str.rsplit(sep=None, maxsplit=- 1)
Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done

str.split(sep=None, maxsplit=- 1)
Return a list of the words in the string, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done

```
>>> str1="a,b,c,d,e"
>>> l1=str1.split(",")
>>> print(l1)
['a', 'b', 'c', 'd', 'e']
>>> str2="a,b:c,d"
>>> l2=str2.split(",")
>>> print(l2)
['a', 'b:c', 'd']
>>> str3="a,b,c d,e f"
>>> l3=str2.rsplit()
>>> l3=str3.rsplit()
>>> print(l3)
['a,b,c', 'd,e', 'f']
>>> l4=str3.split()
>>> print(l4)
['a,b,c', 'd,e', 'f']
>>> l5=str3.split(" ",1)
>>> print(l5)
['a,b,c', 'd,e f']
>>> l6=str3.rsplit(" ",1)
>>> print(l6)
['a,b,c d,e', 'f']
```

str.startswith(*prefix*[, *start*[, *end*]])
Return True if string starts with the *prefix*, otherwise return False. *prefix* can also be a tuple of prefixes to look for. With optional *start*, test string beginning at that position. With optional *end*, stop comparing string at that position.

Example:

```
namesList=['naresh','ramesh','kishore','rajesh','kiran']
for name in namesList:
    if name.startswith('r'):
        print(name)
for name in namesList:
    if name.startswith(('r','k')):
        print(name)
```

Output:

```
ramesh
rajesh
ramesh
kishore
rajesh
kiran
```

str.strip([*chars*])

Return a copy of the string with the leading and trailing characters removed. The *chars* argument is a string specifying the set of characters to be removed. If omitted or None, the *chars* argument defaults to removing whitespace. The *chars* argument is not a prefix or suffix; rather, all combinations of its values are stripped:

```
>>> str1="  nit   "
>>> str2="nit"
>>> str1==str2
False
>>> str1.strip()==str2
True
>>> str3=str1.strip()
>>> print(str3)
nit
>>> print(str1)
  nit
>>> str4="www.nareshit.com"
>>> str5=str4.strip("w.com")
>>> print(str4)
www.nareshit.com
>>> print(str5)
nareshit
>>> str6="***$%nit&**$"
>>> str7=str6.strip("*$%&")
>>> print(str6)
***$%nit&**$
>>> print(str7)
Nit
```

str.swapcase()
Return a copy of the string with uppercase characters converted to lowercase and vice versa

```
>>> str1="nARESH"
>>> str2=str1.swapcase()
>>> print(str1)
nARESH
>>> print(str2)
Naresh
```

str.title()
Return a titlecased version of the string where words start with an uppercase character and the remaining characters are lowercase

```
>>> str1="python programming language"
>>> str2=str1.title()
>>> print(str1)
python programming language
>>> print(str2)
Python Programming Language
```

str.upper()
Return a copy of the string with all the cased characters converted to uppercase

```
>>> liststr=["java","python","oracle"]
>>> for c in liststr:
    print(c.upper())


JAVA
PYTHON
ORACLE
```

str.zfill(*width*)
Return a copy of the string left filled with ASCII '0' digits to make a string of length *width*

```
>>> listStr=["1200","45678","140","12","4","123999"]
>>> for s in listStr:
...     print(s.zfill(8))
...
...
00001200
00045678
00000140
00000012
00000004
00123999
```

bytes

bytes is an immutable data type
A bytes object is an immutable array. The items are 8-bit bytes, represented by integers in the range 0 <= x < 256. Bytes literals (like b'abc') and the built-in bytes() constructor can be used to create bytes objects. Also, bytes objects can be decoded to strings via the decode() method.

```
>>> str1="ABC"
>>> b1=str1.encode()
>>> print(str1)
ABC
>>> print(b1)
b'ABC'
>>> b2=b'ABC'
>>> print(b2)
b'ABC'
>>> type(b1)
<class 'bytes'>
>>> type(b2)
<class 'bytes'>
type(str1)
<class 'str'>
```

```
>>> list1=[10,20,30,40,50]
>>> b4=bytes(list1)
>>> print(b4)
b'\n\x14\x1e(2'
>>> b5=bytes(65)
>>> print(b5)
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
>>> b6=bytes("ABC",encoding="UTF-8")
>>> print(b6)
b'ABC'
```

bytearray
A bytearray object is a mutable array. They are created by the built-in bytearray() constructor. Aside from being mutable (and hence unhashable), byte arrays otherwise provide the same interface and functionality as immutable bytes objects.

bytearray() -→ create empty byte array
bytearray(iterable) → converting list of integers to byte array
bytearray(string,encoding) → converting string to byte array
bytearray(size) → creating byte array with the specified size

Example:

```
>>> b1=bytearray([1,2,3,4,5])
>>> print(b1)
>>> bytearray(b'\x01\x02\x03\x04\x05')
for x in b1:
    print(x)


1
2
3
4
5
>>> b1[0]
1
```

```
>>> b1[1]
2
>>> b1[0]=6
>>> print(b1)
bytearray(b'\x06\x02\x03\x04\x05')
>>> del b1[0]
>>> print(b1)
bytearray(b'\x02\x03\x04\x05')
```

sequence data types

1. List → Mutable
2. Tuple
3. String
4. Range
5. Bytes
6. Bytesarray → Mutable

Set types

1. Set → Mutable
2. Frozenset → Immutable

Mapping

1. Dictionary – Mutable

Functions