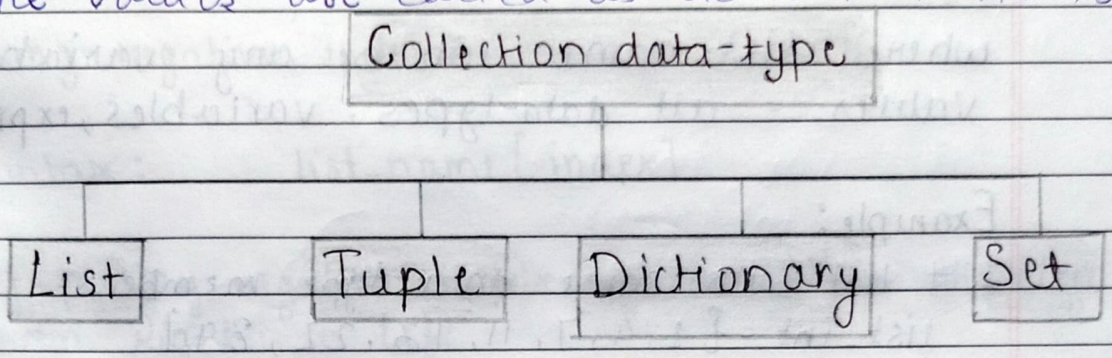
Chap - 4: Collection Data-Type

Collection data-type are the collection of same or different type of data together in a single entity.

collection-type = {val-1, val-2, val-3, . . . , val-n}

The values are called as element or items.



1) List data-type :

List is a collection of same or different type of data enclosed within square bracket [ ] and the elements are separated by commas ,.

list = [val-1, val-2, val-3, . . . , val-n]

e.g. list-1 = [1, 2, 3, 4, 5]
list-2 = ['a', 'e', 'i', 'o', 'u']
list-3 = [1.7, 8-5j, 'hello', 0b1011]

List are mutable, that means we can change or modify the elements of list.
There are various operation that can be performed on list.

1) Creating or Declaring a list :

In python we can create or declare a list by using following syntax

list_name = [val-1, val-2, val-3, . . . , val-n]

e.g.

list_1 = [11, 17, 21, 27, 32]

list_on = ['a', 2.7, 'e', 5-7j, 'hello']

all_data = [[1, 2, 'OK'], (0, 1, 2), 'hello', 7+7j, 0o52]

where list_name can be any variable/Identifie

values = all data-types, variables, expression

Example:

1. # list with same data-type members
   list_int = [1, 4, 7, 11, 13, 22, 89]
2. # list with different data-members
   list = [1, 2.5-4j, 0b1011, "hello", True, None, 7.4]
3. # list with advanced data-members.
   list_avd = [22, "hii", (2, 4, 7, 9), {'hii', 'hello', bool(4)}, [2, 4, bool(None)], 7-4j, 0x12A]

2] Accessing element of list:

As list is a collection elements/values, the element are placed at unique indexes in the list.

"The Position of element's in sequence is called as index."

The index usually strat from 0 and goes upto n, from left to right

From Right to Left index start from -1 and upto -(n+1).

There are Three ways to access the element of list.

a) Using positive index:

Syntax: list_name[index]

by puting the index in sq. bracket we can access that elem.

Example:

```
list = [4, 6, 8, 7, 11, 13]
print(list[0]) ; print(list[4])
print(list[2]) ; print(list[5])
```

b) Using negative index:

Syntax: list_name[index]

by puting negative index we can access the elements from Right to Left.

Example:

```
list = [4.4, "hii", 0x12B, "None", bool(1), "101"]
print(list[-1]) ; print(list[-2])
print(list[-4]) ; print(list[-6])
```

c) Using slice ':' operator:

Syntax: list_name[starting index : ending index]

by using slice operator we can access elements from starting and ending index.

Example:

```
list = [5, 'hello', [1,2,3], 0b101, (1,2,3), 'By']
print(list[0:2])
print(list[0:5])
print(list[-1:-4])
print(list[-3:-5])
```

3) Traversing a List:

Traversing of list is the method of printing all elements one-by-one.
List can be traversed using loop (usually for loop)

Syntax:

```
for variable in list:
    print(variable)
```

Example:

1)
```
# Print the element of list.
list = [2, 4, 6, 8, 10]
for var in list:
    print(var)
print("end")
```

2)
```
# Printing days of week
week = ['Sunday', 'Monday', 'Tuesday', 'Wednesday
    'Thursday', 'Friday', 'Saturday']
for days in week:
    print(days)
print("stop")
```

3)
```
# Print vowels
vowels = ['a', 'e', 'i', 'o', 'u']
for alphabate in vowels:
    print(alphabate)
print("all other alphabate are consonant")
```

There are two more ways to traverse a list.

