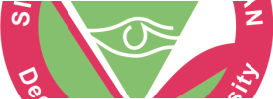
SIKSHA ‘O’ ANUSANDHAN



## DEEMED TO BE UNIVERSITY

**Admission Batch: 2019-23** **Session: 2022**

# Laboratory Record Programming in Python (CSE 3142)

## Submitted by

Name: **SASWAT MOHANTY**

Registration No.: **1941012407**

Branch: **CSE**

Semester: **5th** Section: **D**

# Department of Computer Science & Engineering

**Faculty of Engineering & Technology (ITER)**

**Jagamohan Nagar, Jagamara, Bhubaneswar, Odisha - 751030**

# INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Name of Program** | **Page No** | **Remarks** |
| 1. | MINOR ASSIGNMENT- 12 | 3-15 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**MINOR ASSIGNMENT-12: LIST MANIPULATION**

**Q1. How many passes are required for the elements to be sorted in Insertion sort?**

**Ans:-** Insertion sort requires n – 1 pass to sort an array of n elements. In each pass we insert the current element at the appropriate place so that the elements in the current range is in order.

**Q2. Mention some methods you can use to choose the pivot element in Quick Sort.**

**Ans:-**

1. Always pick first element as pivot.
2. Always pick last element as pivot (implemented below)
3. Pick a random element as pivot.
4. Pick median as pivot.

**Q3. Mention the main idea that lies behind selection sort.**

**Ans:-** The Selection sort algorithm is based on the idea of finding the minimum or maximum element in an unsorted array and then putting it in its correct position in a sorted array.

Q4. Following are some examples of a list. Find out which one of them will work if we want to implement Binary Search. Also explain the reason:

a. [2,3,4,5,9,10] Search element: -9

b. [1,8,-10,8,1,2] Search element: -10

c. [’a’,’k’,4,5,0] Seacrch element: 4

### Program:-

def binary\_search(array, low, high, x): if high >= low:

mid = (high + low) // 2 if array[mid] == x:

return mid

elif array[mid] > x:

return binary\_search(array, low, mid - 1, x) else:

return binary\_search(array, mid + 1, high, x) else:

return -1 array=eval(input('Enter: '))

x= eval(input('Enter the search element: ')) result = binary\_search(array, 0, len(array)-1, x) if result != -1:

print("The index of the Element is", str(result)) else:

print("This element is not present in your Array.")

**Output :-**

**(a)** Enter: [2,3,4,5,9,10]

Enter the search element: -9

This element is not present in your Array.

Because -9 is not present in the list.

**(b)** Enter: [1,8,-10,8,1,2]

Enter the search element: -10 The index of the Element is 2

**(c)** Enter: [’a’,’k’,4,5,0]

SyntaxError: invalid character in identifier Because strings are present in the list.

**Q5. Develop a program to sort the employee data on the basis of pay of the employees using**

1. **Selection sort**
2. **Bubble sort algorithm**
3. **Insertion sort.**

**Consider a list L containing objects of class Employee having empNum, name, and salary**

### Program:-

class Employees:

def

init

(self, empNum, name, salay):

self.empNum = empNum self.name = name self.salary = salay

def convert(self): lst = []

lst.extend([self.empNum, self.name, self.salary]) return lst

# Selection Sort

def selectionSort(list1):

for i in range(0, len(list1)-1): minindex = i

for j in range(i+1, len(list1)):

if list1[j][2] < list1[minindex][2]: minindex = j

if minindex != i:

list1[i], list1[minindex] = list1[minindex], list1[i]

lst = []

e1 = Employees(1, "John", 80000) e2 = Employees(2, "Mike", 50000) e3 = Employees(3, "Derek", 30000) e4 = Employees(4, "Raj", 25000)

lst.extend([e1.convert(), e2.convert(), e3.convert(), e4.convert()])

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Selection Sort: ') selectionSort(lst)

print(lst)

# Bubble Sort

def bubbleSort(lst): n = len(lst)

for i in range(n):

for j in range(0, n-i-1):

if lst[j][2] > lst[j+1][2]:

lst[j], lst[j+1] = lst[j+1], lst[j]

lst = []

e1 = Employees(1, "John", 80000) e2 = Employees(2, "Mike", 50000) e3 = Employees(3, "Derek", 30000) e4 = Employees(4, "Raj", 25000)

lst.extend([e1.convert(), e2.convert(), e3.convert(), e4.convert()])

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Bubble Sort: ') bubbleSort(lst)

print(lst)

