Cookbook Tkinter: Construindo Interfaces Gráficas com Python

Marcos A. Koslinski

Julho 2025

Contents

Chapter 1

Introdução ao Tkinter

1.1 O que é Tkinter?

O Tkinter é a biblioteca padrão do Python para criar interfaces gráficas (GUIs). Baseada na biblioteca Tcl/Tk, ela é leve, portátil e ideal para iniciantes e desenvolvedores intermediários. Este cookbook expande o tutorial *Pensando em Tkinter* de Steven Ferg, oferecendo uma abordagem mais detalhada e prática para aprender a criar GUIs com Tkinter.

1.2 Configurando o Ambiente

O Tkinter está incluído na instalação padrão do Python. Para verificar sua disponibilidade, execute:

```
1 import tkinter
2 print("Tkinter está pronto!")
```

1.3 As Quatro Questões Básicas da Programação de GUIs

Conforme descrito em *Pensando em Tkinter*, a programação de interfaces gráficas envolve quatro questões fundamentais:

1. **Aparência**: Como a interface será exibida (janelas, botões, etc.)?
2. **Event Handlers**: Quais procedimentos serão executados em resposta a eventos?
3. **Binding**: Como associar eventos (cliques, teclas) a widgets e handlers?
4. **Event Loop**: Como o programa aguarda e processa eventos do usuário?

1.4 Exemplo 1: Janela Simples

Este exemplo cria uma janela simples, ilustrando o conceito de *event loop*.

```
import tkinter as tk

# Criando a janela principal
janela = tk.Tk()
janela.title("Minha Primeira Janela")
janela.geometry("300x200")  # Tamanho: 300px de largura x 200px de
   altura

# Iniciando o loop principal
janela.mainloop()
```

1.4.1 Explicação do Código

- `tk.Tk()`: Cria a janela principal, chamada de *root* ou *toplevel*.
- `janela.title()`: Define o título na barra superior.
- `janela.geometry()`: Configura o tamanho da janela (largura x altura).
- `janela.mainloop()`: Inicia o loop de eventos, mantendo a janela aberta e responsiva.

Chapter 2

Widgets Fundamentais

Widgets são os elementos interativos de uma GUI, como botões, rótulos e campos de texto. Este capítulo explora os widgets mais comuns e suas propriedades.

2.1 Exemplo 2: Botão e Rótulo

Este exemplo cria uma janela com um rótulo e um botão que altera o texto do rótulo ao ser clicado.

```
import tkinter as tk

# Criando a janela principal
janela = tk.Tk()
janela.title("Exemplo com Botão")
janela.geometry("400x300")

# Criando um rótulo
rotulo = tk.Label(janela, text="Bem-vindo ao Tkinter!",
   font=("Noto", 14))
rotulo.pack(pady=10)

# Função para alterar o texto do rótulo
def clique_botao():
    rotulo.config(text="Botão clicado!")

# Criando um botão
botao = tk.Button(janela, text="Clique Aqui", command=clique_botao,
   font=("Noto", 12))
botao.pack(pady=10)

# Iniciando o loop principal
janela.mainloop()
```

2.1.1 Explicação do Código

- `tk.Label`: Cria um rótulo com texto estático. O parâmetro `font` define a fonte e o tamanho.
- `pack()`: Posiciona widgets verticalmente. `pady` adiciona espaçamento vertical.

- `tk.Button`: Cria um botão interativo. `command` associa o botão à função `clique`$_botao$.`config(`$Alterapropriedadesdeumwidget, comootextodortulo.$

2.2 Exemplo 3: Campo de Texto (Entry)

Este exemplo adiciona um campo de texto para entrada de dados.

```
import tkinter as tk

# Criando a janela principal
janela = tk.Tk()
janela.title("Exemplo com Entry")
janela.geometry("400x300")

# Criando um rótulo
rotulo = tk.Label(janela, text="Digite seu nome:", font=("Noto",
    12))
rotulo.pack(pady=10)

# Criando um campo de texto
entrada = tk.Entry(janela, font=("Noto", 12))
entrada.pack(pady=10)

# Função para exibir o texto inserido
def mostrar_nome():
    nome = entrada.get()
    rotulo.config(text=f"Olá, {nome}!")

# Criando um botão
botao = tk.Button(janela, text="Enviar", command=mostrar_nome,
    font=("Noto", 12))
botao.pack(pady=10)

# Iniciando o loop principal
janela.mainloop()
```

2.2.1 Explicação do Código

- `tk.Entry`: Cria um campo de texto editável.
- `entrada.get()`: Obtém o texto inserido pelo usuário.
- `rotulo.config()`: Atualiza o texto do rótulo com base na entrada.

