PYTHON: LIST VS TUPLE

What is a List in Python?

Definition: A **List** in Python is a collection of values that is **mutable** and **ordered**. Being mutable means that you can change the contents of the list after it has been created. It can store multiple items in a single variable, and the items are kept in the same order in which they were added.

```python
# -------------list
"""
List in python is mutable and ordered means you can update or delete the items from list after creation.
List contains all type of data like string, integer, float, boolean.
"""
lst: list = ["apple", "banana", 21, 32.4, True, False]
```

Properties:

- **Mutable:** You can update, delete, or add elements after creation.
- **Ordered:** The order in which you insert items is preserved.
- **Dynamic:** List size can grow or shrink.
- **Can hold mixed data types:** Integers, strings, booleans, etc.

What is a Tuple in Python?

Definition: A **Tuple** in Python is a collection of elements that is **ordered** and **immutable**. Ordered means the elements are stored in the same sequence as they were added, and immutable means once the tuple is created, you **cannot change, add, or remove** any of its elements. This makes tuples a good choice when you want to store data that should remain **constant** throughout the program.

```python
# -------------tuple
"""
tuple in python is immutable and ordered means you cannot update tuple after creation.
tuple contains all type of data like string, integer, float, boolean.
"""
tuple = ("pakistan", "sky", 2, 4.1, True, False)
```

Properties:

- **Immutable:** Cannot change its elements after creation.
- **Ordered:** Maintains insertion order.
- **Faster:** Due to immutability, Python optimizes them better.
- **Can hold mixed data types:** Integers, strings, booleans, etc.

Mutable vs Immutable:

Feature	List (Mutable)	Tuple (Immutable)
Can you change elements?	Yes	No
Can you add/remove items?	Yes (append, pop, etc.)	No
Use as dictionary key?	No (not hashable)	Yes (hashable)
Performance	Slower (more flexible)	Faster (fixed size/structure)
Memory Usage	Higher	Lower

List Methods:

Method	Description
append(x)	Adds element at the end
insert(i, x)	Inserts at a specific index
pop()	Removes and returns the last item
remove(x)	Removes the first occurrence of x
sort()	Sorts the list
reverse()	Reverses the list
clear()	Empties the list

Example:

```python
# Example
fruits = ["apple", "orange", "strawberry"]

fruits.append("mango")
print(fruits)  # ['apple', 'orange', 'strawberry', 'mango']

fruits.pop()
print(fruits)  # ['apple', 'orange', 'strawberry']
```

Tuple Methods:

Method	Description
count(x)	Returns number of times x appears
index(x)	Returns the index of first x

Example:

```python
# Example
nums = (1, 2, 2, 3)

# Counts how many times 2 appears in the tuple
print(nums.count(2))  # Output: 2

# Prints the index of given number
print(nums.index(1))  # Output: 0
```

When to Use List vs Tuple?

Use a List when:

- You expect to **change**, **add**, or **delete** elements.
- The data is **dynamic** in nature.
- You need to use many **list methods** (like sort, reverse, etc.).

Example Use Cases:

- Shopping cart items
- User input collection
- Dynamic configurations

Use a Tuple when:

- Data should **not change** (for safety).
- You're returning a fixed group of values from a function.
- You want **better performance**.
- You want to use it as a **key in a dictionary**.

Example Use Cases:

- Coordinates (latitude, longitude)
- RGB color values
- Immutable settings or constants
- Database row records

Security & Safety:

- **Tuples are safer** when you don't want data to be accidentally changed.
- Tuples are preferred in **multi-threaded applications** due to immutability.

Presented by:

Narmeen Asghar