetros do widget “`variable=...`” ou “`textVariable=...`” e na função coloca `nomeDaVariavel.get()` pra pegar o valor add `nomeDaVariavel.get("1.0", "end")` pra pegar o valor da linha toda

pra pegar o valor de uma entry (`entry.get()`)

```
# pra definir a variavel pra entry
num = StringVar()
entryNumEstudante = Entry(windowMovimentos, width=20, textvariable = num)

# pra pegar o valor dessa variavel
numeroEstudante = num.get()
```

```
entry_nome.delete(0, END)
entry_nome.insert(0, "Novo valor")
```

O método `delete(0, END)` apaga o conteúdo atual da Entry, enquanto o método `insert(0, "Novo valor")` insere o novo valor na posição 0 (ou seja, no início) da Entry.

- **Text**

(mais do que 1 linha de texto)

width: comprimento do componente,

bg : background; fg : foreground

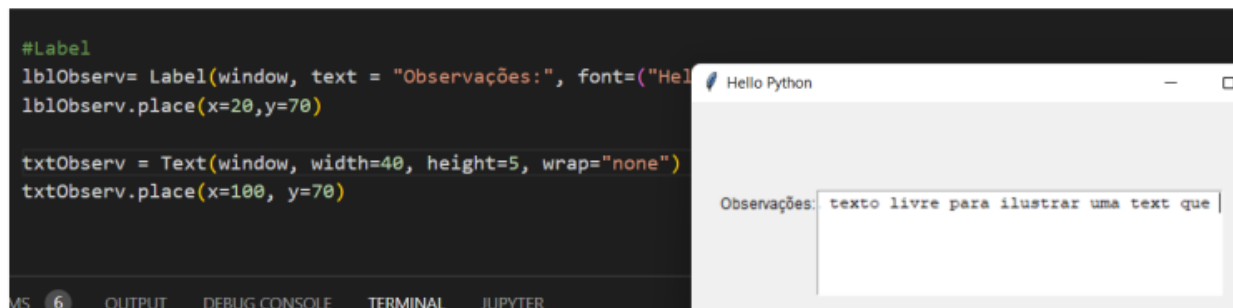
font : font name , size

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"



Para deletar info dentro de uma widget de texto: `nomeWidget.delete("0.0", "end")`

Para adicionar info dentro de uma widget de texto: `nomeWidget.insert("0.0",`

`variavelTxtParaAdd)` → "0,0" é o lugar onde vai inserir a info, pode ser:

"1.0" - linha 1, character 0

"end" – final da Text

"end-1c" – final da Text

sem \n no final da Text

ou user `set` - `linha = "cdfwefw" nomeWidget.get("end", linha)`

Para pegar a info dentro de uma widget de texto: `novaVar = nomeWidget.get("0.0", "end")`

- **Button**

Os botões são usados para criar ações e/ou interações com o usuário. Tkinter oferece diversos parâmetros para personalizar os botões:

- `command` - Uma função que é chamada quando o botão é pressionado.
- `state` - Estado do botão (ativo, desativo). Por padrão, o botão é ativo.
- `text` - Legenda do botão.
- `bg` - Cor de fundo do botão.
- `fg` - Cor do texto do botão.
- `font` - Fonte do texto do botão.
- `image` - Imagem a ser exibida no lugar do texto.

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

- `width` - Largura do botão em pixels.
- `relief` - Raised, groove, sunken(sombra parecendo que o botão tá pra baixo) ou flat(sem borda nenhuma).
- `bitmap` - Colocar um ícone no lugar do texto ou da imagem (os que já fazem parte do algoritmo: error, hourglass, info, question, warning, questhead)

- **Imagem**

Para add uma imagem

```
global imgPrincipal
imgPrincipal = PhotoImage(file = "../ficha12//images//img-gestao.png")
labelImgPrincipal = Label(window, image=imgPrincipal, width = 600)
labelImgPrincipal.place(x=0, y=0)
```

- **Data e hora**

* Y maiusculo = ano completo ex 2023

H e M maiusculo = indica a hora e minuto, minusculo são outras formas de demonstrar meses

```
from datetime import datetime

dataHora= datetime.now()


data= dataHora.strftime("%d/%m/%Y")
hora =dataHora.strftime("%H:%M")

print(data)
print(hora)
```

Link base:

Python Dates

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.

