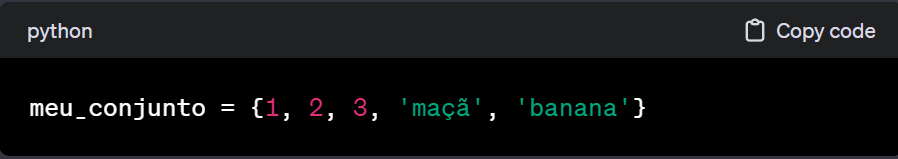
PYTHON sets

[Set in Python](https://www.w3schools.com/python/python_sets.asp)

A set in Python is an unordered collection of unique elements. It is defined by enclosing elements in curly braces {}, separated by commas.

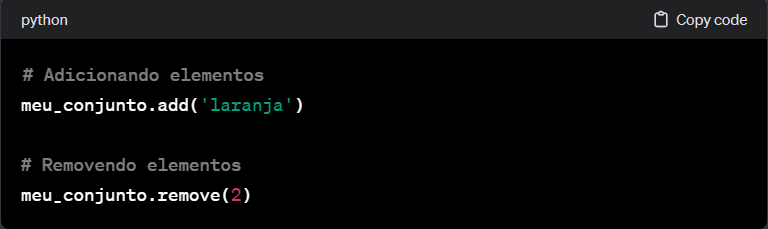
Sets are similar to lists and tuples, but don't allow duplicate elements.

**CREATING SET**

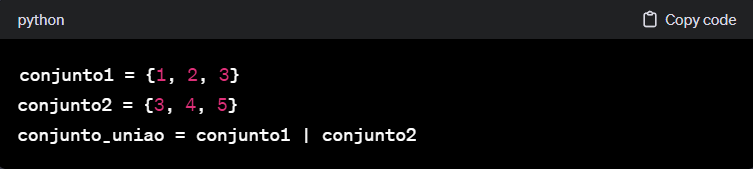
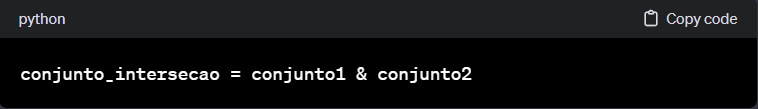
 You can create a set using the constructor set( ) or simply enclosing the elements in curly brackets { }

**ACCESSING AND MODIFYING**

Sets are unordered, so they don't support indexing. However, you can iterate through the elements using a loop. You can't modify elements individually, but you can add or remove elements.

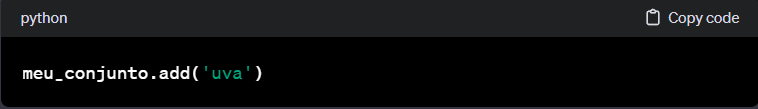


**OPERATIONS**

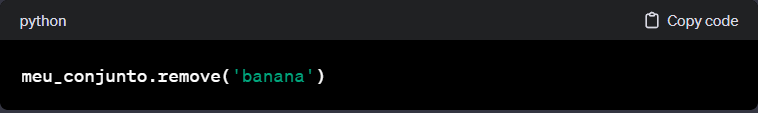
* **Union (|):** Combines two sets, returning a new set with all the unique elements of both sets.
* **Intersection (&):** Returns a new set with elements common to both sets.
* **Difference (-):** Returns a new set with elements from the first set that are not present in the second set.

[**SET METHODS**](https://www.w3schools.com/python/python_sets_methods.asp)

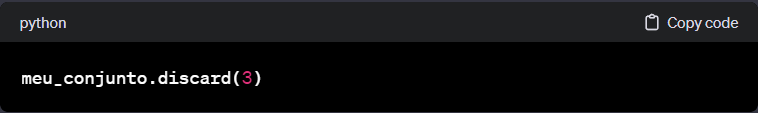
* **Add():** Adds an element to the set.



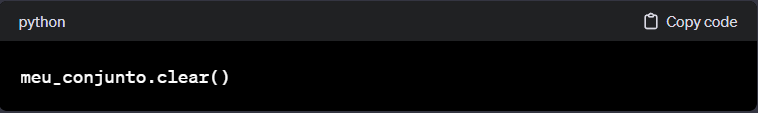
* **Remove():** Removes a specified element from the set. Generates an error if the element is not present.



* **Discard():** Removes a specified element from the set. It does not generate an error if the element is not present.



* **Clear(): Removes** all elements from the set.



**WHEN TO USE SETS**

* When you need to store a collection of unique elements;
* When the order is not important;
* When you want to perform set operations such as join, intersection, and difference.

Sets are particularly useful in scenarios where you need to work with single elements and perform mathematical operations on sets efficiently.