Worksheet 3.1

Student Name: Sahil Kaundal UID: 21BCS8197

Branch: CSE (Lateral Entry) Section/Group:20BCS-807\_B Semester: 4th Date of Performance: 04/04/2022

Subject Name: Programming in Python Lab Subject Code: 20CSP-259

1. Aim/Overview of the practical:
2. Python program to implement linear search.
3. Python program to implement bubble sort.
4. Python program to implement binary search without recursion.
5. Python program to implement selection sort.

1. Task to be done/ Which logistics used:

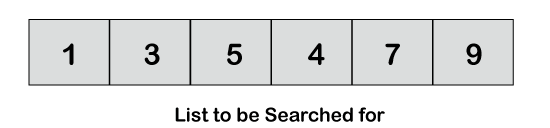


* Python program to implement linear search.

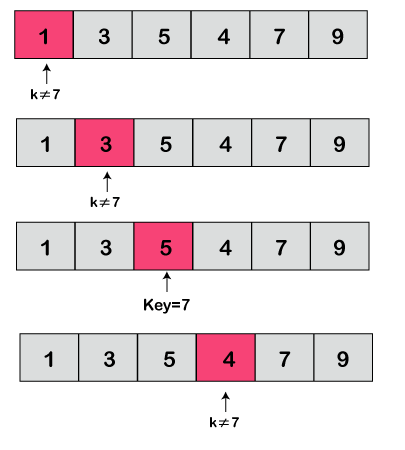
Linear search is a method of finding elements within a list. It is also called a sequential search.  It is the simplest searching algorithm because it searches the desired element in a sequential manner.

It compares each and every element with the value that we are searching for. If both are matched, the element is found, and the algorithm returns the key's index position.

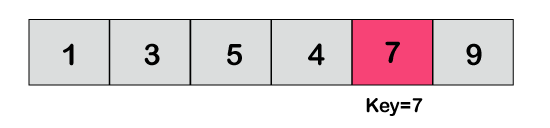
Step - 1: Start the search from the first element and Check key = 7 with each element of list x.



Step - 2: If element is found, return the index position of the key.



Step - 3: If element is not found, return element is not present.



Linear Search Algorithm

There is list of n elements and key value to be searched.

1. LinearSearch(list, key)
2. for each item in the list
3. if item == value
4. return its index position
5. return -1

* Python program to implement bubble sort.

The bubble sort uses a straightforward logic that works by repeating swapping the adjacent elements if they are not in the right order. It compares one pair at a time and swaps if the first element is greater than the second element; otherwise, move further to the next pair of elements for comparison.

