

Python - Programação GUI

Python oferece várias opções para desenvolver interfaces gráficas de usuário (GUIs). Os recursos mais importantes estão listados abaixo.

- **Tkinter** - Tkinter é a interface Python para o kit de ferramentas Tk GUI fornecido com Python. Veríamos essa opção neste capítulo.
- **wxPython** - Esta é uma interface Python de código aberto para o kit de ferramentas GUI wxWidgets. Você pode encontrar um tutorial completo sobre WxPython [aqui](#).
- **PyQt** - Esta também é uma interface Python para uma popular biblioteca Qt GUI multiplataforma. TutorialsPoint tem um tutorial muito bom sobre PyQt5 [aqui](#).
- **PyGTK** - PyGTK é um conjunto de wrappers escritos em Python e C para a biblioteca GTK + GUI. O tutorial completo do PyGTK está disponível [aqui](#).
- **PySimpleGUI** - PySimpleGui é uma biblioteca GUI de plataforma cruzada de código aberto para Python. Seu objetivo é fornecer uma API uniforme para a criação de GUIs de desktop baseadas nos kits de ferramentas Tkinter, PySide e WxPython do Python. Para um tutorial detalhado do PySimpleGUI, clique [aqui](#).
- **Pygame** - Pygame é uma biblioteca Python popular usada para desenvolver videogames. É um wrapper gratuito, de código aberto e multiplataforma em torno da Simple DirectMedia Library (SDL). Para um tutorial abrangente sobre Pygame, [visite este link](#).
- **Jython** - Jython é uma porta Python para Java, que fornece aos scripts Python acesso contínuo às bibliotecas de classes Java na máquina local <http://www.jython.org>.

Existem muitas outras interfaces disponíveis, que você pode encontrar na rede.

Programação Tkinter

Tkinter é a biblioteca GUI padrão para Python. Python, quando combinado com Tkinter, fornece uma maneira rápida e fácil de criar aplicativos GUI. Tkinter fornece uma interface poderosa orientada a objetos para o kit de ferramentas Tk GUI.

O pacote tkinter inclui os seguintes módulos -

- **Tkinter** - Módulo principal do Tkinter.
- **tkinter.colorchooser** - Caixa de diálogo para permitir ao usuário escolher uma cor.
- **tkinter.commondialog** - Classe base para os diálogos definidos nos demais módulos listados aqui.
- **tkinter.filedialog** - Diálogos comuns para permitir ao usuário especificar um arquivo para abrir ou salvar.

- **tkinter.font** - Utilitários para ajudar a trabalhar com fontes.
- **tkinter.messagebox** - Acesso às caixas de diálogo padrão do Tk.
- **tkinter.scrolledtext** - Widget de texto com barra de rolagem vertical integrada.
- **tkinter.simpledialog** - Diálogos básicos e funções convenientes.
- **tkinter.ttk** - Conjunto de widgets temáticos introduzido no Tk 8.5, fornecendo alternativas modernas para muitos dos widgets clássicos no módulo principal do tkinter.

Criar um aplicativo GUI usando Tkinter é uma tarefa fácil. Tudo que você precisa fazer é executar as seguintes etapas.

- Importe o módulo Tkinter.
- Crie a janela principal do aplicativo GUI.
- Adicione um ou mais dos widgets mencionados acima ao aplicativo GUI.
- Entre no loop de eventos principal para agir em relação a cada evento acionado pelo usuário.

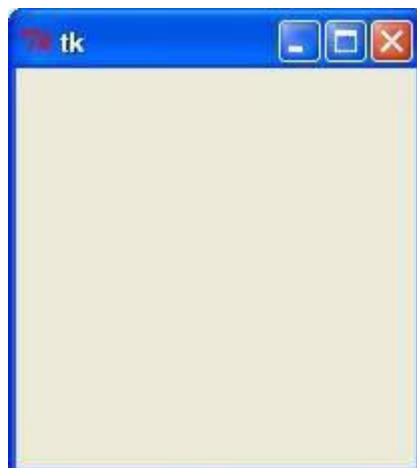
Exemplo

```
# note that module name has changed from Tkinter in Python 2
# to tkinter in Python 3

import tkinter
top = tkinter.Tk()

# Code to add widgets will go here...
top.mainloop()
```

Isso criaria a seguinte janela -



Quando o programa se torna mais complexo, o uso de uma abordagem de programação orientada a objetos torna o código mais organizado.

```
import tkinter as tk

class App(tk.Tk):
    def __init__(self):
        super().__init__()

app = App()
app.mainloop()
```

Widgets do Tkinter

Tkinter fornece vários controles, como botões, rótulos e caixas de texto usados em um aplicativo GUI. Esses controles são comumente chamados de widgets.