Chapter 3

Gerenciadores de Layout

O Tkinter oferece três gerenciadores de layout: `pack`, `grid` e `place`. Este capítulo explora cada um com exemplos práticos.

3.1 Exemplo 4: Usando Pack com Opções

Este exemplo demonstra o uso do gerenciador `pack` com opções como `side`, `expand` e `fill`.

```
import tkinter as tk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.buttons_frame = tk.Frame(parent,
           background="lightgray")
        self.buttons_frame.pack(padx=10, pady=10, fill=tk.BOTH,
           expand=True)

        self.botao1 = tk.Button(self.buttons_frame, text="Botão 1",
           background="green")
        self.botao1.pack(side=tk.LEFT, padx=5, pady=5)

        self.botao2 = tk.Button(self.buttons_frame, text="Botão 2",
           background="yellow")
        self.botao2.pack(side=tk.LEFT, padx=5, pady=5)

        self.botao3 = tk.Button(self.buttons_frame, text="Botão 3",
           background="red")
        self.botao3.pack(side=tk.LEFT, padx=5, pady=5)

root = tk.Tk()
root.title("Exemplo com Pack")
root.geometry("400x300")
app = MinhaApp(root)
root.mainloop()
```

3.1.1 Explicação do Código

- `tk.Frame`: Cria um contêiner para organizar widgets.
- `pack(side=tk.LEFT)`: Posiciona botões horizontalmente, da esquerda para a direita.
- `fill=tk.BOTH, expand=True`: Faz o frame ocupar todo o espaço disponível.
- `padx, pady`: Adicionam espaçamento externo aos widgets.

3.2 Exemplo 5: Usando Grid

O gerenciador `grid` organiza widgets em uma grade bidimensional.

```
import tkinter as tk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

        tk.Label(self.frame, text="Nome:", font=("Noto",
           12)).grid(row=0, column=0, padx=5, pady=5, sticky="e")
        self.entrada_nome = tk.Entry(self.frame, font=("Noto", 12))
        self.entrada_nome.grid(row=0, column=1, padx=5, pady=5)

        tk.Label(self.frame, text="Idade:", font=("Noto",
           12)).grid(row=1, column=0, padx=5, pady=5, sticky="e")
        self.entrada_idade = tk.Entry(self.frame, font=("Noto", 12))
        self.entrada_idade.grid(row=1, column=1, padx=5, pady=5)

        tk.Button(self.frame, text="Enviar", command=self.enviar,
           font=("Noto", 12)).grid(row=2, column=0, columnspan=2,
           pady=10)

    def enviar(self):
        nome = self.entrada_nome.get()
        idade = self.entrada_idade.get()
        print(f"Nome: {nome}, Idade: {idade}")

root = tk.Tk()
root.title("Exemplo com Grid")
root.geometry("400x300")
app = MinhaApp(root)
root.mainloop()
```

3.2.1 Explicação do Código

- `grid(row, column)`: Posiciona widgets em uma grade.

- `sticky="e"`: Alinha o widget à direita (east) na célula.
- `columnspan`: Faz o botão ocupar duas colunas.

Chapter 4

Manipulando Eventos

Eventos permitem que widgets respondam a ações do usuário, como cliques ou pressionamento de teclas.

