TKINTER E LISTAS:

Exercício:

Desenvolva um script em Python utilizando a biblioteca Tkinter para criar uma interface gráfica que permita ao usuário inserir e armazenar 5 nomes em uma lista. Utilize o layout place para posicionar os elementos na tela.

Requisitos:

1. Crie um campo de entrada de texto para que o usuário possa digitar um nome.
2. Insira um botão denominado "Capturar Nome". A cada clique neste botão, o nome inserido deve ser capturado e adicionado a uma lista.
3. Após a inserção de 5 nomes, exiba os nomes na janela principal, logo abaixo do frame, com um nome em cada linha, respeitando a ordem de inserção na lista.

Dicas:

- Certifique-se de que o campo de entrada seja limpo após cada captura de nome.
- Utilize o layout place para posicionar os elementos na tela de maneira organizada.
- Use uma estrutura de repetição para exibir os nomes na interface após capturar os 5 nomes.

Objetivo: Praticar o uso de estruturas de repetição, listas e manipulação de interfaces gráficas em Python.

WIDGET TREEVIEW

O widget Treeview da biblioteca Tkinter em Python 3.x é uma ferramenta para a exibição de dados tabulares. Ele permite organizar e visualizar informações de forma hierárquica, com a capacidade de exibir múltiplas colunas e cabeçalhos, além de possibilitar a expansão da informações. Isso faz com que o Treeview seja ideal para representar estruturas como sistemas de arquivos, tabelas de banco de dados, ou listas organizadas.

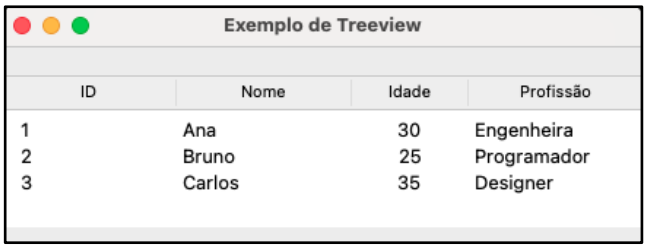
Principais Funcionalidades do Treeview

1. **Múltiplas Colunas:** Diferente de outros widgets, o Treeview permite a exibição de múltiplas colunas, onde cada coluna pode ter seu próprio cabeçalho. Isso facilita a visualização de informações adicionais associadas a cada item.
2. **Manipulação de Itens**: É possível adicionar, remover e atualizar itens dinamicamente. Além disso, o Treeview oferece funcionalidades para a seleção de itens, permitindo interações com o usuário.

3. **Scrollbars:** Quando o conteúdo do Treeview excede o espaço disponível, barras de rolagem (scrollbars) podem ser adicionadas automaticamente, tanto horizontal quanto verticalmente.

 # Exemplo:

```python
# TreeView

import tkinter as tk
from tkinter import ttk

# Criação da janela principal
root = tk.Tk()
root.title("Exemplo de Treeview")

# Criação do widget Treeview
tree = ttk.Treeview(root)

# Definição das colunas
tree['columns'] = ('Nome', 'Idade', 'Profissão')

# Formatação das colunas
tree.column("#0", width=120, minwidth=25)
tree.column("Nome", anchor=tk.W, width=120)
tree.column("Idade", anchor=tk.CENTER, width=80)
tree.column("Profissão", anchor=tk.W, width=120)

# Definição dos cabeçalhos
tree.heading("#0", text="ID", anchor=tk.W)
tree.heading("Nome", text="Nome", anchor=tk.W)
tree.heading("Idade", text="Idade", anchor=tk.CENTER)
tree.heading("Profissão", text="Profissão", anchor=tk.W)

# Adicionando dados
tree.insert(parent='', index='end', iid=0, text='1', values=('Ana', 30, 'Engenheira'))
tree.insert(parent='', index='end', iid=1, text='2', values=('Bruno', 25, 'Programador'))
tree.insert(parent='', index='end', iid=2, text='3', values=('Carlos', 35, 'Designer'))

# Posicionando o Treeview na janela
tree.pack(pady=20)

# Executando a janela
root.mainloop()
```



 # Exercício 2:

Desenvolva um script em Python utilizando a biblioteca Tkinter para criar uma interface gráfica que permita ao usuário inserir e armazenar 5 nomes em uma lista. Utilize o layout place para posicionar os elementos na tela.

