String Functions

1. join()
    - s= '-'.join(t)
2. len()
3. rstrip()
4. lstrip()
5. strip()
6. upper()
7. lower()
8. swapcase()
9. title()
10. capitalize()
11. isalnum()
12. isalpha()
13. isdigit()
14. islower()
15. isupper()
16. istitle()
17. isidentifier()
18. isspace()
19. find()
    - s.find('i',7,15)
20. count()
21. replace()
    - string.replace(oldstring,newstring)
22. split()

Function to take list and tupple as input

eval()

In [2]:
```
tuple_ = eval(input("Enter tuple -:"))
print(tuple_,'\t',type(tuple_))
```

```
Enter tuple -:10,20,30
(10, 20, 30)     <class 'tuple'>
```

In [3]:
```
list_ = eval(input("Enter tuple -:"))
print(list_,'\t',type(list_))
```

```
Enter tuple -:[10,20,30]
[10, 20, 30]     <class 'list'>
```

List Functions

1. len()
2. count()
3. index()
4. append()
5. index()
6. extend()
   - merge two list syntax:l1.extend(l2)
7. remove()
   - Remove first occurrence of value.
8. pop()
   - we can also give index
9. clear()
   - Remove all items from list.
10. reverse()
11. sort(reverse=True/False)
    - gives error when used on hetrogeneous list
12. copy()
    - copys list to another variable (note: id is diffrent)

List Comprehension

syntax : list = [expresion "for" item in list "if" condition]

```
s = [x*x for x in range(1,11)]
```

```
def square(n):
    return n**2
s = [square(x) for x in range(1,11)]
```

Dictionary Functions

1. dict()
   - creates a empty dictionary
2. del
   - to delete pair from dict syntax : del dict[key]
   - to delete dict syntax : del dict
3. clear()
   - to remove all values from dict
4. update()
   - merge two dictionary syntax : dict1.update(dict2)
5. get()
   - gives value ,if not present it doesen't give error ,we can also give default value syntax : get(key,default_value)
6. popitem()
   - removes an item from last of dictionary
7. keys()
   - gives all keys of dict in list form
8. values()
   - gives all values of dict in list form
9. items()
   - gives pair of key and value in form of tuple inside a parent list
10. setdefault()

Dictionary Comprehension

In [ ]:
```
square = {x:x*x for x in range(1,6)}
```

Set Functions

1. union() or |
   - returns all elements present in both set
2. intersection() or &
   - returns common elements in both set
3. difference() or -
   - returns elements present in x but not in y if syntax is x.difference(y)
4. symmetric_difference()
   - returns uncomman elements from both set
5. issubset()
6. issuperset()
7. add()
   - to add element to set
8. update()
   - to add multiple elements in set syntax : set.update(x,y,z)
9. copy()
10. pop()
11. remove()
12. discard()
13. clear()

Lambda Function

syntax -: lambda argumentlist:expretion
eg:

1. s = lambda n:n*n
   print(s(2))
2. s = lambda a,b:a+b
   print(s(30,40))

Map Function

for every element present in given sequence apply some functionality and generate new elements with required modification
map(function,sequence)
eg:

1. l = [1,2,3,4,5]
   def double(x):
   return 2*x
   l1 = list(map(double,l))
   print(l1)
2. l = [1,2,3,4,5]
   l1 = list(map(lambda n:2*n,l))
   print(l1)

Reduce Function

reduces sequence of elements into a single elements by appling spetial function
syntax:reduce(function,sequence)
eg:

1. from functools import *
   l = [1,2,3,4,5]
   sum = reduce(lambda x,y:x+y,l)
   print(sum)
   mul = reduce(lambda x,y:x*y,l)
   print(mul)

Filter Function

filter values from given sequence based on condition
filter(function,sequence)
eg:

1. def isEvne(x):
   if x%2 == 0:
   return True
   else:
   return False
   l = [0,5,10,15,20,25,30]
   l1 = list(filter(isEvne,l))
   print(l1)
2. l1 = list(filter(lambda x:x%2 == 0,l))
   print(l1)

Frozenset Function

- it creates immutable set object from an itrable
- dont have duplicate values
  eg:

1. m = ['apple','banana','chery']
   x = frozenset(m)
   x.add('stra')