4.1 Exemplo 6: Binding de Eventos

Este exemplo associa eventos de mouse e teclado a botões.

```
import tkinter as tk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

        self.botao1 = tk.Button(self.frame, text="OK",
           background="green")
        self.botao1.pack(side=tk.LEFT, padx=5)
        self.botao1.bind("<Button-1>", self.botao1_clique)
        self.botao1.bind("<Return>", self.botao1_clique)
        self.botao1.focus_force()

        self.botao2 = tk.Button(self.frame, text="Cancelar",
           background="red")
        self.botao2.pack(side=tk.LEFT, padx=5)
        self.botao2.bind("<Button-1>", self.botao2_clique)

    def botao1_clique(self, event):
        self.botao1.configure(background="yellow" if
           self.botao1["background"] == "green" else "green")

    def botao2_clique(self, event):
        self.myParent.destroy()

root = tk.Tk()
```

```
root.title("Exemplo de Binding")
root.geometry("400x300")
app = MinhaApp(root)
root.mainloop()
```

4.1.1 Explicação do Código

- `bind("<Button-1>", ...)`: Associa um clique do botão esquerdo do mouse ao método.
- `bind("<Return>", ...)`: Associa a tecla Enter ao método.

- `focus`$_{f}orce() : Definefocoinicialnobotao OK.$`myParent.destroy()` $: Fechaajanela.$

4.2 Exemplo 7: Usando Lambda para Binding

Este exemplo usa funções `lambda` para passar argumentos a um handler genérico.

```
import tkinter as tk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

        botoes = [("Botão 1", "green", 1, "Bom!"), ("Botão 2",
           "red", 2, "Ruim!")]
        for nome, cor, id, msg in botoes:
            botao = tk.Button(self.frame, text=nome, background=cor)
            botao.pack(side=tk.LEFT, padx=5)
            botao.bind("<Button-1>", lambda event, n=nome, i=id,
               m=msg: self.handler(n, i, m))

    def handler(self, nome, id, mensagem):
        print(f"Botão: {nome}, ID: {id}, Mensagem: {mensagem}")

root = tk.Tk()
root.title("Exemplo com Lambda")
root.geometry("400x300")
app = MinhaApp(root)
root.mainloop()
```

4.2.1 Explicação do Código

- `lambda event, n=nome, i=id, m=mensagem`: Cria uma função anônima que passa argumentos ao handler.
- `handler`: Função genérica que processa eventos de múltiplos botões.

Chapter 5

Estruturando Aplicações com Classes

Estruturar aplicações como classes melhora a organização e a manutenção do código, especialmente em projetos complexos.

5.1 Exemplo 8: Formulário Estruturado

Este exemplo cria um formulário com validação, usando uma classe.

```
import tkinter as tk
from tkinter import messagebox

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

        tk.Label(self.frame, text="Nome:", font=("Noto",
           12)).grid(row=0, column=0, padx=5, pady=5, sticky="e")
        self.entrada_nome = tk.Entry(self.frame, font=("Noto", 12))
        self.entrada_nome.grid(row=0, column=1, padx=5, pady=5)

        tk.Label(self.frame, text="Idade:", font=("Noto",
           12)).grid(row=1, column=0, padx=5, pady=5, sticky="e")
        self.entrada_idade = tk.Entry(self.frame, font=("Noto", 12))
        self.entrada_idade.grid(row=1, column=1, padx=5, pady=5)

        tk.Button(self.frame, text="Enviar", command=self.enviar,
           font=("Noto", 12)).grid(row=2, column=0, columnspan=2,
           pady=10)

    def enviar(self):
        nome = self.entrada_nome.get()
        idade = self.entrada_idade.get()
```

```
23         if nome and idade.isdigit():
24             messagebox.showinfo("Sucesso", f"Nome: {nome}\nIdade:
                   {idade}")
25         else:
26             messagebox.showerror("Erro", "Preencha todos os campos
                   corretamente!")
27
28 root = tk.Tk()
29 root.title("Formulário Completo")
30 root.geometry("400x400")
31 app = MinhaApp(root)
32 root.mainloop()
```

5.1.1 Explicação do Código

- `messagebox`: Exibe caixas de diálogo para mensagens de sucesso ou erro.