Existem atualmente 15 tipos de widgets no Tkinter. Apresentamos esses widgets, bem como uma breve descrição na tabela a seguir -

Sr. Não.	Operador e Descrição
1	Botão O widget Button é usado para exibir os botões em seu aplicativo.
2	Tela O widget Canvas é usado para desenhar formas, como linhas, ovais, polígonos e retângulos, em seu aplicativo.
3	Botão de verificação O widget Checkbutton é usado para exibir diversas opções como caixas de seleção. O usuário pode selecionar várias opções ao mesmo tempo.
4	Entrada O widget Entrada é usado para exibir um campo de texto de linha única para aceitar valores de um usuário.
5	Quadro O widget Quadro é usado como widget contêiner para organizar outros widgets.
6	Rótulo O widget Label é usado para fornecer uma legenda de linha única para outros widgets. Também pode conter imagens.
7	Caixa de listagem O widget Listbox é usado para fornecer uma lista de opções para um usuário.
8	Botão de menu O widget Menubutton é usado para exibir menus em seu aplicativo.
9	Cardápio

O widget Menu é usado para fornecer vários comandos a um usuário. Esses comandos estão contidos no Menubutton.

Mensagem

O widget Mensagem é usado para exibir campos de texto multilinha para aceitar valores de um usuário.

Botao de radio

O widget Radiobutton é usado para exibir diversas opções como botões de opção. O usuário pode selecionar apenas uma opção por vez.

Escala

O widget Escala é usado para fornecer um widget deslizante.

Barra de rolagem

O widget Barra de rolagem é usado para adicionar capacidade de rolagem a vários widgets, como caixas de listagem.

Texto

O widget Texto é usado para exibir texto em várias linhas.

Nível superior

O widget Toplevel é usado para fornecer um contêiner de janela separado.

Caixa giratória

O widget Spinbox é uma variante do widget Tkinter Entry padrão, que pode ser usado para selecionar um número fixo de valores.

Janela Panorâmica

Um PanedWindow é um widget contêiner que pode conter qualquer número de painéis, organizados horizontal ou verticalmente.

LabelFrame

Um labelframe é um widget de contêiner simples. Seu objetivo principal é atuar como espaçador ou contêiner para layouts de janelas complexos.

tkMessageBox

Este módulo é usado para exibir caixas de mensagens em seus aplicativos.

Vamos estudar esses widgets em detalhes.

Atributos Padrão

Vejamos como alguns dos atributos comuns, como tamanhos, cores e fontes, são especificados.

- **Dimensões**
- **Cores**

- Fontes
- Âncoras
- Estilos de relevo
- Bitmaps
- Cursores

Vamos estudá-los brevemente -

Gerenciamento de Geometria

Todos os widgets Tkinter têm acesso aos métodos específicos de gerenciamento de geometria, que têm o objetivo de organizar os widgets em toda a área do widget pai. Tkinter expõe as seguintes classes de gerenciamento de geometria: pack, grid e place.

- **The pack() Method** - Este gerenciador de geometria organiza widgets em blocos antes de colocá-los no widget pai.
- **The grid() Method** - Este gerenciador de geometria organiza widgets em uma estrutura semelhante a uma tabela no widget pai.
- **The place() Method** - Este gerenciador de geometria organiza widgets colocando-os em uma posição específica no widget pai.

Vamos estudar brevemente os métodos de gerenciamento de geometria -

DE ANÚNCIOS

SimpleDialog

O módulo `simpledialog` no pacote `tkinter` inclui uma classe de diálogo e funções de conveniência para aceitar entradas do usuário por meio de um diálogo modal. Consiste em um rótulo, um widget de entrada e dois botões Ok e Cancelar. Essas funções são -

- **askfloat(title, prompt, **kw)** - Aceita um número de ponto flutuante.
- **askinteger(title, prompt, **kw)** - Aceita uma entrada inteira.
- **askstring(title, prompt, **kw)** - Aceita uma entrada de texto do usuário.

