COLMEIA - Grupo de extensão em software e hardware livre

COLMÉIA

UDESC JOINVILLE

Minicurso de Tkinter
Interface gráfica utilizando Python

Ministrantes:

- Ediney Mendonça Da Silva Junior
- Matias Giuliano Gutierrez Benitez
- Pedro Kenzo Kawasaki

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA
JOINVILLE
CENTRO DE CIÊNCIAS TECNOLÓGICAS

QUEM SOMOS?

- O programa de extensão denominado Socialização em Software e Hardware Livre, tradicionalmente é realizado no ambiente do Grupo de pesquisa e desenvolvimento e é composto por projetos inter-relacionados cujas ações visam a divulgação do SW e HW livres e sua filosofia na comunidade em geral.

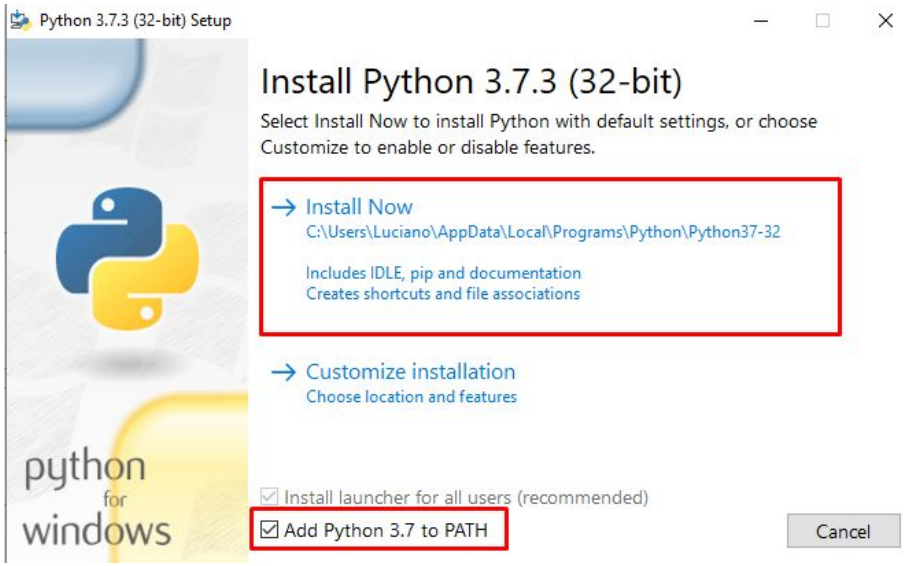
COLMÉIA

Módulo I – Resumo de Python

- **Programando em Python:**
  - Para instalar o python na máquina não é necessário um IDE (ambiente de desenvolvimento integrado), é possível criar um programa utilizando apenas do bloco de notas. Entretanto, IDEs possuem ferramentas que auxiliam na hora de programar, como compilação do código diretamente na interface, aviso de erros, sugestões de preenchimento, e muitas outras.
  - Por este motivo, utilizaremos o **VS Code** para facilitar a programação durante o minicurso. O VS Code não é a única IDE para Python, seguem exemplos de algumas das IDEs mais utilizadas para programação em Python.

Módulo I – Resumo de Python

- **Instalando a linguagem Python:**
  - Primeiro, será necessária a instalação do Python e da IDE selecionada para o curso.
  - Para ser feita a instalação da linguagem de programação, vá na aba Downloads no site https://www.python.org/
  - Assim que o arquivo for executado, deverá ser marcado a *checkbox* referente ao PATH antes de prosseguir com a instalação.

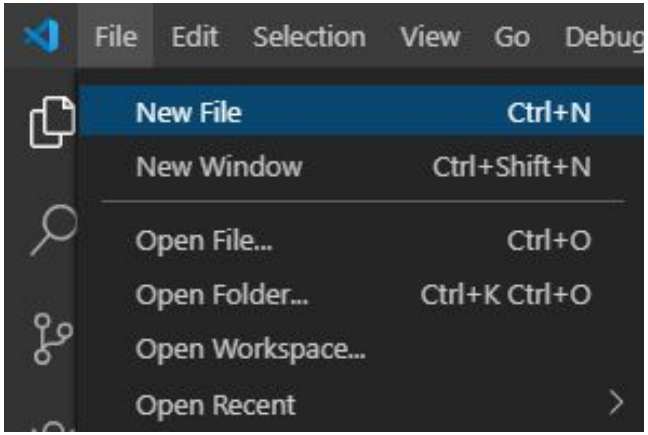


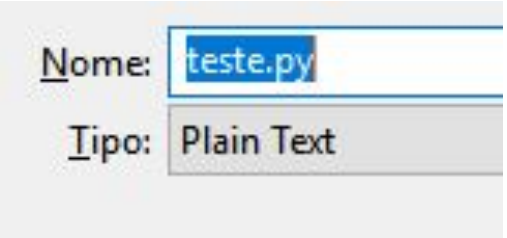
Módulo I – Resumo de Python

- Instalando a IDE VS Code:
  - Com o Python instalado, pode ser feita a instalação da IDE selecionada para o curso.
  - No site oficial do VS Code, https://code.visualstudio.com, faça o download do instalador.
  - A instalação é bem simples: Apenas siga os passos informados na aplicação. Caso utilize uma distribuição Linux verifique a loja de aplicativos nativa do sistema.

Módulo I – Resumo de Python

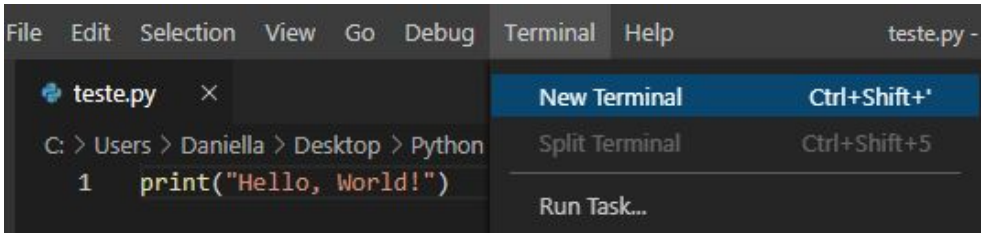
- Começando um projeto:
  - Para começar a programar em Python usando o VS Code, crie um novo File.
  - Vá na barra de ferramentas no canto superior esquerdo, selecione File → New File (também é possível usar o atalho Ctrl+N). Salve o arquivo com qualquer nome que quiser, não esquecendo do sufixo .py
  - A partir deste ponto você já pode programar em Python!

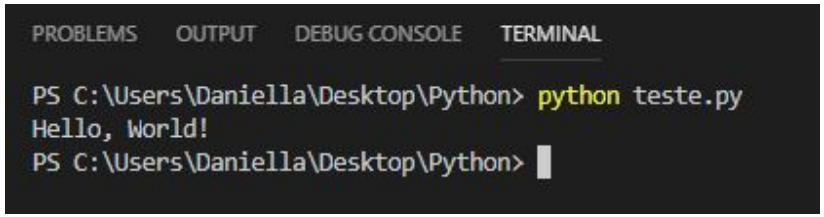




Módulo I – Resumo de Python

- Executando o código:
  - Para executar o código, basta ir até o terminal que a própria IDE apresenta, no canto superior direito (ou utilizando o comando Ctrl+Shift+´), e digitar o comando **python3 NomeDoCodigo.py**
  - Lembre-se de estar executando o arquivo na mesma pasta em que o salvou!





Módulo I – Resumo de Python

- Variáveis:
  - O recurso utilizado nos programas para escrever e ler dados da memória do computador é conhecido como *variável*. É simplesmente um espaço na memória o qual reservamos e nomeamos.
  - É um objeto (uma posição, frequentemente localizada na memória) capaz de reter e representar um valor ou expressão.
  - Para atribuir um valor a uma variável em Python só é preciso nomeá-la e definir seu valor utilizando de um caractere "=".

