```
"""
Claudinei de Oliveira - pt-br - utf-8 - 22-04-2024
010aula2204.py
"""

from tkinter import *
import tkinter.font as tkFont

# Função que redimensiona a fonte dos widgets dentro do frame
def resize_font(event):
    # Calcula o novo tamanho da fonte baseado na altura do labelframe
    new_size = max(10, int(frame.winfo_height() * 0.05))
    # Atualiza o tamanho da fonte
    font.configure(size=new_size)
    # Aplica o novo tamanho de fonte a todos os labels, entries e display_label
    label.config(font=font)
    entry.config(font=font)
    display_label.config(font=font)
    # Aplica a nova fonte aos novos widgets (labels e entries adicionados)
    for widget in additional_entries:
        widget.config(font=font)

# Função que captura o texto digitado no campo 'entry' e exibe no display_label
def adicionar():
    name = entry.get()
    display_label.config(text='O nome digitado foi: "{}"'.format(name))
```

```
# Criação da janela principal do aplicativo
app = Tk()
app.title('Análise e Desenvolvimento de Sistemas')  # Define o título da janela
app.configure(background='#F8F8FF')  # Configura a cor de fundo da janela principal
```

```
# Configuração do layout da janela principal usando grid
app.grid_rowconfigure(0, weight=4)  # Configura a primeira linha da grid com peso 4
app.grid_rowconfigure(1, weight=6)  # Configura a segunda linha da grid com peso 6
app.grid_columnconfigure(0, weight=1)  # Configura a coluna da grid com peso 1
```

```
# Criação do LabelFrame (uma moldura com um rótulo)
frame = LabelFrame(app, text='Cadastro', borderwidth=1, relief='solid', background='white')
frame.grid(row=0, column=0, sticky='NSEW', padx=3, pady=3)  # Posiciona o frame na grid
frame.bind('<Configure>', resize_font)  # Liga o evento de redimensionamento à função resize_font
```

```
# Criação de uma fonte inicial com tamanho 10
font = tkFont.Font(size=10)
```

```
# Criação do Label para instrução ao usuário
label = Label(frame, text='Digite um nome:', font=font, background='white')
label.grid(row=0, column=0, sticky='W', padx=(3, 20))  # Posiciona o label na grid do frame
```

```
# Criação do Label para instrução ao usuário
label = Label(frame, text='Digite um nome:', font=font, background='white')
label.grid(row=0, column=0, sticky='W', padx=(3, 20))  # Posiciona o label na grid do frame
```

```
# Criação do botão para submeter o nome digitado
submit_button = Button(frame, text='Adicionar', command=adicionar, font=font)
submit_button.grid(row=1, column=1, sticky='S', pady=10)  # Posiciona o botão na grid do frame

# Criação de um Label que exibirá o nome digitado após clicar em "Adicionar"
display_label = Label(frame, text='', font=font, background='white')
display_label.grid(row=2, column=0, columnspan=2, sticky='W', padx=3)  # Posiciona o display_label na grid

# Lista para armazenar os novos labels e entries adicionados
additional_entries = []

# Loop para criar 4 pares de labels e entries adicionais
for i in range(4):
    # Criação de um novo Label para cada par
    new_label = Label(frame, text=f'Campo {i*2 + 1}:', font=font, background='white')
    new_label.grid(row=3 + i, column=0, sticky='W', padx=(3, 20))  # Posiciona o novo label na grid do frame

    # Criação de um novo Entry para cada par
    new_entry = Entry(frame, font=font)
    new_entry.grid(row=3 + i, column=1, sticky='EW')  # Posiciona o novo entry na grid do frame

    # Adiciona os novos widgets à lista para serem redimensionados posteriormente
    additional_entries.extend([new_label, new_entry])

# Configura a coluna 1 do frame para expandir junto com a janela
frame.grid_columnconfigure(1, weight=1)
```

```
# Define o tamanho mínimo para a janela principal
app.minsize(width=800, height=480)

# Inicia o loop principal da aplicação para manter a janela aberta
app.mainloop()
```

ANÁLISE DETALHADA

1. **Importação dos Módulos**:
   - from tkinter import *: Importa todos os componentes da biblioteca Tkinter.
   - import tkinter.font as tkFont: Importa o módulo para manipular fontes em Tkinter.

