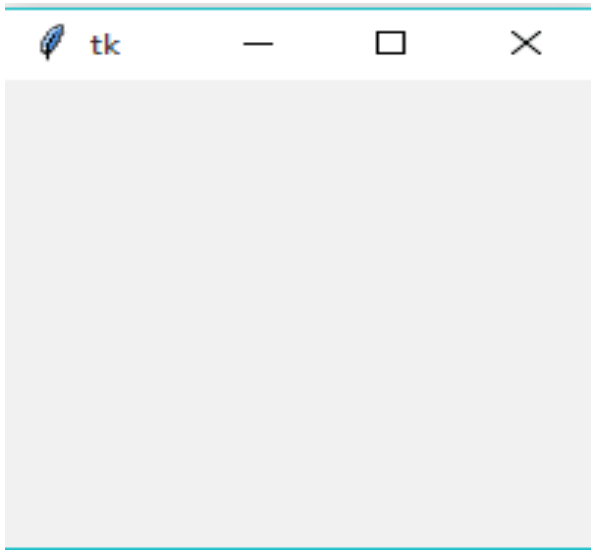
Tkinter (GUI) - Python

Prof. Dr. Alex Sandro Roschildt Pinto

Mateus Manoel Pereira

Tkinter

- Tkinter é a biblioteca padrão do python para construir uma interface gráfica para usuário. É muito simples e prático de usar, além de ter uma interface orientada a objeto que facilita a utilização e contribui para reutilização de código.



Visual de uma janela criada no Tkinter usando o SO Windows 10.

Exemplo de formulário (simples)

```
>> janelaPrincipal = Tk()
>> stringEmail = StringVar(value='Email')
>> email = Entry(janelaPrincipal, textvariable=stringEmail)
>> email.pack(pady=15, side=TOP)
>> stringUsername = StringVar(value='Username')
>> username = Entry(janelaPrincipal, textvariable=stringUsername)
>> username.pack(pady=15, side=TOP)
>> stringPassword = StringVar(value="********")
>> password = Entry(janelaPrincipal, textvariable=stringPassword, show='*')
>> password.pack(pady=15, side=TOP)
>> submit = Button(janelaPrincipal, text='SUBMIT', command= lambda:
enviarFormulario())
>> submit.pack(pady=15, side=TOP)
>> janelaPrincipal.mainloop()
```

Exemplo de formulário (simples)

Tkinter

- Vamos dividir sua utilização em 3 passos:
  - Criar a janela principal.
  - Adicionar elementos de interação.
  - Iniciar o laço de evento principal para rastrear eventos gerados pelo clique do utilizador.

Hands-on (Mão na massa)

- Com o intuito de demonstrar como construir uma aplicação fazendo uso da ferramenta vamos a partir de agora começar a montar um programa de criação de personagem. Quem sabe você não faz um jogo para esse pergonagem jogar!

Criando a janela principal

- O primeiro passo é muito simples, vamos instanciar a janela principal da nossa aplicação chamando o construtor da classe Tkinter.

```
>> janelaPrincipal = tkinter.Tk()
```

- Obs.: Não esqueça de importar todos os módulos do tkinter.

```
>> from tkinter import *
```

Elementos de intereção (Widgets)

- Widgets são elementos gráficos que geralmente executam uma ação ou acessam um serviço, mas também temos widgets que servem apenas para fins estéticos.

Label com imagem.

- A seguir vamos ver a lista completa de widgets que o Tkinter fornece.

Elementos de intereção (Widgets)

- Button
- Canvas
- Checkbutton
- Entry
- Frame
- Label
- Listbox
- Menubutton
- Menu
- Message
- Radiobutton
- Scale
- Scrollbar
- Text
- Toplevel
- Spinbox
- PanedWindow
- LabelFrame
- tkMessageBox

Inserindo Widgets

- Agora vamos inserir nosso primeiro widget na JanelaPrincipal criada anteriormente, a janela do personagem, essa janela vai servir para posicionar outros widgets em uma determinada área, para fazer isso temos como opção o Frame.
- Construtor de um Frame:

```
>> Frame(master, option, ...)
```

- Sendo que:
  - **master** = Janela na qual o Frame estará contido.
  - **option** = Opções de customização do Frame, como tamanho e cor do plano de fundo.

