set

1. sets are mutable
2. sets are represented by using {}, but we use {} for dictionary also, hence s={} will not create empty set, so to create empty set use s=set()
3. It is unordered collection of immutable data.
4. It is a collection of heterogeneous data, which is immutable, hence we can store numbers, strings, tuples, frozensets in the set, but we cannot store list, set, and dictionary.
5. It stores only unique values.
6. Since it is unordered, we cannot use index to access data, and hence we cannot access data randomly from the sets.

to create a empty set

s=set()

using set we can remove duplicate characters from strings

s="this is string"

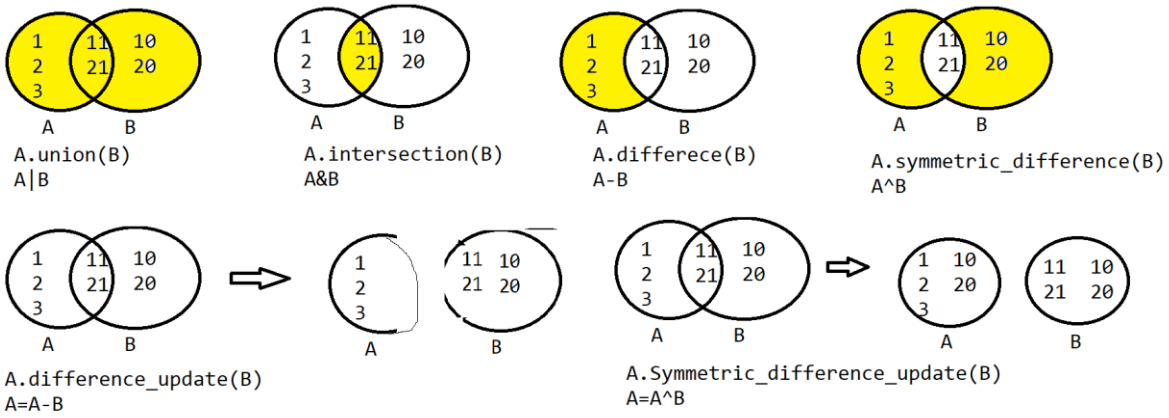
s1=set(s)

to find length of the set

len(s1)

s.add(value)	add the value in the set if it doesnot exists
s.update(iterable)	to add multiple values in the set
s.pop()	it will delete the random value, and it will return the value
s.remove()	if you want delete by value, then pass the value to remove function, it will remove the value if exists, otherwise throw an exception
s.discard()	if you want delete by value, then pass the value to discard function, it will remove the value if exists, otherwise it will ignore
s.copy()	to create a shallow copy of the set
s.clear()	to delete all the values from the set
s.union(s1) s|s1	it will return a new set, which all the values from both the sets
s.intersection(s1) s&s1	it will return a new set, which has only common values in the both the set
s.difference(s1) s-s1	it returns a new set, which has values in s but not in s1
s.symmetric_difference(s1) s^s1	it returns a new set, which has all the values either only in s or only in s1
s.difference_update(s1) s=s-s1	it will overwrite set s, it will replace it with all values which are only in s

s.symmetric_difference_update(s1) s=s^s1	it overwrites set s, which has all the values either only in s or only in s1
s.issubset(s1) s<s1	If s1 has all the values of s and some extra values, then it will return true
s.isdisjoint(s1)	If no elements are common in 2 sets, the it will return true, otherwise return false
s.issuperset(s1) s>s1	If s has all the values of s1 and some extra values, then it will return true, otherwise return false



Dictionary

When you need to store the data in the key: value format then use dictionary

1. dictionary is represented as {}
2. all the keys in the dictionary are unique and immutable objects, so a key can be of type number, string, tuple, frozenset
3. dictionary is an ordered collection.
4. keys are unique but values can be duplicate.
5. keys can be used as index, so random access is possible with keys

d[“aaa”]=34	It will add a key:value pair in the dictionary if key is not there. and if if key is there then it will overwrite the value of the key
d.update(d1)	it will all the keys of d1 into dictionary d, d={“a”:10,”b”:100} d1={“b”:200,”c”:345,”d1”:456} d.update(d1) then the dictionary d will be d={“a”:10,”b”:200,”c”:345”,”d1”:456}
d3={**d,**d1}	d3={“a”:10,“b”:200,”c”:345,”d1”:456}
d.pop(key,[default value])	It will delete the key:value pair and return the value if key exists

	but if key does not exists, and default value is not given, then it will throw exception. but if key doesnot exists and default value is given, then it will return default value
d.popitem()	it will delete the last key:value pair, and it will return a tuple (key,value)
d.get(key,[default value])	It will return the value of the given key, if key exists. but if key is not there and default value is not given, then it will return None, and if key does not exists, and the default value is given then it will return default value
d.setdefault(key,default value)	if key exists, then it will return the value of the key, but if the key does not exists, then it will add key:defaultvalue pair in the dictionary and returns default value
d.keys()	it will return list of all keys
d.values()	it will return list of values
d.items()	It will return a iterable objects in which (key,value) are returned one by one, in the form of tuple
d.copy()	It will create a shallow copy of the dictionary
d.clear()	it will delete all (key,value) pairs and keep empty dictionary
dict.fromkeys(lst,data)	it will add all values from the list as keys in the dictionary, and every keys value will be = data lst=[1,2,3] dict.fromkeys(lst,100)-→ {1:100,2:100,3:100}