2. **Função resize_font(event)**:
   - Essa função é chamada automaticamente quando o frame é redimensionado. Ela ajusta dinamicamente o tamanho da fonte dos elementos dentro do frame, garantindo que o texto permaneça legível e proporcional ao tamanho da janela.

3. **Função adicionar()**:
   - Captura o texto inserido pelo usuário no campo entry e o exibe no display_label quando o botão "Adicionar" é clicado.

4. **Configuração da Janela Principal (app)**:
   - Cria a janela principal e define seu título e cor de fundo.

- Configura o layout da janela usando grid, com 2 linhas e 1 coluna. As linhas têm pesos diferentes para controlar como o espaço é distribuído verticalmente.

5. **Criação do LabelFrame (frame)**:
   - Um LabelFrame é usado para agrupar os widgets relacionados ao cadastro. Ele tem uma borda sólida e um fundo branco, com um título "Cadastro".
   - O LabelFrame é posicionado na grid da janela principal.
6. **Criação e Posicionamento dos Widgets**:
   - Um Label, Entry, e Button são criados e posicionados dentro do frame. Esses widgets formam o formulário básico de entrada de dados.
   - Um display_label é criado para exibir o texto digitado após o botão ser clicado.
7. **Adição de Mais Campos**:
   - O loop for cria mais quatro pares de Label e Entry, que são posicionados em novas linhas dentro do frame. Esses novos widgets são adicionados à lista additional_entries para que também sejam redimensionados pela função resize_font.
8. **Finalização**:
   - O frame é configurado para que a segunda coluna (coluna 1) se expanda dinamicamente.
   - Define o tamanho mínimo da janela principal e inicia o loop principal da aplicação, mantendo a interface aberta até que o usuário a feche.

O script acima constrói uma interface gráfica em Python usando Tkinter, com um layout organizado e que se adapta ao redimensionamento da janela.

Um segundo tipo de Layout que podemos utilizar para criar Interfaces gráficas utilizando o Tkinter é o layout: **place()**

- Entenda que o **Tkinter** é uma biblioteca gráfica ou (GUI) integrada ao Python. Isso significa que não é preciso instalá-la.

- Normalmente para criar uma GUI usando o Python e Tkinter aplique a seguinte lógica:

1º - Importe a biblioteca tkinter;

2º - Defina o elemento (widgets) que receberá a a classe Tk(), que é a janela gráfica;

3º - Defina outros elementos (widgets) da janela

4º - Defina a função que irá manter a janela ativa: (nome_do_app.manillop())

Definindo uma janela simples no tkinter

```
from tkinter import *

app =  Tk()


app.mainloop()
```

- Ao aplicar os comandos acima, a janela fica responsiva. A função mainloop() executa o *script*.

Configurações iniciais de uma janela no tkinter

```
from tkinter import *

app =  Tk()

app.title('Análise e Desenvolvimento de Sistemas')

app.geometry('800x480')


app.mainloop()
```

- Sobre acesse: https://html-color-codes.info/Codigos-de-Cores-HTML/

Pilares do desenvolvendo Interfaces Gráficas: **Python** e **Tkinter**

- Gerenciador de layout: trata da forma que os *widget's* serão organizados na tela.

- ***Widget's***: são os componentes de tela.

- **Eventos:** é qualquer interação captada pela biblioteca tkinter e que pode acionar uma função. Em Python temos os eventos de clique e de teclado por exemplo. Fonte: https://docs.python.org/pt-br/dev/library/tkinter.html

- **Gerenciador de *Layout***: partindo da premissa que em interfaces gráficas temos *containers* e/ou *widget's* sobrepostos em camadas, para cada um deles o gerenciador de *layout* será o responsável por posicioná-los na tela conforme o projeto da janela. Em *tkinter* temos três gerenciadores de layout, **place()**, pac() e grid().