 https://www.w3schools.com/python/python_datetime.asp



<https://www.alura.com.br/artigos/lidando-com-datas-e-horarios-no-python>

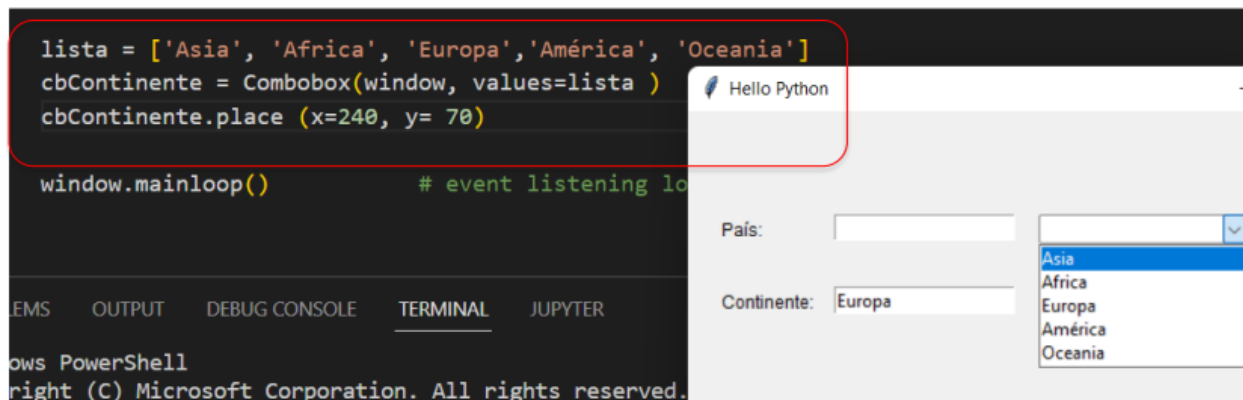
- **Combobox**

Widget definido no módulo ttk (importar) `from tkinter.ttk import Combobox`

Preenche uma lista dropdown a partir de uma lista, ou uma coleção de dados
Só é possível selecionar um item da Combobox

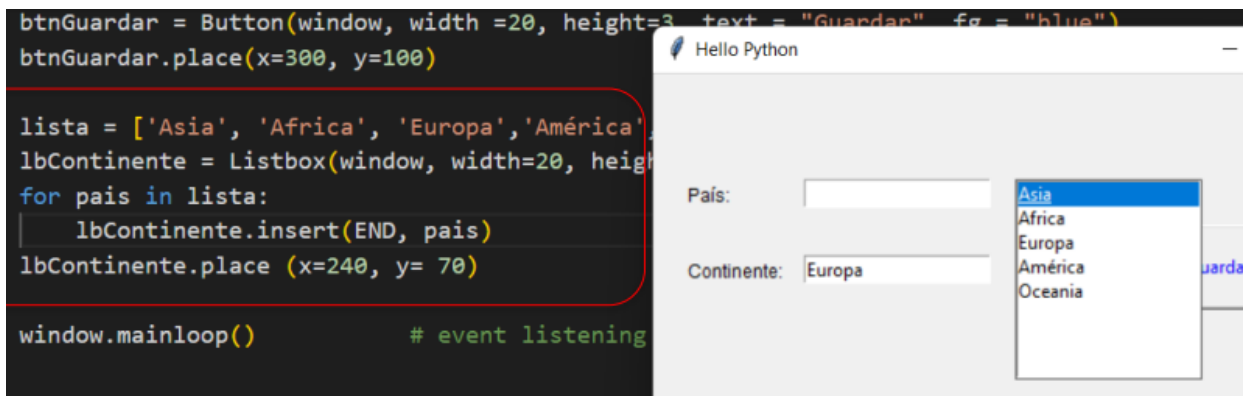
Selectmode: qts items podem ser selecionados:

- ☐ 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)



- **Listbox**

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



→ Para pegar a linha ativa do listbox

```

def remover():
    """Apaga a tarefa selecionada da listbox"""
    pos = tarefas.curselection() # obtém o índice da linha ativa
                                # OU: lbboxTarefas.focus()

    if len(pos) > 0:
        # Remove o item selecionado da Listbox
        tarefas.delete(tarefas.curselection()) # índice do item selecionado
        with open(".\\ficha11\\ex03\\tarefas.txt", "r", encoding="utf-8") as f:
            lines = f.readlines()
        with open(".\\ficha11\\ex03\\tarefas.txt", "w", encoding="utf-8") as f:
            for i, line in enumerate(lines):
                if i != pos[0]:
                    f.write(line)

```

```
else:
    messagebox.showerror("info", "Não existem dados seleccionados")
```

