Resumo sobre a Lib Tkfull na construção rápida de Interfaces Gráficas (version 0.0.3)

Autor: Adjailson Ferreira de Melo
prof.adjailson.ete@gmail.com

Desenvolvida para ser usada nas aulas de Desenvolvimento de Aplicação Desktop, disciplina do módulo I do curso de Desenvolvimento de Sistemas da ETEAVS – Escola Técnica Estadual Ariano Vilar Suassuna.

Atenção! A biblioteca está apenas na fase inicial, vários objetos são limitados ainda, a mesma está sendo construída aos poucos na medida em que precisamos para outros objetivos em sala. É apenas para agilizar outros conteúdos da disciplina, o foco não é a qualidade das interfaces e sim, manipulação dos dados sobre outras telas ou classes.

O Tkfull

O Tkfull é apenas uma classe que usa da biblioteca Tcl/Tk, a Tkinter para o desenvolvimento de Interface Gráfica no Python. O objetivo é apenas reduzir a codificação com Tkinter reescrevendo algumas funções padrões e reduzir o uso de variáveis dentro da classe herdada, usando assim apenas a variável da classe principal para manipulação de tudo dentro do objeto construído.

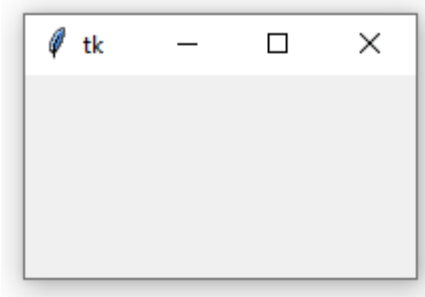
Como funciona? Você apenas precisa baixar a lib Tkfull (link no GitHub) e colocar dentro da pasta de seus projetos, considerando que você já tenha Pyhton 3 instalado em sua máquina, em seguida só construir seus novos programas fazendo o import como mostra a imagem logo abaixo:

```
from Tkfull import Janela
```

Nome do arquivo .py baixado. Nome da classe ou objeto dentro do Tkfull.py.

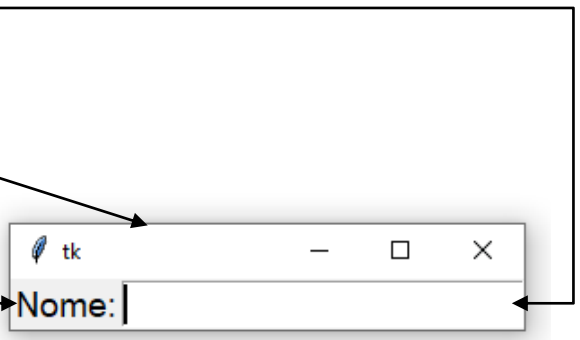
Para construir apenas a janela só precisamos fazer uma chamada da classe que foi importada. (Todos os exemplos são construídos na estrutura orientada a objetos) Ex.:

```
from Tkfull import Janela

class Exemplo:

    def __init__(self):
        self.classe = Janela()

        self.classe.start()

Exemplo()
```



A variável '**classe**' ou o objeto '**classe**' do tipo *Janela*, deve ser utilizada em todo nosso programa para manipular uma série de comportamentos dentro dessa janela. A mesma vai construir uma interface gráfica com base em um vetor bidimensional, onde o que tem nesse vetor são tipos de dados para apelidar os objetos reais como: *Label, Button, Entry, Combobox*. Ex.:

```
from Tkfull import Janela

class Exemplo:

    objetos = [['Nome:',input]]

    def __init__(self):
        self.classe = Janela()

        self.classe.gerar(self.objetos)

        self.classe.start()

Exemplo()
```



A função **gerar**() é quem faz toda a transformação (veja mais detalhes no uso dos sets), gerar() pode ser chamado várias vezes dentro do construtor da classe, mas, para cada chamada um novo Frame é gerado dentro da janela principal, o gerar() comporta os objetos no mesmo formato do *grid*() do tkinter, isso porque os dois são bidimensional, ou seja, para cada linha e coluna da matriz um objeto será gerado dentro de cada *row* e *column* do *grid*() sobre um Frame.