WEEK 4

a) Using range() function:
Using index of list we can traverse it by range function

Syntax:

```
for variable in rage(list length):
    |
```

Example:

1) # Printing the list element
```
list = ['a', 'b', 'c', 'd', 'e']
for i in range(len(list)):
    print(list[i])
print('stop')
```

b) # Using Enumerate() function:
Using enumerate function we can print both index and item of list.

Syntax:

```
for var in enumerate(list_name):
```

Example:

1) # Printing index and corresponding value.
```
list = [3, 7, 13, 17, 23, 27]
for val in enumerate(list):
    print(val)
print('stop')

list2 = [1, 11, 21, 31, 41, 51]
for i in enumerate(list2):
    print(i)
print('stop')
```

4] Updating elements of list:

As list are mutable data-type we can change the element of list, by assigning new value to particular index we can the element of list.

Syntax:

```
list_name [index] = new value
```

Example:

1)
```
list = [1, 2, 3, 4, 5]
list[0] = 11 ; list[1] = 21
list[2] = 31 ; list[3] = 41
list[3] = 51 ; list[6] = 61
print (list)
```

2)
```
list2 = [2, 4, 9, 6, 25, 36]
list2[-3] = 32 ; list2[-5] = 39
list [1:5] = 22, 23, 55, 27
print [list2]
```

5] Deleting element of list:

Deletion of elements is carried out by using various functions like pop, remove, del.

a) pop: using pop function we can delete an element at a particular index.

The index is passed as argument to pop.

Syntax:

```
list_name.pop(index)
```

Example:
```
short = ['url', 'IBM', 'TCS', 'http', 'GKM', 'OYO']
short.pop(0) ; short.pop(3)
shor.pop(-5) ; short.pop(-4)
```

b) remove():

remove function is used to delete a particular value from the list. The value is passed as argument to remove function.

Syntax:

list_name.remove(value)

Example: symbol = ['$', '#', '!', 3, 4, '@', B, 'π', 'hii']

symbol.remove(3) ; symbol.remove(4)
symbol.remove(B) ; symbol.remove('hii')
⟶ symbol = ['$', '#', '!', '@', 'π']

c) del():

del function is used to delete more than one element at a time, starting and ending index are passed as argument to del function

Syntax:

del listname[start : end index]

Examples:

1) # delete a group of values
list_even = [2, 4, 6, 8, 1, 3, 5, 7, 9, 10]
del list_even[4:9]
del list_even[-6 : -2]

2) # delete offline games
games = ['PUBG', 'Free Fire', 'Temple run', 'Dr. Driving', 'BGMI', 'Candy Crush']
del games[2]
del games[3]
del games[-1]

6] List Operations:
There are two operations that can be perform on the list.

a) Concatenation: '+'
This is used to join two or more list, the combined list will contain all elements together in single list.

Example:
```
1 # join two list
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]
list3 = list1 + list2
→ list3 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

b) Repeatation: '*'
It is used to repeate a list no. of times the list will contain that no. of elements.

Examples:
```
list = [10, 20, 30]
list1 = list * 3
print(list)
→ [10, 20, 30, 10, 20, 30, 10, 20, 30]
```

7] Different Methods and Built in function's:
There are various Built in functions in python which performs different operations on list.

1) Count():

Count() function returns the count i.e. no. of times of a particular element present in list.

list name.count(element)

Examples: list = ["h", 2, 0b101, None, 'OK', 'Fun']

print(list.count("h")), print(list.count(2))

a = list.count() , b = list.count('Fun')

print(a, b, list.count(None))

2) index():

Index function return the index of a particular element. It only return the positive index.

list_name.index(element)

Example:

letter = ['a', 'b', 'c', 'd', 'e', 'f']

print(letter.index('d')) ; print(letter.index('a'))

print(letter.inde('z') ; letter.index('f'))

c = letter.index('b')

print(c)

3) len():

len function return the length of list i.e. no. element present in list.

len(list_name)

Example:

colour = ['red', 'violet', 'indigo', 'blue', 'green']

two_multiple = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

print(len(colour)) ; print(len(two_multiple))

print(len([1, 3, 5, 7, 9, 11, 13]))

4) max():

max. function return the maximum or largest element of the list.

max(list name)

Example:

```
list = [1, 2, 4, 5, 6, 7]
print(max(list))
arr = [2, 4, 6, 99, 99.9]
print(max(arr))
```

5) min():

min. function return the minimum or smallest element of list.

min(list name)

Example:

```
list = [0, 0.0, 0.00, 0.000]
print(min(list))
print(min(arr))
```

6) Sum():

sum function return the sum of all element of list

sum(list name)

Example:

```
# sum of integers list
listn = [1, 5, 6.2, 8, 0.01, 0b101, 0o52, 0xA5]
print(sum(listn))

list_string = ['fun', '#', 'lucky', '$', 'on my own']
print(sum(list_string)) — concatenation.
```

7) Append():

append method is used to add one element at a time in the list at the last of list.

listname.append(element)

Example:

list_num = [5, 8, 12, 'hi', 9.9, 7-5.2j]

listnum.append(72) ; list_num.append(5.01)

list_num.append('101') ; list_num.append('Joy')

→ print(list_num)

∴ → list_num[5, 8, 12, 'hi', 72, 5.01, '101', 'Joy']

8) Extend():

extend method is used to add more than one element at a time to list.

can only take collection data as argument

listname.extend(elements)

Example:

a = [10, 20, 30] ; b = ['a', 'b', 'c']

a.extend(b) ; a.extend(22, 33)

a.extend('2', 2, 2.2, 0b101)

print(a)

9) Sort():

Sort method arrange the list in increasing order. Sorted() method is also used to sort the list.

listname.sort()/sorted(listname)

Example:

number = [2, 8, 7, 9, 3, 1, 4, 5]

number.sort()

letters = ['a', 'e', 'i', 'o', 'b', 'c', 'f', 'd']

letters.sort()

print(number, letters)

10) all():

If all the element of list are true or if the list is empty then this function return true value.

all(list name)

Example:

```
bool=[True, bool(5), bool('hii')]
print(all(bool))
print(all([]))
```

11) any():

This function is used to check whether list is empty or any element of list is true, then it return true value.

any(list name)

Example:

```
list=['hello', 5, True, False, bool(None)]
print(any(list))
Empty=[None]
print(any(Empty))
```

12) insert():

insert() function is used to add new value to a particular index

listname.insert(index, value)

it takes two argument first index and second is corresponding value.

Example:

```
list=[2, 4, 5, 6, 9]
list.insert[2, 3]; list.insert(3, 0.2)
list.insert(0, 5) ; list.insert(4, 0.112)
print(list)
```

13) clear():

It is used to clear or empty the entier list.

listname.clear()

Example:

```
list = [1,2,3,4,5]
list.clear()
print(list)
```

14) List(): it is a built in function

list() function is used to convert any other data-type or sequence of data into list

list(sequence or data)

Example

```
string = 'kabaddi'
print(list(string)
numeric = 9, 0.5, 5-4j, 0b101, 0xA5
print(list(numeric)
tuple = (5, 0.5, 5-0.5j, bool(5))
print(list(tuple)
```

15) Split() function:

split function is used to separate the string or sequence through an enumerate (' ', ':', '#', '-', '.', _) and return the sequence in the form of list.

listname.split('enumerate')

Example:

```
sequence = 5-4j, 2, 0b101, 7.14, 99
string = 'India is my country'
print(sequence.split('-') -> values get separate by dash
print(string.split('*'))
```

List Comprehension:

List comprehension is an elegent way to create and define a new list from existing list.
It is based on methodology of writing the code inside the square bracket and after execution it add the elements, accordingly in the list.

Example:

1) list = [x. for x in range (1, 11)]
print (list)

2) # Printing letters of string value in list
list = [lett for lett in 'String']
print (list)

Syntax:

[expression for item in list]

3) # Printing elem. of even no.
even = [x. for x in rang[20) if x%.2==0]
print [even)

4) # Program to Print and combine two list.

print [(x,y) for x in ['a','b'] for y in['b,d']
if x != y]

note > first loop is outer loop.