|  |  |
| --- | --- |
| **| B  A.union(B)** | Returns a set which is the union of sets A and B. |
| **A |= B  A.update(B)** | Adds all elements of array B to the set A. |
| **A & B  A.intersection(B)** | Returns a set which is the intersection of sets A and B. |
| **A &= B  A.intersection\_update(B)** | Leaves in the set A only items that belong to the set B. |
| **A - B  A.difference(B)** | Returns the set difference of A and B (the elements included in A, but not included in B). |
| **A -= B  A.difference\_update(B)** | Removes all elements of B from the set A. (discard) |
| **A ^ B  A.symmetric\_difference(B)** | Returns the symmetric difference of sets A and B (the elements belonging to either A or B, but not to both sets simultaneously). |
| **A ^= B  A.symmetric\_difference\_update(B)** | Writes in A the symmetric difference of sets A and B. |
| **A <= B  A.issubset(B)** | Returns true if A is a subset of B. |
| **A >= B  A.issuperset(B)** | Returns true if B is a subset of A. |
| **A < B** | Equivalent to A <= B and A != B |
| **A > B** | Equivalent to A >= B and A != B |

**1. [What is a set](https://snakify.org/en/lessons/sets/" \l "section_1)**

*Set* in Python is a data structure equivalent to sets in mathematics. It may consist of various elements; the order of elements in a set is undefined. You can add and delete elements of a set, you can iterate the elements of the set, you can perform standard operations on sets (union, intersection, difference). Besides that, you can check if an element belongs to a set.

Unlike arrays, where the elements are stored as ordered list, the order of elements in a set is undefined (moreover, the set elements are usually not stored in order of appearance in the set; this allows checking if an element belongs to a set faster than just going through all the elements of the set).

Any immutable data type can be an element of a set: a number, a string, a tuple. Mutable (changeable) data types cannot be elements of the set. In particular, list cannot be an element of a set (but tuple can), and another set cannot be an element of a set. The requirement of immutability follows from the way how do computers represent sets in memory.

s already mentioned, Python sets are similar to Python Strings, Lists and Tuples, as they are sequence of Python objects. But, unlike all of them, Python sets are Unordered collections, just like [Python Dictionaries](http://www.codeninja.in/2016/08/python-dictionaries-introduction-and-basic-operations.html), as elements in Python sets are not associated with any positional parameters which we call as offset or index. Like Python Lists and unlike Python Strings and Tuples, Python sets are mutable, which means that, we can add, remove or modify the contents of a Python set. Unlike other Python data structures, Python sets have a unique and important property - every element in a Python set is unique across the sequence. To rephrase this, every element in a Python set can have only one occurrence in that particular set, no multiple occurrences are allowed. Also, another very important thing is that, every element in the Python set is a Python immutable object - which means it may be a str type or an int type, but it cannot be a list type.