In [2]:
```
#copy()
lis = [4,5,6,7,8,34]
a = lis.copy()
print(a)
print(lis)
```

```
[4, 5, 6, 7, 8, 34]
[4, 5, 6, 7, 8, 34]
```

In [3]:
```
#count()
lis = [4,5,6,7,8,34,5,3,5]
lis.count(5)
```

Out[3]: 3

In [4]:
```
#index()
lis = [4,5,6,7,8,34,5,3,5]
lis.index(5)
```

Out[4]: 1

In [6]:
```
lis.index(5,2)
```

Out[6]: 6

In [7]:
```
#reverse()
lis = [4,5,6,7,8,34,5,3,5]
lis.reverse()
print(lis)
```

```
[5, 3, 5, 34, 8, 7, 6, 5, 4]
```

In [8]:
```
#sort()
lis = [4,5,6,7,8,34,5,3,5]
lis.sort()
print(lis)
```

```
[3, 4, 5, 5, 5, 6, 7, 8, 34]
```

In [9]:
```
#sort(reverse=True)
lis = [4,5,6,7,8,34,5,3,5]
lis.sort(reverse=True)
print(lis)
```

```
[34, 8, 7, 6, 5, 5, 5, 4, 3]
```

In [12]:
```
lis = [2,3,[12,13,14,[20,40,60],90,89],30,20]
lis[0]
```

Out[12]: 2

In [13]:
```
lis[2]
```

Out[13]: [12, 13, 14, [20, 40, 60], 90, 89]

In [14]:
```
lis[2][3][0]
```

Out[14]: 20

In [15]:
```
lis[2][5]
```

Out[15]: 89

In [16]:
```
lis[4]
```

Out[16]: 20

Other Operations on list

In [ ]:

In [ ]:

In [ ]:

len() :- It is a built-in function that gives the length of any sequence.

```
Syntax :
    len(var)
```

In [10]:
```
a = 'heloo'
len(a)
```

Out[10]: 5

In [11]:
```
lis = [45,6,7,8,9,34]
len(lis)
```

Out[11]: 6

Q. Wapp to enter your fullname and print the surname.

```
input >> 'Gitanjali Digal'
output >> 'Digal'
```

Q. Wapp to add 7000 after 6000 in the given list:

lis = [10,20,[300,400,[5000,6000],500],30,40]

In [22]:
```
lis = [10,20,[300,400,[5000,6000],500],30,40]
lis[2][2].insert(2,7000)
print(lis)
```

```
[10, 20, [300, 400, [5000, 6000, 7000], 500], 30, 40]
```

Q. Marks obtain by the students are given in a list. Find the student who stood 3rd in the class.

```
Marks = [85,86,78,88,89,90,87,65,76,56,98,95]
```

Q. Given a list of animals perform all the operations on the same list.

animals=['cat','dog','parrot','tiger','zebra']

1. add a new animal 'lion' to the list.
2. add 'bear' in between 'dog' and 'parrot'.
3. add some more animals like: 'tiger','rabbit','horse'.
4. delete the 2nd tiger from the list.
5. print the final list of animals.
6. print the total no.of animals in the list.

In [ ]:

In [ ]:

In [ ]:

Tuple :- It is an immutable ordered sequence of items in a ().

1. Items are indexed.
2. Items can't be changed.

In [17]:
```
t = (67,89,45,34)
type(t)
```

Out[17]: tuple

In [18]:
```
a = (4,)
type(a)
```

Out[18]: tuple

In [ ]:

In [ ]:

Tuple Attributes :

count()	The number of times the specified element occurs in the tuple
index()	Returns index place of the first occurance of item

In [20]:
```
#count()
a = (78,54,34,2,4,6,2)
a.count(2)
```

Out[20]: 2

In [21]:
```
#index()
a = (78,54,34,2,4,6,2)
a.index(2)
```

Out[21]: 3

In [ ]:

In [ ]:

Set :- It is a collection of items inside a {}.