Ex:

```
nome = "Fernanda"
idade = 12
```

Módulo I – Resumo de Python

- Tipos:
  - Uma variável possui um tipo, que é uma identificação sobre seus atributos.
  - O interpretador alocará a memória da variável dependendo de seu tipo, tipos comuns de variáveis são inteiro (*int*), ponto flutuante (*float*), frase (*string*), e booleano (*bool*).
  - Entretanto, em Python, ao contrário de outras línguas de baixo nível, não é necessária a identificação de tipo durante a declaração de qualquer variável. Para fazer a conversão de um tipo a outro é somente necessário utilizar de <u>funções</u> pré-disponibilizadas pela linguagem.

Módulo I – Resumo de Python

- Operações com variáveis:
  - As variáveis, além de armazenar dados, podem ser manipuladas através de operadores.

Ex:
```
saldo = 100
divida = 80
result = saldo - divida
```

  - No exemplo acima, foi atribuída a uma variável o resultado de uma operação entre outras duas. No caso, a variável "result" neste momento deve armazenar o número inteiro 20.

Módulo I – Resumo de Python

- Operadores Aritiméticos:
  - Além da subtração, outras operações estão integradas à linguagem de programação, como por exemplo:
    - Soma: +
    - Subtração: -
    - Multiplicação: *
    - Divisão: /
    - Exponenciação: **
    - Divisão inteira: //
    - Resto de divisão inteira: %

Módulo I – Resumo de Python

- **Saída e entrada de dados:**
  - Assim como a maioria das linguagens, é possível receber dados através de um *input* pelo console.
  - Através do comando print, você poderá visualizar dados no console.

Ex:
```
idade = int(input("Qual é sua a idade?"))
print(idade)
```

- Esse programa receberá um número inteiro, inserido pelo usuário, armazena o valor na variável "idade" e depois imprime no console o valor digitado.

Módulo I - Resumo de Python

- **Saída e entrada de dados:**
  - A função print não imprime somente variáveis já declaradas. Para imprimir um texto ou o resultado de uma expressão numérica, é possível fazê-lo ao definir diretamente na função print().

Ex:
```
print(3+32)
Console: 35
print("A soma de três e trinta e dois é trinta e cinco.")
```

Console: A soma de três e trinta e dois é trinta e cinco.

Nota-se que para referir-se a strings, sempre é necessário a utilização de aspas!

Módulo I – Resumo de Python

- **Saída e entrada de dados:**
  - Além disso é possível juntar variáveis e texto utilizando “,”

Ex:
```
print(“A idade de Fernanda é”, idade)
```

Console: A idade de Fernanda é 16

  - Outra maneira de fazer isso é utilizando o método .format que funciona substituindo os termos “{}” com as variáveis determinadas.

Ex:
```
print("A idade de {} é {}.".format("Fernanda",16))
```

Console: A idade de Fernanda é 16.

Módulo I – Resumo de Python

- Saída e entrada de dados:

  - Exemplo de print 'antigo' 2.x
  - print('A %s tem %s anos' % (nome, idade))
  - print('A %s tem %.2f anos' %(nome,float(idade)))

  - Exemplo de print 'moderno' 3.x
  - print('A {0} tem {1} anos'.format(nome, idade))
  - print('A {0} tem {1:.4f} anos'.format(nome,float(idade)))

Módulo I – Resumo de Python

- TypeCast:
  - Por padrão o python declara a variável como String caso não especificarmos o tipo da variável.
  - O typecast serve para transformar a variável “string A” para outro tipo de variável como pro exemplo “int A”.

```
a = input("qual sua idade? ")
#como padrão o python irá pegar o valor como string
a = int(a)
#ou já no input
a = int(input("qual a sua idade? ")
```

Módulo I – Resumo de Python

<u>Exercício I</u>:

- Utilizando os conhecimentos anteriores, crie um programa que receba três notas de um aluno e seu nome, imprimindo no console o resultado da seguinte maneira:

Ex:

```
Aluno: Mark Zuckerberg
Media: 5.0
```

Módulo I – Resumo de Python

- Operadores Lógicos:
  - Para fazer comparações será necessário o uso de operadores lógicos como os seguintes:
    - Maior que: >
    - Menor que: <
    - Igual: ==
    - Diferente: !=
    - E: and
    - Ou: or
    - Não (Operador que inverte a verdade sobre um termo): not
    - Maior ou igual/Menor ou igual: >=, <=

Módulo I – Resumo de Python

- **Operadores Lógicos:**
  - Todas as afirmações a seguir são verdades.

A = 3 B=1 C=3

1. A <= C
2. B < A
3. A == C
4. C != B
5. A > B and C > B
6. B < C or C != A
7. B < A and not C < B
8. not B < A and not B < C
9. A == C and not A == B
10. ((not A<B or A<B) and (not B<C or B<C)) or A == C

Módulo I – Resumo de Python

- Condicionais:
  - Agora entramos na área de lógica de programação, utilizando de condicionais podemos decidir que partes do código serão executadas dependendo de certos parâmetros.

  - **if** (parâmetro): Caso (afirmação) seja verdade, execute.
  - **elif** (parâmetro): Caso a primeira não seja, e essa sim, execute.
  - **else**: Caso nenhuma das afirmações anteriores seja verdadeira, execute.

Caso algum dos casos ocorra o interpretador executará o código indentado no próximo slide:

Módulo I – Resumo de Python

Ex:
```
if (2 > 1):
    a = 1

elif (3>2):
    a = 2

else:
    a = 3
```

Esse código deve retornar a = 1, pois o primeiro teste já é uma verdade e por este motivo executa seu código, não permitindo a segunda validação e nem escapando pelo else.

Módulo I – Resumo de Python

- **Laços de repetição:**
  - Dada a necessidade do programa de executar o mesmo comando mais de uma vez, existem os laços de repetição. Eles funcionam como *loops* que executam um bloco de código um número definido de vezes.

Ex:
```
for i in range(0,5,1):
    print(i)
```

Dado o código acima, o programa gera uma contagem de 0 a 4, pois i aumentará em 1 a cada repetição do *loop*.

Módulo I – Resumo de Python

- Laços de repetição:
  - O formato então da estrutura <u>for</u> é for (variável definida pelo usuário) in range(a,b,c):
    - a = Começo do intervalo.
    - b = Final do intervalo.
    - c = Ritmo do intervalo.

  - Não é necessário passar todos os valores do for, a linguagem já toma como padrão 0 como começo e 1 como ritmo, portanto:
  - for i in range(0,5,1): = for i in range(5):

Módulo I – Resumo de Python

- **Laços de repetição:**
  - Além da estrutura <u>for</u>, outro laço de repetição é a estrutura <u>while</u> que traz conceitos de laços de repetição e condicionais juntos. Ou seja, executa um bloco de código até uma condição se tornar verdade.

Ex:
```
c = 0
while (c != 5):
    print(c)
    c = c+1
```

Esse código deve retornar a mesma saída de um for c in range(5). Porém, utilizando dos mesmos conceitos de condicionais é possível atingir outros resultados bem diferentes utilizando da estrutura <u>while</u>.

Módulo I – Resumo de Python

- Vetores:
  - Um vetor é uma sequência de dados ocupando posições consecutivas de memória e por isso existe uma ordem natural (o primeiro elemento, o segundo e assim por diante). A grande vantagem de usar vetores é poder trabalhar com um grande número de variáveis utilizando um único nome.

