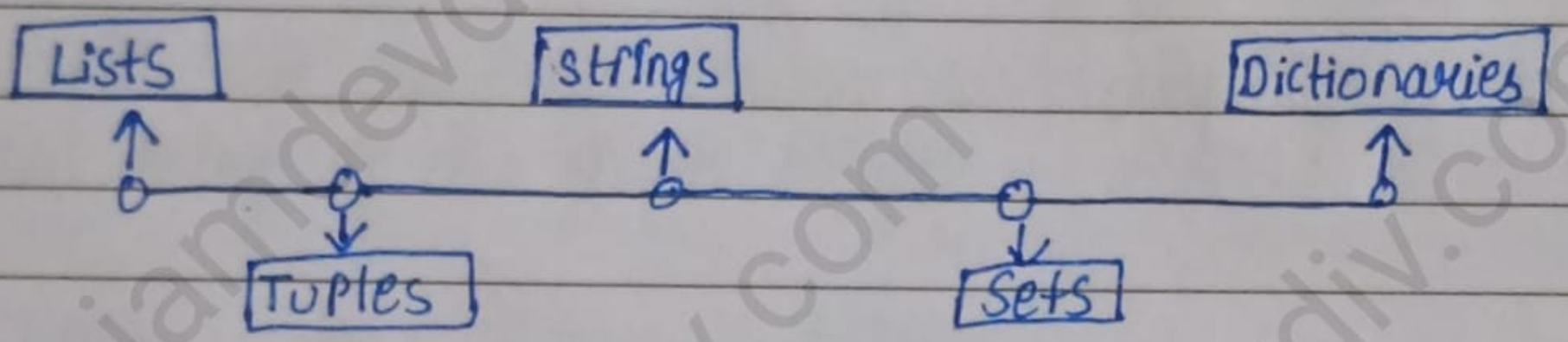
Evergreen
Page No
Date : / /20

Chapter-7

Lists in Python

- Sequence is an object that contains multiple items of data.
- The items are stored in a sequence one after the other such as a sequence of events, movements, items.

⇒ Sequences



List

- A list is a collection of values or an ordered sequence of values/items.
- The elements in a list can be of any type such as string, integers, float, objects, or even a list.
- Elements of a list are enclosed in square brackets [ ]. separated by commas.
- Lists are mutable and heterogeneous (of multiple types) in nature.

⇒ Declaring / Creating / Initializing List

- List can be created by putting comma-separated values/ items within square brackets [ ].
- Square brackets indicate start and end of the list, consisting of elements separated by a comma.

Syntax:-

<List_name> = [ ]

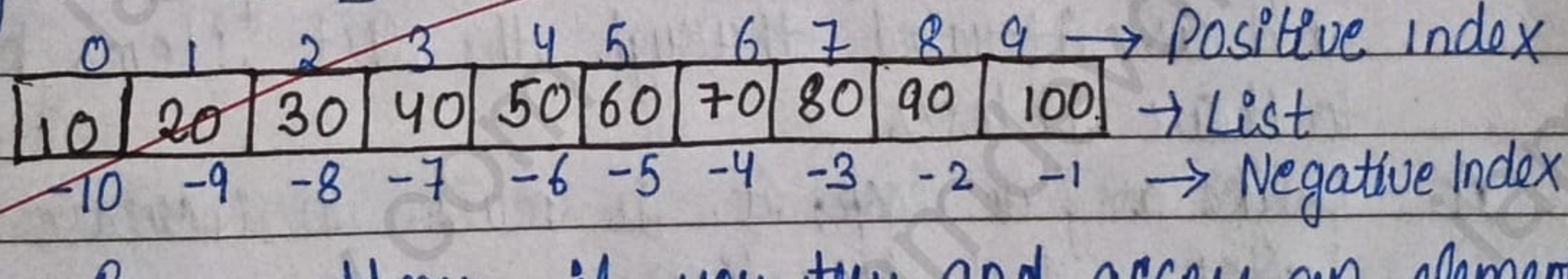
- List types and Examples:-

1. Empty lists
2. Long lists
3. Nested lists

☆ Lists() function can convert certain types of objects into lists.

⇒ Accessing List Elements (Indexing)

- A list consist of any collection of items which are stored according to its index.
- List indices start with 0 (zero) and go ~~of~~ to length -1.
- List index can be a positive or negative integer. i.e. you can access the elements from a list either moving from left to right [positive index] or from to left [negative index].