1. Set items are unordered.
2. Do not allow duplicate values.
3. Supports all mathematical set operation.

In [25]:
```
s = {4,5,6,7,84,23,24,4345,322,1,4,4,6,7}
type(s)
```

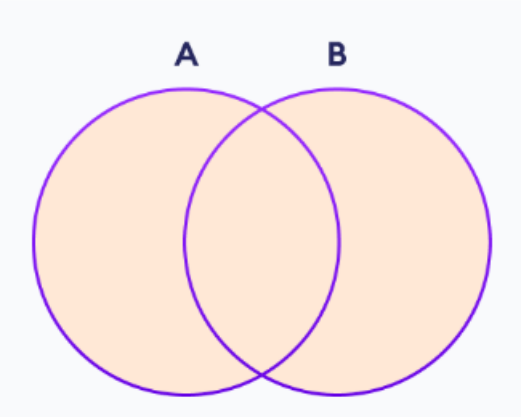
Out[25]: set

In [26]:
```
print(s)
```

```
{1, 322, 4, 5, 6, 7, 84, 23, 24, 4345}
```

Set Attributes :

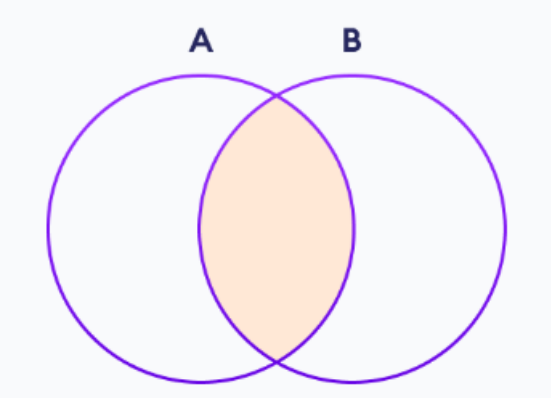
1. Union of Two Sets : The union of two sets A and B include all the elements of set A and B.



In [27]:
```
s1 = {3,4,5,6,7,8,9}
s2 = {4,5,3,2,4,5,9}
s1.union(s2)
```

Out[27]: {2, 3, 4, 5, 6, 7, 8, 9}

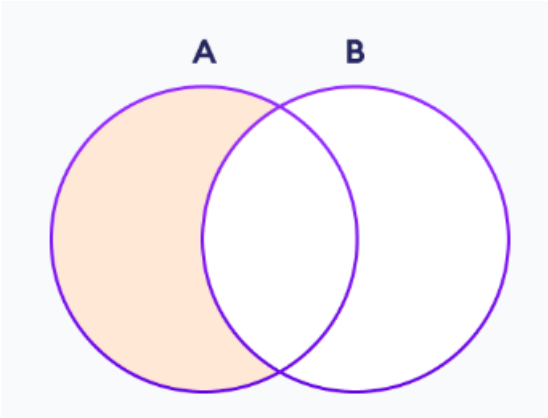
2. Set Intersection : The intersection of two sets A and B include the common elements between set A and B.



In [28]:
```
s1 = {3,4,5,6,7,8,9}
s2 = {4,5,3,2,4,5,9}
s1.intersection(s2)
```

Out[28]:
```
{3, 4, 5, 9}
```

3. Difference : The difference between two sets A and B include elements of set A that are not present on set B.



In [29]:
```
s1 = {3,4,5,6,7,8,9}
s2 = {4,5,3,2,4,5,9}
s1.difference(s2)
```

Out[29]:
```
{6, 7, 8}
```

In [31]:
```
#isdisjoint()
s = {1,2,3,4}
s2 = {4,5,6,7}
s3 = {8,9}
s.isdisjoint(s2)
```

Out[31]:
```
False
```

In [32]:
```
s.isdisjoint(s3)
```

Out[32]:
```
True
```

In [ ]:

Dictionary : consist of key:value pairs inside a curly braces.