Ex:
```
vet = [21, 22, 23, 24]      //declara-se uma lista fornecendo todos seus elementos
vet = [ ]                   //define um vetor vazio!
```

Módulo I - Resumo de Python

- Vetores:
  - Para inserir valores em um vetor vazio, basta usar o método **.insert(a, b)**
    - Nele, insere-se dois parâmetros, sendo o primeiro (a) a posição e o segundo (b) o que será inserido na dada posição do vetor.

Ex:
```
vetor = [ ]
vetor.insert(0, 1)
print(vetor[0])
```

Output: 1

Módulo I – Resumo de Python

- **Vetores:**
  - Outro método de inserir valores em uma lista é o .append, utilizando deste método o valor será inserido o valor ao final da lista.

Ex:
```
vetor = [1]
vetor.append(2)
print(vetor[0], vector[1])
```

Output: 1 2

Módulo I – Resumo de Python

<u>Exercício II</u>:

- a) Utilizando os conhecimentos de laços de repetição, vetores e operadores condicionais, faça um programa que o usuário insira seis nomes, imprimindo-os em seguida.
- b) Imprima apenas os nomes que estão nas posições pares do vetor.

Ex de input: Fernanda Daniella Luciano Vinícius Matias Rafael

Ex de output: Fernanda, Luciano, Matias

Módulo I – Resumo de Python

- Funções:
  - Na programação, funções são blocos de código que realizam determinadas tarefas que normalmente precisam ser executadas diversas vezes dentro de uma aplicação.

Existem funções nativas do programa que são fundamentais para muitos aspectos da programação em console, entretanto não as usaremos neste minicurso devido a seu direcionamento a biblioteca de tkinter.

Módulo I – Resumo de Python

- Funções:
  - Funções podem receber parâmetros antes de sua execução, de maneira a funcionarem para situações diferentes a partir de um mesmo código.

Ex:
```
def soma(a, b):
    print(a + b)

soma(5,5)
```

O código escrito acima deve retornar o valor 10.

Módulo I – Resumo de Python

- Funções:
  - Funções inicialmente não alteram as variáveis no programa, sendo totalmente isoladas caso não sejam indicadas a alterar no programa inicial.

Ex:
```
def soma(a,b):
    c = a+b
    c = 1

soma(12,8)
```

Neste caso o valor de c ainda é um, pois o interpretador não faz ligação ao programa inicial quando iniciado em função

Módulo I – Resumo de Python

- **Funções:**
  - Um dos métodos de retorno de uma função é utilizando de *return* ao seu final.

Ex:

```
def soma(a,b):
    c = a+b
    return c


d= soma(2,3)
print(d)
```

Console: 5

Módulo I – Resumo de Python

- Funções e variáveis globais:
  - Para alterar uma variável no programa, só é necessário a declaração de variáveis globais antes de suas operações.
  - Variáveis globais funcionam como ponteiros em funções e quando mencionadas ou referidas serão possíveis de serem manipuladas mesmo dentro de uma função.

Módulo I – Resumo de Python

Ex:
```
def soma(a,b):
    global c
    c = a+b


c=1
soma(12, 8)
print(c)
```

Neste momento a variável a retorna o valor 20, pois durante a função ela soma recebe o valor da soma de a e b.

Módulo I – Resumo de Python

<u>Exercício III</u>:

- Utilizando os conhecimentos anteriores, crie um programa contendo uma função que altere o valor de um número elevado pelo outro dado o formato: nomeDaFuncao(Base,Expoente), o programa deve alterar o valor da variável na main.

Ex:

```
r = nomeDaFuncao(3,2)
print("3 elevado a 2 é {}".format(r))
```

Fim do Módulo I

Agora entraremos mais a fundo na programação direcionada ao Tkinter, por este motivo iremos agora para um coffee break antes de retomarmos

Módulo II – Biblioteca Tkinter

As interfaces gráficas do usuário (GUI – Graphic User Interface) são bastante populares no uso de softwares em geral, e os programadores devem estar aptos a trabalhar com a criação de interfaces, já que torna o uso mais fácil além de aumentar a produtividade.

Tkinter é uma biblioteca da linguagem Python nativa e permite desenvolver interfaces gráficas. Isso significa que qualquer computador que tenha o interpretador Python instalado é capaz de criar interfaces gráficas usando o Tkinter, com exceção de algumas distribuições Linux, exigindo que seja feita o download do módulo separadamente.

Módulo II – Biblioteca Tkinter

- Iniciando o Tkinter:
  - Assim como todas as bibliotecas, antes de ser utilizado, o Tkinter deve ser inicializado no início do código, para fazer isso deve se usar o comando import da seguinte maneira:

from tkinter import *

- Desta maneira importaremos todos os recursos da biblioteca, para importar um recurso específico deve-se somente substituir o asterisco pelo nome da função que deseja utilizar.

Módulo II – Biblioteca Tkinter

- Iniciando o Tkinter:
  - Importada a biblioteca, seu interpretador já deve ser capaz de reconhecer as funções. Para iniciar uma janela, deve ser iniciado o loop do programa. Para fazer isto atribui-se a função Tk() a uma variável.

  - Para finalizar, você determina o final do loop com o método mainloop.

Ex:
```
main = Tk()
main.mainloop()
```

Módulo II – Biblioteca Tkinter

- Definições:
  - É possível alterar ou definir informações sobre através de alguns meios, sendo eles:
  - Através de parâmetros;
    - Durante a inicialização de objetos é possível definir algumas informações.

      Ex: Button(text="Ok")

  - text é um dos parâmetros deste botão. Também é possível alterar esse parâmetro mais tarde usando o *método* .config()
  - Através de métodos; Métodos geralmente se referem a objetos já iniciados ou criados.

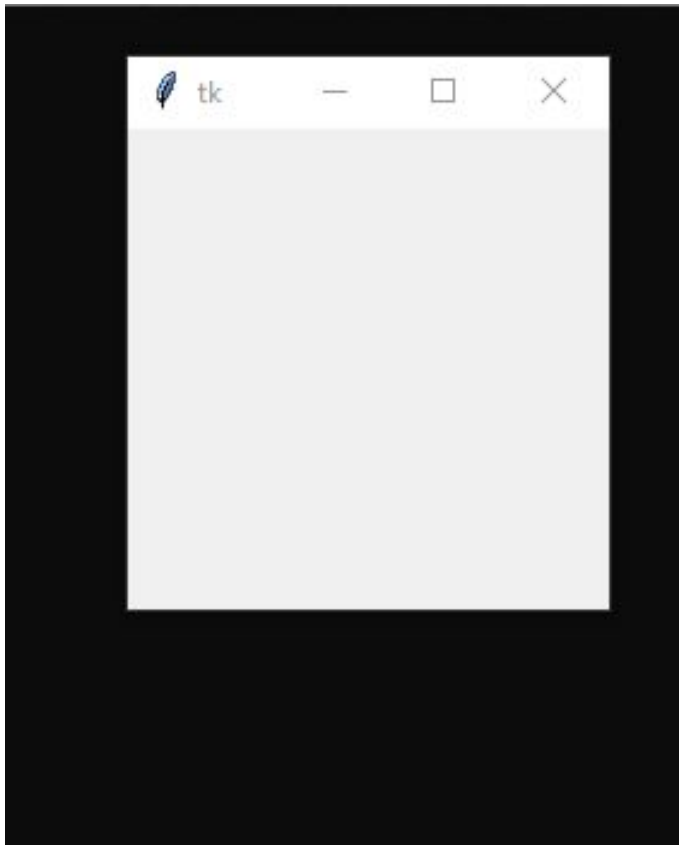
    Ex: var.get().

  - Através de Enum; Possuem um rótulo identificador.

    Ex: label['text'] = "Olá mundo!"

