Aula 21

Frame

No Tkinter, o widget Frame é um recipiente retangular que pode conter outros widgets. Ele é usado para organizar e agrupar widgets relacionados em um layout mais complexo. O Frame em si não é visível, mas fornece uma estrutura para posicionar e organizar os widgets internos.

Aqui está como o widget Frame funciona em detalhes:

- Criação do Frame: primeiro, criamos uma instância do widget Frame usando a classe Frame do módulo Tkinter. Fazemos isso chamando o construtor Frame(parent, opções) e fornecendo o widget pai (geralmente a janela principal) como o primeiro argumento. Você também pode fornecer várias opções para personalizar o Frame, como largura, altura, cor de fundo, etc.

- Adição de widgets ao Frame: depois de criar o Frame, podemos adicionar outros widgets a ele. Isso é feito criando instâncias de widgets, como botões, rótulos ou caixas de texto, e usando o método widget.pack() ou widget.grid() para colocá-los dentro do Frame.

- Posicionamento dos widgets: dentro do Frame, podemos controlar o posicionamento dos widgets usando os métodos pack() ou grid() mencionados acima. Por exemplo, com pack(), podemos definir a opção side para 'top', 'bottom', 'left' ou 'right' para posicionar os widgets vertical ou horizontalmente. Com grid(), você especifica a linha e a coluna em que o widget deve ser colocado, usando os argumentos row e column.

- Gerenciamento de layout: o Frame pode ajudar a organizar os widgets internos, ajustando automaticamente seu tamanho e posição. Por exemplo, podemos usar opções como fill e expand para fazer os widgets ocuparem todo o espaço disponível dentro do Frame. Além disso, podemos usar o método widget.place() para posicionar widgets com precisão, fornecendo coordenadas x e y.

- Aninhamento de Frames: também podemos criar hierarquias de Frames aninhando-os uns dentro dos outros. Isso permite criar layouts complexos e organizar os widgets de forma hierárquica. Basta criar um Frame dentro de um Frame existente e adicionar widgets a ele da mesma maneira.

Geralmente, o widget Frame é usado para criar interfaces de usuário mais avançadas no Tkinter, permitindo organizar e agrupar widgets relacionados. Ele fornece flexibilidade no design do layout e facilita a manutenção e a organização do código.

Veja um exemplo abaixo:

```python
import tkinter as tk

def mensagem():
    """Função de exemplo para um botão"""
    print("Botão clicado!")

# criação da janela principal
janela = tk.Tk()

# criação do Frame
frame = tk.Frame(janela, width=200, height=200, bg="gray")
frame.pack()

# adição de um botão ao Frame
botao = tk.Button(frame, text="Clique Aqui", command=mensagem)
botao.pack(pady=20)

# execução do loop principal da janela
janela.mainloop()
```

Neste exemplo, criamos uma janela principal usando tk.Tk(). Em seguida, criamos um Frame com um tamanho de 200x200 pixels e cor de fundo cinza. O Frame é adicionado à janela principal usando frame.pack().

Dentro do Frame, adicionamos um botão usando tk.Button(). O botão é colocado no Frame usando button.pack(pady=20), com um espaçamento de 20 pixels acima e abaixo do botão.