☆ An Index Error appears if you try and access an element that does not exist in the list. It will display an error message:

Index Error: list index out of range

⇒ Traversing a list

- Traversing a list means accessing each elements of a list.
- Traversing can be done in the following three ways:

Evergreen
Page No.
Date : / /20

1. Using 'in' operator inside for loop

- The elements of a list can be accessed individually using for loop.
- 'in' operator using for loop iterates each element of a list in sequence.

2. Using range() function

- This function can be used for accessing individual elements of a list, using indexes of elements only, along with len() method which gives length of the list and specifies how many items for loop will iterate

3. Using while loop

- The list elements/items in Python can also be iterated or accessed using a while loop.
- Firstly the len() function is used to determine the length of the list, then the while loop starts from index 0 and is iterated through the list items by referring to their indexes.

☆ Unlike a for loop, it is must to increase the index by 1 after each iteration.

⇒ Aliasing

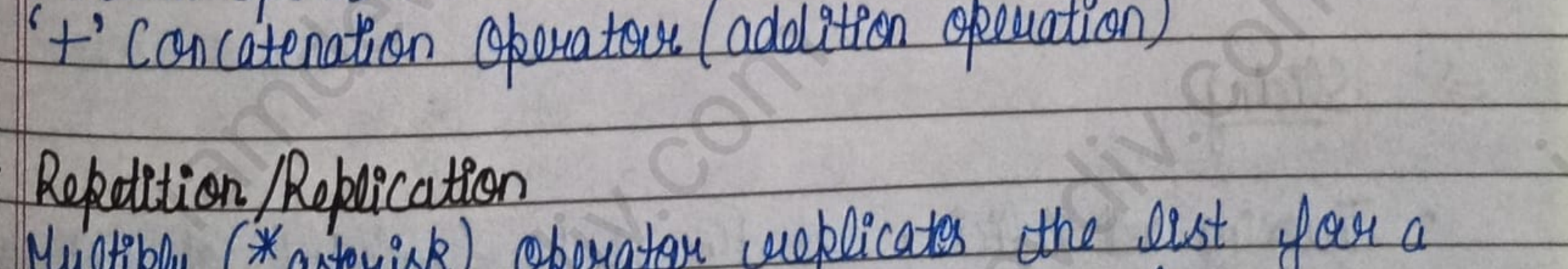
- In python, variables refer to object. The lists we create in python are also objects we are referring to.
- If we assign the elements of one list to another list, both shall refer to the same object.

Evergreen
Page No.
Date : / /20

⇒ Comparing Lists

- Python allows us to compare two lists.
- Each element is individually compared in lexicographical (alphabetical or dictionary) order using relational operators (>, < ==, !=)
- Python generates the final result of non-~~eq~~ equality comparison as either True/false from corresponding Elements comparision.
- If the corresponding elements are equal, it goes on to the next element and so on, until it finds the element that differs.

Operations on lists



⇒ Concatenation

- Concatenation or joining is a process in which multiple sequences/lists can be combined together with the help of certain operators
- '+' Concatenation Operator (addition operation)

⇒ Repetition/Replication

- Multiply (*asterisk) operator replicates the list for a specified no. of times & creates a new list

* We cannot multiply one list with another.

Evergreen
Page No.
Date : / /20

⇒ Membership Testing

- Membership Testing is an operation carried out to check whether a particular element / item is a member of that sequence or not.
- The 'in' operator checks whether a given element is contained in a list. It returns True if the element appears in the list, otherwise returns false.
- The 'not in' operator returns True if the element does not appear in the list, otherwise returns false.

'in' operator used with for loop
the membership operator in can be used to display the elements of a list through for loop.

⇒ Indexing

- Index is nothing but an integer value for each item present in the sequence.
- Index is a no. specifying the position of an element in a list. It enables elements in the list.
- Index of first element in the list is 0, second element is 1, and n^{th} element is n-1.

⇒ Slicing

- Indexing and slicing are two index-related operations in lists. Slicing is done through index.
- Slicing is an operation in which we can slice a particular range from a sequence and retrieve a subset of values.
- List slices are sub-parts extracted from a list.

Evergreen
Page No.
Date : / /20

- List slices can be created through the use of indexes.
Syntax:

List[start : stop : step]

- start → starting point
- stop → stopping point, which is not included
- step → the step size - also known as stride - and is optional with depends value!

- In case the resulting index lies outside the list, python raises an Index error: list index out of range
- Slice range (start and stop argument) is treated as boundary and result will simply contain all the elements which lie between the boundary.