As três funções acima fornecem caixas de diálogo que solicitam ao usuário que insira um valor do tipo desejado. Se Ok for pressionado, a entrada será retornada; se Cancel for pressionado, None será retornado.

perguntarinteiro

```
from tkinter.simpledialog import askinteger
from tkinter import *
from tkinter import messagebox

top = Tk()

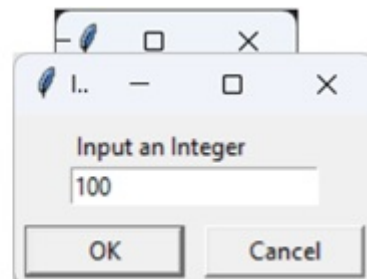
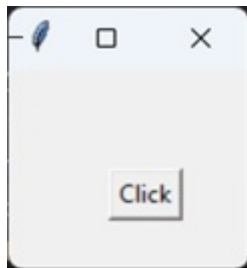
top.geometry("100x100")

def show():
    num = askinteger("Input", "Input an Integer")
    print(num)

B = Button(top, text ="Click", command = show)
B.place(x=50,y=50)

top.mainloop()
```

Ele produzirá a seguinte **saída** -



perguntarfloat

```
from tkinter.simpledialog import askfloat
from tkinter import *
top = Tk()

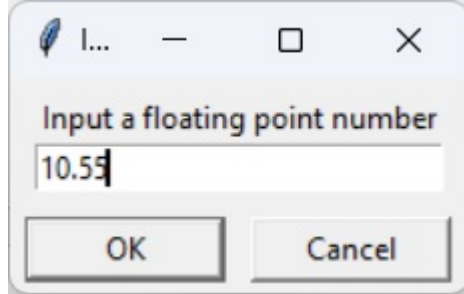
top.geometry("100x100")

def show():
    num = askfloat("Input", "Input a floating point number")
    print(num)

B = Button(top, text ="Click", command = show)
B.place(x=50,y=50)

top.mainloop()
```

Ele produzirá a seguinte **saída** -



pergunta

```
from tkinter.simpledialog import askstring
from tkinter import *

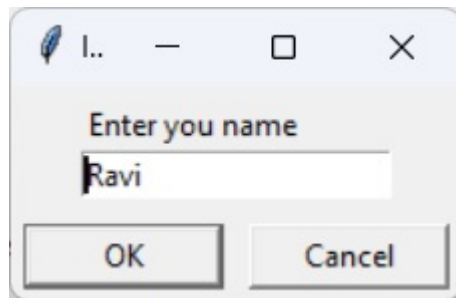
top = Tk()

top.geometry("100x100")
def show():
    name = askstring("Input", "Enter you name")
    print(name)

B = Button(top, text ="Click", command = show)
B.place(x=50,y=50)

top.mainloop()
```

Ele produzirá a seguinte **saída** -



DE ANÚNCIOS

O Módulo FileDialog

O módulo filedialog no pacote Tkinter inclui uma classe FileDialog. Ele também define funções de conveniência que permitem ao usuário executar atividades de abrir arquivos, salvar arquivos e abrir diretórios.

- `filedialog.asksaveasfilename()`
- `arquivodialog.asksaveasfile()`

- `filedialog.askopennome do arquivo()`
- `arquivodialog.askopenfile()`
- `arquivodialog.askdirectory()`
- `filedialog.askopennomesdearquivos()`
- `arquivodialog.askopenfiles()`