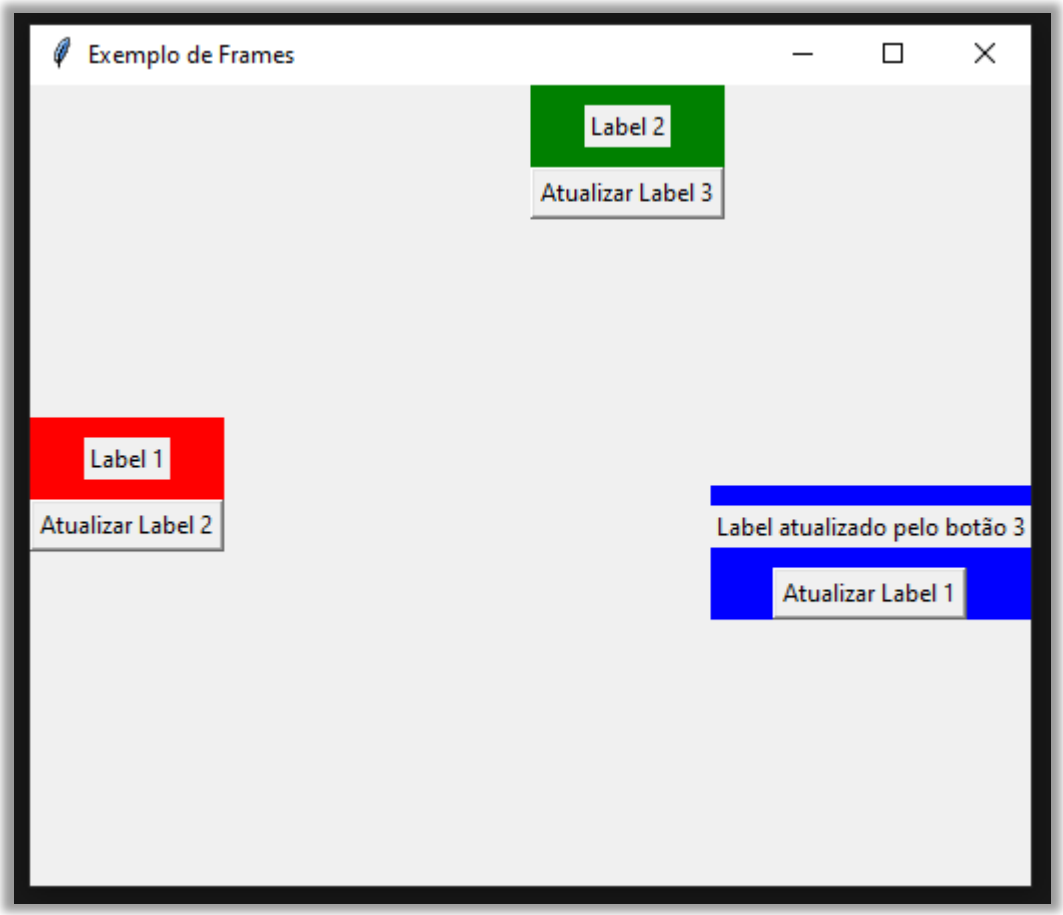
Finalmente, chamamos janela.mainloop() para iniciar o loop principal da janela e exibir a interface gráfica. Quando o botão é clicado, a função mensagem() é chamada e imprime uma mensagem no console.

Veja um exemplo de uma janela com vários Frames dentro dele:

```
import tkinter as tk


def atualiza_lbl1():
    """função que atualiza o label1"""
    label1.config(text="Label atualizado pelo botão 1")


def atualiza_lbl2():
    """função que atualiza o label2"""
    label2.config(text="Label atualizado pelo botão 2")


def atualiza_lbl3():
    """função que atualiza o label3"""
    label3.config(text="Label atualizado pelo botão 3")


# Criação da janela principal
janela = tk.Tk()
janela.title("Exemplo de Frames")
janela.geometry("500x400")
```

```
# Criação dos Frames
frame1 = tk.Frame(janela, width=100, height=100, bg="red")
frame2 = tk.Frame(janela, width=100, height=100, bg="green")
frame3 = tk.Frame(janela, width=100, height=100, bg="blue")

# Posicionamento dos Frames
frame1.pack(side="left")
frame2.pack(side="top")
frame3.pack(side="right")

# Labels nos Frames
label1 = tk.Label(frame1, text="Label 1")
label1.pack(pady=10)
label2 = tk.Label(frame2, text="Label 2")
label2.pack(pady=10)
label3 = tk.Label(frame3, text="Label 3")
label3.pack(pady=10)

# Botões nos Frames
btn1 = tk.Button(frame1, text="Atualizar Label 2", command=atualiza_lbl2)
btn1.pack()
btn2 = tk.Button(frame2, text="Atualizar Label 3", command=atualiza_lbl3)
btn2.pack()
btn3 = tk.Button(frame3, text="Atualizar Label 1", command=atualiza_lbl1)
btn3.pack()

# Execução do loop principal da janela
janela.mainloop()
```

