



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 09 – TKINTER I

O QUE IREMOS APRENDER

01

FRAMEWORK

02

GUI

03

WIDGETS

04

MÉTODOS DE POSICIONAMENTO

04

MÃOS NO CÓDIGO



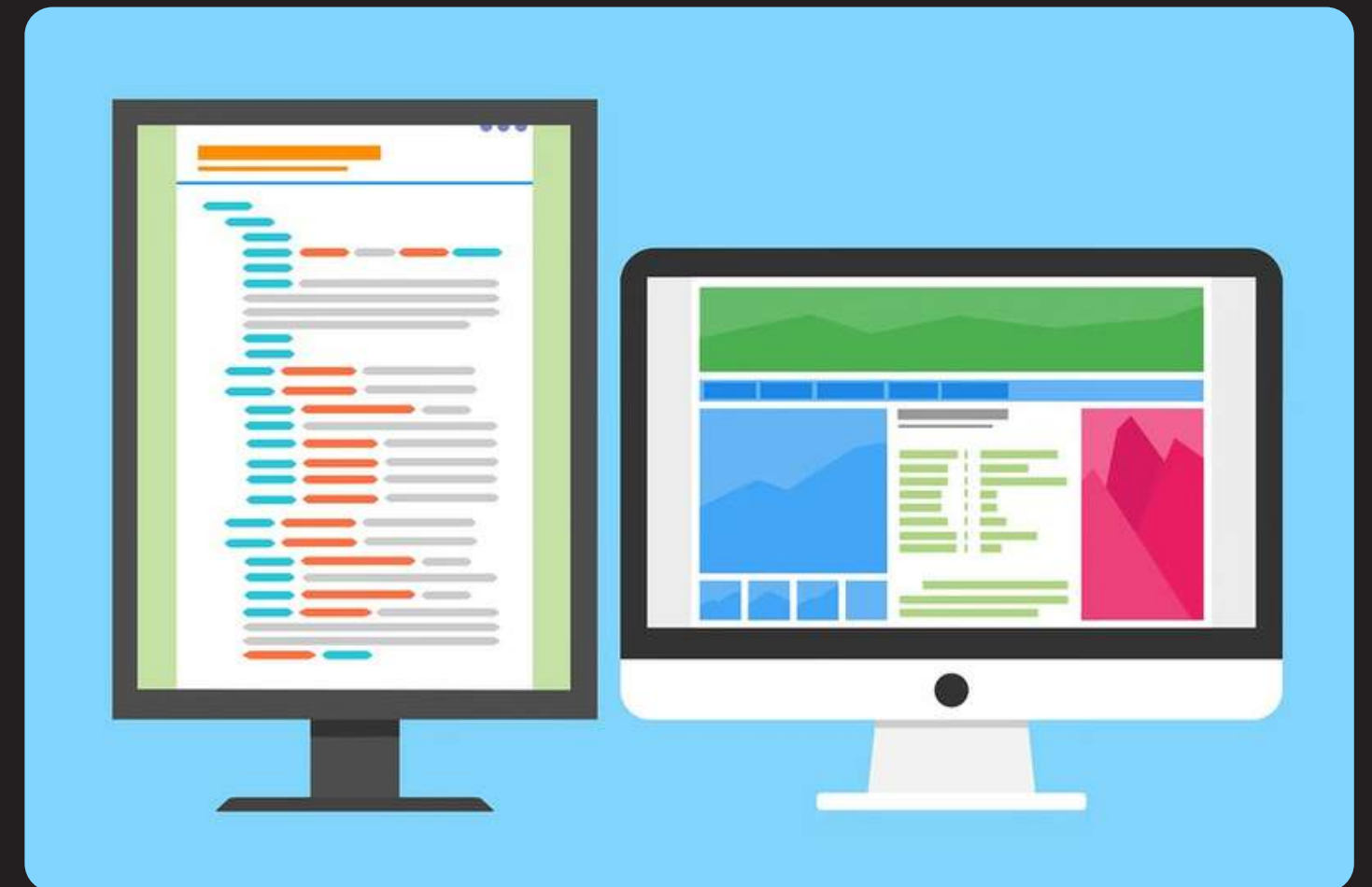
Tkinter I

Nesta aula vamos aprender sobre o único **framework** para criação de **interfaces gráficas** que está embutido na biblioteca padrão da linguagem Python. Este framework é Multiplataforma (Windows, macOS, Linux, etc...) e os elementos visuais são renderizados usando elementos nativos do sistema operacional.