Módulo II – Biblioteca Tkinter

- Configuração de janela:
  - A partir de agora, o programa já cria uma janela, porém seu tamanho é pré-definido assim como seu título. Para alterar esses fatores, temos que declarar métodos no programa.



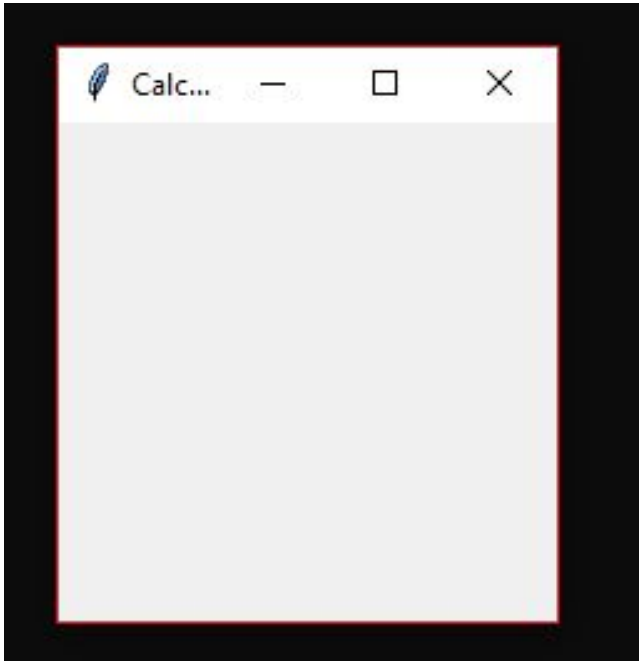
Módulo II – Biblioteca Tkinter

- Métodos de janela:
  - Por se tratar de um contato inicial, somente será apresentado de maneira mais aprofundada os métodos necessários para o minicurso ministrado. Seguem exemplos:
    - main.title("Título do programa")
    - main.geometry("LARGURAxALTURA")
    - main.resizable(width=False,height=False)

Módulo II – Biblioteca Tkinter

- **main.title:**
  - O método .title será utilizado para alterar o título, que é o nome que aparece na barra superior do seu programa.

Ex: `main.title("Calculadora")`

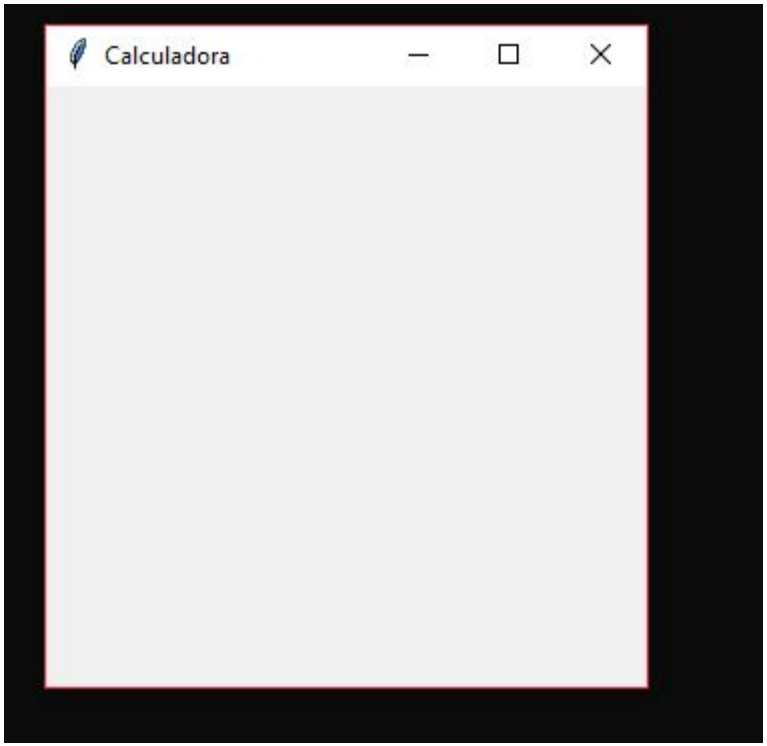


Módulo II – Biblioteca Tkinter

- main.geometry:
  - No slide anterior vimos que o título foi alterado. Entretanto ele não pode ser visto por completo, a janela estava pequena demais.
  - Para alterar o tamanho inicial da janela, utilizaremos o **main.geometry**

Ex:

```
main.title("Calculadora")

main.geometry("500x500")
```



Módulo II – Biblioteca Tkinter

- **main.geometry:**
  - Para se definir o tamanho da janela, passamos dois parâmetros: largura e altura. Também é possível determinar a posição inicial do programa com:

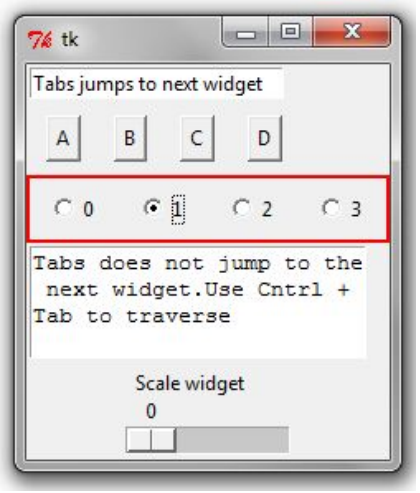
```
main.geometry("LarguraxAltura+DistanciaEmX+DistanciaEmY")
```

- **main.resizable:**
  - Muitas vezes, não é interessante ao programador que o programa tenha a tela ajustável, então para resolver esse problema é necessário a declaração de:

```
main.resizable(width=False,height=False)
```

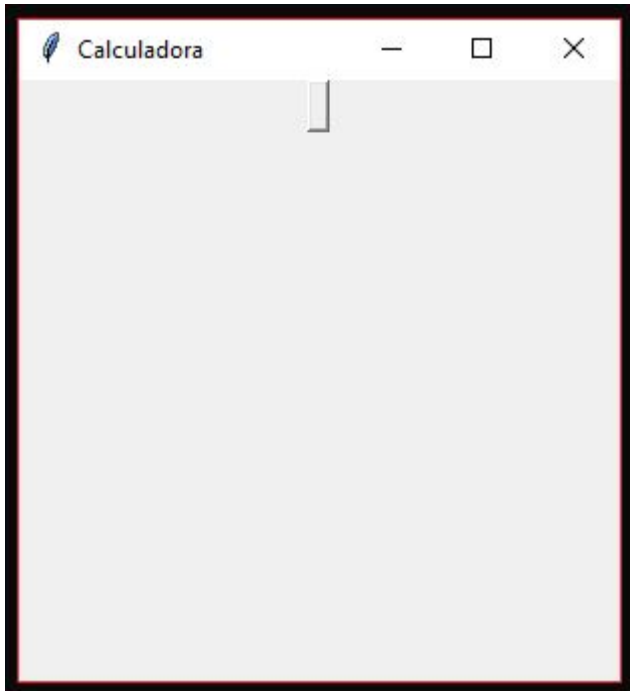
Módulo II – Biblioteca Tkinter

- **Widgets:**
  - Widgets são objetos pré-programados pela biblioteca. Eles possuem diversas funções, como botões, texto, caixas de opção, entre outros. Através deles e de sua manipulação, é possível atingir diversos resultados no programa. Não entraremos em detalhes em toda sua listagem pois esta informação é facilmente adquirida na documentação da biblioteca junto a seus métodos.



Módulo II – Biblioteca Tkinter

- Botões:
  - Ao não estipular texto ou posição exata este será o resultado.