- `self.entrada`$_{n}ome.get() : Obtmotextodocampodeentrada.Classes : Encapsulamwidgetselgica, fac$

Chapter 6

Uso de Widgets TTK

Os widgets do módulo `ttk` oferecem uma aparência mais moderna e suporte a temas, compatíveis com diferentes sistemas operacionais.

6.1 Exemplo 9: Combobox e Treeview

Este exemplo cria uma interface com um `ttk.Combobox` para selecionar categorias e um `ttk.Treeview` para exibir dados.

```
import tkinter as tk
from tkinter import ttk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

        # Combobox para selecionar categorias
        tk.Label(self.frame, text="Categoria:", font=("Noto",
           12)).grid(row=0, column=0, padx=5, pady=5, sticky="e")
        self.categoria = ttk.Combobox(self.frame,
           values=["Eletrônicos", "Roupas", "Livros"],
           font=("Noto", 12))
        self.categoria.grid(row=0, column=1, padx=5, pady=5)
        self.categoria.set("Eletrônicos")

        # Treeview para exibir produtos
        self.tree = ttk.Treeview(self.frame, columns=("Nome",
           "Preço"), show="headings")
        self.tree.heading("Nome", text="Nome do Produto")
        self.tree.heading("Preço", text="Preço (R$)")
        self.tree.grid(row=1, column=0, columnspan=2, pady=10)

        # Botão para adicionar item
```

```
23          ttk.Button(self.frame, text="Adicionar",
               command=self.adicionar).grid(row=2, column=0,
               columnspan=2, pady=5)
24
25      def adicionar(self):
26          categoria = self.categoria.get()
27          self.tree.insert("", tk.END, values=(f"Produto
               {categoria}", "99.90"))
28
29  root = tk.Tk()
30  root.title("Exemplo com TTK")
31  root.geometry("500x400")
32  app = MinhaApp(root)
33  root.mainloop()
```

6.1.1 Explicação do Código

- `ttk.Combobox`: Cria uma lista suspensa com opções predefinidas.
- `ttk.Treeview`: Exibe dados em formato de tabela, com colunas configuráveis.
- `tree.insert`: Adiciona linhas à tabela.

Chapter 7

Animações no Canvas

O widget Canvas permite criar animações dinâmicas, úteis para jogos ou visualizações interativas.

7.1 Exemplo 10: Mini-Jogo de Nave

Este exemplo implementa um jogo simples onde uma nave se move com as teclas de seta e dispara projéteis.

```
import tkinter as tk

class JogoNave:
    def __init__(self, parent):
        self.myParent = parent
        self.canvas = tk.Canvas(parent, width=600, height=400,
           background="black")
        self.canvas.pack(padx=10, pady=10)

        # Criando a nave
        self.nave = self.canvas.create_rectangle(280, 360, 320,
           380, fill="blue")
        self.velocidade = 5
        self.tiros = []

        # Vinculando eventos de teclado
        self.canvas.bind_all("<Left>", self.mover_esquerda)
        self.canvas.bind_all("<Right>", self.mover_direita)
        self.canvas.bind_all("<space>", self.disparar)

        # Iniciando a animação
        self.animar()

    def mover_esquerda(self, event):
        x1, _, x2, _ = self.canvas.bbox(self.nave)
        if x1 > 0:
            self.canvas.move(self.nave, -self.velocidade, 0)
```

```
26
27      def mover_direita(self, event):
28          _, _, x2, _ = self.canvas.bbox(self.nave)
29          if x2 < 600:
30              self.canvas.move(self.nave, self.velocidade, 0)
31
32      def disparar(self, event):
33          x1, y1, x2, _ = self.canvas.bbox(self.nave)
34          tiro = self.canvas.create_oval(x1+15, y1-10, x2-15, y1,
                fill="red")
35          self.tiros.append(tiro)
36
37      def animar(self):
38          for tiro in self.tiros[:]:
39              self.canvas.move(tiro, 0, -10)
40              if self.canvas.bbox(tiro)[1] < 0:
41                  self.canvas.delete(tiro)
42                  self.tiros.remove(tiro)
43          self.myParent.after(50, self.animar)
44
45  root = tk.Tk()
46  root.title("Mini-Jogo de Nave")
47  root.geometry("620x420")
48  app = JogoNave(root)
49  root.mainloop()
```

7.1.1 Explicação do Código

- `create`$_{r}ectangle$: $Criaanavecomoumretnguloazul.$

Chapter 8

Integração com PIL (Pillow)

A biblioteca PIL (Pillow) permite manipular imagens, como redimensionar e exibir em um Canvas ou como fundo de uma janela.

