# Widgets básicos

Este capítulo apresenta os widgets Tk básicos que você encontrará em praticamente qualquer interface de usuário: frames, rótulos, botões, botões de verificação, botões de opção, entradas e caixas de combinação. No final, você saberá como usar todos os widgets de que precisa para um tipo de formulário de preenchimento típico de interface de usuário.

## Frame (Quadro, ou Armação)

Um **quadro** é um widget exibido apenas como um retângulo simples. Os quadros são usados principalmente como um contêiner para outros widgets, que estão sob o controle de um gerenciador de geometria, como uma grade.

Como criar um frame:

nomeObjetoFrame = ttk.Frame(‘componente pai’)

Os quadros podem ter várias opções de configuração diferentes que podem alterar a forma como são exibidos:

#### **Tamanho Solicitado**

Como qualquer outro widget, após a criação, ele é adicionado à interface do usuário por meio de um gerenciador de geometria (pai). Normalmente, o tamanho que o quadro solicitará do gerenciador de geometria será determinado pelo tamanho e layout de quaisquer widgets contidos nele (que estão sob o controle do gerenciador de geometria que gerencia o conteúdo do próprio quadro).

#### **Preenchimento - Padding**

A opção de configuração "padding" é usada para solicitar espaço extra dentro do widget; dessa forma, se você estiver colocando outros widgets dentro do quadro, haverá uma pequena margem ao redor. Um único número especifica o mesmo preenchimento em toda a volta, uma lista de dois números permite especificar o preenchimento horizontal e depois o vertical, e uma lista de quatro números permite especificar o preenchimento esquerdo, superior, direito e inferior, nessa ordem.

Exempo de como configurar um padding um espaço interno dentro de um frame:

nomeObejtoFrame['padding'] = (5,10)

#### **Borders - Fronteiras**

Você pode exibir uma borda ao redor do widget de quadro; você vê muito isso onde pode ter uma parte da interface do usuário parecendo "afundada" ou "elevada" em relação ao seu entorno. Para fazer isso, você precisa definir a opção de configuração "borderwidth" o padrão é 0, portanto, sem borda), bem como a opção (“relief”) - "relevo" , que especifica a aparência visual da borda: (flat) - "plana" (padrão), (raised) - "elevado" ,(sunken) -  "afundado” ,(solid) -  "sólido" , (ridge) - "cume" ou (groove) - "sulco".

Exemplo configurando um espaço externo ao redor da caixa frame um conteiner:

nomeObejtoframe['borderwidth'] = 2

nomeObejtoframe['relief'] = 'sunken'

## Label - Rótulo

## Um **rótulo** é um widget que exibe texto ou imagens, normalmente que o usuário apenas visualizará, mas não interagirá de outra forma.

Os rótulos são criados usando a função **ttk.Label** e, normalmente, seu conteúdo é configurado ao mesmo tempo:

nomeObejtoLabel = ttk.Label(‘componente pai’, text='text que vai dentro da label:')

Como os quadros, os rótulos podem ter várias opções de configuração diferentes que podem alterar a forma como são exibidos.

#### **Exibindo Texto**

A opção de configuração (text) - "texto" mostrada acima ao criar a etiqueta é a mais comummente usada, principalmente quando a etiqueta é puramente decorativa ou explicativa. É claro que você pode alterar essa opção a qualquer momento, não apenas ao criar o rótulo pela primeira vez.

Você também pode fazer com que o widget monitore uma variável em seu script, de modo que, sempre que a variável for alterada, o rótulo exibirá o novo valor da variável; isso é feito com a opção "textvariable" :

resultsVariavel = StringVar()

nomeObejtoLabel['textvariable'] = resultsVariavel

resultsVariavel.set('Novo valor para mostar na tela')

#### **Exibindo Imagens**

Você também pode exibir uma imagem em um rótulo em vez de texto; se você deseja apenas uma imagem em sua interface, normalmente é a maneira de fazê-lo. Veremos as imagens com mais detalhes em um capítulo posterior, mas, por enquanto, vamos supor que você deseja exibir uma imagem GIF que está armazenada em um arquivo no disco. Este é um processo de duas etapas, primeiro criando uma imagem "objeto" e, em seguida, dizendo ao rótulo para usar esse objeto por meio de sua opção de configuração de "imagem" :