Inserindo Widgets

- Agora que temos o Frame criado vamos adicionar o personagem a ele, para isso precisamos de um widget que possa conter uma imagem, assim podemos optar entre uma Label ou uma área Canvas, neste exemplo vamos utilizar uma Label.
- Construtor de uma Label:

```
>> Label(master, option, ...)
```

- Sendo que:
  - **master** = Janela na qual a Label estará contida.
  - **option** = Opções de customização da Label, como tamanho e cor do plano de fundo.

Posicionamento

- No Tkinter existem três formas de posicionar um widget em uma janela.
- **Pack(options):**
  - Posiciona os widgets em blocos, um do lado do outro.
- **Grid(options):**
  - Posiciona os widgets no estilo matriz, com linhas e colunas.
- **Place(options):**
  - Posiciona o widget na posição determinada.

Inserindo Widgets

```
>> janelaPersonagem = Frame(janelaPrincipal, bg='black', width='300', height='480')
>> janelaPersonagem.place(x='-10' ,y='160')
>> imgCorpo = Image.open('./corpo.png')
>> imgCorpo = imgCorpo.resize((550,300), Image.ANTIALIAS)
>> imgTkCorpo = ImageTk.PhotoImage(imgCorpo)
>> corpo = Label(janelaPersonagem, image=imgTkCorpo, bg='black')
>> corpo.pack()
```

Resultado...

Inserindo Widgets

- Que tal adicionarmos um campo de texto para inserir o nome do personagem? Vamos usar o widget Entry e um tipo especial de variável para armazenar o nome, StringVar.

- StringVar é uma das variáveis de rastreamento do Tkinter, ou seja, se o valor contido nessa variável mudar alguma ação pode ser tomada. Isso é muito útil para limitar o número de caracteres inseridos.

Inserindo Widgets

```
>> janelaNome = Frame(janelaPrincipal, width='300', height='60')
>>janelaNome.place(x='560', y='480')
>> textoNome = Label(janelaPrincipal, text='Nome', font=('Quicksand', 24, 'bold'), bg='black', fg='white')
>> textoNome.place(x='670', y='436')
>> nome = StringVar()
>> nome.trace('w', limitaEntrada)  #Próximo Slide
>> campoNome = Entry(janelaNome, textvariable=nome, font=('Quicksand', 24, 'bold'))
>> campoNome.place(width='300', height='60')
```

Inserindo Widgets

```
>>> def limitaEntrada(*args):
>>>     aux = nome.get()
>>>     if len(aux) > 16:
>>>         nome.set(aux[:16])
```

Resultado...

Inserindo Widgets

- Ok, já temos o nosso personagem e o nome dele, mas isso não é uma criação de personagem se não pudermos ao menos trocar seu cabelo não é mesmo? Está na hora de colocarmos alguns Buttons na nossa interface e configurá-los para trocar o cabelo do nosso personagem.
- É possível customizar cada Button separadamente através do seu constrututor.

```
>> botao = Button(janelaPrincipal, width='70', relief='raised',
bd=2, bg='#202020', activebackground='#202020',
highlightcolor='#202020', highlightbackground='black')
```

- Mas isso significa que se houver 300 Buttons tenho que fazer isso para cada um deles?

- Não, você pode criar sua própria classe herdando de Button.

```
class Botao(Button):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs, width='70', relief='raised', bd=2,
bg='#202020', activebackground='#202020',
highlightcolor='#202020', highlightbackground='black', command=
lambda: pressionaBotao(self))
        self.__isPressed = False
    def reseta(self):
        self.config(relief='raised', bg='#202020',
activebackground='#202020')
        self.__isPressed = False
    def isPressed(self):
        return self.__isPressed
    def setIsPressed(self):
        self.   isPressed = True
```