Requisitos:

1. Crie um campo de entrada de texto para que o usuário possa digitar um nome.
2. Insira um botão denominado "Capturar Nome". A cada clique neste botão, o nome inserido deve ser capturado e adicionado a uma lista.
3. Após a inserção de 5 nomes, insira esses nomes em um widget Treeview, onde serão exibidos na janela principal, organizados em uma tabela.

Dicas:

- Certifique-se de que o campo de entrada seja limpo após cada captura de nome.
- Utilize o layout place para posicionar os elementos na tela de maneira organizada.
- Use uma estrutura de repetição para exibir os nomes na Treeview após capturar os 5 nomes.

Objetivo: Praticar o uso de estruturas de repetição, listas e manipulação de interfaces gráficas em Python, incluindo o uso do widget Treeview.

====== /// ======

2ª Coleção - Tuplas: As tuplas são estruturas muito semelhantes às listas. A principal diferença é que tuplas são estruturas que armazenam dados e são **imutáveis**. Principais características das tuplas:

- Seu tamanho não pode ser alterado, diferentemente dos dicionários e listas.

- Seus elementos são imutáveis, iteráveis e ordenados.

- Os valores em uma tupla ficam entre parênteses e são separados por vírgulas:

- Tuplas devem ser utilizadas na construção dos algoritmos somente quando a **imutabilidade** dos dados ser um requisito importante, pois, ela fornece uma maneira segura de garantir a integridade de dados.

- Portanto, quando uma tupla for criada não podemos fazer mais nada que venha a modificar seus valores.

- Note que as opções de métodos para esse tipo de estrutura são bem limitadas comparadas com as outras estruturas de dados estudadas até então.

- Se tentarmos adicionar um item ou deletar um elemento, numa tupla, teremos uma mensagem de erro pois estamos tentando modificar algo imutável.

Os principais métodos disponíveis para cada um dos elementos de uma tupla são:

Criando uma tupla vazia :

```
tupla = ()

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA variável tupla é do tipo: {type(tupla)} \n')
```

```
A tupla tem: 0 elementos

Os elementos da tupla são: ()

A variável tupla é do tipo: <class 'tuple'>
```

Criando e Populando uma tupla

```
tupla = ('a', 'b', 'c', 'd', 'f', 'g', 1, 2, 3, 4, 5, 6, 7, 8, 9)

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA variável tupla é do tipo: {type(tupla)} \n')
```

```
A tupla tem: 15 elementos

Os elementos da tupla são: ('a', 'b', 'c', 'd', 'f', 'g', 1, 2, 3, 4, 5, 6, 7, 8, 9)

A variável tupla é do tipo: <class 'tuple'>
```

Criando e Populando uma tupla a partir da função range()

```
tupla = tuple(range(11))

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA variável tupla é do tipo: {type(tupla)} \n')
```

```
A tupla tem: 11 elementos

Os elementos da tupla são: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

A variável tupla é do tipo: <class 'tuple'>
```

Observação: tuplas são imutáveis (depois de criadas seus valores não podem ser alterados). Qualquer tentativa de alteração, deleção do valor de um elemento da tupla irá gerar *TypeError: 'tuple' object does not support item assigment.*

No entanto podemos deletar um tupla inteira com todos os seus elementos:

Deletando uma tupla inteiras e todos os seus elementos:

```
tupla = ('a', 'b', 'c', 'd', 'f', 'g', 1, 2, 3, 4, 5, 6, 7, 8, 9)

del tupla

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA variável tupla é do tipo: {type(tupla)} \n')
```

```
Traceback (most recent call last):
  File "/Users/proj_alp_23/aula_27_04.py", line 59, in <module>
    print(f'\nA tupla tem: {len(tupla)} elementos')
NameError: name 'tupla' is not defined. Did you mean: 'tuple'?
```

Porém, você pode inserir novos elementos dentro da tupla

Inserindo elementos em uma tupla:

```
tupla = ('a', 'b', 'c', 'd', 'f', 'g', 1, 2, 3, 4, 5, 6, 7, 8, 9)

tupla = tupla+('h', 'i', 'j', 10,11,12 )

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA variável tupla é do tipo: {type(tupla)} \n')
```

```
Os elementos da tupla são: ('a', 'b', 'c', 'd', 'f', 'g', 1, 2, 3, 4, 5, 6, 7, 8, 9, 'h', 'i', 'j', 10, 11, 12)
```

Fatiamento de tupla - Transformando uma tupla em string / int...