1. It can have keys and values of different data types.

In [33]:
```
d = {1:'A',2:'B',3:'C'}
print(d)
type(d)
```

```
{1: 'A', 2: 'B', 3: 'C'}
```

Out[33]:
```
dict
```

In [ ]:

In [ ]:

Dictionary Attributes :-

```
keys()      Returns the list of the dictionary's keys.
values()    Returns the list of the dictionary's values.
items()     Returns the list of all key value pairs.
update()    If the key exists in the dictionary this will update the value of that key, otherwise this will add the
            key:value pair to the dictionary.
```

In [34]:
```
#keys()
d = {1:'A',2:'B',3:'C'}
d.keys()
```

Out[34]:
```
dict_keys([1, 2, 3])
```

In [35]:
```
#values()
d.values()
```

Out[35]:
```
dict_values(['A', 'B', 'C'])
```

In [36]:
```
#items()
d.items()
```

Out[36]:
```
dict_items([(1, 'A'), (2, 'B'), (3, 'C')])
```

In [37]:
```
d.get(1)
```

Out[37]:
```
'A'
```

In [38]:
```
#update()
d = {1:'A',2:'B',3:'C'}
d.update({4:'d'})
print(d)
```

```
{1: 'A', 2: 'B', 3: 'C', 4: 'd'}
```

In [39]:
```
d.update({4:'abc'})
print(d)
```

```
{1: 'A', 2: 'B', 3: 'C', 4: 'abc'}
```

In [ ]:

In [ ]:

Operators:

An operator is a special symbol upon which we perform some operations on variables and values.

Assignment Operators:

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

In [40]:
```
a = 5
print(a)
```

```
5
```

In [41]:
```
b = 45
b = b+5
print(b)
```

```
50
```

In [42]:
```
b = 45
b+= 5
print(b)
```

```
50
```

Arithmetic Operators:

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

In [43]:
```
#addition
2+6
```

Out[43]: 8

In [44]:
```
#subtraction
7-2
```

Out[44]: 5

In [45]:
```
#multiplication
7*3
```

Out[45]: 21

In [46]:
```
#division
7/3
```

Out[46]: 2.3333333333333335

In [47]:
```
#moduls
8%2
```

Out[47]: 0

In [48]:
```
#floor division
7//3
```

Out[48]: 2

In [49]:
```
#power
5**3
```

Out[49]: 125

Comparison Operators:

Comparison operators are used to compare two values and returns a boolean value:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

In [50]:
```
#equal
5==5
```

Out[50]: True

In [51]:
```
5==10
```

Out[51]: False

In [52]:
```
#not equal(!=)
5!=6
```

Out[52]: True

In [53]:
```
5!=5
```

Out[53]: False

In [54]:
```
#Less than
5<6
```

Out[54]: True

In [55]:
```
#Less than or equal to
5<=5
```

Out[55]: True

In [56]:
```
5<=7
```

Out[56]: True

In [57]:
```
#greater than
6>2
```

Out[57]: True

In [58]:
```
#greater than or equal to
6>=6
```

Out[58]: True

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Logical Operators:

logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

In [59]:
```
#and
8<10 and 6>2
```

Out[59]: True

In [60]:
```
8<10 and 6<2
```

Out[60]: False

In [61]:
```
#or
8<10 or 6>2
```

Out[61]: True

In [62]:
```
8<10 or 6<2
```

Out[62]: True

In [63]:
```
8>10 or 6<2
```

Out[63]: False

In [64]:
```
#not
not 8>10 or 6<2
```

Out[64]: True

In [65]:
```
not 8<10 or 6<2
```

Out[65]: False

In [ ]:

Membership Operators:

in : True if var is the sequence.
False if var is not in the sequence

Syntax : var in sequence

not in : True if var is not in the sequence.
False if var is in the sequence

Syntax : var not in sequence

In [66]:
```
#in
a = 'python'
'o' in a
```

Out[66]: True

In [67]:
```
'b' in a
```

Out[67]: False

In [68]:
```
#not in
'b' not in a
```

Out[68]: True

In [69]:
```
'o' not in a
```

Out[69]: False

In [71]:
```
a = int(input('enter three digit number: '))
print('last digit number is',a%10)
```

```
enter three digit number: 675
last digit number is 5
```

In [72]:
```
#String format
a = int(input('enter a number '))
b = int(input('Enter another number '))
print('sum of ',a,'and ',b,' is ',a+b)
```

```
enter a number 5
Enter another number 6
sum of  5 and  6  is  11
```

In [74]:
```
a = int(input('enter a number '))
b = int(input('Enter another number '))
print(f'sum of {a} and {b} is {a+b}')
```

```
enter a number 4
Enter another number 8
sum of 4 and 8 is 12
```

In [ ]:
```

```