8.1 Exemplo 11: Exibindo Imagens

Este exemplo carrega uma imagem e a exibe como fundo de um formulário, ajustando seu tamanho.

```
import tkinter as tk
from PIL import Image, ImageTk

class MinhaApp:
    def __init__(self, parent):
        self.myParent = parent
        self.canvas = tk.Canvas(parent, width=800, height=600)
        self.canvas.pack(fill=tk.BOTH, expand=True)

        # Carregando e redimensionando a imagem
        imagem = Image.open("background.jpg")  # Substitua pelo
            caminho da sua imagem
        imagem = imagem.resize((800, 600), Image.LANCZOS)
        self.background_image = ImageTk.PhotoImage(imagem)
        self.canvas.create_image(0, 0, image=self.background_image,
            anchor="nw")

        # Adicionando um formulário
        self.frame = tk.Frame(self.canvas, background="white")
        self.canvas.create_window(400, 300, window=self.frame)

        tk.Label(self.frame, text="Usuário:", font=("Noto",
            12)).grid(row=0, column=0, padx=5, pady=5)
        self.entrada_usuario = tk.Entry(self.frame, font=("Noto",
            12))
        self.entrada_usuario.grid(row=0, column=1, padx=5, pady=5)

```

```
24          tk.Button(self.frame, text="Entrar", command=lambda:
               print("Entrar clicado")).grid(row=1, column=0,
               columnspan=2, pady=5)
25
26  root = tk.Tk()
27  root.title("Formulário com Imagem")
28  root.geometry("800x600")
29  app = MinhaApp(root)
30  root.mainloop()
```

8.1.1 Explicação do Código

- `Image.open`: Carrega uma imagem do disco.
- `resize`: Redimensiona a imagem para caber na janela.
- `ImageTk.PhotoImage`: Converte a imagem para um formato compatível com Tkinter.

- `create`$_{w}indow : Posicionaumframesobreocanvascomaimagemdefundo.$

Chapter 9

Formulários Complexos e Banco de Dados

Este capítulo apresenta um formulário avançado integrado com MySQL para gerenciar produtos, com validação e exibição de imagens.

9.1 Exemplo 12: Cadastro de Produtos com MySQL

Este exemplo cria um formulário para cadastrar produtos, salvar no banco de dados e exibir imagens.

```
import tkinter as tk
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
import mysql.connector

class CadastroProdutos:
    def __init__(self, parent):
        self.myParent = parent
        self.conexao = mysql.connector.connect(
            host="localhost", user="root", password="sua_senha",
               database="estoque"
        )
        self.cursor = self.conexao.cursor()
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS produtos (
                id INT AUTO_INCREMENT PRIMARY KEY,
                nome VARCHAR(255),
                preco FLOAT,
                imagem VARCHAR(255)
            )
        """)

        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10)