→ Para apagar tudo do listBox

```
def remover():
    """Apaga tudo da listBox"""
    tarefas.delete(0, "end")
```

The screenshot shows a Python Tkinter application titled "ToDoList". On the left, there is a listbox containing three items: "Compras de Natal", "Trabalho de AED", and "Jantar de Natal". To the right of the listbox is a text entry field labeled "Tarefa:" containing the text "Corrida de S. Silvestre". Below the text entry are six buttons: "Adicionar", "Remove", "Clear", "Upload", "Download", and "Ordenar". At the bottom right, there is a label "Nº de Tarefas Pendentes:" followed by a text entry field showing the number "3".

Annotations with arrows point to specific parts of the code and the GUI:

- An arrow points from the text "Nº de linhas da ListBox" to the line `cont = lbox_tarefas.size()` in the `num_tarefas()` function.
- An arrow points from the text "Adicionar à ListBox: (posição, conteúdo)" to the line `lbox_tarefas.insert("end", tarefa)` in the `adicionar()` function.
- An arrow points from the text "Remover da posição 0 à última (tudo!)" to the line `lbox_tarefas.delete(0, "end")` in the `limpar()` function.
- An arrow points from the text "Remover linha seleccionada" to the line `lbox_tarefas.delete(lbox_tarefas.curselection())` in the `remover()` function.
- An arrow points from the text "Índice da linha seleccionada" to the line `indice = lbox_tarefas.curselection()` in the `selecao_item(event)` function.

remove, adiciona, seleciona item listBox

• Radiobutton

seleção exclusiva (apenas 1 opção pode estar seleccionada)

font

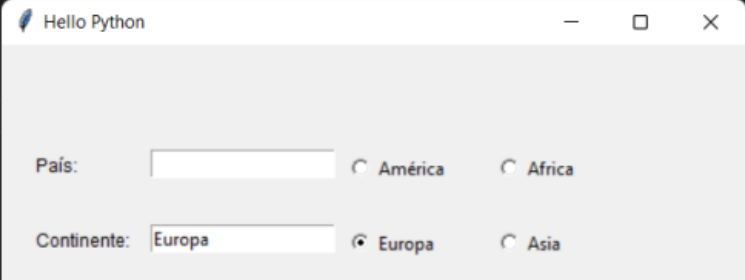
state

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

```



`variableRadiobtn.get()` → pega o valor associado a variavel desse radiobutton

```


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

```

```

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

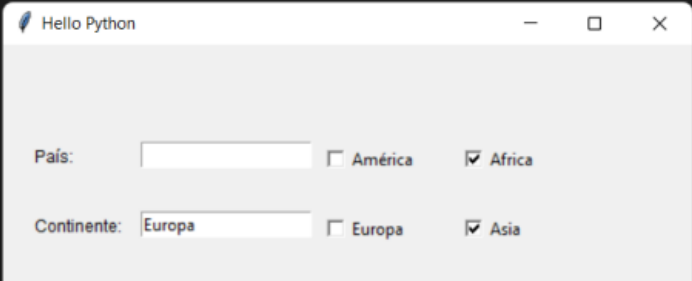
```



- **Checkbutton**

podemos selecionar mais do que uma opção
variavel tem que ser uma pra cada

```
47 #CheckButtons
48 checkVar1 = IntVar()
49 checkVar2 = IntVar()
50 checkVar3 = IntVar()
51 checkVar4 = IntVar()
52 checkVar2.set(1)      # 1 - significa que o checkbutton está ativo
53 checkVar4.set(1)      # Neste exemplo, os checkbuttons 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
```



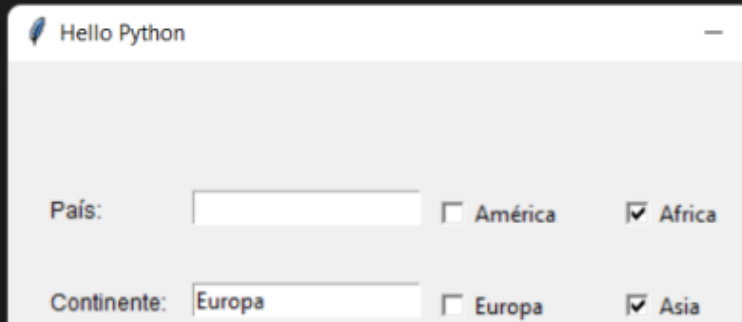
checkVar.GET()

→ variável de controlo que quando roda o programa o checkbutton já aparece clicado

`nomeButton.set(1)` - 1 para selecionado e 0 para não selecionado #TEM Q SER O NOME DA VARIABLE N TO BOTAO

```
#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)

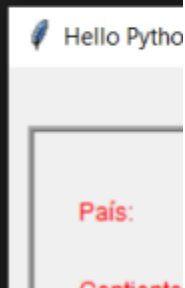
window.mainloop()
```



Se variável associada a cada Checkbutton == 1:
significa que selecionei o objeto (→ aí dá pra usar o valor em uma def, entry, ...)

