FAP 2024 – UFRN – Softex – Prof José Alfredo Costa

**Interfaces Gráficas em Python com Tkinter**

**Introdução às Interfaces Gráficas**

Interfaces gráficas permitem que usuários interajam com sistemas de software de forma visual e intuitiva. Ao invés de comandos de texto, que podem ser complicados para alguns usuários, interfaces gráficas oferecem botões, menus e outros elementos visuais que facilitam a interação. Em muitos casos, isso resulta em uma melhor experiência do usuário e maior acessibilidade.

A importância das interfaces gráficas é evidente em diversas aplicações, desde software comercial até sistemas de controle e ferramentas de análise. Elas permitem que usuários realizem tarefas complexas de forma mais fácil e compreensível.

**Tkinter: A Biblioteca para Interfaces Gráficas em Python**

Tkinter é a biblioteca padrão para a criação de interfaces gráficas em Python. Ela oferece uma maneira simples e eficaz de criar interfaces com diversos componentes, como botões, menus, campos de texto e mais. Tkinter é integrada ao Python e é uma escolha popular devido à sua simplicidade e eficácia.

**Principais Comandos do Tkinter**

1. **tk.Tk()**: Cria a janela principal da aplicação.
2. **tk.Frame()**: Cria um contêiner para outros widgets, ajudando a organizar a interface.
3. **tk.Label()**: Exibe texto ou imagens.
4. **tk.Button()**: Cria botões que podem executar ações quando clicados.
5. **pack()**: Gerenciador de layout que organiza widgets dentro de um contêiner.
6. **grid()**: Outro gerenciador de layout que organiza widgets em uma grade.
7. **place()**: Permite posicionar widgets em coordenadas específicas.

**Tipos de Gráficos Usados**

Para visualização de dados, são usados diversos tipos de gráficos. Neste código, os seguintes gráficos são apresentados:

1. **Gráfico de Dispersão (Scatter Plot)**: Mostra a relação entre duas variáveis contínuas usando pontos.
2. **Gráfico de Barras (Bar Chart)**: Utilizado para comparar categorias diferentes por meio de barras verticais.
3. **Gráfico de Pizza (Pie Chart)**: Representa partes de um todo, mostrando a proporção de cada categoria em relação ao total.
4. **Gráfico de Linhas (Line Chart)**: Ideal para mostrar a evolução de uma variável ao longo do tempo.
5. **Histograma (Histogram)**: Exibe a distribuição de uma variável contínua dividida em intervalos.

**Código Completo com Comentários**

import tkinter as tk

from tkinter import ttk

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

# Definir um DataFrame de exemplo

data = {

    'total\_bill': [16.99, 10.34, 21.01, 23.68, 24.59],

    'tip': [1.01, 1.66, 3.50, 3.31, 3.61],

    'day': ['Sun', 'Sun', 'Sun', 'Sat', 'Sat'],

    'size': [2, 3, 3, 2, 4]

}

tips = pd.DataFrame(data)

# Função para atualizar o gráfico com base na seleção do menu

def update\_graph(selected\_graph):

    # Limpa o gráfico existente

    for widget in graph\_frame.winfo\_children():

        widget.destroy()

    # Gera um novo gráfico com base na seleção

    fig, ax = plt.subplots()

    if selected\_graph == "Scatter Plot":

        sns.scatterplot(data=tips, x='total\_bill', y='tip', hue='day', size='size', ax=ax)

    elif selected\_graph == "Bar Chart":

        categories = ['A', 'B', 'C', 'D']

        values = [4, 7, 1, 8]

        ax.bar(categories, values)

    elif selected\_graph == "Pie Chart":

        labels = ['A', 'B', 'C', 'D']

        sizes = [15, 30, 45, 10]

        ax.pie(sizes, labels=labels, autopct='%1.1f%%')

    elif selected\_graph == "Line Chart":

        x = [1, 2, 3, 4, 5]

        y = [2, 3, 5, 7, 11]

        ax.plot(x, y, marker='o')

    elif selected\_graph == "Histogram":

        sns.histplot(data=tips['total\_bill'], kde=True, ax=ax)

    # Renderiza o gráfico na área de exibição

    canvas = FigureCanvasTkAgg(fig, master=graph\_frame)

    canvas.draw()

    canvas.get\_tk\_widget().pack(fill=tk.BOTH, expand=True)

