Collection Data types

Python data types are classified into two categories

1. Scalar Data types
2. Collection Data Types

Scalar data types are used to allocate memory for one value

1. Int
2. Float
3. Complex
4. Bool
5. NoneType

Collection Data types are used to allocate memory for more than one value.
Collection object is used to group individual objects as a one object.

Collection data types are used,

1. To perform aggregate operations (Sum, Avg, Min, Max, Median,…)
2. In application development these are used to transport data from one place to another place
3. More than one value is referred with one name
4. Collections are dynamic in size

Python collection types are classified into 3 categories

1. Sequences
    a. List
    b. Tuple
    c. String
    d. Range
    e. Bytes
    f. Bytearray
2. Sets
    a. Set
    b. Frozenset
3. Mapping
    a. Dictionary

Every collection type uses one data structure for organizing of data.

Sequences

- Sequences are ordered collection, where insertion is preserved or insertion order does not change form one insertion to another.
- Sequences are index based collection, where reading and writing is using index.
- Sequences allow duplicate values.
- Sequences can be homogenous or heterogeneous.

Mutable Sequences

1. List
2. ByteArray

Immutable Sequences

1. Tuple
2. Range
3. String
4. Bytes

List Data type or Collection

Lists are mutable sequences, typically used to store collections of homogeneous items or heterogeneous items.
"list" class or data type represents list object

How to create list?

1. Using a pair of square brackets to denote the empty list: []
2. Using square brackets, separating items with commas: [a], [a, b, c]
3. Using a list comprehension: [x for x in iterable]

4. Using the type constructor: list() or list(iterable)

Example:

```
>>> a=[]
>>> print(a,type(a))
[] <class 'list'>
>>> b=[10]
>>> print(b,type(b))
[10] <class 'list'>
>>> c=[10,20,30,40,50]
>>> print(c,type(c))
[10, 20, 30, 40, 50] <class 'list'>
>>> d=[1,1.5,1+2j,"python"]
>>> print(d,type(d))
[1, 1.5, (1+2j), 'python'] <class 'list'>
```

Example of converting other iterables/collections into list type

```
>>> e=list()
>>> print(e,type(e))
[] <class 'list'>
>>> f=list(10)
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    f=list(10)
TypeError: 'int' object is not iterable
>>> f=list("NIT")
>>> print(f,type(f))
['N', 'I', 'T'] <class 'list'>
>>> g=list(range(1,6))
>>> print(g,type(g))
[1, 2, 3, 4, 5] <class 'list'>
>>> h=list(range(10,110,10))
>>> print(h,type(h))
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100] <class 'list'>
>>> i=list(h)
>>> print(i,type(i))
```

[10, 20, 30, 40, 50, 60, 70, 80, 90, 100] <class 'list'>

How to read content of list?

Content of list can be read in different ways

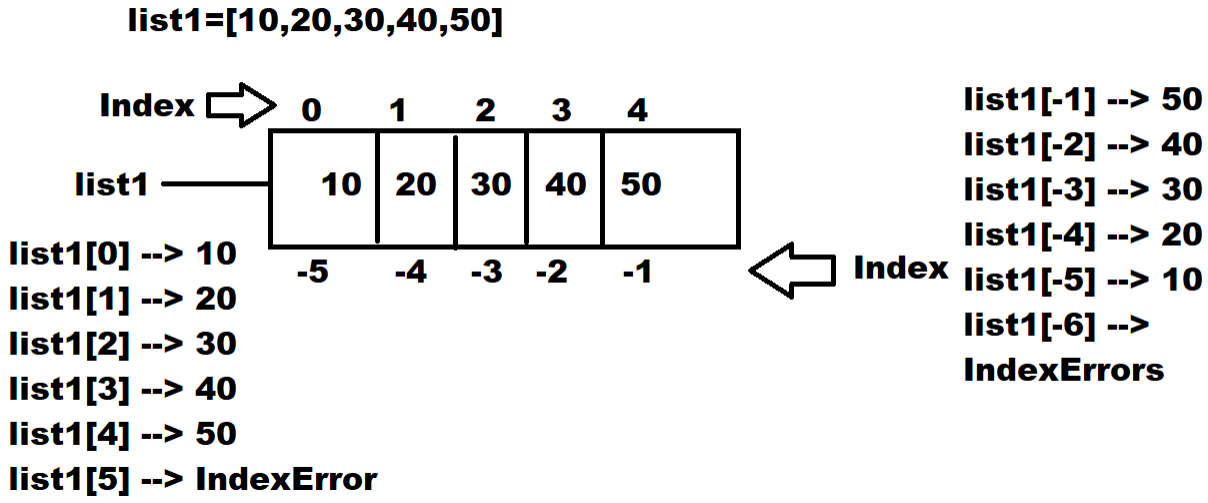
1. Index
2. Slicing
3. For loop
4. Iterator
5. Enumerate

Index

What is index?

Index is an integer value.
Index represents position of value in sequence (OR) each value in sequence is identified with unique integer number called index.
Using index data from sequence can be read sequentially or randomly.
Index is used as a subscript to read or write.
This index can be +ve or –ve

The index start at 0, when reading from L-R (forward Direction)
The index start at -1, when reading from R-L (backward Direction)



Example	**Output**

list1=[10,20,30,40,50]
print(list1[0],list1[1],list1[2],list1[3],list1[4])
print(list1[-1],list1[-2],list1[-3],list1[-4],list1[-5])
 	
list1=[10,20,30,40,50,60,70,80,90,100]
for i in range(10): # 0 1 2 3 4 5 6 7 8 9
  print(list1[i],end=' ')
 	
print()
for i in range(-1,-11,-1):
  print(list1[i],end=' ')
 	
print()
for i in range(0,10,2): # 0 2 4 6 8
  print(list1[i],end=' ')
	10 20 30 40 50
	50 40 30 20 10
	10 20 30 40 50 60 70 80 90 100
	100 90 80 70 60 50 40 30 20 10
	10 30 50 70 90