- Gerenciador de layout ***place()***: os *containers* e/ou ***widget's*** serão posicionados com base no plano cartesiano aplicando as coordenadas x e y. Nesse gerenciador as coordenadas iniciam no canto superior direito da janela e os valores em pixels.

Definido padrão de nossa janela base, é o momento de definir quais *Widgets* (elementos) irão compor a janela.

WIDGETS: **em áreas lógicas**: é todo componente que compões uma GUI: janelas, botões, rótulos de texto, botões de opção e campos de texto, caixa de mensagens, frames etc.:

- ***Container*:** é uma parte gráfica que pode conter outros *widget's;*

- ***Top Level*:** é uma janela que se sobrepões à janela principal e normalmente é independente. Exemplo uma janela de mensagem onde o usuário deve tomar uma decisão como: continua ou volta, salva ou cancela.

- ***FRAME*:** é um quadro que divide a janela em diversas partes isoladas, permitindo a manipulação de *widget's* de maneira independente.

- *Child-Parent*: sempre que inserimos um *widget* em um *container* estamos definindo uma relação *child-parent*. O pai, do *widget* será a janela que ele foi inserido. Caso o *widget* seja um *container frame* que foi inserido sobre outro *widget* tipo *janela* o pai do *frame* será a *janela*.

- ***Label:*** é uma função usada para inserir textos não editáveis nas janelas ou frames. Características como cor da fonte, tipo da fonte, tamanho da fonte, estilo da fonte pode ser aplicados sobre este *widget*. Mais informações sobre esse *widget* estão disponíveis em: https://www.delftstack.com/pt/howto/python-tkinter/how-to-set-font-of-tkinter-text-widget/

- Para entrada de texto em variáveis usamos as funções **Entry()** e **Text()**:

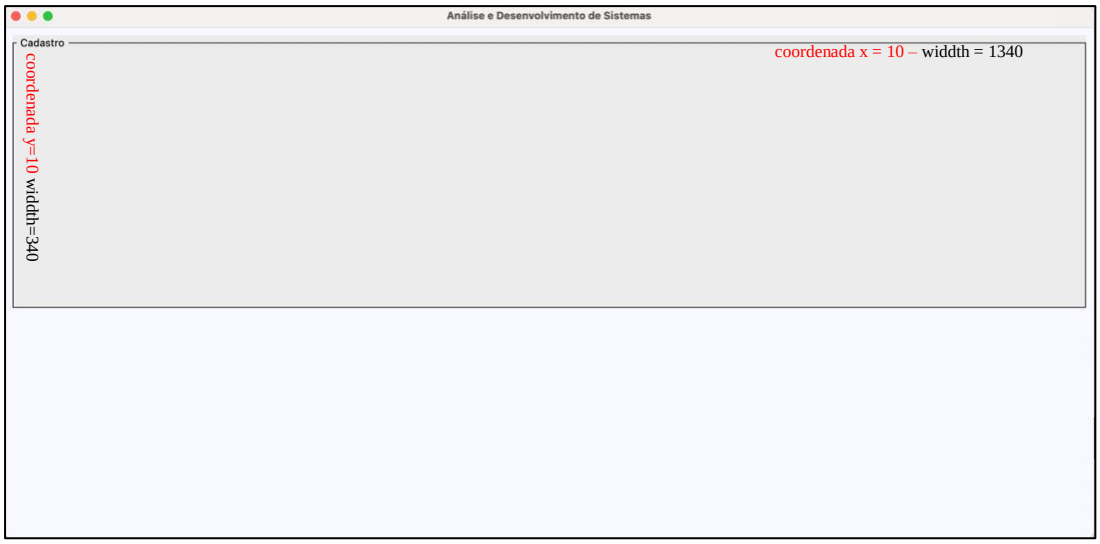
- **Entry()** é usado quando precisamos preencher o valor de um campo ou linha.