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, 4.3, False)

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nOs elementos da tupla na posição 0 são: {tupla[0]}')

print(f'\nOs elementos da tupla na posição 1 são: {tupla[1]}')

print(f'\nOs elementos da tupla na posição 2 são: {tupla[2]}')

print(f'\nOs elementos da tupla na posição 3 são: {tupla[3]}')

valor = tupla[4]
print(f'\nOs elementos da tupla na posição 4 são: {valor}')
print(f'O Tipo do elementos da tupla na posição 4 são: {type(valor)}')

print(f'\nOs elementos da tupla na posição 5 são: {tupla[5]} \n')
```

```
Os elementos da tupla são: ('Algoritmos', 'e', 'Lógica de Programação', 2023, 4.3, False)

Os elementos da tupla na posição 0 são: Algoritmos

Os elementos da tupla na posição 1 são: e

Os elementos da tupla na posição 2 são: Lógica de Programação

Os elementos da tupla na posição 3 são: 2023

Os elementos da tupla na posição 4 são: 4.3
O Tipo do elementos da tupla na posição 4 são: <class 'float'>

Os elementos da tupla na posição 5 são: False
```

Desempacotamento de tuplas

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, 4.3, False)

dados_0, dados_1, dados_2, dados_3, dados_4, dados_5 = tupla

print(f'\nO elemento de dados_0 capturado da tupla é: {dados_0}')

print(f'\nO elemento de dados_1 capturado da tupla é: {dados_1}')

print(f'\nO elemento de dados_2 capturado da tupla é: {dados_2}')

print(f'\nO elemento de dados_3 capturado da tupla é: {dados_3}')

print(f'\nO elemento de dados_4 capturado da tupla é: {dados_4}')

print(f'\nO elemento de dados_5 capturado da tupla é: {dados_5}')

print(f'\nO tipo de elemento de dados_5 capturado da tupla é: {type(dados_5)}')
```

```
O elemento de dados_0 capturado da tupla é: Algoritmos

O elemento de dados_1 capturado da tupla é: e

O elemento de dados_2 capturado da tupla é: Lógica de Programação

O elemento de dados_3 capturado da tupla é: 2023

O elemento de dados_4 capturado da tupla é: 4.3

O elemento de dados_5 capturado da tupla é: False

O tipo de elemento de dados_5 capturado da tupla é: <class 'bool'>
```

- Se colocarmos menos ou mais variáveis que o número de elementos da tupla irá gerar *ValueError* no momento do desempacotamento.

Lendo quantos elementos tem a tupla

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre', 4.3, False)

print(f'\nA tupla tem: {len(tupla)} elementos')

print(f'\nOs elementos da tupla são: {tupla}')
```

```
A tupla tem: 7 elementos

Os elementos da tupla são: ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre', 4.3, False)
```

Operações com tuplas: soma, máximo, minimo

```
tupla = (8, 15, 7, 10, 12.5, 0.5, 25, 99, 1080.99, -298)

print(f'\nOs elementos da tupla são: {tupla} \n')

print(f'\nA soma dos elementos da tupla é: {sum(tupla)}')

print(f'\nO maior valor que está na tupla é: {max(tupla)}')

print(f'\nO menor valor que está na tupla é: {min(tupla)}')
```

```
Os elementos da tupla são: (8, 15, 7, 10, 12.5, 0.5, 25, 99, 1080.99, -298)


A soma dos elementos da tupla é: 959.99

O maior valor que está na tupla é: 1080.99

O menor valor que está na tupla é: -298
```

Concatenando tuplas

```
tupla_1 = ('Gertrudez', 'Genoveva', 'Gerimunda', 'Gastronildo', 5, 6, 7)

tupla_2 = (11, 12, 13, 'Marilia', 'Marcelo', 'Matias')

print(f'\nOs elementos da tupla_1 são: {tupla_1}')

print(f'\nOs elementos da tupla_2 são: {tupla_2}')

print(f'\nOs elementos concatenados da tupla_1 e 2 são: {tupla_1 + tupla_2}')
```

```
Os elementos da tupla_1 são: ('Gertrudez', 'Genoveva', 'Gerimunda', 'Gastronildo', 5, 6, 7)

Os elementos da tupla_2 são: (11, 12, 13, 'Marilia', 'Marcelo', 'Matias')

Os elementos concatenados da tupla_1 e 2 são: ('Gertrudez', 'Genoveva', 'Gerimunda', 'Gastronildo', 5, 6, 7, 11, 12, 13,
'Marilia', 'Marcelo', 'Matias')
```

Verificando se um elemento está contido na tupla:

```
tupla = ('Gertrudez', 'Genoveva', 'Gerimunda', 'Gastronildo', 5, 6, 7)

print(f'\nOs elementos da tupla são: {tupla}')


print(f"\nVerificando se o elemento 'b' está na tupla: {'b' in tupla} \n")
```

```
Os elementos da tupla_1 são: ('Gertrudez', 'Genoveva', 'Gerimunda', 'Gastronildo', 5, 6, 7)

Verificando se o elemento 'b' está na tupla: False
```

Iterando sobre tupla:

```
tupla = (11, 12, 13, 'Marilia', 'Marcelo', 'Matias')

print(f'\nOs elementos da tupla são: {tupla} \n')

for elementos in tupla:
    print(elementos )
```

```
Os elementos da tupla são: (11, 12, 13, 'Marilia', 'Marcelo', 'Matias')

11
12
13
Marilia
Marcelo
Matias
```

Iterando sobre tupla com for(), **enumerate()** - gerando índice:

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre',)

print(f'\nOs elementos da tupla são: {tupla} \n')

for indice, valor in enumerate(tupla):
    print('Indice: ',indice, 'Elemento da tupla: ',valor)
```

```
Os elementos da tupla são: ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre')

Indice:  0 Elemento da tupla:  Algoritmos
Indice:  1 Elemento da tupla:  e
Indice:  2 Elemento da tupla:  Lógica de Programação
Indice:  3 Elemento da tupla:  2023
Indice:  4 Elemento da tupla:  1º semestre
```

Iterando sobre tupla - gerando índice para os elementos - while:

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre',)

print(f'\nOs elementos da tupla são: {tupla} \n')

total = len(tupla)
contador = 0

while contador < total:
    print(f'Indice: {contador + 1}, Elemento da tupla: {tupla[contador]}')
    contador +=1
```

```
Os elementos da tupla são: ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre')

Indice: 1, Elemento da tupla: Algoritmos
Indice: 2, Elemento da tupla: e
Indice: 3, Elemento da tupla: Lógica de Programação
Indice: 4, Elemento da tupla: 2023
Indice: 5, Elemento da tupla: 1º semestre
```

Iterando sobre tupla - Contando elementos específicos de uma tupla:

```
tupla = ('a', 4, 5, 6, 'c', 'd', 'c', 'a', 'a', 'b', 'b', 1, 2, 3, 'a','A', 'a')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA tupla contém {tupla.count("a")} elementos = a \n')
```

```
Os elementos da tupla são: ('a', 4, 5, 6, 'c', 'd', 'c', 'a', 'a', 'b', 'b', 1, 2, 3, 'a', 'A', 'a')

A tupla contém 5 elementos = a
```

Iterando sobre tupla - Contando elementos específicos de uma tupla:

```
tupla = ('Instituto Federal de Rondônia - Campus Ariquemes')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nA tupla contém {tupla.count("e")} elementos = e \n')
```

```
Os elementos da tupla são: Instituto Federal de Rondônia - Campus Ariquemes

A tupla contém 5 elementos = e
```

Acessando os elementos de uma tupla pelo seu índice:

```
tupla = ('Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nApresentando o elemento que está no índice  "4" da tupla = {tupla[4]}')
```

```
Os elementos da tupla são: ('Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho')

Apresentando o elemento que está no índice  "4" da tupla: Maio
```

Verificando em qual índice um elemento está na tupla:

```
tupla = ('Domingo', 'Segunda', 'Terça', 'Quarta', 'Quinta', 'Sexta', 'Sábado')

print(f'\nOs elementos da tupla são: {tupla}')

print(f'\nO elemento Quinta da tupla está no índice = {tupla.index("Quinta")} \n')
```

Atenção: *caso o elemento não exista será gerado um ValueError. E caso haja elementos repetidos na tupla, o Python retornará o valor do índice do primeiro elemento contando da esquerda para a direita*.

Aplicando *slice* em uma tupla => tupla(início: fim: passo)

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre',)

print(f'\nElementos da tupla: {tupla}')

print(f'\nQuantidade de elementos na tupla:  {len(tupla)}')

print(f'\nElementos a partir da posição 0 da tupla {tupla[0:]}')

print(f'\nElementos a partir da posição 0 até a posição 3 da tupla {tupla[:3]}')

print(f'\nElementos a partir da posição 30 até a final tupla {tupla[3:]}')

print(f'\nTodos os elementos da tupla em passo 2 {tupla[::2]}')

print(f'\nTodos os elementos da tupla em passo 4 {tupla[::4]} \n')
```

```
Elementos da tupla: ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre')

Quantidade de elementos na tupla:  5

Elementos a partir da posição 0 da tupla ('Algoritmos', 'e', 'Lógica de Programação', 2023, '1º semestre')

Elementos a partir da posição 0 até a posição 3 da tupla ('Algoritmos', 'e', 'Lógica de Programação')

Elementos a partir da posição 30 até a final tupla (2023, '1º semestre')

Todos os elementos da tupla em passo 2 ('Algoritmos', 'Lógica de Programação', '1º semestre')

Todos os elementos da tupla em passo 4 ('Algoritmos', '1º semestre')
```