```

```
25          # Formulário
26          tk.Label(self.frame, text="Nome:", font=("Noto",
               12)).grid(row=0, column=0, padx=5, pady=5)
27          self.entrada_nome = ttk.Entry(self.frame, font=("Noto", 12))
28          self.entrada_nome.grid(row=0, column=1, padx=5, pady=5)
29
30          tk.Label(self.frame, text="Preço:", font=("Noto",
               12)).grid(row=1, column=0, padx=5, pady=5)
31          self.entrada_preco = ttk.Entry(self.frame, font=("Noto",
               12))
32          self.entrada_preco.grid(row=1, column=1, padx=5, pady=5)
33
34          tk.Label(self.frame, text="Imagem:", font=("Noto",
               12)).grid(row=2, column=0, padx=5, pady=5)
35          self.entrada_imagem = ttk.Entry(self.frame, font=("Noto",
               12))
36          self.entrada_imagem.grid(row=2, column=1, padx=5, pady=5)
37          ttk.Button(self.frame, text="Selecionar Imagem",
               command=self.selecionar_imagem).grid(row=2, column=2,
               padx=5, pady=5)
38
39          ttk.Button(self.frame, text="Cadastrar",
               command=self.cadastrar).grid(row=3, column=0,
               columnspan=3, pady=10)
40
41          # Visualização da imagem
42          self.canvas = tk.Canvas(self.frame, width=200, height=200)
43          self.canvas.grid(row=4, column=0, columnspan=3, pady=10)
44
45      def selecionar_imagem(self):
46          from tkinter import filedialog
47          caminho = filedialog.askopenfilename(filetypes=[("Imagens",
               "*.jpg;*.png")])
48          if caminho:
49              self.entrada_imagem.delete(0, tk.END)
50              self.entrada_imagem.insert(0, caminho)
51              imagem = Image.open(caminho).resize((200, 200),
                   Image.LANCZOS)
52              self.imagem_tk = ImageTk.PhotoImage(imagem)
53              self.canvas.create_image(100, 100, image=self.imagem_tk)
54
55      def cadastrar(self):
56          nome = self.entrada_nome.get()
57          preco = self.entrada_preco.get()
58          imagem = self.entrada_imagem.get()
59          if nome and preco.replace(".", "").isdigit() and imagem:
60              self.cursor.execute("INSERT INTO produtos (nome, preco,
                   imagem) VALUES (%s, %s, %s)", (nome, float(preco),
                   imagem))
61              self.conexao.commit()
62              messagebox.showinfo("Sucesso", "Produto cadastrado!")
```

```
63         else:
64             messagebox.showerror("Erro", "Preencha todos os campos
                   corretamente!")
65
66 root = tk.Tk()
67 root.title("Cadastro de Produtos")
68 root.geometry("600x500")
69 app = CadastroProdutos(root)
70 root.mainloop()
```

9.1.1 Explicação do Código

- `mysql.connector`: Conecta ao banco de dados MySQL.
- `filedialog.askopenfilename`: Permite selecionar uma imagem do disco.
- `ttk.Entry`: Usa widgets estilizados para entrada de dados.
- `messagebox`: Exibe mensagens de sucesso ou erro após validação.

Chapter 10

Projetos Práticos

10.1 Exemplo 13: Editor de Texto Simples

Este exemplo cria um editor de texto com funcionalidades básicas.

```
import tkinter as tk
from tkinter import filedialog, messagebox

class EditorTexto:
    def __init__(self, parent):
        self.myParent = parent
        self.frame = tk.Frame(parent)
        self.frame.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

        self.texto = tk.Text(self.frame, font=("Noto", 12),
           height=10)
        self.texto.pack(padx=5, pady=5, fill=tk.BOTH, expand=True)

        self.menu = tk.Menu(parent)
        parent.config(menu=self.menu)
        self.menu_arquivo = tk.Menu(self.menu, tearoff=0)
        self.menu.add_cascade(label="Arquivo",
           menu=self.menu_arquivo)
        self.menu_arquivo.add_command(label="Salvar",
           command=self.salvar)
        self.menu_arquivo.add_command(label="Sair",
           command=self.sair)

    def salvar(self):
        arquivo =
           filedialog.asksaveasfilename(defaultextension=".txt")
        if arquivo:
            with open(arquivo, "w") as f:
                f.write(self.texto.get("1.0", tk.END))
            messagebox.showinfo("Sucesso", "Arquivo salvo com
               sucesso!")

```

```
27     def sair(self):
28         self.myParent.destroy()
29 
30 root = tk.Tk()
31 root.title("Editor de Texto")
32 root.geometry("600x400")
33 app = EditorTexto(root)
34 root.mainloop()
```

10.1.1 Explicação do Código

- `tk.Text`: Cria uma área de texto multilinha.
- `tk.Menu`: Adiciona um menu à janela.
- `filedialog`: Abre uma caixa de diálogo para salvar arquivos.