- **Text()** é usado quando precisamos preencher múltiplas linhas.

Implementando e posicionando um *frame* dentro de uma *janela*:

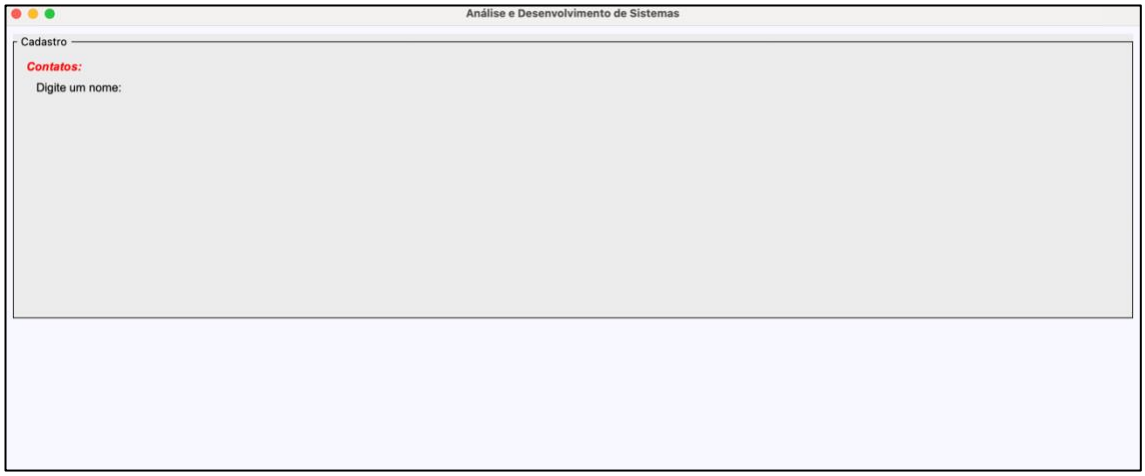
```
from tkinter import *

app = Tk()
app.title('Análise e Desenvolvimento de Sistemas')
app.geometry('800x480')  # largura e altura
app.configure(background='#F8F8FF')
app.resizable(True, True)
app.maxsize(width=1360, height=670)
app.minsize(width=800, height=480)

# Valores possíveis para O Frame: relief = flat, raised, sunken, solid
# frame = Frame(app, borderwidth = 1, relief = 'solid')
# frame.place(x = 10, y = 10, width = 1340, height = 340)

# Adicionando um argumento 'text' para dar nome Frame = LabelFrame
frame = LabelFrame(app, text=" Cadastro ", borderwidth=1, relief='solid')
frame.place(x=10, y=10, width=1340, height=340)

app.mainloop()  # Correção: adicionar parênteses
```

Resultado



Implementando e posicionando um *Label* dentro do *Frame*:

```
from tkinter import *

app =  Tk()

app.title('Análise e Desenvolvimento de Sistemas')
app.geometry('800x480') # largura e altura
app.configure(background = '#F8F8FF')
app.resizable(True, True)
app.maxsize(width=1360, height=670)

frame = LabelFrame(app, text=" Cadastro ", borderwidth = 1, relief = 'solid')
frame.place(x = 10, y = 10, width = 1340, heigh = 340)

#  inserindo e posicionando um Label's dentro do frame

lb_1 = Label(frame, text = 'Contatos: ',fg = 'red', font=("Arial",14, "italic", "bold"))
lb_1.place(x = 15, y = 10, width = 70, height = 20)

lb_2 = Label(frame, text = 'Digite um nome: ',font=("Arial",14))
lb_2.place(x = 20, y = 35, width = 120, height = 20)


app.mainloop()
```