```
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)
```

```
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")
```



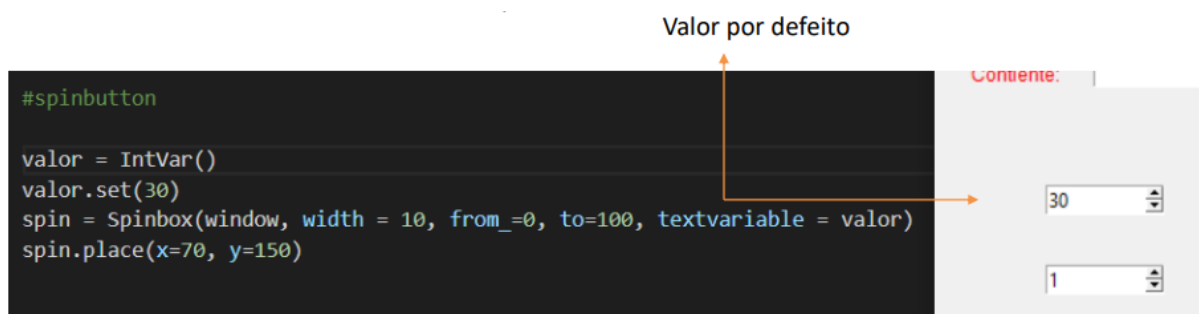
- **Spinbox**

Número com setinha pra ir pra cima ou pra baixo

Select a value from a fixed range of fixed options with up and down arrows. It can be used to set a variable to a certain value, or to execute a function when the value is changed.

- `textvariable` : used to associate the Spinbox's value to a variable, which can then be used to access the value in a program. The
- `command` : used to specify a function that will be called whenever the value of the Spinbox is changed.
- `from_` and `to` : used to specify the range of values the Spinbox can select from.
- `increment` parameter is used to specify the amount by which the value of the Spinbox will be incremented or decremented when the up or down arrows are pressed (default value = 1)
- `state` used to control whether the Spinbox is enabled or disabled.
- `values` associar a uma lista ou dicionário com todos os números que podem ser escolhidos no spinbox

Para atribuir um valor por defeito:

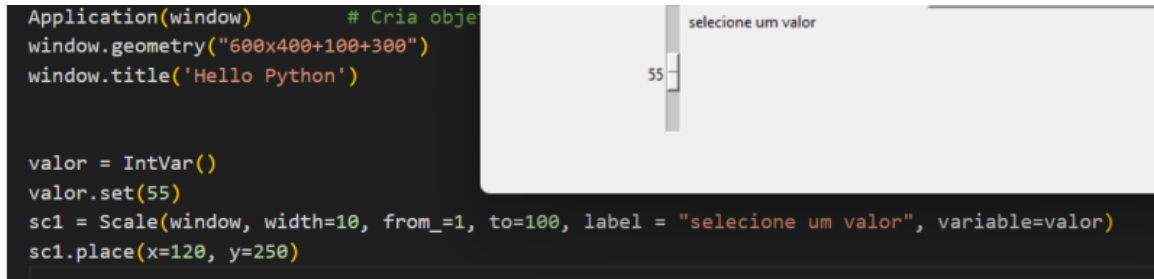


- **Scale**

Selecionar um número dentro de um numero fixo de números, mas arrastando