perguntar arquivo aberto

Esta função permite ao usuário escolher um arquivo desejado do sistema de arquivos. A janela de diálogo do arquivo possui botões Abrir e Cancelar. O nome do arquivo junto com seu caminho é retornado quando Ok é pressionado, Nenhum se Cancel for pressionado.

```
from tkinter.filedialog import askopenfile
from tkinter import *

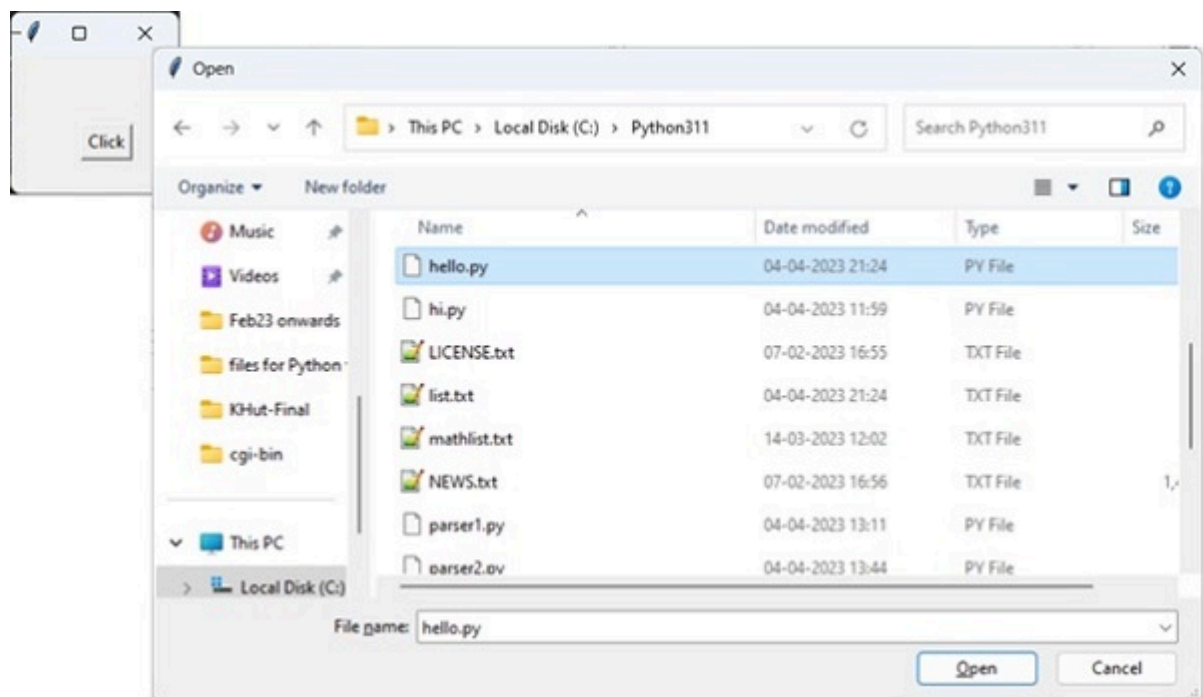
top = Tk()

top.geometry("100x100")
def show():
    filename = askopenfile()
    print(filename)

B = Button(top, text ="Click", command = show)
B.place(x=50,y=50)

top.mainloop()
```

Ele produzirá a seguinte **saída** -



Seletor de cores

O módulo `colorchooser` incluído no pacote `tkinter` tem a característica de permitir ao usuário escolher um objeto de cor desejado através da caixa de diálogo de cores. A função `askcolor()` apresenta uma caixa de diálogo de cores com amostras de cores predefinidas e facilidade para escolher cores personalizadas definindo valores RGB. A caixa de diálogo retorna uma tupla de valores RGB da cor escolhida, bem como seu valor hexadecimal.

```
from tkinter.colorchooser import askcolor
from tkinter import *

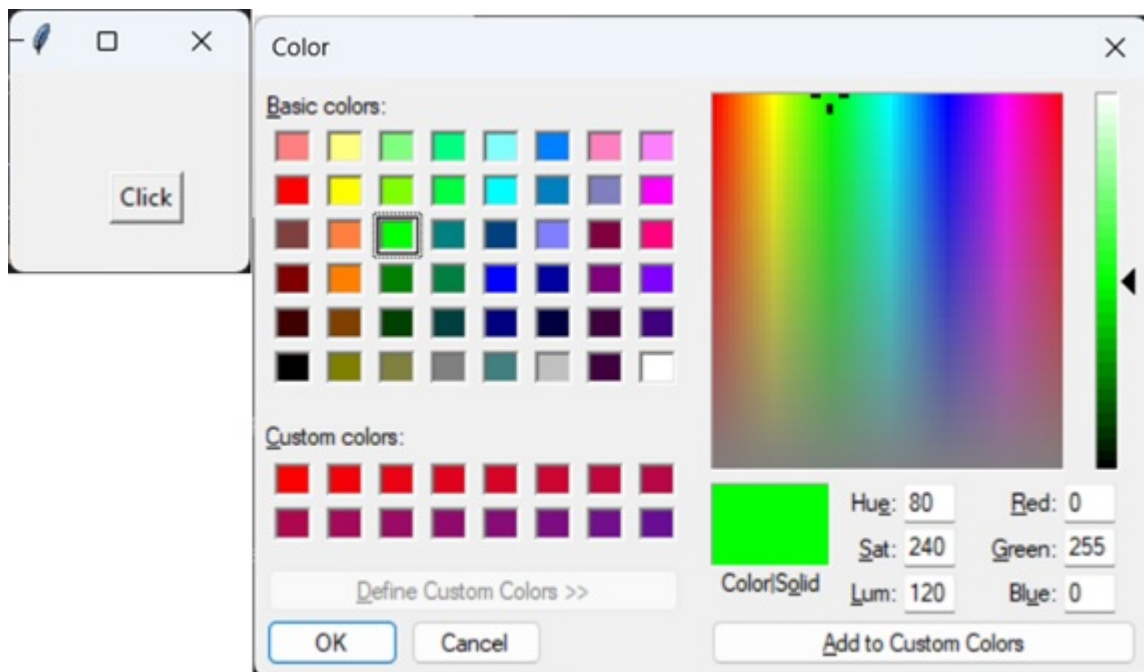
top = Tk()

top.geometry("100x100")
def show():
    color = askcolor()
    print(color)

B = Button(top, text ="Click", command = show)
B.place(x=50,y=50)

top.mainloop()
```