O que é Framework

Um framework é um conjunto de ferramentas, bibliotecas e convenções que fornecem uma estrutura para o desenvolvimento de software. Existem vários tipos de frameworks, desde frameworks de desenvolvimento web até frameworks de interface gráfica (Local).

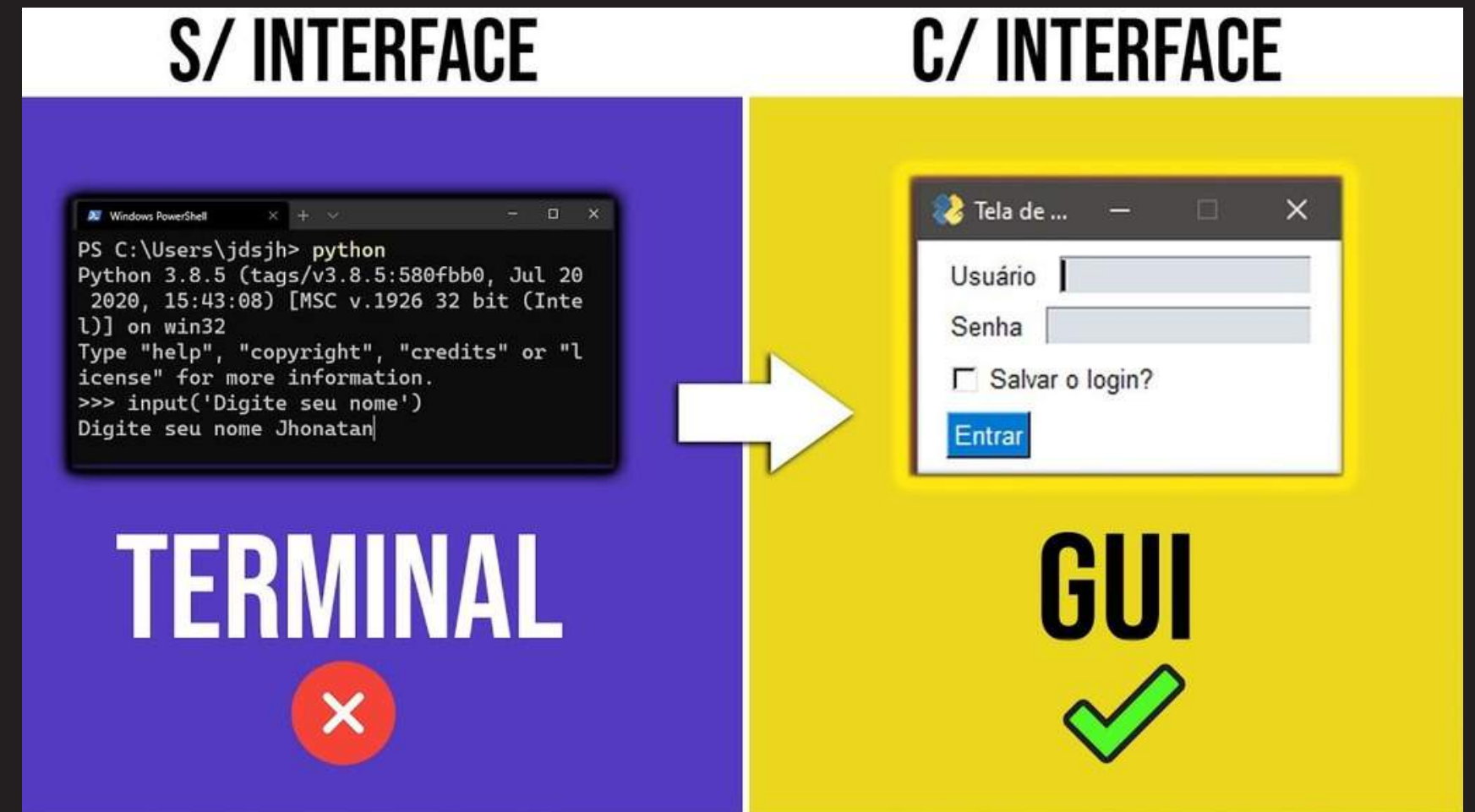


Tkinter

Já desejou criar uma interface visual altamente interativa e dinâmica que interage de forma perfeita com o código em segundo plano? Se a resposta for sim, você veio ao lugar certo. Nesta aula, vamos mergulhar fundo no mundo do Tkinter, uma incrível biblioteca de interface gráfica para Python que vai te proporcionar todas as ferramentas necessárias para criar aplicativos visualmente atraentes e funcionais.


