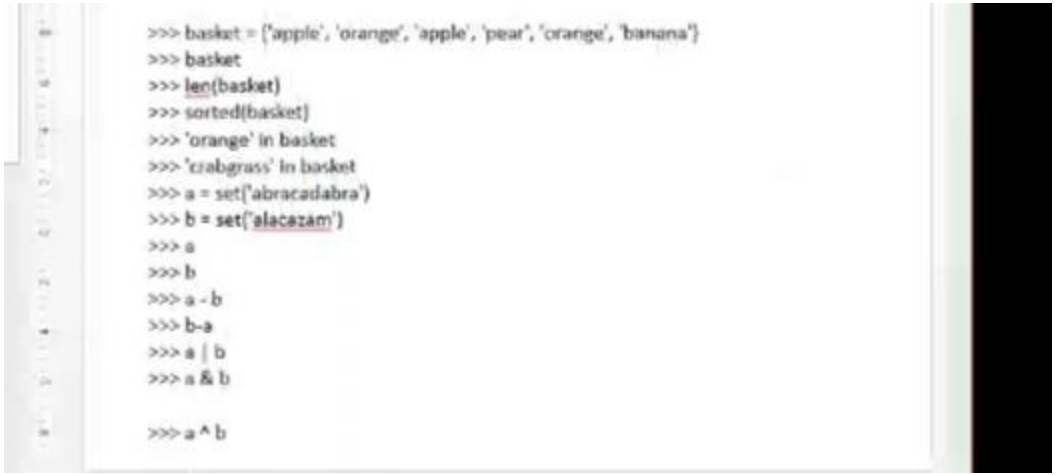
Sets

- Python also includes a data type for sets.
- A set is an unordered collection with no duplicate items. Primary uses of sets include membership testing and eliminating duplicate entries. Sets also support mathematical operations, such as union, intersection, difference, and symmetric difference.
- Curly braces **{ }** or the set() function can be used to create sets with a comma-separated list of items inside curly brackets **{ }**.

Sets

- A set is a collection of unique items. Duplicate items will be removed
- Note: to create an empty set you have to use set() and not **{ }** as the latter creates an empty dictionary.
- Sets are mutable.
- Indexing is not possible in sets, since set items are unordered.
- You cannot access or change an item of the set using indexing or slicing.


>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> basket
>>> len(basket)
>>> sorted(basket)
>>> 'orange' in basket
>>> 'crabgrass' in basket
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a
>>> b
>>> a - b
>>> b-a
>>> a | b
>>> a & b
>>> a ^ b

issubset()	set_name.issubset(*other*)	The *issubset()* method returns True if every item in the set *set_name* is in *other* set.
issuperset()	set_name.issuperset(*other*)	The *issuperset()* method returns True if every element in *other* set is in the set *set_name*.
pop()	set_name.pop()	The method *pop()* removes and returns an arbitrary item from the set *set_name*. It raises KeyError if the set is empty.
remove()	set_name.remove(*item*)	The method *remove()* removes an *item* from the set *set_name*. It raises KeyError if the *item* is not contained in the set.
symmetric_ difference()	set_name. symmetric_difference(other)	The method *symmetric_difference()* returns a new set with items in either the set or other but not both.
union()	set_name.union(**others*)	The method *union()* returns a new set with items from the set *set_name* and all *others* sets.
update()	set_name.update(**others*)	Update the set *set_name* by adding items from all *others* sets.

Note: Replace the words *"set_name", "other" and "others"* mentioned in the syntax with your *actual set names* in your code.

Traversing of Sets

Program to Iterate Over Items in Sets Using *for* Loop

```
warships = {"u.s.s._arizona", "hms_beagle", "ins_airavat", "ins_hetz"}
for i in warships:
    print(f"{i} is a Warship")
```

Write a Function Which Receives a Variable Number of Strings as Arguments. Find Unique Characters in Each String

```
def find_unique(*all_words):
    for i in all_words:
        unique_list = list(set(i))
        print(f"Unique characters in the word {i} are {unique_list}")

find_unique("egg", "immune", "feed", "vacuum", "goddessship")
```

Frozenset

- A frozenset is basically the same as a set, except that it is immutable.
- Once a frozenset is created, then its items cannot be changed. Since they are immutable, they can be used as members in other sets and as dictionary keys.
- The frozensets have the same functions as normal sets, except none of the functions that change the contents (update, remove, pop,etc.) are available.

- >>> dir(frozenset)