No exemplo acima, temos a criação de 3 Frames e, dentro de cada um, a criação de um botão e um label. Quando um botão é clicado, um label de outro Frame é atualizado.
O resultado do código acima pode ser conferido abaixo:



Toplevel

No Tkinter, o widget Toplevel é usado para criar uma nova janela superior (top-level window) separada da janela principal. Ele permite criar múltiplas janelas independentes em um aplicativo. O Toplevel é semelhante à janela principal (Tk) em muitos aspectos, mas possui algumas características específicas.

Aqui está como o widget Toplevel funciona em detalhes:

- Criação do Toplevel: para criar uma nova janela Toplevel, usamos a classe Toplevel do módulo Tkinter. Basta chamar o construtor Toplevel(parent, opções) e fornecer a janela pai (geralmente a janela principal) como o primeiro argumento. Você também pode fornecer várias opções para personalizar o Toplevel, como largura, altura, título, ícone, etc.

- Configurações do Toplevel: podemos personalizar várias configurações do Toplevel usando métodos e atributos. Por exemplo, podemos usar toplevel.title("Título da Janela") para definir o título da janela, toplevel.geometry("500x300") para definir sua geometria (largura e altura), toplevel.iconbitmap("icone.ico") para definir um ícone personalizado para a janela, entre outras opções.

- Adição de widgets ao Toplevel: assim como na janela principal, podemos adicionar vários widgets à janela Toplevel usando métodos como pack(), grid() ou place(). Por exemplo, podemos adicionar botões, rótulos, campos de entrada e outros widgets ao Toplevel para construir uma interface do usuário interativa.

- Tratamento de eventos: o Toplevel suporta o tratamento de eventos, como cliques em botões, pressionamento de teclas, movimento do mouse, entre outros. Podemos usar métodos como bind() para

associar funções a eventos específicos, permitindo que você responda a interações do usuário na janela Toplevel.

- Destruir o Toplevel: quando você não precisa mais do Toplevel, podemos destruí-lo chamando o método toplevel.destroy(). Isso fechará a janela Toplevel e liberará os recursos associados a ela.

O widget Toplevel é útil quando desejamos criar janelas adicionais no aplicativo, como caixas de diálogo, janelas de configuração ou janelas secundárias. Ele permite que exibamos informações e interaja com o usuário em diferentes contextos, mantendo uma separação clara entre as diferentes janelas.

Veja um exemplo de uso:

```
import tkinter as tk

def abre_nova_janela():
    """Abre uma nova janela"""
    nova_janela = tk.Toplevel(janela)
    nova_janela.title("Nova Janela")
    nova_janela.geometry("300x200")
    label = tk.Label(nova_janela, text="Esta é uma nova janela!")
    label.pack(pady=50)

# Criação da janela principal
janela = tk.Tk()

# Botão para abrir nova janela
botao = tk.Button(janela, text="Abrir Nova Janela", command=abre_nova_janela)
botao.pack(pady=20)

# Execução do loop principal da janela
janela.mainloop()
```

Neste exemplo, temos uma janela principal criada com tk.Tk(). Dentro dela, definimos uma função abre_nova_janela() que é chamada quando o botão é clicado. Essa função cria uma nova janela Toplevel usando tk.Toplevel(window), onde janela é a janela principal.

A nova janela Toplevel é configurada com um título "Nova Janela" e uma geometria de 300x200 pixels. Em seguida, adicionamos um label à nova janela usando tk.Label(nova_janela, text="Esta é uma nova janela!"), e o posicionamos com label.pack(pady=50).