GUI (Graphical UserInterface)

Uma GUI é uma **forma de interação** entre os usuários e os programas de computador que utilizam elementos gráficos, como janelas, botões, menus e outros controles visuais. As GUIs tornam os aplicativos mais intuitivos e fáceis de usar, permitindo que os usuários interajam com o software de forma visual e sem a necessidade de comandos complexos.



O que é Tkinter

Tkinter é uma biblioteca Python usada para **criar interfaces gráficas de usuário** (GUI, do inglês "Graphical User Interface"). Ela fornece um conjunto de ferramentas e widgets que permitem criar aplicativos de desktop com uma interface gráfica. Tkinter é frequentemente usada para criar janelas, botões, caixas de diálogo, campos de entrada, e outros elementos visuais em aplicativos Python.



```
1 from tkinter import *
2 '''
3 1. Importando toda biblioteca tkinter para nosso projeto.
4 2. Atribuindo a classe onde contém todos os métodos para
5 construção de nossa GUI.
6 3. O ultimo passo é usar o método mainloop() Isso permite
7 que a janela seja atualizada continuamente e responda aos eventos.
8 '''
9 janela = Tk()
10
11 janela.mainloop()
```

Primeira janela com o Tkinter



```
1  from tkinter import *  
2  
3  janela = Tk()  
4  
5  janela.mainloop()
```

Passo 1: Importe o módulo Tkinter

Passo 2: Crie uma Janela principal

Passo 3: Inicie um loop de evento

Widgets

Os **widgets** seriam elementos que interagem com o usuário, na usabilidade da experiência do usuário como: botões, caixas de seleção para opções (checkbox), eventos ao clicar do botão, labels, imagens e outros...

Alguns exemplos de widgets:

- labels(Label): Usado para exibir texto na tela.
- caixas de texto(Entry): Widget de entrada de dados que permite apenas uma única linha de texto.

Widgets

- botões(Button): Um botão que pode conter texto e realizar uma ação quando clicado.
- checkbox(Checkbutton): Uma caixa de seleção para implementar ações do tipo liga/desliga.
- radiobutton(Radiobutton): Uma caixa de seleção para implementar uma ação de muitas seleções.

Widgets

Label (Rótulo):

- **text**: Define o texto exibido no rótulo.
- **font**: Define a fonte do texto.
- **foreground** ou **fg**: Define a cor do texto.

```
1 from tkinter import *
2
3 janela = Tk()
4
5 # Label (Rótulo)
6 label = Label(janela, text="Isso é um rótulo", font=("Arial", 12), fg="#6A5ACD")
7 label.pack()
8
9 janela.mainloop()
```

Widgets

Button (Botão):

- **text**: Define o texto exibido no botão.
- **command**: Especifica a função a ser executada quando o botão é clicado.

```
1 from tkinter import *
2
3 janela = Tk()
4
5 # Label (Rótulo)
6 label = Label(janela, text="Isso é um rótulo", font=("Arial",12))
7 label.pack()
8
9 # Button (botão)
10 def button_click():
11     label.config(text="Botão clicado!")
12
13 button = Button(janela, text="Clique em mim!", command=button_click)
14 button.pack()
15
16 janela.mainloop()
```

Widgets

Entry (Campo de Entrada):


- **textvariable**: Associa uma variável de controle a este campo de entrada.
- **show**: Esconde os caracteres digitados (útil para senhas).

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 #bg é para mudar o background color do campo de entrada
6
7 entry_text = tk.StringVar()
8 entry = tk.Entry(root, textvariable=entry_text, bg='red')
9 entry.pack()
10
11 root.mainloop()
```

Widgets

Text (Caixa de Texto):

- **height e width:** Define as dimensões da caixa de texto.
- **wrap:** Define como o texto é quebrado (por exemplo, 'word' ou 'char').



```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 # Text (Caixa de Texto)
6
7 text = tk.Text(root, height=5, width=30, wrap='word')
8 text.pack()
9
10 text.insert(tk.END, "Este é um exemplo de caixa de texto.")
11
12 root.mainloop()
```


Widgets

Frame (Quadro):

- **relief**: Define o estilo de relevo da moldura (por exemplo, 'flat', 'raised', 'sunken').
- **borderwidth ou bd**: Define a largura da borda.