Implementando e posicionando Entry dentro do Frame:

```
from tkinter import *

app =  Tk()

app.title('Análise e Desenvolvimento de Sistemas')
app.geometry('1360x670')
app.configure(background = '#F8F8FF')
app.resizable(True, True)
app.maxsize(width=1360, height=670)

frame = LabelFrame(app, text=" Cadastro ", borderwidth = 1, relief = 'solid')
frame.place(x = 10, y = 10, width = 1340, heigh = 340)

lb_1 = Label(frame, text = 'Contatos: ',fg = 'red', font=("Arial",14, "italic", "bold"))
lb_1.place(x = 15, y = 10, width = 70, height = 20)

lb_2 = Label(frame, text = 'Digite um nome: ',font=("Arial",14))
lb_2.place(x = 20, y = 35, width = 120, height = 20)

#  inserindo e posicionando um Entry dentro do frame - entrando com valores

nome = Entry(frame, font=('Arial', 14))  # altera a fonte para Arial e o tamanho para 14
nome.place(x =135, y = 35, width = 400, height = 20)

app.mainloop()
```

- A função **Entry(),** permite que o usuário digite dados dentro de um campo e que estes sejam adicionados em uma variável.

- Para definir o foco automaticamente em um campo quando estamos utilizando **Entry** quando a janela é aberta, aplicamos o método focus_set() logo após a criação do campo **Entry**. Isso garantirá que o campo de entrada estará pronto para receber a digitação assim que a interface gráfica for exibida.

```
# Definindo o foco no campo "nome"
nome.focus_set()
```

- Para capturar os dados digitados que estão na variável, será necessário criar um **BOTÃO** e aplicar um evento de clique para que ele chame a ação / função. Para isso execute os seguintes passos:

Capturando dados digitados:

- Antes de criar a janela, defina uma função que será acionada ao clique no botão no início do script logo após as importações.
- No final do código antes da função mainloop(), defina um label que irá receber os dados e depois implemente o botão “Capturar dados”.

Capturando dados digitados - implementando um botão

```
from tkinter import *

# Definindo a função capturar ao clicar no botão
def capturar():
    Ib_3['text'] = nome.get()

app = Tk()
app.title('Análise e Desenvolvimento de Sistemas')
app.geometry('1360x680')  # largura e altura
app.configure(background='#F8F8FF')
app.resizable(True, True)
app.minsize(width=1360, height=670)
app.maxsize(width=1360, height=670)

# Adicionando um argumento 'text' para dar nome Frame = LabelFrame
frame = LabelFrame(app, text=" Cadastro ", borderwidth=1, relief='solid')
frame.place(x=10, y=10, width=1340, height=340)

# Inserindo e posicionando um Label dentro do frame
Ib_1 = Label(frame, text='Contatos: ', fg='red', font=("Arial", 14, "bold italic"))
Ib_1.place(x=15, y=10, width=70, height=20)

Ib_2 = Label(frame, text='Digite um nome: ', font=("Arial", 14))
Ib_2.place(x=20, y=35, width=120, height=20)

# Inserindo e posicionando um Entry dentro do frame - entrando com valores
nome = Entry(frame, font=('Arial', 14))  # Altera a fonte para Arial e o tamanho para 14
nome.place(x=135, y=35, width=400, height=20)

# Definindo o foco no campo "nome"
nome.focus_set()

# Definindo a label que irá capturar os dados digitados ao clicar no
Ib_3 = Label(app, text='', font=("Arial", 14), background='#F8F8FF')  # Fundo branco
Ib_3.place(x=135, y=370, width=400, height=20)

# implementando um botão:
btn_captura = Button(app, text = 'Capturar dados', font=("Arial", 14, "bold"), command = capturar)
btn_captura.place(x = 490, y = 300, width = 125, height = 40)

app.mainloop()
```

Descrição Detalhada da linha de comandos button()

```
btn_captura = Button(app, text = 'Capturar dados', font=("Arial", 14, "bold"), command = capturar)
```

1. **btn_captura = Button(...)**:
   - **Button**: É uma classe do módulo Tkinter que cria um widget de botão. Esse botão pode ser clicado pelo usuário para realizar uma ação.
   - **btn_captura**: É a variável que armazena a instância do botão criado. Esse nome pode ser qualquer nome válido de variável, e você o usará para manipular o botão posteriormente no código (como para posicioná-lo na janela).