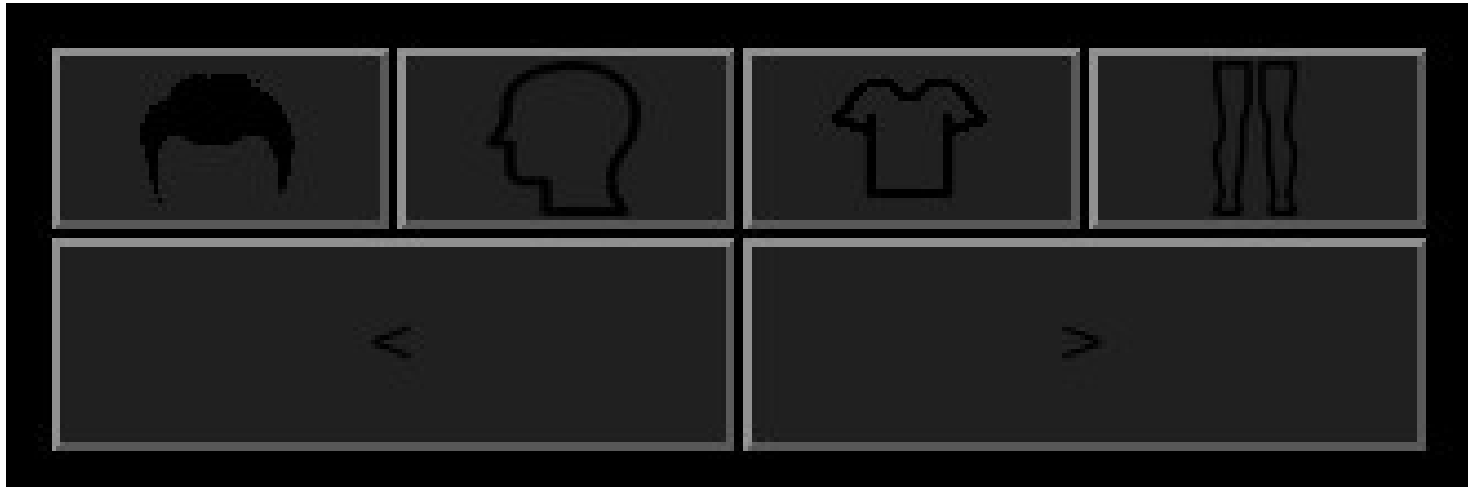
>>> pressionaBotao(self)

```
>>> def pressionaBotao(botao):
>>>     botaoCabelo.reseta( )
>>>     botaoFace.reseta( )
>>>     botaoTorso.reseta( )
>>>     botaoPerna.reseta( )
>>>     botao.config(relief='sunken', bg='#161616',
activebackground='#161616')
>>>     botao.setIsPressed( )
```

Inserindo Widgets

```
>> janelaEscolha = Frame(janelaPrincipal, bg='black',
width='300', height='420')
>> janelaEscolha.place(x='560', y='60')
>> botaoCabelo = Botao(janelaEscolha, image=imgTkCabelo)
>> botaoCabelo.grid(row=0, column=0)
>> botaoFace = Botao(janelaEscolha, image=imgTkFace)
>> botaoFace.grid(row=0, column=1)
>> botaoTorso = Botao(janelaEscolha, image=imgTkTorso)
>> botaoTorso.grid(row=0, column=2)
>> botaoPerna = Botao(janelaEscolha, image=imgTkPerna)
>> botaoPerna.grid(row=0, column=3)
```

Resultado...



Funcionamento do Botão

- Na classe Botao criada anteriormente definimos que o Button executa a função pressionaBotao( ), agora vamos ver como essa função funciona.

```
def pressionaBotao(botao):
    botaoCabelo.reseta()
    botaoFace.reseta()
    botaoTorso.reseta()
    botaoPerna.reseta()
    botao.config(relief='sunken', bg='#161616',
activebackground='#161616')
    botao.setIsPressed()
```

