LAB # 09

Searching & sorting

# OBJECTIVE

Searching and sorting data in the form of list.

# THEORY

**Searching:**

Searching is a very basic necessity when you store data in different data structures/list. The simplest appraoch is to go across every element in the data structure/list and match it with the value you are searching for. This is known as ***Linear search***.

In Python, the easiest way to search for an object is to use [***Membership Operators***](https://docs.python.org/3/reference/expressions.html#membership-test-operations) , to check whether a value/variable exists in the sequence like string, list, tuples, sets, dictionary or not.

**Syntax:**

* **in** - Returns True if the given element is a part of the sequence.
* **not in** - Returns True if the given element is not a part of the sequence.

**Example:**

|  |
| --- |
| # declare a list and a string  str1 = "Hello world"  list1 = [10, 20, 30, 40, 50]  # Check 'w' (capital exists in the str1 or not  if 'w' in str1:  print ("Yes! w found in ", str1)  else:  print("No! w does not found in " , str1)  # check 30 exists in the list1 or not  if 30 in list1:  print ("Yes! 30 found in ", list1)  else:  print ("No! 30 does not found in ", list1) |

**Output:**

|  |
| --- |
| >>> %Run task1.py  Yes! w found in Hello world  Yes! 30 found in [10, 20, 30, 40, 50] |

**Sorting:**

Sorting, like searching, is a common task in computer programming.

A Sorting is used to rearrange a given list/array elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of element in the respective data structure.

Many different algorithms have been developed for sorting like selection sort, insertion sort , bubble sort etc.

**Bubble sort:**

Bubble sort is one of the simplest sorting algorithms. The two adjacent elements of a list are checked and swapped if they are in wrong order and this process is repeated until we get a sorted list. The steps of performing a bubble sort are:

* Compare the first and the second element of the list and swap them if they are in wrong order.
* Compare the second and the third element of the list and swap them if they are in wrong order.
* Proceed till the last element of the list in a similar fashion.
* Repeat all of the above steps until the list is sorted.

**Example:**

|  |
| --- |
| a = [16, 19, 11, 15, 10, 12, 14]  #repeating loop len(a)(number of elements) number of times  for j in range(len(a)):  #initially swapped is false  swapped = False  i = 0  while i<len(a)-1:  #comparing the adjacent elements  if a[i]>a[i+1]:  #swapping  a[i],a[i+1] = a[i+1],a[i]  #Changing the value of swapped  swapped = True  i = i+1  #if swapped is false then the list is sorted  #we can stop the loop  if swapped == False:  break  print (a) |

**Output:**

|  |
| --- |
| >>> %Run task2.py  [10, 11, 12, 14, 15, 16, 19]  >>> |

**EXERCISE**

1. **Point out the errors, if any, and paste the output also in the following Python programs.**
2. Code

|  |
| --- |
| 'apple' is in ['orange', 'apple', 'grape'] |

Output

|  |
| --- |
|  |

2. Code

|  |
| --- |
| def countX(lst, x):  return lst.count(x) |

Output:

|  |
| --- |
|  |

**What will be the output of the following programs:**

1. Code

|  |
| --- |
| strs = ['aa', 'BB', 'zz', 'CC']  print (sorted(strs))  print (sorted(strs, reverse=True)) |

Output

|  |
| --- |
|  |

1. Code

|  |
| --- |
| test\_list = [1, 4, 5, 8, 10]  print ("Original list : " , test\_list)    # check sorted list  if(test\_list == sorted(test\_list)):  print ("Yes, List is sorted.")  else :  print ("No, List is not sorted.") |

Output

|  |
| --- |
|  |

**C. Write Python programs for the following**:

1. Write a program that take function which implements linear search. It should take a list and an element as a parameter, and return the position of the element in the list. The algorithm consists of iterating over a list and returning the index of the first occurrence of an item once it is found. If the element is not in the list, the function should return ‘not found’ message.

2. Write a program that create function that takes two lists and returns True if they have at least one common member. Call the function with atleast two data set for searching.

3. Write a program that create function that merges two sorted lists and call two list with random numbers.