```
1  import tkinter as tk
2
3  root = tk.Tk()
4
5  #Frame (Quadro)
6
7  frame = tk.Frame(root, relief=tk.RAISED, borderwidth=2)
8  frame.pack()
9
10 root.mainloop()
```

Widgets

Canvas (Tela):

- **width e height:** Define as dimensões da tela.
- Várias opções para desenhar formas, texto e imagens.

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 # Canvas (Tela)
6
7 canvas = tk.Canvas (root, width=200, height=100)
8 canvas.pack()
9
10 canvas.create_rectangle(10, 10, 100, 60, fill="blue")
11
12 root.mainloop()
```

Widgets

Checkbox (Botão de Verificação):

- **variable**: Associa uma variável de controle a este botão de verificação.
- **text**: Define o rótulo do botão de verificação.

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 #Checkbox (Botão de Verificação)
6
7 check_var = tk.BooleanVar()
8
9 checkbox = tk.Checkbutton(root, text="Aceitar Termos", variable=check_var)
10 checkbox.pack()
11
12 root.mainloop()
```

Widgets

Radiobutton (Botão de Opção):

- **variable**: Associa uma variável de controle a este botão de opção.
- **value**: Define o valor associado a este botão de opção.
- **text**: Define o rótulo do botão de opção.

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 #Radiobutton (Botão de Opção)
6
7 radio_var = tk.StringVar()
8
9 radiol = tk.Radiobutton(root, text="Opção 1", variable=radio_var, value="opcao1")
10 radio2 = tk.Radiobutton(root, text="Opção 2", variable=radio_var, value="opcao2")
11 radiol.pack()
12 radio2.pack()
13
14 root.mainloop()
```

Widgets

Scale (Controle Deslizante):

- **from_ e to:** Define os valores mínimo e máximo.
- **orient:** Define a orientação (horizontal ou vertical).
- **variable:** Associa uma variável de controle a este controle deslizante.

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 #Scale (Controle Deslizante)
6
7 scale_var = tk.DoubleVar()
8 scale = tk.Scale(root, from_=8, to=100, orient=tk.HORIZONTAL, variable=scale_var)
9 scale.pack()
10
11 root.mainloop()
```

Widgets

Menu (Menu):

- Opções de configuração para itens de menu, como label e command.

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 # Menu (Menu)
6 menu = tk.Menu(root)
7 root.config(menu=menu)
8 file_menu = tk.Menu(menu)
9 menu.add_cascade(label="Arquivo", menu=file_menu)
10 file_menu.add_command(label="Abrir")
11 file_menu.add_command(label="Salvar")
12 file_menu.add_separator()
13 file_menu.add_command(label="Sair")
14
15 root.mainloop()
```


Métodos de Posicionamento

Os métodos de posicionamento `pack()`, `grid()` e `place()` são utilizados para **posicionar** e **organizar** os elementos gráficos (widgets) dentro de um ou mais contêineres (frame, janela GUI).

pack - O método `pack()` é usado para posicionar os widgets em um contêiner usando o conceito de empacotamento (packing). Ele organiza os widgets em uma linha ou coluna, preenchendo o espaço disponível de acordo com as configurações definidas. Alguns parâmetros comuns para o método `pack()` incluem:

Métodos de Posicionamento

side: Define o lado do contêiner onde o widget será colocado, por exemplo (top, bottom, left, right).

padx e pady: Define o preenchimento interno (espaço em pixels) a ser adicionado ao redor do widget.



```
1 entrada.pack(side="left", padx=10, pady=5)
2 enviar.pack(side="left", padx=10, pady=5)
3
```

Nesse exemplo, os dois widgets são empacotados lado a lado à esquerda.

Métodos de Posicionamento

grid - O método `grid()` é usado para posicionar os widgets em uma grade (grid) dentro do contêiner. Ele organiza os widgets em linhas e colunas, permitindo um controle mais preciso da posição e do tamanho dos widgets. Alguns parâmetros comuns para o método `grid()` incluem:

row e column: Define a posição da célula da grade onde o widget será colocado.

Métodos de Posicionamento

rowspan e colspan: Define o número de linhas e colunas que o widget deve ocupar.

sticky: Define como o widget se expande para preencher a célula (por exemplo, "N" para norte, "E" para leste, "S" para sul, "W" para oeste ou combinações como "NE" para nordeste).



```
1 titulo.grid(row=0, column=0, colspan=2, padx=10, pady=5)
2 nome_label.grid(row=1, column=0, padx=10, pady=5)
3 entrada.grid(row=1, column=1, padx=10, pady=5)
4 enviar.grid(row=2, column=0, colspan=2, padx=10, pady=5)
```

Neste exemplo, os labels (titulo e nome_label) estão na primeira linha, a entry (entrada) está na mesma linha, mas na coluna seguinte, e o button (enviar) está na terceira linha e ocupa as duas colunas disponíveis. O uso do rowspan e colspan permite que o botão ocupe duas colunas no grid.