Ao executar o código, você verá a janela principal com um botão "Abrir Nova Janela". Quando você clicar no botão, uma nova janela Toplevel será aberta, exibindo o label "Esta é uma nova janela!".

Veja outro exemplo de como passar um valor de uma janela para outra:

```python
import tkinter as tk

def abre_nova_janela():
    """abre nova janela"""
    nova_janela = tk.Toplevel(janela)

    def pega_valor():
        valor = entrada.get()
        label.config(text="Valor recebido: " + valor)

    entrada = tk.Entry(nova_janela)
    entrada.pack(pady=10)

    btn_valor = tk.Button(nova_janela, text="Passar Valor", command=pega_valor)
    btn_valor.pack(pady=5)

# Criação da janela principal
janela = tk.Tk()

label = tk.Label(janela, text="Valor recebido: ")
label.pack(pady=10)

btn_janela = tk.Button(janela, text="Abrir Nova Janela",
                       command=abre_nova_janela)
btn_janela.pack(pady=5)

# Execução do loop principal da janela
janela.mainloop()
```

Neste exemplo, quando clicamos no botão "Abrir Nova Janela" na janela principal, um novo Toplevel é criado. Esse Toplevel contém uma Entry para inserir um valor e um botão "Passar Valor". Quando o botão é clicado, a função pega_valor() é chamada.

Dentro da função pega_valor(), obtemos o valor inserido na Entry usando entrada.get(). Em seguida, atualizamos o texto do label na janela principal com o valor recebido.

Ao executar o código, você pode inserir um valor na Entry do Toplevel e clicar no botão "Passar Valor". O valor inserido será passado para a janela principal e exibido no label.

Esse exemplo demonstra como passar um valor de um Toplevel de volta para a janela principal ao clicar em um botão. Você pode adaptar esse conceito para passar valores e informações mais complexas entre diferentes janelas em seu aplicativo.

Herança no Tkinter

Até o momento, vimos como usar o tkinter de uma forma muito crua. Simplesmente declaramos as variáveis e as chamamos dentro do arquivo.

Relembrando as primeiras aulas sobre o assunto, vamos lembrar que o tkinter é composto por diversas classes. Logo, podemos herdar essas classes e encapsular todo o nosso código em uma classe.

Veja um exemplo simples abaixo:

```
import tkinter as tk

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

        self.title("Minha Aplicação")
        self.geometry("300x200")

        self.label = tk.Label(self, text="Olá, Tkinter!")
        self.label.pack()

        self.botao = tk.Button(self)
        self.botao['text'] = "Clique Aqui"
        self.botao['command'] = self.clique_botao
        self.botao.pack()

    def clique_botao(self):
        self.label.config(text="Botão Clicado!")

# Criar uma instância da classe MinhaAplicacao
app = MeuApp()

# Executar o loop principal do Tkinter
app.mainloop()
```

Neste exemplo, definimos uma classe MeuApp() que herda da classe tk.Tk do Tkinter. Dentro do construtor __init__, usamos super().__init__() para chamar o construtor da classe base Tkinter e inicializar a janela principal.

Em seguida, podemos adicionar elementos de interface gráfica à nossa janela. No exemplo, adicionamos um rótulo (Label) e um botão (Button). Usamos o método pack() para posicionar esses elementos na janela.

