A data structure is a specialized format for organizing and storing data, so that various operations can be performed on it efficiently and easily.

3.1 LISTS :

Lists are used to store multiple items in a single variable. List is a sequence of values written in a square bracket and separated by commas.

SYNTAX :

<list_name> = [value1 , value2 , . . . ,valueN]

Ex.

Flowers = [“rose” , “lotus” , “mogra”]
Numbers = [2 , 5 , 8 , 9]
Team = [“Tanvi” , 09 , “Vikas” , 6 , “Tanay” , 2 , “Piyush” , 12]

1. Creating a List :

We can create a list by placing a comma seperated sequence of values in square brackets [ ] .

How to create List ?

```
a=['AAA','BBB','CCC']
b=[10,20,30,40]
```

Here a and b are the lists data structures.

2. Accessing Values in List :

Accessing elements from a list in Python programming is a method to get values that are stored in a list at particular location or index. Individual element of the list can be accessed using index of the list.

Ex.

```
rollNo = [1, 2, 3, 4, 5]
name = ['Shilpa', 'Chinmaya', 'Akash', 'Aditya']
print(rollNo[0], name[0])
print(rollNo[1], name[1])
```

3. Deleting Values in List :

The deletion of any element from the list is carried out using various functions like pop, remove, del .

i) The pop function :

Pop function is used to remove particular item/element from the list by using its index . If we know the index of the element to be deleted then just pass that index as an argument to pop function.

REMEMBER

If you do not specify the index, the pop() method removes the last item.

Ex.

```
a=['u','v','w','y']
print("Original list : ",a)
val=a.pop(0)
print("New list : ",a)
```

ii) The remove function :

If we know the value of the element to be deleted then the remove function is used.

REMEMBER

If there are more than one item with the specified value, the remove() method removes the first occurrence

Ex.

```
a=['u','v','w','y']
print("Orinial list : ",a)
val=a.remove('v')
print("New list : ",a)
```

iii) The del function :

We can delete one or more items from a list using the keyword 'del' . The del keyword can also delete the list completely.

Ex.

```
#delete all the list

a=['u','v','w','y']
print("Orinial list : ",a)
del a

#delete the 2nd item in list

a=['u','v','w','y']
print("Orinial list : ",a)
del a[1]
print("New list : ",a)
```

4. Updating the List :

Lists are mutable. That means it is possible to change the values of list. We can update items of the list by simply assigning the value at the particular index position .

Ex.

```
a = ["apple", "banana", "cherry"]
a[1] = "blackcurrant"
print(a)
```

REMEMBER

We can use assignment operator (=) to change an item or range of items

Basic List Operations :-

(1) Traversing a List :

Transverse in a list means accessing all the elements or items of the list . The loop is used in list for traversing purpose. The for loop is used to traverse the list elements.

Syntax :

```
for VARIABLE in LIST :
BODY
```

Ex.

```
a = ["apple", "banana", "cherry"]
for x in a:
print(x)
```


range() function :

Using range() function we can access each element of the list using index of a list.

Syntax :

```
range(start, stop, step)
```

Ex.

```
x = range(3, 20, 2)
for n in x:
print(n)
```

(2) Finding Length of a list :

The len() function is used to find the number of elements present in the list. It is used to return the number of items in a list .

Syntax :

```
len(obj)
```

Ex.

```
a = ["apple", "banana", "cherry"]
x = len(a)
print(x)
```

(3) Membership :

Using the in operator we can check whether particular element belongs to the list or not.

Ex.

```
x = ["apple", "banana"]
print("banana" in x)
```

(4) List Slices :

The : Operator used within the square bracket that it is a list slice and not the index of the list.

Ex.

```
a=[10,20,30,40,50]
a[2:4]=[111,222,333]
print(a)
```

List Methods :-

(1) append :

The append() method adds an element to the end of a list. We can insert a single item in the list data time with the append().

Ex.

```
a = ["apple", "banana", "cherry"]
a.append("orange")
print(a)
```

(2) insert :

To insert a list item at a specified index, use the insert() method.

Ex.

```
a = ["apple", "banana", "cherry"]
a.insert(1, "orange")
print(a)
```

(3) extend :

The extend() method extends a list by appending items . The extend function takes the list as an argument and appends this list at the end of old list.

Ex.

```
a = ["apple", "banana", "cherry"]
b = ["mango", "pineapple", "papaya"]
a.extend(b)
print(a)
```

(4) sort :

The sort method arranges the elements in increasing order.