Regras dos tipos no vetor

String ou str – Se escrever uma string, um objeto *Label* será gerado com o texto da string, caso antes do texto descrito tenha um prefixo '*' (asterisco), um objeto *Button* será gerado com o texto informado depois do prefixo;

Entrada ou input – Se escrever a chamada do teclado input, um objeto *Entry* será gerado;

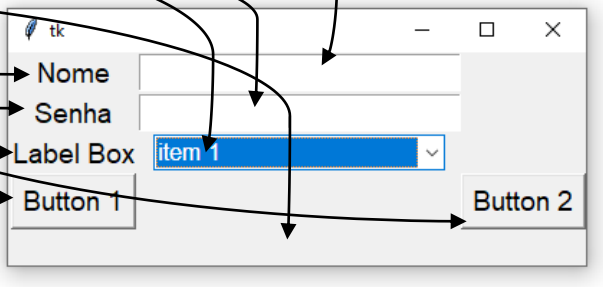
Tipo complex – Se escrever o type complex, um objeto *Entry* configurado para campo de senha será gerado;

Tipo tuplas – Se declaramos uma tupla, um objeto *Combobox* será gerado com os itens da própria tupla;

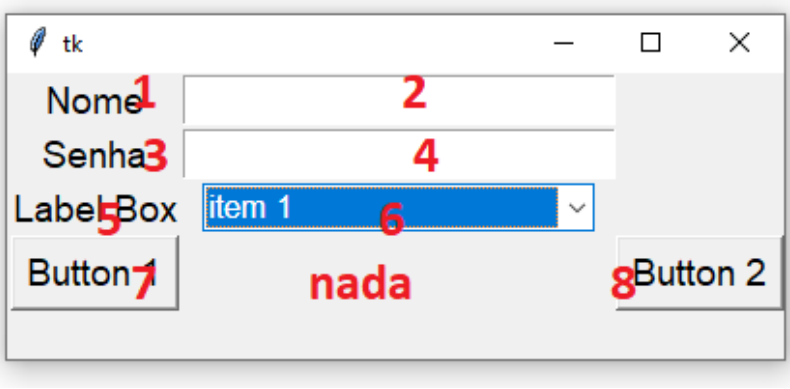
Tipo None – Se escrito sobre o vetor a posição desse grid não pode ser ocupada, ou seja, pula os próximos objetos para outra coluna.

Veja abaixo todos os exemplos:

```
from Tkfull import Janela

class Exemplo:

    objetos = [['Nome',input],
               ['Senha',complex],
               ['Label Box',('item 1','item 2')],
               ['*Button 1',None,'*Button 2']
               ]

    def __init__(self):
        self.classe = Janela()

        self.classe.gerar(self.objetos)

        self.classe.start()

Exemplo()
```



Tudo é respeitado de acordo com a matriz bidimensional, ou seja, para cada lista dentro do vetor uma nova linha também é criada na janela, para cada tipo dentro da mesma lista os objetos ficam um ao lado do outro. Mas, atenção! A ordem dos objetos guardados na classe não é bidimensional, e sim, uma lista simples na ordem de itens por itens construídos. Veja o exemplo da contagem ou ordem na lista:



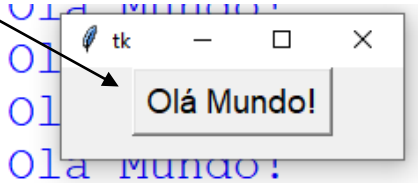
Esses elementos todos estão armazenados dentro da classe *Janela*() e sempre que precisamos devemos chamar ou apontar apenas essa posição numérica, se precisamos de todos, então a classe já tem a função *getObjetos*(), veja depois no Uso dos *gets*.

Uso dos sets

Em todos os sets vamos perceber que sempre existe um primeiro parâmetro do tipo inteiro (‘**posicao**’), ele é o mais importante, pois, é ele quem localiza qual objeto dentro da matriz, lembrando que essa posição não é a mesma de uma posição do vetor bidimensional da matriz, é apenas a ordem continua dos elementos gerados sobre um vetor comum. Reveja a explicação do uso do método gerar(), logo acima.

O método **setEvento**(int posicao, function funcao), os dois parâmetros são obrigatórios, o ‘**funcao**’ recebe um *def* que deve ser executado após aplicar um evento sobre o objeto declarado pela posição. Ex.:

```
from Tkfull import Janela

class Exemplo:

    objetos = [['*Olá Mundo!']]

    def __init__(self):
        self.classe = Janela()

        self.classe.gerar(self.objetos)
        self.classe.setEvento(1,self.click)

        self.classe.start()

    def click(self):
        print("Olá Mundo!")

Exemplo()
```



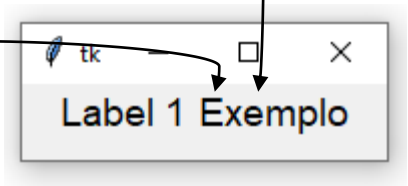

```
Olá Mundo!
Olá Mundo!
Olá Mundo!
```

No exemplo acima um botão chamado ‘*Olá Mundo!*’ foi criado na ordem 1, quando for clicado chame e função chamada ‘*click*’ que imprimi no terminal a mensagem ‘*Olá Mundo!*’.