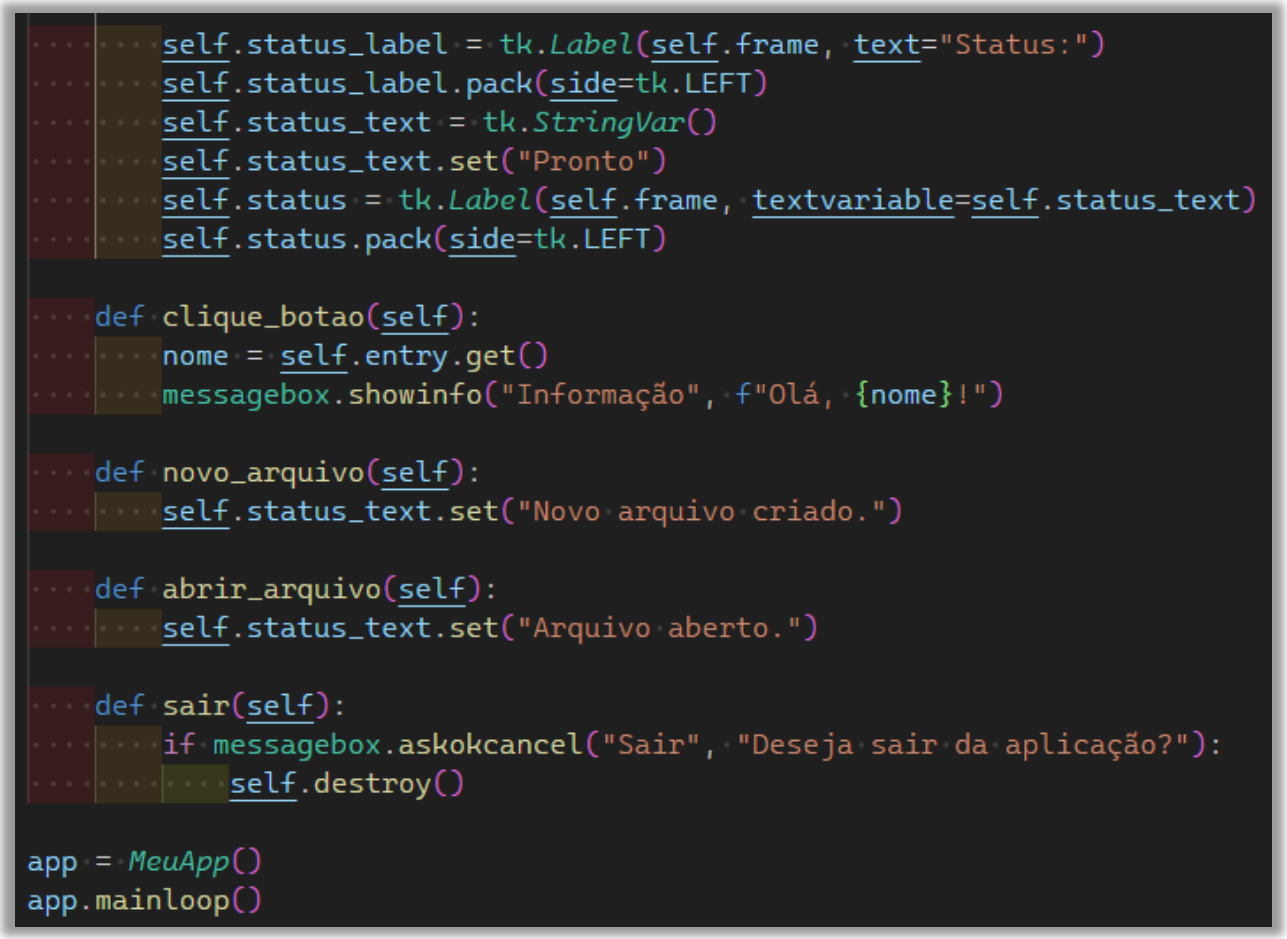
Além disso, definimos um método clique_botao que será chamado quando o botão for clicado. Neste caso, ele simplesmente atualiza o texto do rótulo.

Por fim, criamos uma instância da classe MeuApp e executamos o loop principal do Tkinter chamando o método mainloop(). Isso faz com que a janela seja exibida e o programa aguarde interações do usuário.