# criando um obejto de tipo imagem

nomeObjetoImage = PhotoImage (file = 'minhaImage.gif')

# setando o obejto de tipo imagem na label

nomeObejtoLabel[‘image’] = nomeObjetoImage

Você pode usar uma imagem e um texto, como você verá frequentemente nos botões da barra de ferramentas, por meio da opção de configuração (compound) - "composto" . O valor padrão é (none) - "nenhum" , significando exibir apenas a imagem se presente, caso contrário, o texto especificado pelas opções (text) - "texto" ou (text only) - "variável de texto" . Outras opções são (text) - "texto" (somente texto), (image) -  "imagem" (somente imagem), (center) -  "centro" (texto no centro da imagem), (top) - "topo" (imagem acima do texto),(left) -  "esquerda" , (bottom) - "inferior" e (right) -  "direita " .

#### **Layout**

Enquanto o layout geral do rótulo (ou seja, onde ele está posicionado na interface do usuário e quão grande é) é determinado pelo gerenciador de geometria, várias opções podem ajudá-lo a controlar como o rótulo será exibido dentro da caixa que o gerenciador de geometria fornece isto.

Se a caixa fornecida à Label for maior do que a Label exige para seu conteúdo, você pode usar a opção (anchor) - "âncora" para especificar a qual borda ou canto a Label deve ser fixada, o que deixaria qualquer espaço vazio na borda ou canto oposto . Os valores possíveis são especificados como direções da bússola: "n" (norte ou borda superior), "ne" , (nordeste ou canto superior direito), "e" , "se" , "s" , "sw" , "w" , "nw" ou "center" .

Os rótulos podem ser usados ​​para exibir mais de uma linha de texto. Isso pode ser feito incorporando retornos de carro ( "\ n" ) na string "text" / "textvariable" . Você também pode permitir que o rótulo envolva a string em várias linhas que não sejam maiores do que um determinado comprimento (com o tamanho especificado como pixels, centímetros, etc.), usando a opção "wraplength" .

Você também pode controlar como o texto é justificado, usando a opção (justify) - "justificar" , que pode ter os valores (left) -  "esquerda" , (Center) - "centro" ou (right) - "direita" . Se você tiver apenas uma linha de texto, isso é praticamente o mesmo que usar a opção (anchor) -  "âncora" , mas é mais útil com várias linhas de texto.

#### **Fontes, cores e mais**

Como acontece com as molduras, normalmente você não quer tocar em coisas como a fonte e as cores diretamente, mas se precisar alterá-las (por exemplo, para criar um tipo especial de Label), isso seria feito criando um novo estilo, que é então usado pelo widget com a opção (style) - "estilo" .

Ao contrário da maioria dos widgets temáticos, o widget de Label também fornece opções explícitas específicas do widget como alternativa; novamente, você usaria isso apenas em casos únicos especiais, quando usar um estilo não necessariamente fazia sentido.

Você pode especificar a fonte usada para exibir o texto do rótulo usando a opção de configuração (font) - "fonte" . Embora entremos em fontes com mais detalhes em um capítulo posterior, aqui estão os nomes de algumas fontes predefinidas que você pode usar:

* TkDefaultFont: O padrão para todos os itens da GUI não especificados de outra forma.
* TkTextFont: Usado para widgets de entrada, caixas de listagem, etc.
* TkFixedFont: Uma fonte padrão de largura fixa.
* TkMenuFont: A fonte usada para itens de menu.
* TkHeadingFont: Uma fonte para títulos de coluna em listas e tabelas.
* TkCaptionFont: Uma fonte para barras de legenda de janela e diálogo.
* TkSmallCaptionFont: Uma fonte de legenda menor para subjanelas ou diálogos de ferramentas.
* TkIconFont: Uma fonte para legendas de ícones.
* TkTooltipFont: Uma fonte para dicas de ferramentas.

(foreground text) - O primeiro plano (texto) - (background color) - e a cor do plano de fundo também podem ser alterados por meio das opções (foreground) - "primeiro plano" e (background) - "plano de fundo" . As cores são abordadas em detalhes posteriormente, mas você pode especificá-las como nomes de cores (por exemplo, "vermelho" ) ou códigos RGB hexadecimais (por exemplo, "#f340a" ).

As etiquetas também aceitam a opção de (relief) - "relevo" que foi discutida para os quadros.