Ele produzirá a seguinte **saída** -



```
((0, 255, 0), '#00ff00')
```

módulo ttk

O termo ttk vem de widgets com tema Tk. O módulo ttk foi introduzido a partir do Tk 8.5. Ele oferece benefícios adicionais, incluindo renderização de fonte suavizada no X11 e transparência de janela. Ele fornece suporte a temas e estilos para Tkinter.

O módulo ttk vem com 18 widgets, dos quais 12 já estão presentes no Tkinter. A importação do ttk substitui esses widgets por novos, projetados para ter uma aparência melhor e mais moderna em todas as plataformas.

Os 6 novos widgets no ttk são Combobox, Separator, Sizegrip, Treeview, Notebook e ProgressBar.

Para substituir os widgets Tk básicos, a importação deve seguir a importação Tk -

```
from tkinter import *  
from tkinter.ttk import *
```

Os widgets Tk originais são automaticamente substituídos pelos widgets tkinter.ttk. Eles são Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale e Scrollbar.

Novos widgets que oferecem uma melhor aparência em todas as plataformas; no entanto, os widgets de substituição não são totalmente compatíveis. A principal diferença é que opções de widgets como "fg", "bg" e outras relacionadas ao estilo do widget não estão mais presentes nos widgets Ttk. Em vez disso, use a classe ttk.Style para efeitos de estilo aprimorados.

Os novos widgets no módulo ttk são -

- **Notebook** - Este widget gerencia uma coleção de "abas" entre as quais você pode alternar, alterando a janela exibida no momento.
- **ProgressBar** - Este widget é usado para mostrar o progresso ou o processo de carregamento por meio de animações.
- **Separator** - Usado para separar diferentes widgets usando uma linha separadora.
- **Treeview** - Este widget é usado para agrupar itens em uma hierarquia semelhante a uma árvore. Cada item possui um rótulo textual, uma imagem opcional e uma lista opcional de valores de dados.
- **ComboBox** - Usado para criar uma lista suspensa de opções na qual o usuário pode selecionar uma.
- **Sizegrip** - Cria uma pequena alça próxima ao canto inferior direito da tela, que pode ser usada para redimensionar a janela.

Widget de caixa de combinação

O Python ttk Combobox apresenta uma lista suspensa de opções e as exibe uma de cada vez. É uma subclasse do widget Entry. Conseqüentemente, ele herda muitas opções e métodos da classe Entry.

Sintaxe

```
from tkinter import ttk

Combo = ttk.Combobox(master, values.....)
```

A função get() para recuperar o valor atual do Combobox.

Exemplo

```
from tkinter import *
from tkinter import ttk

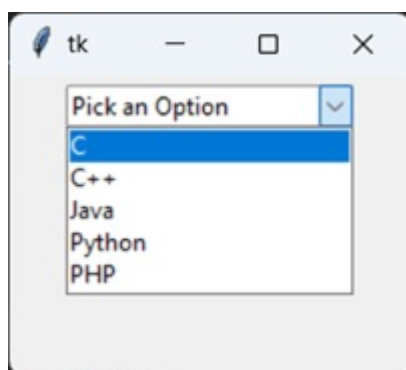
top = Tk()
top.geometry("200x150")

frame = Frame(top)
frame.pack()

langs = ["C", "C++", "Java",
         "Python", "PHP"]

Combo = ttk.Combobox(frame, values = langs)
Combo.set("Pick an Option")
Combo.pack(padx = 5, pady = 5)
top.mainloop()
```

Ele produzirá a seguinte **saída** -



Barra de progresso

O widget ttk ProgressBar e como ele pode ser usado para criar telas de carregamento ou mostrar o progresso de uma tarefa atual.