Veja agora um exemplo bem maior e envolvendo muito do que já foi visto:

```
import tkinter as tk
from tkinter import messagebox

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

        self.title("Minha Aplicação")
        self.geometry("400x300")

        self.label = tk.Label(self, text="Digite seu nome:")
        self.label.pack()

        self.entry = tk.Entry(self)
        self.entry.pack()

        self.button = tk.Button(self, text="Enviar", command=self.clique_botao)
        self.button.pack()

        self.var_checkbutton = tk.BooleanVar()
        self.checkbutton = tk.Checkbutton(self, text="Opção",
                                          variable=self.var_checkbutton)
        self.checkbutton.pack()
```

```
        self.var_radiobutton = tk.StringVar()
        self.radiobutton1 = tk.Radiobutton(self, text="Opção 1",
                                           variable=self.var_radiobutton,
                                           value="opcao1")
        self.radiobutton1.pack()
        self.radiobutton2 = tk.Radiobutton(self, text="Opção 2",
                                           variable=self.var_radiobutton,
                                           value="opcao2")
        self.radiobutton2.pack()

        self.menu_bar = tk.Menu(self)
        self.file_menu = tk.Menu(self.menu_bar, tearoff=0)
        self.file_menu.add_command(label="Novo", command=self.novo_arquivo)
        self.file_menu.add_command(label="Abrir", command=self.abrir_arquivo)
        self.file_menu.add_separator()
        self.file_menu.add_command(label="Sair", command=self.sair)
        self.menu_bar.add_cascade(label="Arquivo", menu=self.file_menu)
        self.config(menu=self.menu_bar)

        self.frame = tk.Frame(self, relief=tk.SOLID, borderwidth=1)
        self.frame.pack(pady=10)
```



```
        self.status_label = tk.Label(self.frame, text="Status:")
        self.status_label.pack(side=tk.LEFT)
        self.status_text = tk.StringVar()
        self.status_text.set("Pronto")
        self.status = tk.Label(self.frame, textvariable=self.status_text)
        self.status.pack(side=tk.LEFT)

    def clique_botao(self):
        nome = self.entry.get()
        messagebox.showinfo("Informação", f"Olá, {nome}!")

    def novo_arquivo(self):
        self.status_text.set("Novo arquivo criado.")

    def abrir_arquivo(self):
        self.status_text.set("Arquivo aberto.")

    def sair(self):
        if messagebox.askokcancel("Sair", "Deseja sair da aplicação?"):
            self.destroy()

app = MeuApp()
app.mainloop()
```

Neste exemplo, criamos uma classe MeuApp que herda da classe tk.Tk do Tkinter. Dentro do construtor __init__, criamos vários elementos de interface gráfica, como rótulos (Label), campos de entrada (Entry), botões (Button), checkbox (Checkbutton), radiobuttons (Radiobutton), menu (Menu) e frames (Frame).

Também definimos vários métodos de manipulação de eventos, como clique_botao, novo_arquivo, abrir_arquivo e sair, que são chamados quando os elementos interativos são acionados.

Além disso, utilizamos a classe messagebox do Tkinter para exibir caixas de diálogo de informação e confirmação.

Esse exemplo completo demonstra como você pode usar vários componentes e recursos do Tkinter, como botões, rótulos, campos de entrada, variáveis do Tkinter, menu e frames, em uma aplicação GUI.

.

Para todos os próximos exercícios, use o tkinter como uma classe herdada.