Copiando Tuplas:

```
tupla = ('Algoritmos', 'e', 'Lógica de Programação' )

print(f'\nElementos da tupla: {tupla}')

tupla_2 = tupla

print(f'\nOs elementos da tupla_2 copiado da tupla são : {tupla_2} \n')
```

```
Elementos da tupla: ('Algoritmos', 'e', 'Lógica de Programação')

Os elementos da tupla_2 copiado da tupla são : ('Algoritmos', 'e', 'Lógica de Programação')
```

Criando tuplas a partir de listas - função *tuple():*

```
lista = ['Campus', 'Ariquemes', 2023, True, 532.15]

print(f'\nLista original: {lista}')
print(type(lista))

tupla  = tuple(lista)

print(f'\nTupla gerada a partir da lista = {tupla}')
print(type(tupla), '\n')
```

```
Lista original: ['Campus', 'Ariquemes', 2023, True, 532.15]
<class 'list'>

Tupla gerada a partir da lista = ('Campus', 'Ariquemes', 2023, True, 532.15)
<class 'tuple'>
```

Alterando uma lista que está dentro de uma tupla: *Quando uma tupla tiver uma lista ou outra coleção que pode ter seus elementos, inseridos, alterados, ou excluídas, essas ações poderão ocorrer normalmente*.

Alterando uma lista que está dentro de uma tupla:

```
tupla = ('a', 'b', 12, ['Campus', 'Ariquemes', 2023, True, 532.15, 15])

print(f'\nElementos da tupla: {tupla}')

print(f'\nA quantida de elementos da tupla é: {len(tupla)} \n')

print(input('Tecle <ENTER>'))

print(f'\nInserindo um elemento na lista que está na tupla')

tupla[3].append('Curso de ADS')

print(f'\nTupla modificada: {tupla}\n')

print(input('Tecle enter para continuar<ENTER>'))

tupla[3][4] = False

print(f'\nTupla com valor False alterandoa posição 4 {tupla} \n')

print(input('Tecle <ENTER>'))

tupla[3].insert(1,'IFRO')
print(f'\nTupla com o novo elemento inserido no índice 1 da lista: {tupla} \n')

print(input('Telce <ENTER>'))
tupla[3].pop()
print(f'\nTupla após retirado o último elemento da lista: {tupla} \n')

print(input('tecle <ENTER>'))
tupla[3].pop(3)
print(f'\nTupla após retirado o elemento do índice 3 da lista: ', tupla)
```

```
Lista original: ['Campus', 'Ariquemes', 2023, True, 532.15]
<class 'list'>

Tupla gerada a partir da lista = ('Campus', 'Ariquemes', 2023, True, 532.15)
<class 'tuple'>

Claudinei@Marcias-MBP proj_alp_23 % Python aula_27_04.py

Elementos da tupla: ('a', 'b', 12, ['Campus', 'Ariquemes', 2023, True, 532.15, 15])

A quantida de elementos da tupla é: 4

Tecle <ENTER>


Inserindo um elemento na lista que está na tupla

Tupla modificada: ('a', 'b', 12, ['Campus', 'Ariquemes', 2023, True, 532.15, 15, 'Curso de ADS'])

Tecle enter para continuar<ENTER>


Tupla com valor False alterandoa posição 4 ('a', 'b', 12, ['Campus', 'Ariquemes', 2023, True, False, 15, 'Curso de ADS'])

Tecle <ENTER>


Tupla com o novo elemento inserido no índice 1 da lista: ('a', 'b', 12, ['Campus', 'IFRO', 'Ariquemes', 2023, True, False, 15, 'Curso de ADS'])

Telce <ENTER>


Tupla após retirado o último elemento da lista: ('a', 'b', 12, ['Campus', 'IFRO', 'Ariquemes', 2023, True, False, 15])

tecle <ENTER>


Tupla após retirado o elemento do índice 3 da lista:  ('a', 'b', 12, ['Campus', 'IFRO', 'Ariquemes', True, False, 15])
```