O método **setTexto**(int posicao, str texto) – os dois parâmetros são obrigatórios, o parâmetro ‘**texto**’ recebe uma string para ser escrita no objeto que ta na posição passada, o mesmo

vai sobrescrever o ***text*** do objeto do tipo *Label, Button* ou o ***insert*** para inserir esse texto dentro de uma *Entry*. Ex.:

```
class Exemplo:
    objetos = [['Label 1','Label 2']]

    def __init__(self):
        self.classe = Janela()
        self.classe.gerar(self.objetos)

        self.classe.setTexto(2,'Exemplo')
        self.classe.start()

Exemplo()
```



Escreve o texto "*Exemplo*" no objeto da ordem 2, no caso a Label "*Label 2*".

O método **setEstilo**(int posicao, dict estilo) - os dois parâmetros são obrigatórios, o parâmetro '**estilo**' recebe um tipo dicionário para configurar ou aplicar estilo no objeto da posição da matriz. Ex.:

```
class Exemplo:
    objetos = [['*Botão']]
    estilo = {'width':10,'height':5,'font':('Arial',14)}

    def __init__(self):
        self.classe = Janela()
        self.classe.gerar(self.objetos)

        self.classe.setEstilo(1,self.estilo)
        self.classe.start()

Exemplo()
```

Aplique um estilo no objeto que ta na ordem 1 da matriz, no caso só existe um o *Button* '**Botão*', o que ele faz e aplicar o *config*() ou *configure*() do tkinter.

Uso dos gets

O método **getObjetos**() - retorna list de objetos criado na matriz;

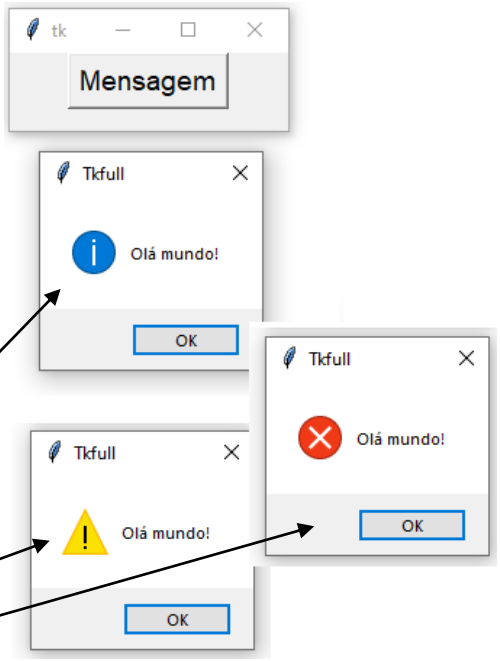
O método **getTexto**(int posicao) – retorna str, o texto do objeto escolhido na ordem ou posição escolhida;

Outras funções

O método **apagarTexto**(int posicao, int acao) – apenas o parâmetro '**posicao'** é obrigatório, se passar algo para '**acao'** deve ser os valores -1 ou 1, -1 apaga cada caractere da esquerda para a direita o 1 apaga cada caractere da direita para esquerda, nada passado para o parâmetro '**acao'** tudo será apagado.

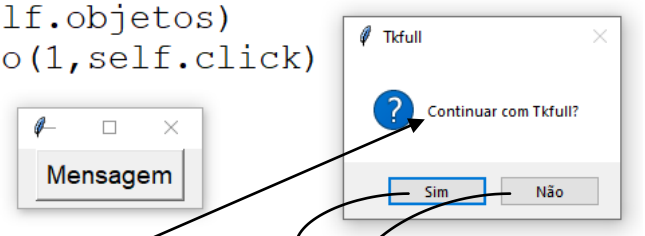
O método **mensagem**(str msg, int tipo) – o primeiro parâmetro é obrigatório você passa uma mensagem para ser aberta numa segunda tela, as chamadas alertas, o parâmetro 'tipo' você pode escolher entre dois inteiros 1 ou 2, o 1 manda uma mensagem para notificar advertência, 2

uma mensagem para informar notificar um erro, nada passado mensagem padrão apenas para notificar uma informação. Veja os exemplos dos tipos:




```
def click(self):
    self.classe.mensagem('Olá mundo!')
    self.classe.mensagem('Olá mundo!',1)
    self.classe.mensagem('Olá mundo!',2)
```

O método **pergunta**(str msg) – retorna True ou False dependendo da sua escolha sobre os botões apresentados. Ex.:




```
    self.classe.gerar(self.objetos)
    self.classe.setEvento(1,self.click)

    self.classe.start()

def click(self):
    if(self.classe.pergunta('Continuar com Tkfull?')):
        pass#Sim
    else:
        pass#Não
```