# Função para exibir o DataFrame de exemplo

def display\_dataframe():

    for widget in data\_frame.winfo\_children():

        widget.destroy()

    columns = list(tips.columns)

    tree = ttk.Treeview(data\_frame, columns=columns, show="headings")

    for col in columns:

        tree.heading(col, text=col)

        tree.column(col, width=100)

    for \_, row in tips.iterrows():

        tree.insert("", "end", values=list(row))

    tree.pack(fill=tk.BOTH, expand=True)

# Cria a janela principal

root = tk.Tk()

root.title("Data Visualization Interface")

root.geometry("1000x600")

# Menu lateral para selecionar o tipo de gráfico

menu\_frame = tk.Frame(root, width=200, bg="lightgrey")

menu\_frame.pack(side="left", fill="y")

menu\_label = tk.Label(menu\_frame, text="Select Graph Type", bg="lightgrey")

menu\_label.pack(pady=10)

graph\_types = ["Scatter Plot", "Bar Chart", "Pie Chart", "Line Chart", "Histogram"]

for graph in graph\_types:

    button = tk.Button(menu\_frame, text=graph, command=lambda g=graph: update\_graph(g))

    button.pack(fill="x", padx=10, pady=5)

# Título centralizado

title\_label = tk.Label(root, text="Data Visualization", font=("Helvetica", 16))

title\_label.pack(pady=10)

# DataFrame com dados de exemplo

data\_frame = tk.Frame(root)

data\_frame.pack(fill="x", padx=20, pady=10)

display\_dataframe()

# Área de exibição do gráfico

graph\_frame = tk.Frame(root)

graph\_frame.pack(fill="both", expand=True, padx=20, pady=10)

# Inicializa a interface com um gráfico de dispersão padrão

update\_graph("Scatter Plot")

# Inicia a interface gráfica

root.mainloop()

**Comentários sobre o Código**

1. **Imports Necessários**:
   * tkinter e ttk são usados para criar a interface gráfica.
   * FigureCanvasTkAgg de matplotlib.backends.backend\_tkagg para renderizar gráficos Matplotlib no Tkinter.
   * matplotlib.pyplot e seaborn para criação dos gráficos.
   * pandas para manipulação de dados.
2. **Função update\_graph**:
   * Limpa o gráfico existente e gera um novo gráfico baseado na seleção do menu.
   * Utiliza seaborn e matplotlib para criar diferentes tipos de gráficos.
3. **Função display\_dataframe**:
   * Exibe um DataFrame de exemplo na interface usando ttk.Treeview.
   * Configura as colunas e adiciona as linhas do DataFrame ao Treeview.
4. **Janela Principal**:
   * Configurada com um menu lateral, um título central, um frame para o DataFrame, e uma área de exibição para o gráfico.
5. **Menu Lateral**:
   * Botões para selecionar o tipo de gráfico. Cada botão chama a função update\_graph com o tipo de gráfico correspondente.
6. **Inicialização da Interface**:
   * A interface é inicializada com um gráfico de dispersão padrão.

O código acima fornece uma interface gráfica completa para visualização de dados e criação de gráficos.

É uma base para projetos que requerem visualização interativa e pode ser personalizado conforme as necessidades específicas.

**Materiais para continuar os estudos, sugestões:**

**Leituras:**

* Python Tkinter: Aprenda a criar interfaces gráficas com Python (introdução com códigos, passo a passo):
  + <https://awari.com.br/python-tkinter-aprenda-a-criar-interfaces-graficas-com-python/>
* Tkinter: Interfaces gráficas em Python
  + <https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956>
* Como criar Interfaces Gráficas (GUI) com Tkinter e Python
  + <https://pythonacademy.com.br/blog/interfaces-graficas-com-tkinter-e-python>

**Vídeos – com códigos – passo a passo:**

* Criando uma interface Grafica Moderna em Python com 3 Linhas de Codigo - Aula01 || #python #tkinter <https://www.youtube.com/watch?v=hIJ6sf0x3Yw>
* Como Criar uma Tela em Python Para Seus Códigos - [Interface Gráfica Intuitiva com Tkinter] - <https://www.youtube.com/watch?v=AiBC01p58oI>