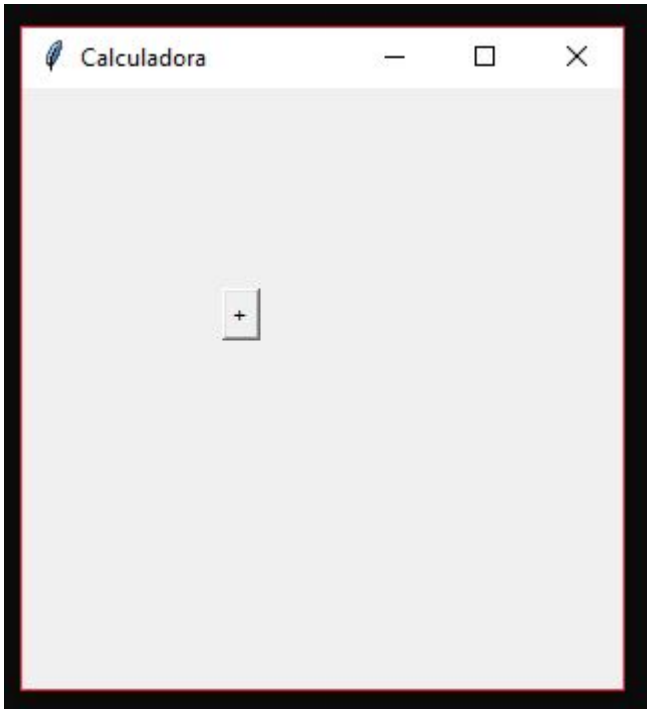
- Parâmetros de botões:
  - **text = "texto"**, ao definir dentro dos parênteses ele apresentará um texto. Seu tamanho o acompanhará visto que a largura de um botão é determinada em caracteres.
  - **width = l** e **height = a,** ambos representam o tamanho do botão, largura e altura respectivamente.
  - **command = funcao**; este recebe o nome da função que será executada ao ser pressionado.

Módulo II – Biblioteca Tkinter

- Parâmetros de botões:
  - Como mencionado antes, um botão receberá uma função, a qual deve ser declarada no início do código para que o programa possa ser executado. No exemplo anterior foi dado:

```
bt_soma = Button(main, text="+",
        command=soma)
```

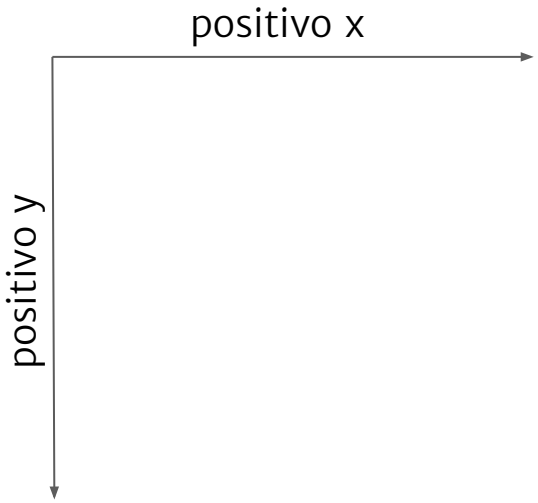
  - Portanto, caso não seja definida a função soma, o programa não poderá continuar.



Módulo II – Biblioteca Tkinter

- Para mudar a posição de qualquer widget temos duas opções: Place e Pack
- Place: Permite definir a posição em x e y.

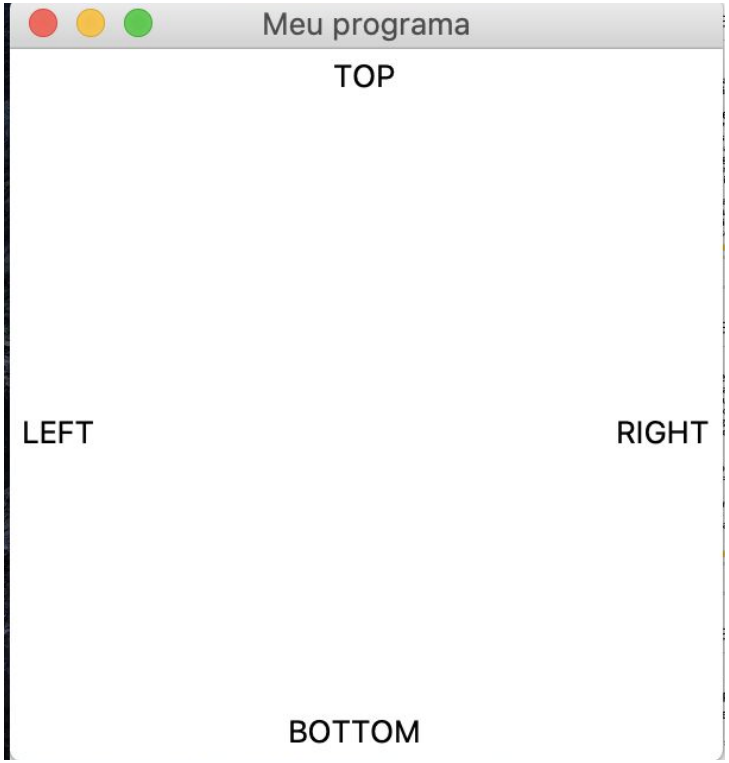
```
button.place(x=100,y=100)
```



Módulo II – Biblioteca Tkinter

- Pack: empacota os widgets na horizontal ou vertical.

```
button.pack(side = TOP)

button.pack(side = BOTTOM)

button.pack(side = RIGHT)

button.pack(side = LEFT)
```

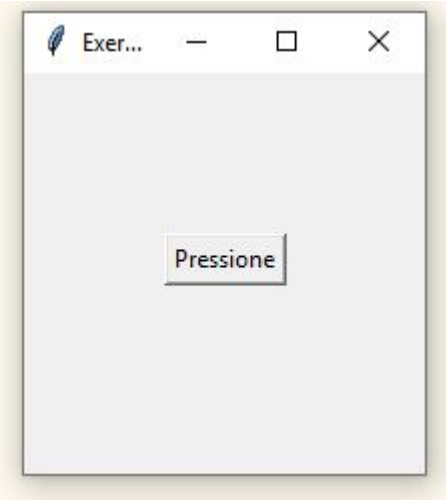


Módulo II – Biblioteca Tkinter

<u>Exercício IV</u>:

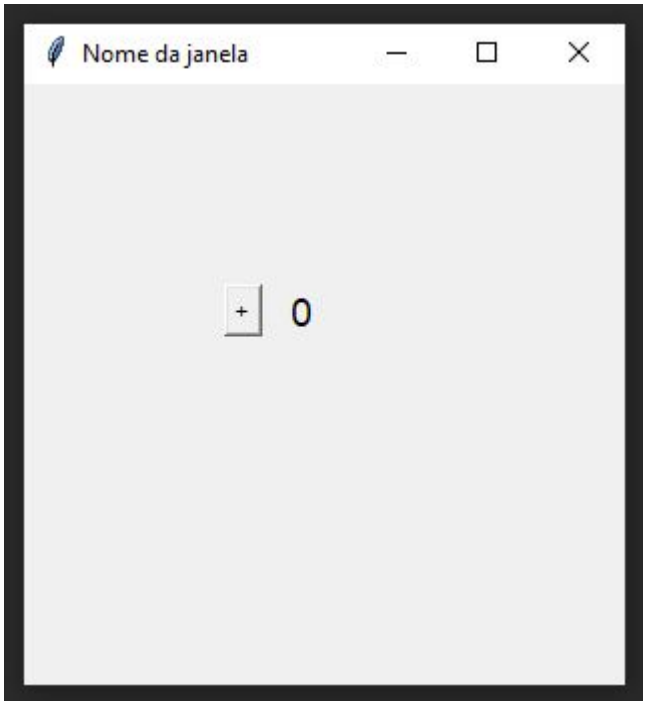
- Utilizando os conhecimentos anteriores, crie um programa que imprima um aviso no console toda vez que o botão for pressionado:

Ex: "O botão foi pressionado."
"O botão foi pressionado."
"O botão foi pressionado."
"O botão foi pressionado."
"O botão foi pressionado."