Ex.

```
a = ["orange", "mango", "kiwi", "pineapple", "banana"]
a.sort()
print(a)
```

Built - in List Functions :-

Function	Purpose
all()	If all the elements of the list are true or if the list is empty then this function returns true
any()	If the list contains any element true or if the list is empty then this function returns true.
len()	This function returns the length of the string.
max()	This function returns maximum element present in the list.
min()	This function returns minimum element present in the list.
sum()	This function returns the sum of all the elements in the list.
sorted()	This function returns a list which is sorted one.

(1) List function :

String is a sequence of characters and list is sequence of values. But list of characters is not the string. We can convert the string to list of characters.

Ex.

```
str="hello"
myList=list(str)
print(myList)
```

(2) Split function :

If the string contains multiple words then we need to use the built in function split to split the words into the list.

Ex.

```
msg="I love Python Programming very much"
myList=msg.split()
print(myList)
```

(3) Join function :

The join function is exactly reverse to the split function. That means the join function takes the list of strings and concatenate to from a string.

Ex.

```
msg = ['I', 'love', 'Python', 'programming']
ch = '#'
result = ch.join(msg)
print(result)
```

3.2 TUPLE :

Tuple is a collection of element which are enclosed within the parenthesis and this elements are separated by comma's .

Syntax:

<tuple_name> = (value1, value2, ... valueN).

Ex.

```
Emp (22, "Tanvi", 'T', 50)
```

REMEMBER

A Tuple in the Python is an immutable data type which means a tuple once created cannot be altered/modified.

1. Accessing Values in Tuple :

Accessing items or elements from the tuple is a method to get values stored in the tuple from a particular location on index. The elements in tuple can be accessed by using the index of the element.

Ex.

```
a = ("apple", "banana", "cherry")
print(a[1])
```

2. Deleting Values in Tuple :

Tuples are unchangeable , so we cannot remove items from it but we can delete the tubel completely .We can delete the tuple by using the del statement .

Ex.

```
a = ("apple", "banana", "cherry")
print(a)
del a
print(a)
```

3. Updating Values in Tuple :

Tuples are immutable , this means values in the tuple cannot be changed . we can only extract values of one Tuple to create another Tuple .

Ex.


```
a = ("A", "B", "C")
b = ("D", "E")
c = a + b
print(c)
```

Basic Tuple Operations :

1. len :

The length function is used to find the number of elements in the tuple .

Ex.

```
a = ["apple", "banana", "cherry"]
x = len(a)
print(x)
```

2. Concatenation :

The operator (+) is used to concatenate two tuples . This is also called as concatenation operation .

Ex.

```
a = ("A", "B", "C")
b = ("D", "E")
c = a + b
print(c)
```


3. Repetition :

For repetition of the elements in the tuple , * operator is used.

Ex.

```
a = ( 1 , 2 , 3 )
print( a * 3 )
```

4. Membership :

Membership operation is used to check whether the element is present in the tuple or not . Membership operation is performed by using in operator .

Ex.

```
a = ( 1 , 2 , 3 )
print( 4 in a )
```


5. Iteration :

Iteration means accessing the individual elements in the Tuple. Elements in the tuple can be accessed by using the for loop .

Ex.

```
a = (1, 2, 3)
for i in a:
print(i)
```

Built - in List Functions :-

Function	Purpose
cmp(a,b)	This function compares tuple a with tuple b .
len(a)	This function returns the length of the tuple ..
max(a)	This function returns the maximum value from the tuple a .
min()	This function returns minimum value from the tuple a .
tuple(sequence)	This function converts the list into tuple .

3.3 SETS :

Sets data structure in Python programming is implemented to support mathematical set operations . Sets are based on specific rules they define .

The specific rules that are followed by sets are :
i) items in a set are unique
ii) items in a set are not ordered

1. Creating the Set :

There are two ways for creation of sets in Python :-

i) Set is Defined by the values separated by, inside braces { }

Ex.

```
Emp = { “Tanvi” , 9 , ‘T’ }
```

ii) set can be created by calling a type constructor called set()

Ex.

```
a = set( 'abc' )
```

2. Accessing the Set :

We cannot access the values in the set using index as the set is unorder and has no index .

Ex.

```
a = { 1 , 2 , 3 , 4 }
print(a)
```

3. Deleting the Set :

For removing the item from the set either remove() method or discard() method is used .

Ex.

```
# using remove method for deleting item in set
a = { 1 , 2 , 3 , 4 }
a.remove(2)
print(a)

# using discard method for deleting item in set
a = { 1 , 2 , 3 , 4 }
a.discard(3)
print(a)
```

4. Updating the Set :

Once the set is created we cannot change the values in set but we can add the element to the set we can update the set by using the add() method or update() method .

Ex.

```
# using update method for updating the set
a = { 1 , 2 , 3 , 4 }
a.update([ 5 , 7 ])
print(a)

# using add method for updating the set
b = { 1 , 2 , 3 , 4 }
b.add(8)
print(b)
```

Basic Set Operations :

1. Union :

Union operation perform on two sets return all the elements from both the sets .

