LIST : An ordered collection of 'n' values (n>=0)

*I/p - l = [ ] → O/p - <class 'list'>

Print(type(l))

*I/p - sea_creatures = ['shark', 'cuttlefish', 'squid', 'mantisshrimp', 'anemone']

Print(sea_creatures)

Print(type(sea_creatures))

Print(len(sea_creatures))

O/p - ['shark', 'cuttlefish', 'squid', 'mantisshrimp', 'anemone']

<class 'list'>

5

NOTE : Print('*' * 20) → duplicates for 20 times.

'''
This is a multi line comment demonstrated by using triple single-quote

Below mentioned list 'd' is a collection of heterogeneous data types.
'''

```
#I/P - d = []
d.append(10)                    O/P - 10, 'Hi', 20.2, True
d.append('Hi')
d.append(20.20)
d.append(True)
Print(1)
```

""" This is a multi line comment demonstrated by using triple double-quote """

```
*I/P - nest_list = ['Happy', [2, 0, 1, 5]]
       Print(nest_list[0][1])
       Print(nest_list[1][3])

O/P - a
      5
```

* append - Used as a variable in list type which means it adds an argument as a single element to the end of a list. The length of the list increases by one.

* insert - It inserts the specified value at the specified position.

SYNTAX : list_name.insert(index, element)

* remove : It is an inbuilt function that removes a given object from the list. It doesn't return any value.

SYNTAX : list_name.remove(obj)

NOTE: removes the first occurance of the object from the list

For insert

NOTE: If given index >= length(list) is given, then it inserts at the end of the list.

* Sort : this can be used to sort a list in ascending, descending or user defined order.

SYNTAX : List_name.sort( ) - ascending

* Reverse : This can be used to reverse a given list.

SYNTAX : list_name.sort(reverse = True)
└ descending

* Pop : This is an inbuilt function that removes and returns last value from the list or the given index value.

SYNTAX : list_name.pop(index)

* Count : This is an inbuilt function that returns count of how many times a given object occures in list.

SYNTAX : list_name.count(object)

* Slice : It returns a slice object or a sequence that can be used to slice strings, bytes, tuple, list or range.

SYNTAX : slice(start, stop, step).

* I/P - my_list ['a', 'b', 'c', 'd', 'e', 'f']

Print(my_list[:-5])

O/P - a

* I/P - my_list['a', 'b', 'c', 'd', 'e', 'f']

Print(my_list[-4:-2])

O/P - c, d.

Sorting a list

* I/P=c [3, 1, 2, 8, 5, 4]

Print("list before sorting:", 1)

c.sort()

Print("list after sorting:", 1)

O/P - List before sorting : [3, 1, 2, 8, 5, 4]

List after sorting : [1, 2, 3, 4, 5, 8]

Reversing a list:

I/P -

```
a = [3, 1, 2, 8, 5, 4]
Print("list before reversing:", a)
a.reverse()
Print("list after reversing:", a)
```

O/P - list before reversing: [3, 1, 2, 8, 5, 4]

list after reversing: [4, 5, 8, 2, 1, 3]

* I/P -

```
lst_1 = [1, 2, 3]
lst_2 = [4, 5, 6]
Print(lst_1 + lst_2)
```

O/P - [1, 2, 3, 4, 5, 6]

int + int - Addition

str + str
lst + lst } - joining (or) concatenate

TUPLE: It is similar to a list except that it is fixed-length and immutable. So, the values in the tuple can't be changed nor the values be added to or removed from the tuple.

↳ These are commonly used for small collections

of values that will not need to change such as an IP address and port.

↳ Tuples are represented with parentheses instead of square brackets.

> Both homogeneous and heterogeneous can be used in Tuple.