Módulo II – Biblioteca Tkinter

- **Label:**
  - Labels são linhas de texto. Além da troca de informações através de métodos, também passamos através de uma posição nomeada, Enum. Não há necessidade de aprofundamento neste tema, porém segue um exemplo de label.



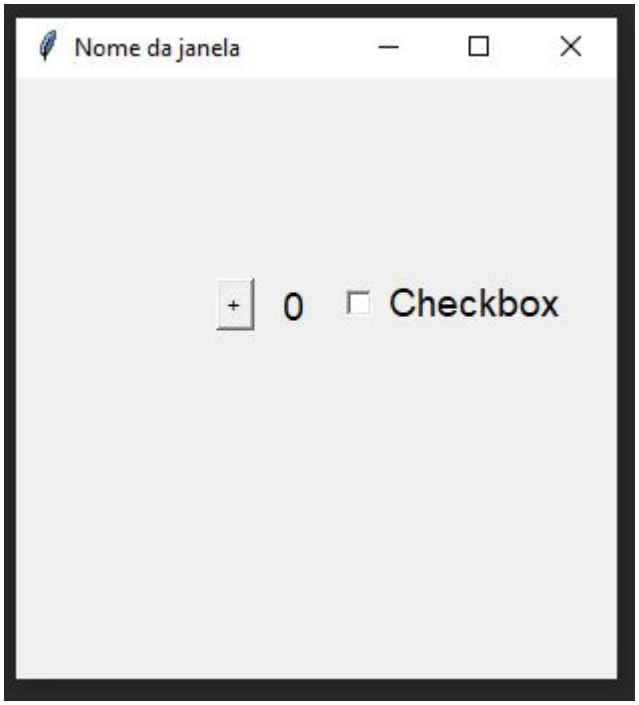

```
Ex:  label = Label(font=("Arial",14))
     label['text']=0
     label.place(x=130,y=100)
```

Os parâmetros entregues a função Label são a fonte e seu tamanho. Já o label['text'] têm a função de armazenar o texto.

Módulo II – Biblioteca Tkinter

- **Checkbox:**
  - Checkboxes ou Checkbuttons podem retornar o estado verdadeiro ou falso. Para receber este valor usa-se o método .get(), ele retornará por padrão 0 para falso (desmarcado) e 1 para verdadeiro (marcado). Para criar uma checkbox é necessário seguir este padrão:

```
var = IntVar()
c = Checkbutton(variable=var)
```

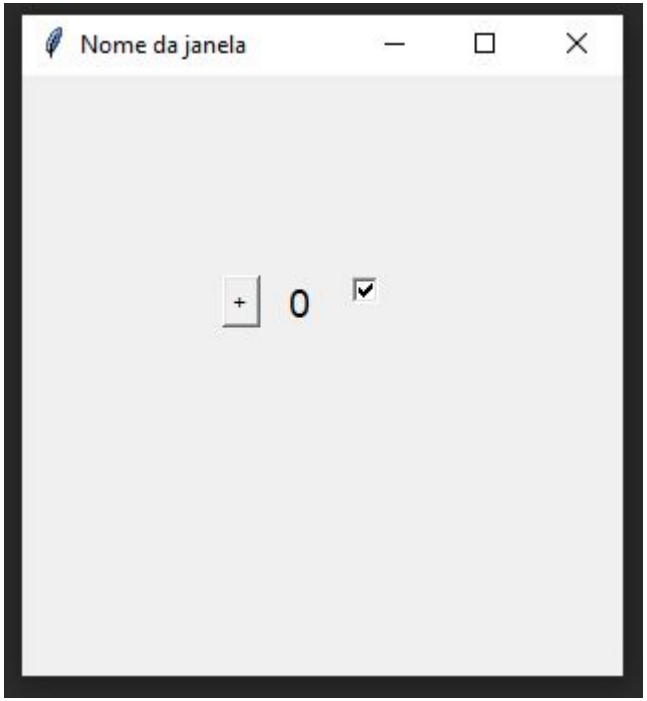


A criação dessa variável que recebe a função IntVar() é necessária para a biblioteca funcionar normalmente.

Módulo II – Biblioteca Tkinter

- **Checkbox:**
  - Segue um exemplo do uso do .get()

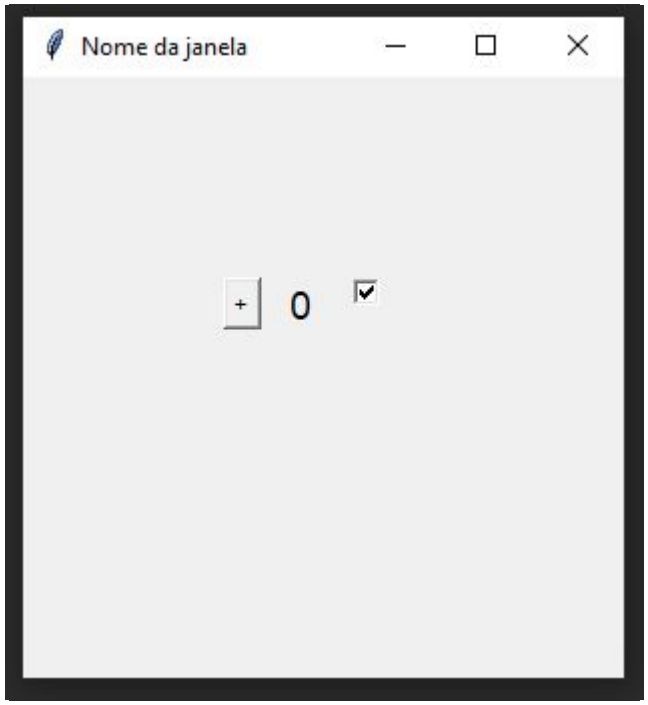
```
def estado():
    print(var.get())
    var = IntVar()

c = Checkbutton(variable=var,command=estado)

c.place(x=160,y=95)
```



No início do programa se define a função <u>estado</u> ela será ativada ao marcar a caixa de seleção, escrevendo no console seu estado referenciado como 0s e 1s.

Módulo II – Biblioteca Tkinter

- Checkbox:
  - Alguns dos parâmetros que a checkbox pode receber são:
    - **text**, e seus relativos. Onde o checkbox irá apresentar um texto a seu lado.
    - **height** e **width**, que irá determinar o tamanho da área sensível a clique, *não alterando o tamanho da caixa.*
    - **state**, sendo ele DISABLED ou NORMAL, se refere ao seu funcionamento, um checkbox disabled fica acinzentado e não pode mudar de estado.