# Insertion Sort

def insertionSort(lst):

for i in range(1, len(lst)): temp = lst[i][2]

j = i-1

while j >= 0 and temp < lst[j][2]: lst[j + 1][2] = lst[j][2]

j -= 1

lst[j + 1][2] = temp lst = []

e1 = Employees(1, "John", 80000) e2 = Employees(2, "Mike", 50000) e3 = Employees(3, "Derek", 30000) e4 = Employees(4, "Raj", 25000)

lst.extend([e1.convert(), e2.convert(), e3.convert(), e4.convert()])

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Insertion Sort: ') insertionSort(lst)

print(lst) print()

Output :-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [[1, 'John', 80000], [2, 'Mike', 50000], [3, 'Derek', 30000], [4, 'Raj', 25000]]

Sorted List Using Selection Sort:

[[4, 'Raj', 25000], [3, 'Derek', 30000], [2, 'Mike', 50000], [1, 'John', 80000]]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [[1, 'John', 80000], [2, 'Mike', 50000], [3, 'Derek', 30000], [4, 'Raj', 25000]]

Sorted List Using Bubble Sort:

[[4, 'Raj', 25000], [3, 'Derek', 30000], [2, 'Mike', 50000], [1, 'John', 80000]]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [[1, 'John', 80000], [2, 'Mike', 50000], [3, 'Derek', 30000], [4, 'Raj', 25000]]

Sorted List Using Insertion Sort:

[[1, 'John', 25000], [2, 'Mike', 30000], [3, 'Derek', 50000], [4, 'Raj', 80000]]

**Q6. Write the recursive version of linear search and binary search algorithms, discussed in the text.**

### Program:-

def linearSearch(lst, l, r, key): if r < l:

return -1

if lst[l] == key: return l

if lst[r] == key: return r

return linearSearch(lst, l+1, r-1, key) lst = [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

n = len(lst) key = 3

index = linearSearch(lst, 0, n-1, key) if index != -1:

print("Element", key, "is present at index %d" % (index)) else:

print("Element %d is not present" % (key))

def binarySearch(lst, low, high, key): if high >= low:

mid = (high + low) // 2 if lst[mid] == key:

return mid

elif lst[mid] > key:

return binarySearch(lst, low, mid - 1, key) else:

return binarySearch(lst, mid + 1, high, key) else:

return -1

lst = [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

key = 3 lst.sort()

index = binarySearch(lst, 0, len(lst)-1, key) if index != -1:

print("Element", key, "is present at index %d" % (index)) else:

print("Element %d is not present" % (key))

Output :-

Element 3 is present at index 1 Element 3 is present at index 5

**Q7. Consider the list : [95,79,19,43,52,3]. Write the passes of bubble sort, sorting the list in ascending**

**order till the third iteration.**

**Ans:-** [79,19, 43, 52, 3, 95]- Pass 1

[19, 43,52,3,79, 95]- Pass 2

[19,43,3, 52, 79, 95]- Pass 3

**Q8. Rewrite selection sort, bubble sort and insertion sort functions using recursion.**

**Program:-**

def minIndex(a, i, j):

if i == j:

return i

k = minIndex(a, i + 1, j) return (i if a[i] < a[k] else k)

def selectionSort(lst, n, index=0): if index == n:

return -1

k = minIndex(lst, index, n-1) if k != index:

lst[k], lst[index] = lst[index], lst[k] selectionSort(lst, n, index + 1)

lst = [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

n = len(lst)

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Selection Sort: ') selectionSort(lst, n)

print(lst)

def bubbleSort(lst, n): if n == 1:

return

for i in range(n - 1): if lst[i] > lst[i + 1]:

lst[i], lst[i + 1] = lst[i + 1], lst[i]

bubbleSort(lst, n - 1)

lst = [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

n = len(lst)

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Bubble Sort: ') bubbleSort(lst, n)

print(lst)

def insertionSort(arr, n): if n <= 1:

return insertionSort(arr, n-1) last = arr[n-1]

j = n-2

while (j >= 0 and arr[j] > last): arr[j+1] = arr[j]

j = j-1 arr[j+1] = last

lst = [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

n = len(lst)

print('\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(lst)

print('Sorted List Using Insertion Sort: ')

insertionSort(lst, n) print(lst)

print()

## Output :-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

Sorted List Using Selection Sort:

[0, 1, 1, 2, 3, 3, 3, 4, 5, 5, 78]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

Sorted List Using Bubble Sort:

[0, 1, 1, 2, 3, 3, 3, 4, 5, 5, 78]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Unsorted List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* [1, 3, 4, 5, 3, 2, 1, 0, 78, 3, 5]

Sorted List Using Insertion Sort:

[0, 1, 1, 2, 3, 3, 3, 4, 5, 5, 78]

**Q9. For the list [10, 15, 22, 24, 45, 55], show the values of the indexes low, high, and mid at each step of**

**the binary search method discussed in the text when we are searching for the key:**

1. **15**
2. **25**
3. **55**
4. **40**
5. **22**

### Program:-

def binary\_search(array, low, high, x): if high >= low:

mid = (high + low) // 2 if array[mid] == x:

return mid

elif array[mid] > x:

return binary\_search(array, low, mid - 1, x) else:

return binary\_search(array, mid + 1, high, x) else:

return -1 array=eval(input('Enter: '))

x= eval(input('Enter the search element: ')) result = binary\_search(array, 0, len(array)-1, x) if result != -1:

print("The index of the Element is", str(result)) else:

print("This element is not present in your Array.")

## Output :-

**(a)** Enter: [10, 15, 22, 24, 45, 55]

Enter the search element: 15 The index of the Element is 1 **(b)** Enter: [10, 15, 22, 24, 45, 55]

Enter the search element: 25

This element is not present in your Array.

**(c)** Enter: [10, 15, 22, 24, 45, 55]

Enter the search element: 55 The index of the Element is 5

**(d)** Enter: [10, 15, 22, 24, 45, 55]

Enter the search element: 40

This element is not present in your Array.

**(e)** Enter: [10, 15, 22, 24, 45, 55]

Enter the search element: 22 The index of the Element is 2

**Q10. Write the program of binary search for a sorted list in descending order.**

### Program:-

def binarySearch(lst, n, key): start = 0

end = n

while (start <= end):

mid = start + (end - start) // 2 if (key == lst[mid]):

return mid

elif (key < lst[mid]): start = mid + 1

else:

end = mid - 1 return -1

lst = [5, 4, 3, 2, 1]

n = len(lst) key = 3

print(binarySearch(lst, n, key))

## Output :-

2

**Q11. Write a function leftCirculate that takes a list as an input and left circulates the values in the list so that in the final list, each value is left shifted by one position and leftmost value in the original list now appears as the rightmost value. For example, on execution of the function on the list [1, 2, 3, 4, 5] it would be transformed to the list [2, 3, 4, 5, 1]. Modify the function to include a numeric argument to specify the number of positions by which left rotation is to be carried out.**

### Program:-

def leftCirculate(lst, n): for i in range(0, n):

first = lst[0]

for j in range(0, len(lst)-1): lst[j] = lst[j+1]

lst[len(lst)-1] = first

lst = [1, 2, 3, 4, 5]

n = 3 leftCirculate(lst, n) print(lst)

## Output :-

[4, 5, 1, 2, 3]

**Q12. Write a program that defines a class Card which can be used to instantiate cards with a particular rank and suit. Create another class DeckOfCards for maintaining a sorted list of cards using a method sortedInsert that takes an object of class Card as an input parameter and inserts it at the suitable position in the sorted list.**

### Program:-

class Card:

def

init

(self, rank, suit):

self.rank = rank self.suit = suit

def getRank(self): return self.rank

def getSuit(self): return self.suit

def value(self):

if self.rank <= 10: return self.rank

else:

return 10

def names(self):

ranks = ["Ace", "Two", "Three", "Four", "Five", "Six",

"Seven", "Eight", "Nine", "Ten", "Jack", "Queen", "King"] suits = ["Diamond", "Clubs", "Hearts", "Spades"]

name = ranks[self.rank-1] if self.suit == "d":

name += suits[0] elif self.suit == "c": name += suits[1] elif self.suit == "h": name += suits[2]

else:

name += suits[3] return name

def

str

(self):

return str.format("({},{})", self.names(), self.value())

class DeckofCards(Card):

def

init

(self, rank, suit):

super().

init

(rank, suit)

def sortedInsert(lst, compare): for index in range(1, len(lst)):

value = lst[index] position = index

while position > 0 and compare(lst[position - 1], value):

lst[position] = lst[position - 1] position = position - 1

lst[position] = value

A = DeckofCards(1, "d") B = DeckofCards(13, "c") C = DeckofCards(12, "s") D = DeckofCards(9, "h") lst = [A, B, C, D]

for i in lst: print(i)

print()

DeckofCards.sortedInsert(lst, lambda a, b: a.rank > b.rank) for i in lst:

print(i)

## Output :-

(AceDiamond,1) (KingClubs,10) (QueenSpades,10) (NineHearts,9)

(AceDiamond,1) (NineHearts,9) (KingClubs,10) (QueenSpades,10)