Desempacotamento de tuplas com listas

```
a, b, c, d = ('IFRO', 'Ariquemes', [1, 2], 2011)

print(f'\nElemento a da tupla: {a}')
print(type(a))

print(f'\nElemento a da tupla: {b}')
print(type(b))

print(f'\nElemento a da tupla: {c}')
print(type(c))

print(f'\nElemento a da tupla: {d}')
print(type(d))
```

```
Elemento a da tupla: IFRO
<class 'str'>

Elemento a da tupla: Ariquemes
<class 'str'>

Elemento a da tupla: [1, 2]
<class 'list'>

Elemento a da tupla: 2011
<class 'int'>
```

Usando * (asterisco) para desempacotar vários elementos dentro de listas em tuplas:

```
print(f'\nIniciando com a fórmula *a, b, c = ([IFRO, Ariquemes, 1, 2, 2011]')

*a, b, c = (['IFRO', 'Ariquemes', 1, 2, 2011])

print(f'\na recebeu todos os elementos da lista, menos o último que foi para c e o penúltimo que foi para b. a = {a}')

print(f'\nO penúlltimo elemento da lista armazenado em b é = {b}')

print(f'\nO úlimo elemento da lista armazenado em c é = {c} \n')

print(input('Tecle <Enter>'))

print(f'\nMudando a fórmula para a, *b, c = ([IFRO, Ariquemes, 1, 2, 2011]')

a, *b, c = (['IFRO', 'Ariquemes', 1, 2, 2011])

print(f'\nPrimeiro elemento da lista armazenado em a é = {a}')

print(f'\n*b recebeu todos os elementos da lista, menos o primeiro que foi para a e o último que foi para c. b = {b}')

print(f'\nÚltimo elemento da lista armazenado em c é = {c} \n')

print(input('tecle <Enter>'))

print(f'\nMudando a fórmula para a, b, *c = ([IFRO, Ariquemes, 1, 2, 2011]')

a, b, *c = ([1, 2, 3, 4,5 ])

print(f'\nPrimeiro elemento da lista armazenado em a é = {a}')

print(f'\nSegundo elemento da lista armazenado em b é = {b}')

print(f'\n*c recebeu todos os elementos da lista, menos o primeiro que foi para a e o segudo que foi para b. c = {c} \n')
```

```
Iniciando com a fórmula *a, b, c = ([IFRO, Ariquemes, 1, 2, 2011]

a recebeu todos os elementos da lista, menos o último que foi para c e o penúltimo que foi para b. a = ['IFRO', 'Ariquemes', 1]

O penúlltimo elemento da lista armazenado em b é = 2

O úlimo elemento da lista armazenado em c é = 2011

Tecle <Enter>


Mudando a fórmula para a, *b, c = ([IFRO, Ariquemes, 1, 2, 2011]

Primeiro elemento da lista armazenado em a é = IFRO

*b recebeu todos os elementos da lista, menos o primeiro que foi para a e o último que foi para c. b = ['Ariquemes', 1, 2]

Último elemento da lista armazenado em c é = 2011

tecle <Enter>


Mudando a fórmula para a, b, *c = ([IFRO, Ariquemes, 1, 2, 2011]

Primeiro elemento da lista armazenado em a é = 1

Segundo elemento da lista armazenado em b é = 2

*c recebeu todos os elementos da lista, menos o primeiro que foi para a e o segudo que foi para b. c = [3, 4, 5]
```

Porque utilizar tuplas:

- Tuplas são mais rápidas que listas;
- Por serem imutáveis, tuplas podem deixar seu código mais seguro;
- Nas tuplas não temos o problema do *Shalow copy*

Quando utilizar tuplas:

- Sempre que não precisarmos mudar os dados contidos em uma coleção. Exemplo: meses do ano, dias por extenso da semana...

Cuidados que devemos tomar ao criar tuplas:

- Parênteses são opcionais, no entanto, sua utilização deixa o *script* mais legível.
- Métodos de adição, alteração e remoção de elementos em tuplas não existem, devido ao fato das tuplas serem imutáveis.

- **Atenção:** se necessário, podemos criar tuplas vazias usando o comando: tupla = ().

Atenção: Poste aqui os exercícios e códigos exemplos da aula de 19-08-2024 até o dia 26-08-2024 às 18:30 – Vale até 10 pontos.