- `orient` - vertical(default) ou horizontal

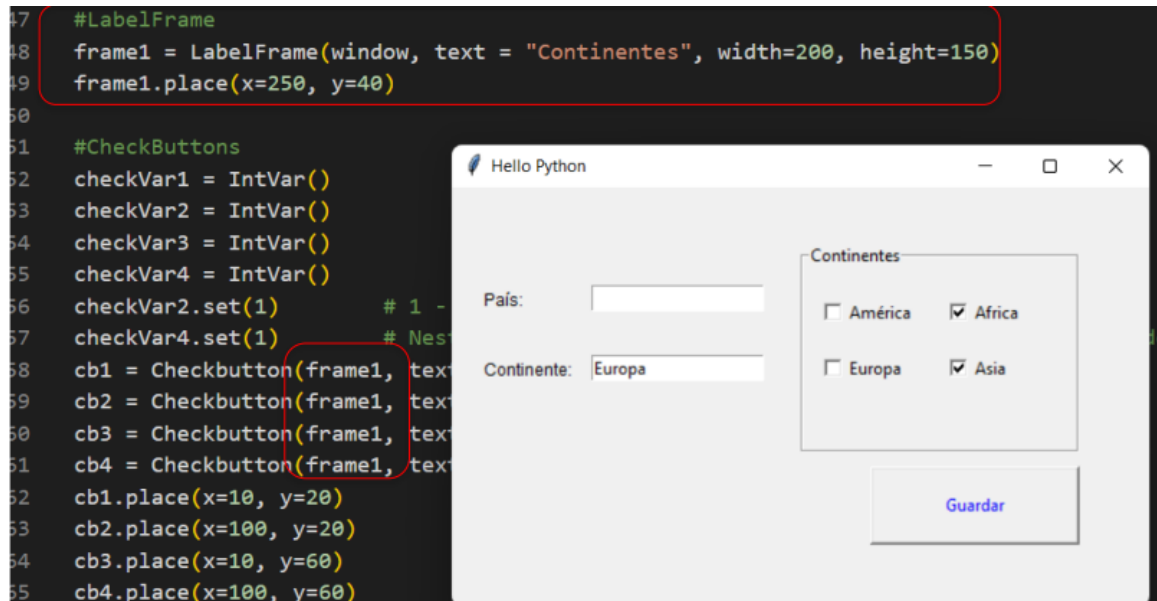
- `label` - texto parametro dentro do proprio widget, que aparece em cima da escala
- `widget`
- `from, to`
- `variable`



- **LabelFrame**

é um container, cujo objetivo é agrupar componentes em layouts mais complexos

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

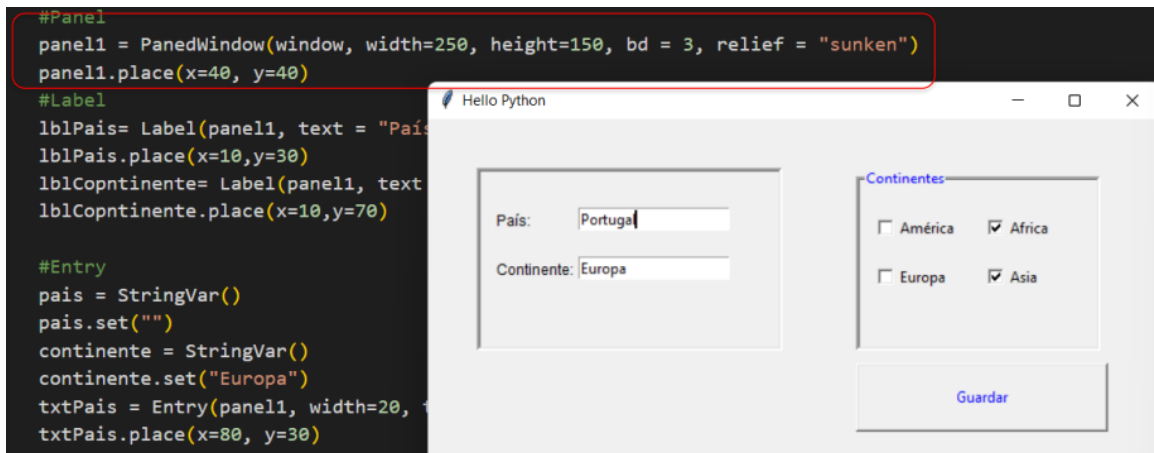


Note que os checkbuttons estão sendo colocados dentro da própria frame, e não por cima dela na window principal

- **PanedWindow**

é um container que permite organizar o layout da aplicação de uma forma lógica

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



Note que a label e a entry estão sendo colocados dentro da própria panedWindow, e não por cima dela na window principal

- **Canvas**

Container especial, que suporta imagens ou desenhos geométricos

Formatos suportados: png e gif

anchor: define onde a imagem vai ficar (pode ser: center, n, s, w, e, nw, ne, sw, se)

- Primeiro cria o canvas e depois adiciona a imagem no canvas (usando o parametro `variavel.create_image(x,y, image = varDaImg)`)

Treeview

Permite apresentar dados em formato agregado, de tabela

Componente do módulo ttk (que deve ser importado)!!!

1. Definir as colunas do componente:

```
tree = ttk.Treeview(window, selectmode= "browse", columns = ("Nome", "Número",  
"Série"), show = "headings")  
tree.place(x= xx, y= yy)
```

→ o nome dessas colunas é o nome interno, que vai ficar no código, não que vai aparecer na interface

2. Definir os nomes, larguras e alinhamentos das colunas na interface do programa, e adicionar os headings

a. `tree.column("coluna", width= xx; anchor = c/e/w)`

```
tree.column("Nome", width = 100, anchor = c) # c - centro; e - direita; w  
- esquerda  
tree.column("Número", width = 100, anchor = c)  
tree.column("Série", width = 100, anchor = c)
```

b. `tree.heading("colunaAreceberOheading", text ="textoDoHeading")`

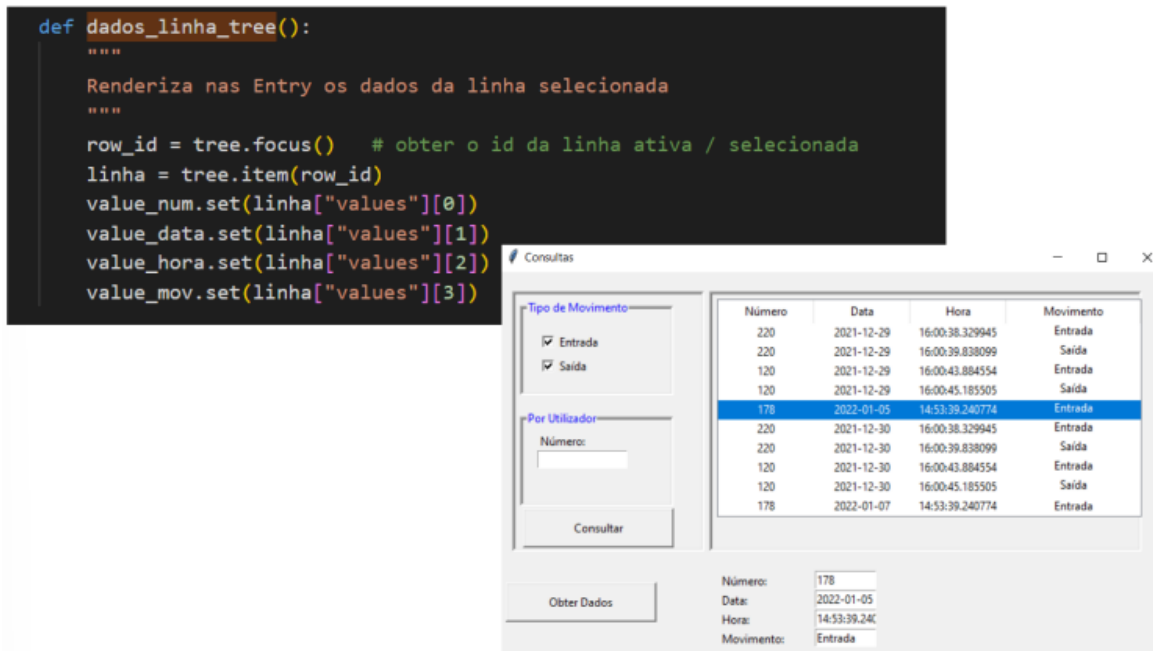
```
tree.heading("Nome", text ="Nome")  
tree.heading("Nome", text ="Número")  
tree.heading("Nome", text ="Série")
```

3. Adicionar os dados da componente Treeview (insert)

`tree.insert("", "end", values = (campos[0], campos[1], [2]))`

- a. `""` → vazio por omissão, indica a ordem hierárquica a inserir os dados
- b. `"end"` → indica em que lugar a linha deve ser inserida na tabela. Se tiver um número, esse número será a linha da tabela na qual a informação será inserida. "0" é na primeira linha; "end" é no final
- c. `values` → são os dados a serem inseridos na Treeview

Para obter dados da linha ativa ou selecionada da TreeView (`tree.focus()`):



Para remover dados da Treeview:

Remover tudo:

```
tree.delete(*tree.get_children())
```

Obter cada linha e remover uma a uma:

```
linhas = tree.get_children()
for linha in linhas:
    tree.delete(linha)
```

Para seleccionar linha da treeview:

```
pos = tree.selection()
```

Devolve o índice da linha seleccionada