Sintaxe

```
ttk.Progressbar(parent, orient, length, mode)
```

Parâmetros

- **Parent** - O contêiner no qual o ProgressBar será colocado, como root ou um quadro Tkinter.
- **Orient** - Define a orientação do ProgressBar, que pode ser vertical ou horizontal.
- **Length** - Define a largura do ProgressBar assumindo um valor inteiro.
- **Mode** - Existem duas opções para este parâmetro, determinado e indeterminado.

Exemplo

O código abaixo cria uma barra de progresso com três botões vinculados a três funções diferentes.

A primeira função incrementa o “valor” ou “progresso” na barra de progresso em 20. Isso é feito com a função `step()` que usa um valor inteiro para alterar a quantidade de progresso. (O padrão é 1,0)

A segunda função diminui o “valor” ou “progresso” na barra de progresso em 20.

A terceira função imprime o nível de progresso atual na barra de progresso.

```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()
frame= ttk.Frame(root)

def increment():
    progressBar.step(20)

def decrement():
    progressBar.step(-20)

def display():
    print(progressBar["value"])

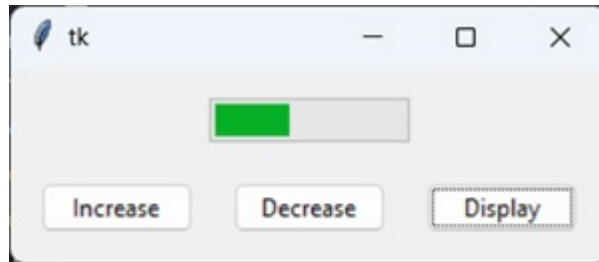
progressBar= ttk.Progressbar(frame, mode='determinate')
progressBar.pack(padx = 10, pady = 10)

button= ttk.Button(frame, text= "Increase", command= increment)
button.pack(padx = 10, pady = 10, side = tk.LEFT)
```

```
button= ttk.Button(frame, text= "Decrease", command= decrement)
button.pack(padx = 10, pady = 10, side = tk.LEFT)
button= ttk.Button(frame, text= "Display", command= display)
button.pack(padx = 10, pady = 10, side = tk.LEFT)

frame.pack(padx = 5, pady = 5)
root.mainloop()
```

Ele produzirá a seguinte **saída** -



Caderno

O módulo Tkinter ttk possui um novo widget útil chamado Notebook. É uma coleção de contêineres (por exemplo, frames) que possuem muitos widgets como filhos dentro.

Cada "guia" ou "janela" possui um ID de guia associado a ela, que é usado para determinar para qual guia alternar.

Você pode alternar entre esses contêineres como faria em um editor de texto normal.

Sintaxe

```
notebook = ttk.Notebook(master, *options)
```

Exemplo

Neste exemplo, adicione 3 janelas ao nosso widget Notebook de duas maneiras diferentes. O primeiro método envolve a função `add()`, que simplesmente anexa uma nova guia ao final. O outro método é a função `insert()` que pode ser usada para adicionar uma guia a uma posição específica.

A função `add()` leva um parâmetro obrigatório que é o widget contêiner a ser adicionado, e o restante são parâmetros opcionais como texto (texto a ser exibido como título da guia), imagem e composto.

A função `insert()` requer um `tab_id`, que define o local onde deve ser inserido. O `tab_id` pode ser um valor de índice ou uma string literal como "end", que o anexará ao final.

```
import tkinter as tk
from tkinter import ttk
```

```

root = tk.Tk()
nb = ttk.Notebook(root)

# Frame 1 and 2
frame1 = ttk.Frame(nb)
frame2 = ttk.Frame(nb)

label1 = ttk.Label(frame1, text = "This is Window One")
label1.pack(pady = 50, padx = 20)
label2 = ttk.Label(frame2, text = "This is Window Two")
label2.pack(pady = 50, padx = 20)

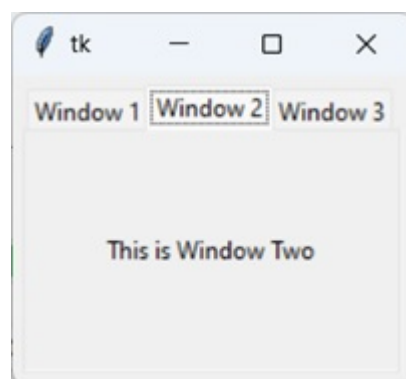
frame1.pack(fill= tk.BOTH, expand=True)
frame2.pack(fill= tk.BOTH, expand=True)
nb.add(frame1, text = "Window 1")
nb.add(frame2, text = "Window 2")

frame3 = ttk.Frame(nb)
label3 = ttk.Label(frame3, text = "This is Window Three")
label3.pack(pady = 50, padx = 20)
frame3.pack(fill= tk.BOTH, expand=True)
nb.insert("end", frame3, text = "Window 3")
nb.pack(padx = 5, pady = 5, expand = True)

root.mainloop()

```

Ele produzirá a seguinte **saída** -



Vista em árvore

O widget Treeview é usado para exibir itens de maneira tabular ou hierárquica. Possui suporte para recursos como criação de linhas e colunas para itens, além de permitir que itens também tenham filhos, levando a um formato hierárquico.