*I/P - t = ()
Print(type(t))
O/P - <class 'tuple'>

I/P - t = tuple()
Print(type(t))
O/P - <class 'tuple'>

Accessing with index :

*I/P - t = (1, 2, 3)
Print(t[-1])
O/P - 3

slicing

*I/P - t = (1, 2, 3)
Print(t[1:3])
O/P - (2,3)

With single element example :

t = tuple([']) or t = (1,) or t = 1,

* i/p - t = 1
Print (type(t))
t = ('rex')
Print (type(t))

o/p - <class 'int'>
<class 'str'>

* i/p - l = [1, 2, 3]
t = [10, 20, 30]
l[1] = 10
Print(l[1])
Print(l)
t[0] = 100
Print(t)

o/p - [10]
[1,10,3]
[100, 20, 30]

* t = (1, 2, 3, 4, 5, 6, 7, 1, 2, 1, 7)

#count(element)
Print(t.count(1))

#index(element)
Print(t.index(7))

#len()
Print(len(t))

#min()
Print(min(t))

#max(t)
Print(max(t))

#sum(t), print(sum(t))

o/p - 3
6
11.
1
7
39.

del the tuple :

```
del t                    or    d.clear()
Print(t)                       Print(d)
```

SETS : A collection of elements with no repeats and without insertion order but sorted order.

↳ They are used in situations where it is only important that some things are grouped together, and not what order they were included.

↳ For large groups of data, it is much faster to check whether or not an element is in a set than it is to do the same for a list.

* I/P - S = set( )
  Print(type(S))
  O/P - <class 'set'>

* I/P - S = set([1, 4, 8, 0])
  Print(S)
  O/P - {8, 1, 4, 0}
  Print(type(S))
  <class 'set'>

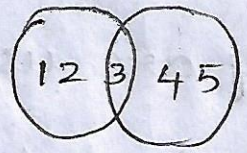
* Defining a set is very similar to defining a dictionary

↳ Duplicates are not allowed in sets. (34)

* I/P - s = {1, 2, 3, 4, 2, 1}
  Print(s)
  O/P - {1, 2, 3, 4}

* I/P - s = {1, 2, 3}
  Print(s[1])
  O/P - 'set' object is not subscriptable.

SET OPERATIONS:



* I/P - set1 = {1, 2, 3}
  set2 = {3, 4, 5}
  Print(set1 | set2)
  (or)
  Print(set1.union(set2))

  Print(set1 & set2)
  (or)
  Print(set1.intersection(set2))
  & - common one.

  Print(set1 - set2)
  (or)
  Print(set1.difference(set2))

  Print(set1 ^ set2)
  (or)
  Print(set1.symmetric_difference(set2))
  ^ - remove common one

O/P - {1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
{3}
{3}
{1, 2}

{1, 2}
{1, 2, 4, 5}
{1, 2, 4, 5}

DICTIONARY: It is a collection of key-value Pairs.

→ It is surrounded by curly braces.

→ Each pair is seperated by a comma and the key and value are seperated by a colon.

* I/p - d = {}
  Print(type(d))
  O/p - <class 'dict'>

* I/p - d = {1: 'abc', 2: 'efg', 'Name': 'xyz'}
  Print(d)
  O/p - {1: 'abc', 2: 'efg', 'Name': 'xyz'}.

Accessing values

* I/p - d = {1: 'abc', 2: 'efg' 'Name': 'xyz'}
  Print(d[1])
  Print(d['name'])
  O/p - abc
  xyz

updating dictionary

* I/P - d={1:'abc', 2:'efg', 'name':'xyz'}

```
Print(d)
d['name'] = 'lmn'
d['age'] = 20
Print(d)
```

O/P - {1:'abc', 2:'efg', 'name':'xyz'}

{1:'abc', 2:'efg', 'name':'lmn', 'age'=20}

deleting elements

* I/P - d={1:'abc', 2:'efg', 'name':~~'xyz'~~ 'lmn', 'age'=20}

```
del d['name']
Print(d)
```

O/P - {1:'abc', 2:'efg', 'age'=20}