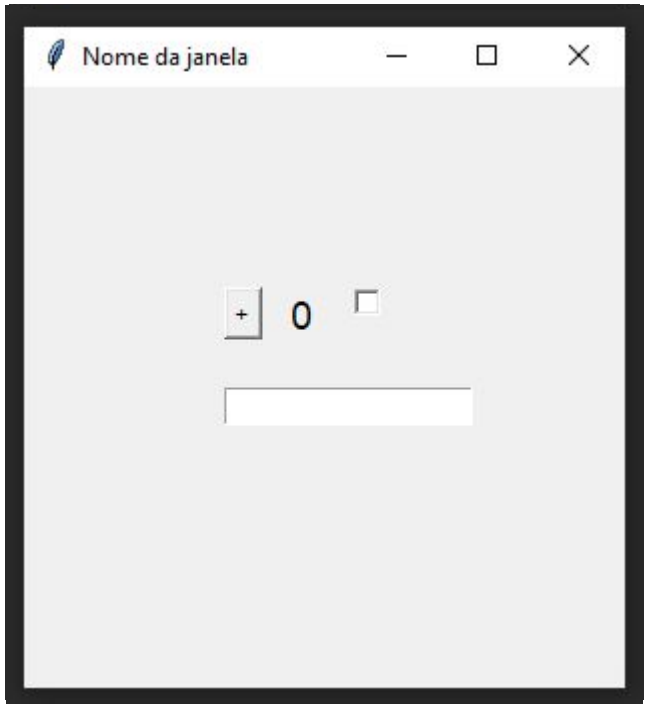


Módulo II – Biblioteca Tkinter

- **Entry:**
  - É uma área de preenchimento de texto, onde o usuário insere um texto através do teclado e o programa pode fazer a manipulação deste texto.
  - Para utilizar o texto digitado usa-se o método .get() como utilizado na checkbox.

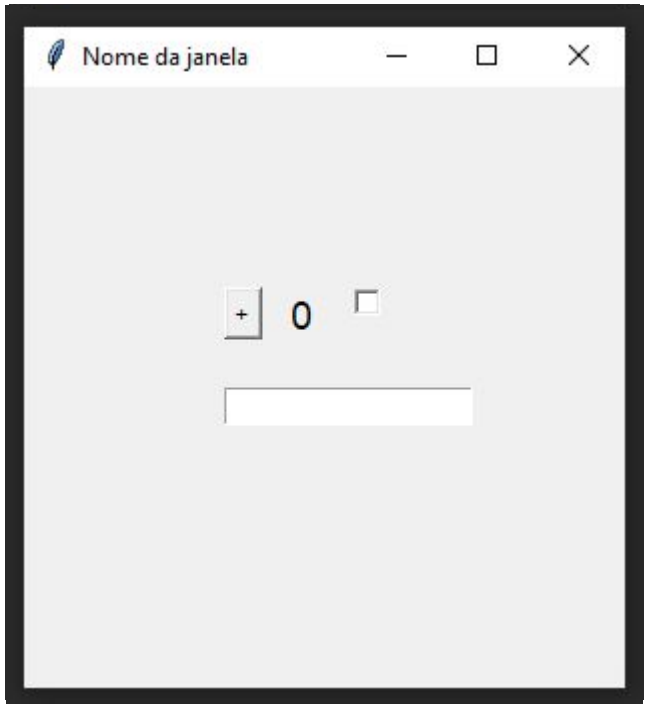
Ex:

```
e = Entry()
e.place(x=100,y=150)
```



Módulo II – Biblioteca Tkinter

- Entry:
  - Alguns dos parâmetros que a entry pode receber são:
    - **width**, que irá determinar o tamanho da área de escrita, height não funciona, para textos de mais de uma linha se usa a text box.
    - **state**, sendo ele DISABLED ou NORMAL, se refere ao seu funcionamento, um entry disabled fica acinzentado e não pode mudar de estado.

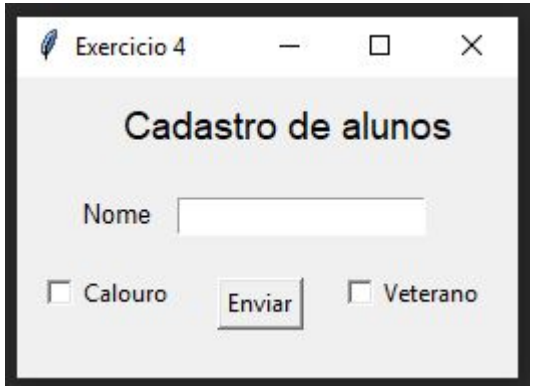


Módulo II – Biblioteca Tkinter

<u>Exercício V</u>:

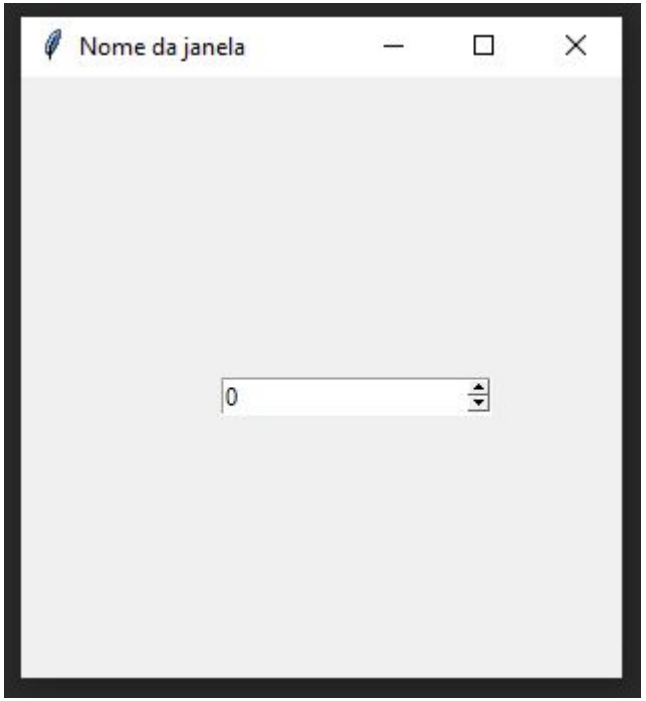
- Utilizando os conhecimentos anteriores, crie um programa que construa um cadastro de alunos, usando checkboxes, entries, botões e labels, e depois imprima essa informação no console.
- Você pode incluir o que quiser, contanto que use os 4 widgets.

Output: Fernanda, Caloura.



Módulo II – Biblioteca Tkinter

- Spinbox:
  - Spinbox é um seletor numérico que varia entre um alcance definido. Para utilizar este widget se declara da seguinte forma:



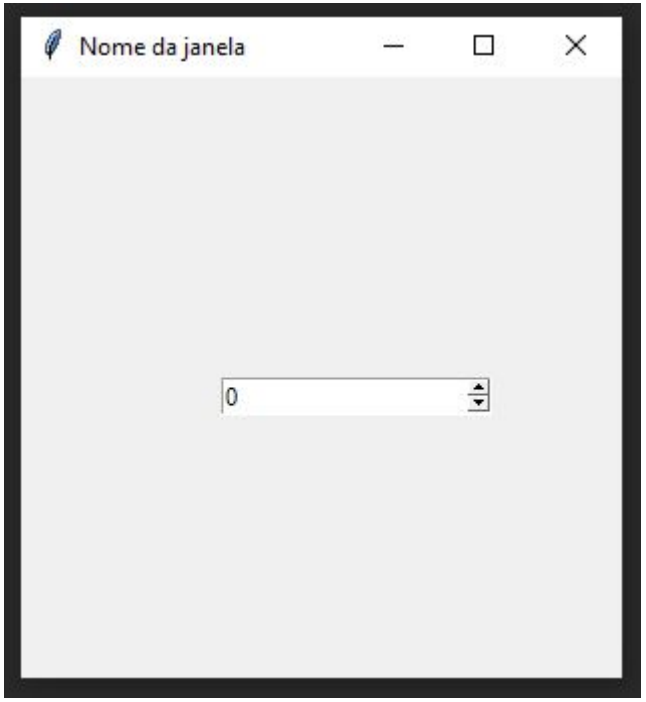
Ex:
```
s = Spinbox(from_=0, to=10)
s.place(x=100,y=150)
```

Onde *from_* é o ponto inicial e *to* é o máximo. Outro jeito de usar, é deixando os valores definidos de começo:

Ex:
```
s = Spinbox(values=(1, 2, 4, 8))
s.place(x=100,y=150)
```

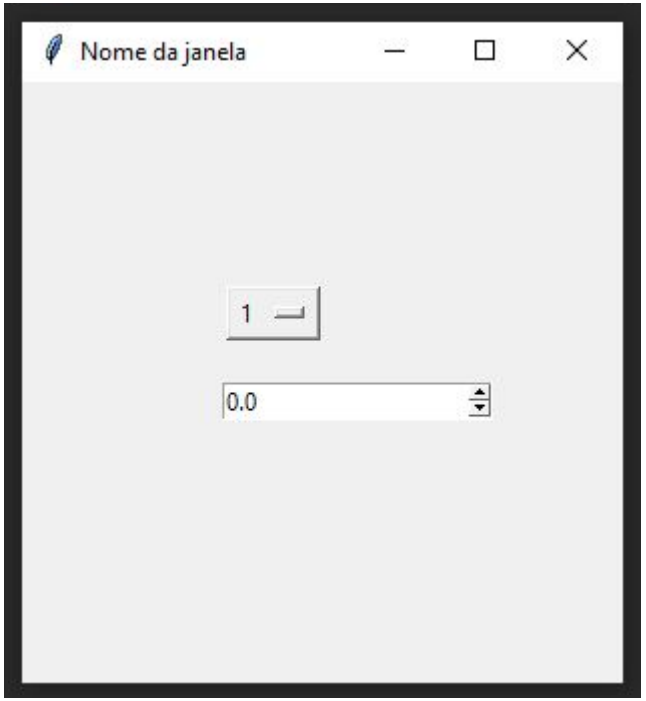
Módulo II – Biblioteca Tkinter

- Spinbox:
  - Parametros interessantes:
    - **increment**: define o valor que será adicionado ou removido ao utilizar as setas. Caso o *increment* seja float, o resultado será float, caso contrário seu padrão de *increment* é 1.
  - Métodos:
    - Assim como os widgets entry e checkbox, para retornar o valor da spinbox pode se usar .get()



Módulo II – Biblioteca Tkinter

- Option menu:
  - O menu de opções armazena todas as opções de forma compacta para utilizá-lo. Como no check box, é necessário a utilização de uma variável específica do TKinter, ou a função **StringVar()**



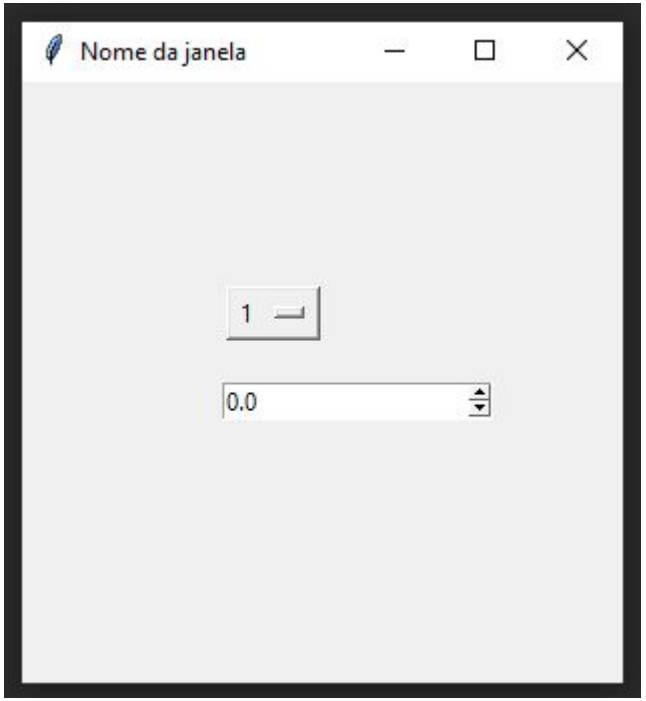
Ex: inicial=StringVar(main)
inicial.set(1)

o = OptionMenu(main, inicial, 1,3.5,"exemplo")

o.place(x=100,y=100)

Módulo II – Biblioteca Tkinter

- Option menu:
  - Não é necessário determinar opções de mesmo tipo, como mostrado anteriormente.
  - Para receber o valor da opção, utilize a primeira variável definida com o método .get()



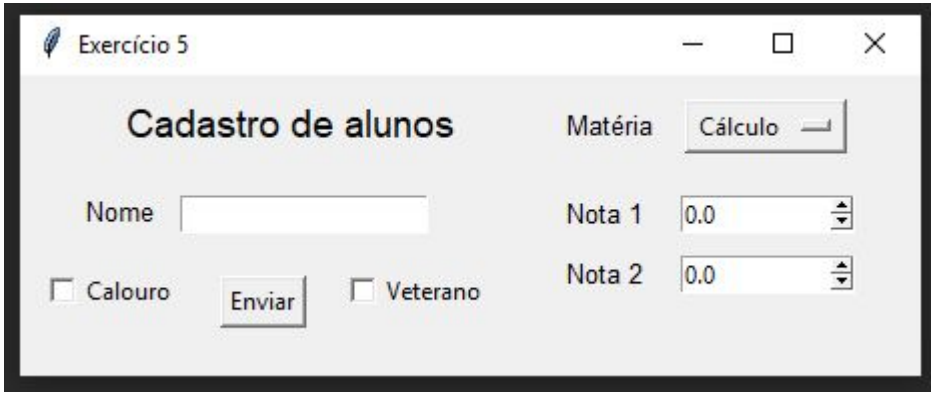

```
Ex:  inicial=StringVar(main)
     inicial.set(1)
     o = OptionMenu(main, inicial, 1,3.5,
“exemplo”)

     o.place(x=100,y=100)
     opcao = inicial.get()
```

Módulo II – Biblioteca Tkinter

<u>Exercício VI</u>:

- Utilizando parte do código anterior, aumente o número de informações coletadas, usando dos novos widgets apresentados, entregando uma saída similar:

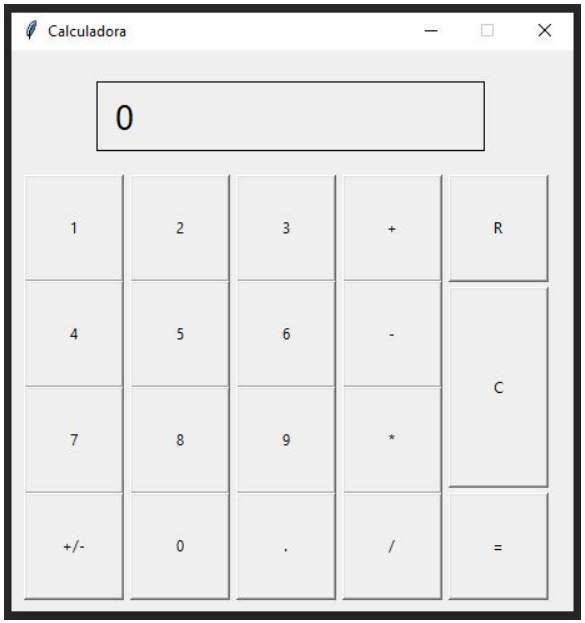
Output: Fernanda, Caloura. Média em Cálculo: 6,25



Módulo II – Biblioteca Tkinter

Desafio:

- Utilizando seus conhecimentos adquiridos durante o minicurso, monte uma calculadora funcional:



Documentação:

Tkinter: http://effbot.org

Python: https://docs.python.org/3/

Contato

Agradecemos sua participação!

Para entrar em contato:

Ediney Silva -edineyjr1@gmail.com
- (47) 98922-8056

Matias Gutierrez - matiguti17@gmail.com
- (47) 99179-6221

Colmeia: https://www.udesc.br/cct/colmeia

Obrigado pela presença!