Sintaxe

```
tree = ttk.Treeview(container, **options)
```

Opções

Sr. Não.	Opção e descrição
1	colunas Uma lista de nomes de colunas
2	colunas de exibição Uma lista de identificadores de coluna (índices simbólicos ou inteiros) especificando quais colunas de dados são exibidas e a ordem em que aparecem, ou a string "#all".
3	altura O número de linhas visíveis.
4	preenchimento Especifica o preenchimento interno do widget. Pode ser um número inteiro ou uma lista de 4 valores.
5	modo de seleção Um entre "estendido", "navegar" ou "nenhum". Se definido como "estendido" (padrão), vários itens poderão ser selecionados. Se "navegar", apenas um item poderá ser selecionado por vez. Se for "nenhum", a seleção não pode ser alterada pelo usuário.
6	mostrar Uma lista contendo zero ou mais dos seguintes valores, especificando quais elementos da árvore serão exibidos. O padrão é "títulos de árvore", ou seja, mostrar todos os elementos.

Exemplo

Neste exemplo, criaremos um widget Treeview ttk simples e preencheremos alguns dados nele. Já temos alguns dados armazenados em uma lista que serão lidos e adicionados ao widget Treeview em nossa função read_data().

Primeiro precisamos definir uma lista/tupla de nomes de colunas. Deixamos de fora a coluna "Nome" porque já existe uma coluna (padrão) com nome em branco.

Em seguida, atribuímos essa lista/tupla à opção de colunas no Treeview, seguido pela definição dos "títulos", onde a coluna é a coluna real, enquanto o título é apenas o título da coluna que aparece quando o widget é exibido. Damos um nome a cada coluna. "#0" é o nome da coluna padrão.

A função tree.insert() possui os seguintes parâmetros -

- **Parent** - que é deixado como uma string vazia se não houver nenhuma.
- **Position** - onde queremos adicionar o novo item. Para anexar, use tk.END
- **Iid** - que é o ID do item usado para rastrear posteriormente o item em questão.
- **Text** - ao qual atribuiremos o primeiro valor da lista (o nome).

Valor que passaremos aos outros 2 valores que obtivemos da lista.

O código completo

```
import tkinter as tk
import tkinter.ttk as ttk
from tkinter import simpledialog

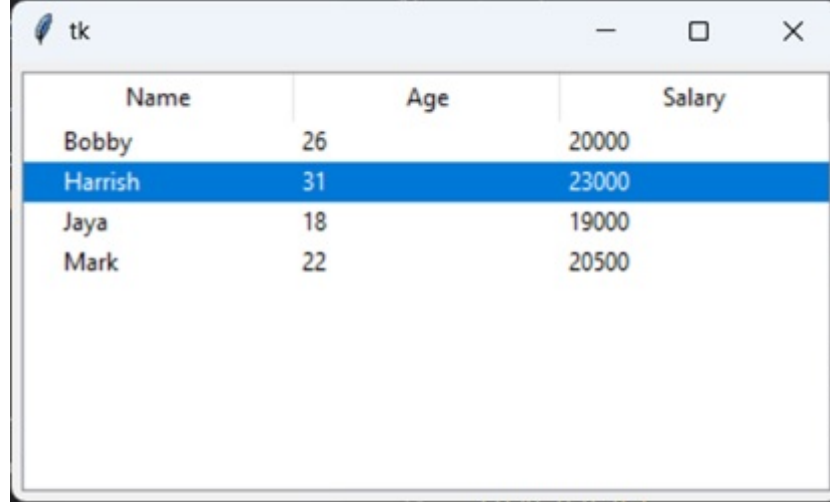
root = tk.Tk()
data = [
    ["Bobby", 26, 20000],
    ["Harrish", 31, 23000],
    ["Jaya", 18, 19000],
    ["Mark", 22, 20500],
]
index=0
def read_data():
    for index, line in enumerate(data):
        tree.insert('', tk.END, iid = index,
                    text = line[0], values = line[1:])
columns = ("age", "salary")

tree= ttk.Treeview(root, columns=columns ,height = 20)
tree.pack(padx = 5, pady = 5)

tree.heading('#0', text='Name')
tree.heading('age', text='Age')
tree.heading('salary', text='Salary')

read_data()
root.mainloop()
```