Trocando o Cabelo

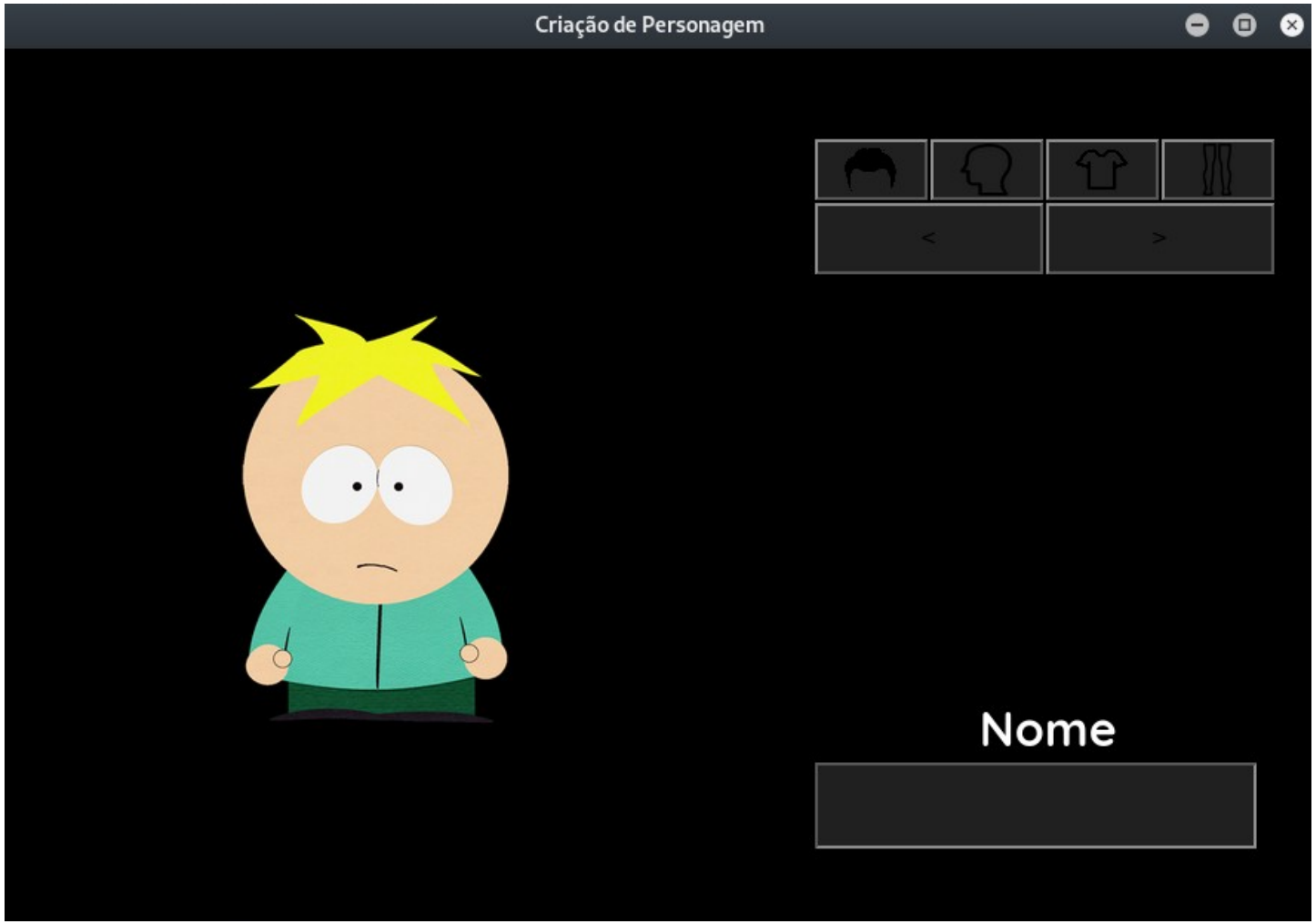
- Vamos agora configurar os botões de seta para trocar o cabelo do personagem.

>> command= lambda: percorreCabide(False)

- Diferente dos botões da classe Botao, esses executaram a função definida como **percorreCabide.**

- Você pode conferir o código completo no Moodle.

Resultado...



Iniciando o laço de evento principal

- No último passo precisamos iniciar um loop na janelaPrincipal, assim é possível capturar os eventos, como um clique, na janela.

```
>> janelaPrincipal.mainloop()
```

Para mais informações sobre a biblioteca...

- Consulte a documentação oficial:

https://docs.python.org/3/library/tk.html