★ In list slicing, if ~~the~~ start index value is missing then it will start from 0 by default and if stop value is missing then it will print till end. Stop index, if given, will stop at index stop - 1.

Nested lists

- When a list appears as an element of another list, it is called a nested list i.e., a list can have one or more lists as its element.
- To access the element of nested list of list1, we have to specify two indices list1[i][j]
  i → take us to the desired nested lists
  j → tak us to the desired element in that nested list.

★ The copy() function returns a new list stored in a different memory location. It doesn't modify the original list or the modifications made in the new list will not be reflected in the base list.

Evergreen
Page No.
Date : / /20

Built-In functions (Manipulating Lists)

- Python offers several built-in 'inplace' functions that can alter the elements of the lists rather than creating a new lists.

Operations →
- Append → List.append(elem)
- Insert → List.insert(index, elem)
- Extent → List.extend(list2)
- Index → List.index(elem)
- Remove → List.remove(elem)
- Sort → List.sort()
- Reverse → List.reverse()

⇒ append()

- The append() methods adds a single item to the end of the list.
- It doesn't create a new list; rather it modifies the original list.

Syntax:

```
List.append(item)
```

- The item can be numbers, strings, another list, dictionary, etc.

⇒ extend()

- extend() methods adds one list at the end of another list
- All the items of a list are added at the end of an already created list.

Syntax:

```
List1.extend(list2)
       |        └→ The list which will be added to list1
       └→ list1 is the primary list which will be extended.
```

Evergreen
Page No.
Date : / /20

⇒ Insert()

- Insert() function can be used to insert an element/object at a specifying index.
- This function takes two arguments: the index where an item/object is to inserted and the item/element itself.

Syntax:

List_name.insert(index_number, value)

- List_name is the name of the list.
- index_number is the index where the new value is to be inserted.
- value is the new value to be inserted in the list.

⇒ reverse()

- The reverse() function in python reverses the order of the elements in a list.
- It replaces a new values "in place" of an item that already exists in the list i.e. it does not create a new list.

⇒ index()

- The index() function in python returns the index of first matched item from the list.
- It returns the first occurences of an item for which the index position is to be searched in the list.
- If the element is not present, Value error is generated.

Syntax:

List.index(<item>)

⇒ Updating list

- Lists are mutable, you can assign new value to existing value. We can use assignment operator (=) to change an item or a range of items.

Syntax:

List [index] = < new value >

⇒ Len()

- The len() function returns the length of the list i.e. the number of elements in a list.

⇒ Sort()

- This function sorts the items of the list by default in ascending / increasing order.
- This modification is done "in place", i.e., it does not create a new list.

Syntax:

List.sort()

- It sorts the string in a lexicographic manner on the basis of their ASCII values
- If we want to sort the list in decreasing order, we need to write: List.sort(reverse = True)

* Sorted() - This function also sorts the items of the list like sort() function.

Syntax: Sorted (List, reverse = True/False)

Note: Sorted() function works in the same way as sort() function, the only difference being that it returns a new list rather than modifying the list() passed as a parameter.

Evergreen
Page No.
Date : / /20

⇒ clear()

- The clear() method remove all items from the list. This method doesn't take any parameters.
- The clear() method only empties the given list. It doesn't return any value.

⇒ count()

- The count() method counts how many times an element has occured in a list and returns it.

Syntax:

```
List.count(element)
```

Deletion operation

- Python provides operator for deleting/removing an item from a list.
- There are many methods for deletion:

1) If the index is known, you can use pop() function or del statement.
2) If the element is known, not the index, remove() can be used
3) To remove more than one element, del with list slice can be used.

⇒ pop()

- It removes the element from the specified index and also returns the element which was removed.

Syntax:-

```
List.pop(index)
```

Evergreen
Page No.
Date : / /20

☆ If no index value is provided in pop() function, then the last element from the list is deleted.

⇒ del statement
del statement removes the specified element from the list but does not return the deleted value

Note: If an out of range index is provided with del and pop(), the code will result in runtime error.

⇒ remove()
The remove() function is used when we know the element to be deleted, not the index of the element.

⇒ Searching list
List can be easily searched for values using the index() method expects a value that is to be searched. The output is the index at which the value is kept.
Syntax:
List_name.index(element)

⇒ max()
Returns the element with the maximum value from the list.
Note:- To use the max() function, all values in the list must be of the same type.

⇒ min()
Returns the element with the minimum value from the list.
Note:- To use the min() function, all values in the list must be of the same type.

DIKSHA
27/01/24