TUPLA VS LISTAS

As duas opções permitem armazenar elementos de forma ordenada. Porém, há algumas diferenças:

**Sintaxe**:  
-> lista = [1, ‘olá, mundo’, True, 9.7]

-> tupla = (1, ‘olá, mundo’, True, 9.7)

**Mutualidade**:

- Aqui está a maior diferença entre as duas.

- As listas são estruturar mutáveis, o que significa que é possível modificar seus elementos, adicionar novos itens ou remover existentes após a criação da lista, podendo inclusive utilizar funções próprias para isso como *append()*, *remove()*, *pop()* e *insert()*.

- Ao contrário das listas, as tuplas são imutáveis. Seus elementos não podem ser alterados, adicionados ou removidos.

**Desempenho**:

- As tuplas têm um desempenho melhor. São mais eficientes em termos de espaço e processamento em determinados cenários. Não significa que seja a melhor opção em qualquer situação. Sempre é necessário analisar antes de utilizar.

- Sendo assim, as listam podem ser menos eficientes em termos de desempenho para operações específicas, especialmente quando se trata de manipulação de grandes conjuntos de dados, devido à sua mutabilidade.

**Quando utilizar cada estrutura?**

- As listas são ideais quando você precisa de uma coleção de elementos que pode ser modificada ao longo do tempo. Por exemplo, ao criar uma lista de tarefas que pode ser atualizada com novos itens ou marcada como concluída.

Além disso, se você precisa adicionar, remover ou modificar elementos com frequência, as listas oferecem métodos próprios convenientes, como mencionado anteriormente. Podemos ver isso com uma lista de compras, como mostrado no código a seguir:

# Criando uma lista de compras

lista\_de\_compras = ["Maçã", "Banana", "Leite", "Pão", "Queijo"]

# Adicionando um item à lista

lista\_de\_compras.append("Ovos")

# Removendo um item da lista

lista\_de\_compras.remove("Banana")

# Exibindo a lista

print("Lista de Compras:")

**for** item **in** lista\_de\_compras:

print("- " + item)

- Já as tuplas são apropriadas quando se deseja garantir que os elementos não sejam alterados após a criação. Isso é útil para dados que devem permanecer constantes ao longo do tempo. Como as tuplas são imutáveis, elas podem ter um desempenho ligeiramente melhor em operações de leitura e acesso a elementos. Isso também as tornam adequadas para situações em que a eficiência é crucial.

Podemos ver um exemplo de uma tupla que armazena as coordenadas geográficas de uma localização específica, que são dados imutáveis que queremos ter acesso, como mostrado a seguir:

# Definindo uma tupla de coordenadas geográficas

coordenadas\_gps = (40.7128, -74.0060)

# Exibindo as coordenadas

print("Coordenadas GPS:")

print("Latitude:", coordenadas\_gps[0])

print("Longitude:", coordenadas\_gps[1])

Obs: Lembre-se de sempre avaliar as condições de seu projeto e escolher qual das duas estruturas é mais adequada para cada situação.