## Button - Botão

Um **botão** , ao contrário de uma moldura ou rótulo, é muito projetado para o usuário interagir e, em particular, pressionar para executar alguma ação. Assim como os rótulos, eles podem exibir texto ou imagens, mas também têm toda uma gama de novas opções usadas para controlar seu comportamento.

Os botões são criados usando a função **ttk.Button** e, normalmente, seu conteúdo e retorno de chamada de comando são configurados ao mesmo tempo:

nomeObjetoBotão = ttk.Button(‘componente pai’, text = 'Ok', command = ‘nome função que se associa e e executada quando o botão for clicado’)

Tal como acontece com outros widgets, os botões podem ter várias opções de configuração diferentes que podem alterar sua aparência e comportamento.

#### **Text or Image - Texto e imagem**

Os botões aceitam as mesmas opções de configuração de (text) - "texto" ,(textvariable) -  "variável de texto" (raramente usado),(image) -  "imagem" e (compound) - "composto" como Labels, que controlam se o botão exibe texto e / ou uma imagem.

Os botões têm uma opção (default) - "padrão" , que informa ao Tk que o botão é o botão padrão na interface do usuário (ou seja, aquele que será chamado se o usuário clicar em Enter ou Return). Algumas plataformas e estilos irão desenhar isso com uma borda ou destaque diferente. Defina a opção como (active) - "ativo" para especificar que este é um botão padrão; o estado regular é "normal". Observe que definir esta opção não cria uma associação de evento que fará com que a tecla Return ou Enter ative o botão; que você tem que fazer sozinho.

#### **O retorno de chamada de comando - The Command Callback**

A opção (command) - "comando" é usada para fornecer uma interface entre a ação do botão e seu aplicativo. Quando o usuário clica no botão, o script fornecido pela opção é avaliado pelo interpretador.

Você também pode pedir ao botão para invocar o retorno de chamada de comando do seu aplicativo. Isso é útil para que você não precise repetir o comando a ser invocado várias vezes em seu programa; para que você saiba que se alterar a opção no botão, não será necessário alterá-la em outro lugar também.

#### **Button State - Estado do Botão**

Os botões e muitos outros widgets podem estar em um estado normal onde podem ser pressionados, mas também podem ser colocados em um estado desativado, onde o botão fica acinzentado e não pode ser pressionado. Isso é feito quando o comando do botão não é aplicável em um determinado momento.

Todos os widgets temáticos carregam consigo um estado interno, que é uma série de sinalizadores binários. Você pode definir ou limpar esses sinalizadores diferentes, bem como verificar a configuração atual usando os métodos "state" e "instate" . Os botões usam o sinalizador (disable) - "desativado" para controlar se o usuário pode ou não pressionar o botão

## Checkbutton - Botão de verificação

Um **botão de verificação** é como um botão normal, exceto que não apenas o usuário pode pressioná-lo, o que invocará um retorno de chamada de comando, mas também contém um valor binário de algum tipo (ou seja, uma alternância). Os botões de verificação são usados ​​o tempo todo quando um usuário é solicitado a escolher entre, por exemplo, dois valores diferentes para uma opção.

## Botao de radio - Radiobutton

## Um **botão de opção** permite escolher entre uma série de opções mutuamente exclusivas; ao contrário de um botão de seleção, ele não se limita a apenas duas opções. Os botões de rádio são sempre usados juntos em um conjunto e são uma boa opção quando o número de opções é bastante pequeno, por exemplo, 3-5.

## Entrada - Entry

Uma **entrada** apresenta ao usuário um campo de texto de uma linha que ele pode usar para digitar um valor de string. Eles podem ser praticamente qualquer coisa: nome, cidade, senha, número do seguro social e assim por diante.

## Caixa combo - COMBO BOX

## Uma **caixa de combinação** combina uma entrada com uma lista de opções disponíveis para o usuário. Isso permite que eles escolham a partir de um conjunto de valores que você forneceu (por exemplo, configurações típicas), mas também colocar seus próprios valores (por exemplo, para casos menos comuns que você não deseja incluir na lista).

Um combobox irá gerar um evento virtual "<ComboboxSelected>" ao qual você pode ligar sempre que seu valor mudar.

country.bind ('<<ComboboxSelected>>', função )