2. **app**:
   - **app**: É o contêiner (ou janela principal) onde o botão será colocado. Isso indica que o botão btn_captura pertence à janela principal app.

3. **text='Capturar dados'**:
   - **text**: É um parâmetro que define o texto exibido no botão. Nesse caso, o botão exibirá "Capturar dados" como rótulo.
   - **'Capturar dados'**: É a string que será mostrada como o texto do botão.

4. **font=("Arial", 14, "bold")**:
   - **font**: Define a fonte do texto exibido no botão.
   - **("Arial", 14, "bold")**:
     - **"Arial"**: Especifica o tipo de fonte a ser usado no texto do botão. Aqui, "Arial" é a fonte escolhida.
     - **14**: Define o tamanho da fonte em pontos. Nesse caso, o tamanho é 14.
     - **"bold"**: Define o estilo da fonte. "bold" torna o texto em negrito.

5. **command=capturar**:
   - **command**: Especifica a função que será chamada quando o botão for clicado.
   - **capturar**: É o nome da função que será executada quando o usuário clicar no botão. A função capturar já deve ter sido definida anteriormente

no código. Note que capturar é passado sem parênteses, o que significa que a função será chamada somente quando o botão for clicado, e não imediatamente ao criar o botão.

Funcionamento: quando essa linha de código é executada, ela cria um botão na janela principal (app) com o nome "Capturar dados". Esse botão usa a fonte Arial, tamanho 14, e em negrito. Quando o usuário clica nesse botão, a função capturar é chamada, executando o que quer que tenha sido definido dentro dessa função.

Lembre-se: esse tipo de estrutura é muito comum em interfaces gráficas, onde botões são usados para iniciar ações específicas dentro do aplicativo.

Mão na massa:

a) A partir do script de exemplo, refatore-o para que ele receba 5 nomes e os armazene em uma lista. Após isso, apresente os nomes na tela, um embaixo do outro.

Norte para a resolução:

1. **Lista nomes:**
   - Uma lista chamada nomes deve ser criada para armazenar os 5 nomes digitados.
2. **Função capturar():**
   - Esta função deve capturar o nome digitado no campo de entrada, armazena-o na lista nomes, e após limpa o campo de entrada para o próximo nome.
   - Quando os 5 nomes forem capturados, crie uma função para exibir_nomes() e não esqueça de chamá-la.
3. **Função exibir_nomes():**
   - Esta função deve exibir todos nomes armazenados na lista, posicionando-os um embaixo do outro na janela.
   - O espaçamento entre os nomes é controlado pela variável y_offset e o índice i.
4. **Campo Entry e Botão Button:**
   - O campo de entrada nome e o botão btn_captura foram ajustados para capturar e processar a entrada dos nomes.
5. **Posicionamento Dinâmico:**
   - Os nomes capturados são exibidos dinamicamente na tela após a inserção do quinto nome.

b) Refatore o script para que ele receba 5 valores booleanos (True/False) e os armazene em uma lista. Após isso, apresente os valores capturados na tela, um embaixo do outro.

c) Refatore

d) Refatore o script para que ele receba 5 valores decimais (float) e os armazene em uma lista. Após isso, apresente os valores capturados na tela, um embaixo do outro.

REFERÊNCIAS

F. V. C. Santos, Rafael. **Python: Guia prático do básico ao avançado (Cientista de dados Livro 2)**. 2ª Edição, 2020. Edição do Kindle.

JUNIOR, Dilermando. Algoritmos e Programação de Computadores (Portuguese Edition) (pp. 326-327). GEN LTC.2ª Edição do Kindle, 2020.

MANZANO Jang, YAMATUMI WY . Free Pascal: **programação de computadores**. São Paulo: Érica ; 2007.

MENEZES, Nilo Ney Coutinho. **Introdução à programação com Python**. 3ª Edição. 2019. Editora Novatec - São Paulo - SP - Brasil.

*