Ele produzirá a seguinte **saída** -



Name	Age	Salary
Bobby	26	20000
Harrish	31	23000
Jaya	18	19000
Mark	22	20500

TamanhoGrip

O widget Sizegrip é basicamente uma pequena alça em forma de seta que normalmente é colocada no canto inferior direito da tela. Arrastar o Sizegrip pela tela também redimensiona o contêiner ao qual ele está anexado.

Sintaxe

```
sizegrip = ttk.Sizegrip(parent, **options)
```

Exemplo

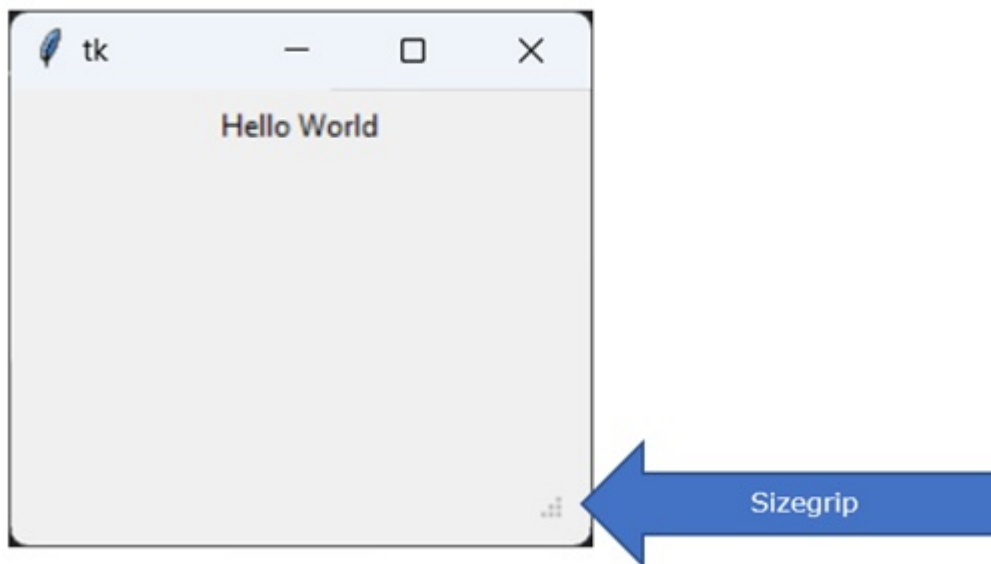
```
import tkinter as tk
import tkinter.ttk as ttk

root = tk.Tk()
root.geometry("100x100")

frame = ttk.Frame(root)
label = ttk.Label(root, text = "Hello World")
label.pack(padx = 5, pady = 5)
sizegrip = ttk.Sizegrip(frame)
sizegrip.pack(expand = True, fill = tk.BOTH, anchor = tk.SE)
frame.pack(padx = 10, pady = 10, expand = True, fill = tk.BOTH)

root.mainloop()
```

Ele produzirá a seguinte **saída** -



Separador

O widget ttk Separator é um widget muito simples, que tem apenas um propósito: ajudar a "separar" widgets em grupos/partições desenhando uma linha entre eles. Podemos alterar a orientação desta linha (separador) para horizontal ou vertical e alterar seu comprimento/altura.

Sintaxe

```
separator = ttk.Separator(parent, **options)
```

O "orient", que pode ser tk.VERTICAL ou tk.HORIZONTAL, para um separador vertical e horizontal respectivamente.

Exemplo

Aqui criamos dois widgets Label e, em seguida, criamos um separador horizontal entre eles.

```
import tkinter as tk
import tkinter.ttk as ttk

root = tk.Tk()
root.geometry("200x150")

frame = ttk.Frame(root)

label = ttk.Label(frame, text = "Hello World")
label.pack(padx = 5)

separator = ttk.Separator(frame, orient= tk.HORIZONTAL)
separator.pack(expand = True, fill = tk.X)

label = ttk.Label(frame, text = "Welcome To Tutorialspoint")
```

```
label.pack(padx = 5)
```

```
frame.pack(padx = 10, pady = 50, expand = True, fill = tk.BOTH)
```

```
root.mainloop()
```

Ele produzirá a seguinte **saída** -