Métodos de Posicionamento

place - O método `place()` é usado para posicionar os widgets usando coordenadas absolutas no contêiner. Ele permite especificar a posição exata (em pixels) do widget no contêiner. Alguns parâmetros comuns para o método `place()` incluem:

x e y: Define as coordenadas x e y onde o widget será colocado.

width e height: Define a largura e altura do widget em pixels.

Métodos de Posicionamento

relx e rely: representam valores relativos entre 0 e 1, indicando a posição do widget em relação às dimensões do contêiner. muito usado para obter widgets responsivos.



```
1 titulo.place(x=200, y=50)
2 nome_label.place(x=100, y=100)
3 entrada.place(x=300, y=100)
4 enviar.place(x=200, y=150)
```

Neste exemplo, usamos os parâmetros x e y no método `place()` para definir as coordenadas absolutas dos widgets na janela. Você pode ajustar os valores de x e y para posicionar os widgets onde desejar.

Ajuste cada widget de acordo com as coordenadas que desejar.

ATIVIDADE PRÁTICA 1

Cadastro de Alunos (com exibição no terminal) Crie um programa utilizando o Tkinter que permita ao usuário cadastrar alunos. O programa deve solicitar o nome, idade e nota do aluno por meio de caixas de entrada (Entry) e exibir um botão para cadastrar. Ao cadastrar um aluno, os dados devem ser exibidos no terminal.

ATIVIDADE PRÁTICA 2

Calculadora Simples (com exibição no terminal)
Desenvolva uma calculadora simples utilizando o Tkinter que permita ao usuário realizar operações matemáticas básicas, como adição, subtração, multiplicação e divisão. O programa deve exibir botões para os números e as operações, e ao realizar o cálculo, o resultado deve ser exibido no terminal.

ATIVIDADE PRÁTICA 3

Conversor de Temperatura (com exibição no terminal)
Crie um conversor de temperatura utilizando o Tkinter que permita ao usuário converter uma temperatura de Celsius para Fahrenheit ou vice-versa. O programa deve fornecer caixas de entrada (Entry) para o usuário inserir a temperatura de origem e exibir um botão para realizar a conversão. Ao realizar a conversão, o resultado deve ser exibido no terminal.

DESAFIO PRÁTICO

Relógio digital

Crie uma aplicação de relógio digital que exiba a hora atual na interface gráfica. A hora deve ser atualizada em tempo real.

A interface gráfica deve incluir um rótulo que exiba a hora atual.

Dica: Use o widget Label para exibir a hora e a biblioteca time para obter a hora atual. Crie uma função que atualize a hora e use a função after do Tkinter para agendar atualizações periódicas.

Material Complementar

- Exploração: Não tenha medo de explorar e testar diferentes códigos. A experimentação é uma grande aliada da aprendizagem.
 - Perguntas: Faça perguntas, seja curioso! Entender o "porquê" das coisas ajuda a consolidar o conhecimento.
 - Revisão: Revise o que aprendeu, tente explicar para si mesmo ou para outras pessoas. Ensinar é uma ótima forma de aprender.
- Prática: A prática leva à perfeição. Quanto mais
- exercícios fizer, mais fácil será lembrar e entender os conceitos.



SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 10 DE
PYTHON. TKINTER II



IN

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Módulo ttk

O módulo **ttk** (Themed Tkinter) é um aprimoramento do Tkinter. Ele fornece uma série de widgets de interface gráfica adicionais com uma aparência mais moderna e consistente em diferentes plataformas. A principal vantagem do **ttk** é a disponibilidade de widgets temáticos. Esses widgets têm uma aparência nativa mais elegante e sofisticada, adaptando-se ao estilo visual do sistema operacional em uso.

```
function decorate(event) {  
  event = event || window.event;  
  var target = event.target || event.srcElement;  
  if (target && (target.getAttribute('action') || target  
    ga(function (tracker) {  
      var linkerParam = tracker.get('linkerParam');  
      document.cookie = '_shopify_ga=' + linkerParam + '  
    }));  
  
  listener(window, 'load', function(){  
    for (i=0; i < document.forms.length; i++) {  
      if (document.forms[i].getAttribute('action')  
        of('/cart') >= 0) {  
        submit, decorate);  
      }  
    }  
  })  
}
```



INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 09 – TKINTER I