

🔗 Tuples and Sets in Python

🔗 1. Tuples in Python

A tuple is a collection of items that is **ordered** and **immutable** (unchangeable). Tuples are similar to lists, but once a tuple is created, you cannot modify it. They are often used to group related data together.

🔗 Syntax:

```
my_tuple = (element1, element2, element3)
```

🔗 Example:

```
my_tuple = ("apple", "banana", "orange")  
numbers_tuple = (1, 2, 3, 4, 5)
```

🔗 2. Accessing Tuple Elements

You can access elements in a tuple using **indexing**, just like with lists.

Tuples also support negative indexing.

🔗 Example:

```
fruits = ("apple", "banana")  
print(fruits[0]) # Output: apple  
print(fruits[-1]) # Output: banana
```

🔗 Slicing Tuples:

You can also slice tuples to access a subset of the elements.

```
print(fruits[1:3]) # Output: ('banana',)
```

🔗 3. Tuple Operations

Although tuples are immutable, you can perform various operations with them.

🔗 Tuple Concatenation:

You can combine two or more tuples using the `+` operator.

```
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
combined_tuple = tuple1 +
print(combined_tuple) #
```

🔗 Tuple Repetition:

You can repeat a tuple multiple times using the `*` operator.

```
repeated_tuple = (1, 2) *
print(repeated_tuple) #
```

🔗 Tuple Repetition:

You can repeat a tuple multiple times using the `*` operator.

```
repeated_tuple = (1, 2) * 3  
print(repeated_tuple) #
```

🔗 Checking Membership:

You can check if an item exists in a tuple using the `in` operator.

```
print("apple" in fruits)
```

🔗 4. Tuple Methods

Though tuples are immutable, Python provides some built-in methods for working with tuples.

- `count()` : Returns the number of times an element appears in the tuple.

```
my_tuple = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
print(my_tuple.count(1))
```

- `index()` : Returns the index of the first occurrence of an element.

```
my_tuple = ("apple", "banana", "cherry", "apple", "orange")
print(my_tuple.index("apple"))
```

5. Advantages of Using Tuples

- **Immutable:** This property ensures that tuple data cannot be modified after creation, making them useful for fixed data.
- **Faster than Lists:** Due to immutability, tuples are generally faster than lists.
- **Can Be Used as Keys in Dictionaries:** Since tuples are hashable, they can be used as keys in dictionaries, unlike lists.

🔗 6. Sets in Python

A set is a collection of **unique** items that is **unordered** and **unindexed**.

Sets do not allow duplicate values.

Sets are useful for performing operations like union, intersection, and difference.

🔗 Syntax:

```
my_set = {element1, element2}
```

🔗 Example:

```
fruits_set = {"apple", "banana"}  
numbers_set = {1, 2, 3, 4}
```

🔗 Empty Set:

To create an empty set, use the `set()` function (not `{}`, which creates an empty dictionary).

```
empty_set = set()
```


🔗 7. Set Operations

Sets support mathematical operations like union, intersection, and difference.

🔗 Union:

The union of two sets combines all elements from both sets, removing duplicates.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1 | set2
```



🔗 Intersection:

The intersection of two sets returns elements that are common to both sets.

```
intersection_set = set1 & set2
```



🔗 Difference:

The difference between two sets returns elements that are in the first set but not in the second.

```
difference_set = set1 - s
```

🔗 Symmetric Difference:

The symmetric difference returns elements that are in either of the sets but not in both.

```
sym_diff_set = set1 ^ set
```

🔗 8. Set Methods


Sets come with several useful methods for performing common tasks.

- `add()` : Adds an element to the set.


🔗 8. Set Methods

Sets come with several useful methods for performing common tasks.


- `add()` : Adds an element to the set.

```
fruits_set.add("orang"   
print(fruits_set) #
```

- `remove()` : Removes a specified element from the set. Raises an error if the element does not exist.

```
fruits_set.remove("ba"   
print(fruits_set) #
```

- `discard()` : Removes a specified element without raising an error if it does not exist.

```
fruits_set.discard("b" 
```

- `pop()` : Removes a random element from the set.

```
fruits_set.pop()
```



- `clear()` : Removes all elements from the set.

```
fruits_set.clear()
```





9. Differences Between Lists, Tuples, and Sets

Feature	List	Tuple
Ordering	Ordered	Ordered
Mutability	Mutable	Immutable
Duplicates	Allows duplicates	Allows duplicates
Indexing	Supports indexing	Support indexing
Operations	List operations	Tuple operation
Common Uses	General collection	Fixed data

Homework

1. Tuple Operations:

- Create a tuple with 5 elements.
- Try to modify one of the elements. What happens?
- Perform slicing on the tuple to extract the second and third elements.
- Concatenate the tuple with another tuple.

2. Set Operations:

- Create two sets: one with your favorite fruits and another with your friend's favorite fruits.
- Find the union, intersection, and difference between the two sets.
- Add a new fruit to your set.
- Remove a fruit from your set using both `remove()` and `discard()`. What happens when the fruit doesn't exist?

3. Tuple and Set Comparison:

- Create a list of elements and convert it into both a tuple and a set.
- Print both the tuple and the set.
- Try to add new elements to the tuple and set. What differences do you observe?