Ex.

```
x = { 'a' , 'b' , 'c' }
y = { 'd' , 'e' }
z = x.union(y)
print(z)
```

2. Intersection :

Intersection operation perform on two sets Returns all the elements which are common or in both the sets .

Ex.

```
x = { 'a' , 'b' , 'c' }
y = { 'a' , 'c' , 'd' , 'e' }
z = x & y
print(z)
```

3. Difference :

Difference operation on two sets set 1 and set to Returns all the elements which are present on Set 1 but not in Set 2 . it is performed by using - operator .

Ex.

```
x = { 'a' , 'b' , 'c' }
y = { 'a' , 'c' , 'd' , 'e' }
z = x - y
print(z)
```

4. Symmetric Difference :

The symmetric represent set of all elements which belong to either to A or B but not both . Difference operation is performed by using ^ operator

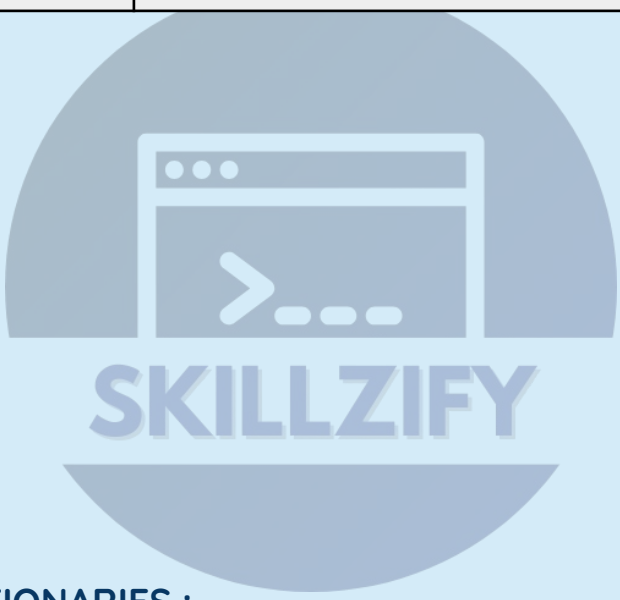
Ex.

```
x = { 'a' , 'b' , 'c' }
y = { 'a' , 'c' , 'd' , 'e' }
z = x ^ y
print(z)
```


Built - in List Functions :-

Function	Purpose
all()	This functions return true if all yments of the set are true this function also returns a true if the value in the set is empty.
any()	This function return true if any element of the set is true.
enumerate()	Function return and enumerate object . it contains the index and the value of all the items of set as a pair .

len()	Is function returns the length of the set .
max(a)	This function returns the maximum value from the set .
min()	This function returns minimum value from the tuple set .
sorted()	This function returns a new sorted list from the elements .
sum()	This function returns the sum of all elements in the set .



3.4 DICTIONARIES :

In Python , dictionary is an unordered collection of elements. This items are in the form of key - value pair . The dictionary contains the collection of indices called keys and collection of values. Each key is associated with single value this association of keys with the values is called as key - value pair .

1. Creating the Dictionary :

Items of the dictionary are written within the {} brackets that are separated by commas . the key - value pair is represented by using : operator .

Ex.

```
a = {
 "name": "Tanvi",
 "age": 17
}
```

2. Accessing values in the Dictionary :

We can access the elements in the dictionary using the keys .

Ex.

```
a = {
 "name": "Tanvi",
 "age": 17
}
print(a["name"])
```

3. Deleting values in the Dictionary :

We can delete the elements in the dictionary by using the keyword del.

Ex.

```
a = {
 "name": "Tanvi",
 "age": 17
}
print(a["name"])
del a["name"]
```

```
print(a)
```

4. Updating values in the Dictionary :

We can update the values in the dictionary by directly assigning them to the corresponding Key position.

Ex.

```
a = {
  "name": "Tanvi",
  "age": 17
}
print(a["name"])
a["name"] = "abc"
print(a)
```


Basic Dictionary Operations :

1. Adding items to the dictionary :

We can add the items in the dictionary .

Ex.

```
a = {
  1: "name",
  2: "age"
}
a[3] = "rollno"
print(a)
```

2. Adding items to the dictionary :

The len() function is used to find the number of pairs in the dictionary.

Ex.

```
a = {
  1: "name",
  2: "age"
}
print(len(a))
```

3. Iterating through dictionary :

For iterating through the dictionary , for loop is used. The corresponding keys and values present in the dictionary can be displayed by using the for loop .

Ex.

```
a = {
   1: "name",
   2: "age"
}
for i in a:
   print(i, a[i])
```