Exercícios para Praticar

1. Crie um Frame com um Label contendo o texto "Bem-vindo!" e um Button com o texto "Sair".
2. Crie um Frame com dois Labels: um com o texto "Nome" e outro com o texto "Idade". Adicione também dois Entry widgets para que o usuário possa digitar o nome e a idade.
3. Crie um Frame com um Label e um Button. Ao clicar no botão, atualize o texto do Label para exibir "Botão clicado!".
4. Crie um Frame com um Label e três Buttons com os textos "Verde", "Vermelho" e "Azul". Ao clicar em cada botão, mude a cor de fundo do Frame correspondente.
5. Crie um Frame com um Label e um Entry. Ao digitar algo no Entry e pressionar Enter, atualize o texto do Label para exibir o conteúdo do Entry.
6. Crie um Frame com dois Labels e um Button. Ao clicar no botão, concatene o texto dos dois Labels em um terceiro Label.
7. Crie um Frame com um Label e um Entry. Ao clicar em um Button, copie o texto do Entry para a área de transferência.
8. Crie um Frame com três Buttons com os textos "Aumentar", "Diminuir" e "Zerar". Ao clicar em "Aumentar", aumente o tamanho do texto de um Label em 2 pontos. Ao clicar em "Diminuir", diminua o tamanho do texto em 2 pontos. Ao clicar em "Zerar", defina o tamanho do texto de volta para o valor inicial.
9. Crie um Frame com um Label e um Entry. Ao clicar em um Button, verifique se o texto do Entry é um número par ou ímpar e exiba a mensagem correspondente no Label.
10. Crie um Frame com três Labels e um Button. Ao clicar no botão, alterne os textos dos Labels em um ciclo, ou seja, o texto do Label 1 passa para o Label 2, o texto do Label 2 passa para o Label 3 e o texto do Label 3 passa para o Label 1.

11. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label contendo o texto "Bem-vindo!".
12. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com dois Labels: um com o texto "Nome" e outro com o texto "Idade". Adicione também dois Entry widgets no Toplevel para que o usuário possa digitar o nome e a idade.
13. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label e um Button. Ao clicar no botão do Toplevel, atualize o texto do Label para exibir "Botão clicado!".
14. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label e três Buttons com os textos "Verde", "Vermelho" e "Azul". Ao clicar em cada botão do Toplevel, mude a cor de fundo do Toplevel correspondente.
15. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label e um Entry. Ao digitar algo no Entry e pressionar Enter, atualize o texto do Label para exibir o conteúdo do Entry.
16. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com dois Labels e um Button. Ao clicar no botão do Toplevel, concatene o texto dos dois Labels em um terceiro Label.
17. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label e um Entry. Ao clicar em um Button do Toplevel, copie o texto do Entry para a área de transferência.
18. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com três Buttons com os textos "Aumentar", "Diminuir" e "Zerar". Ao clicar em "Aumentar", aumente o tamanho do texto de um Label no Toplevel em 2 pontos. Ao clicar em "Diminuir", diminua o tamanho do texto em 2 pontos. Ao clicar em "Zerar", defina o tamanho do texto de volta para o valor inicial.
19. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com um Label e um Entry. Ao clicar em um Button do Toplevel, verifique se o texto do Entry é um número par ou ímpar e exiba a mensagem correspondente no Label.
20. Crie uma janela principal com um Button que, ao ser clicado, abre um Toplevel com três Labels e um Button. Ao clicar no botão do Toplevel, alterne os textos dos Labels em um ciclo, ou seja, o texto do Label 1 passa para o Label 2, o texto do Label 2 passa para o Label 3 e o texto do Label 3 passa para o Label 1.

Extra

Usando todos os conhecimentos adquiridos do tkinter até agora, resolva os problemas abaixo:

- Jogo da Velha: Crie uma interface gráfica para o jogo da velha usando botões em uma matriz 3x3. Os jogadores podem clicar nos botões para fazer suas jogadas e exiba mensagens de vitória, empate ou próximo jogador em um Label.

- Jogo da Forca: Crie uma interface gráfica para o jogo da forca. Exiba uma palavra oculta com underscore (_) para cada letra não revelada. Os jogadores podem digitar letras em um Entry e clicar em um botão para verificar se a letra está correta. Exiba a palavra atualizada e a imagem da forca à medida que o jogador erra.

- Adivinhar o Número: Crie uma interface gráfica para um jogo de adivinhação de números. Gere um número aleatório e deixe os jogadores digitarem sua suposição em um Entry. Forneça um botão para verificar se a suposição está correta e exiba mensagens informando se é muito alta ou muito baixa.