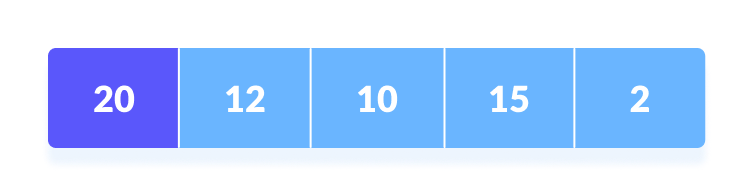
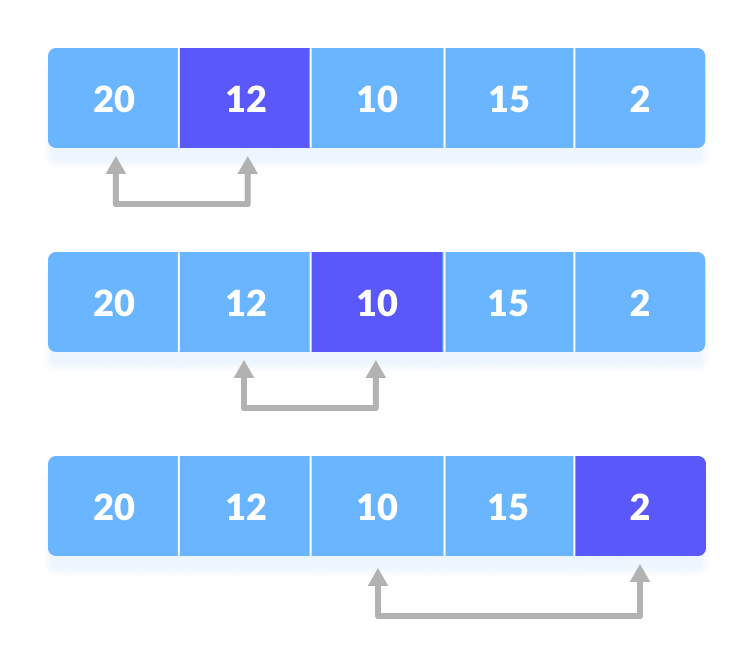
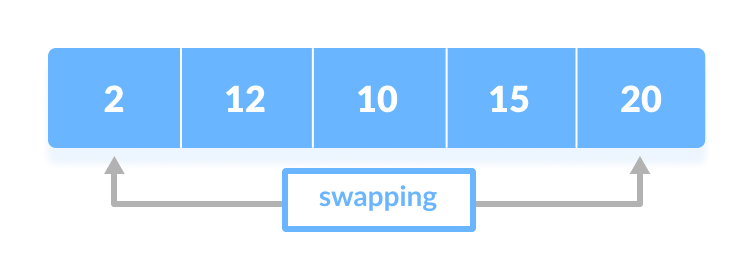
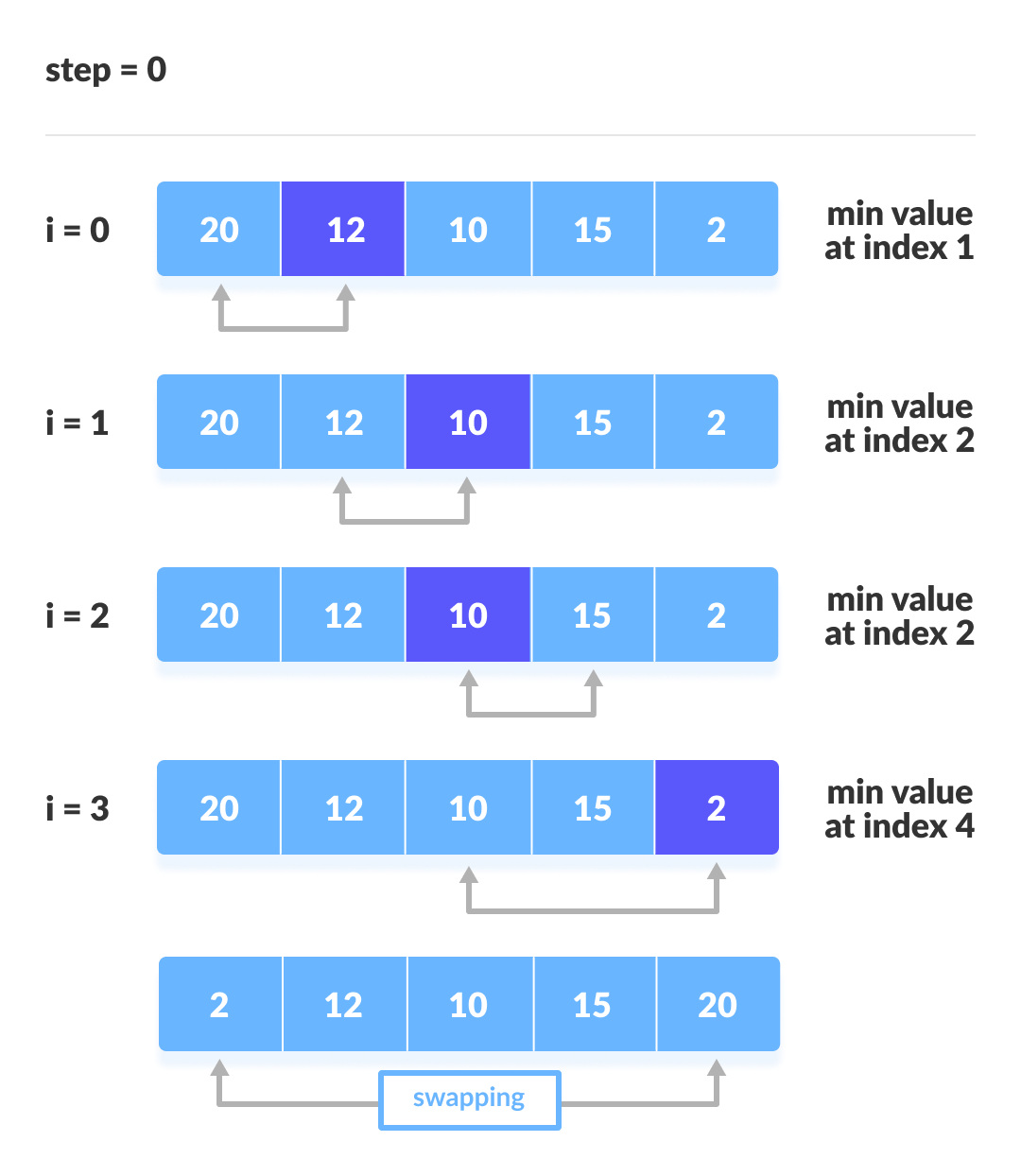
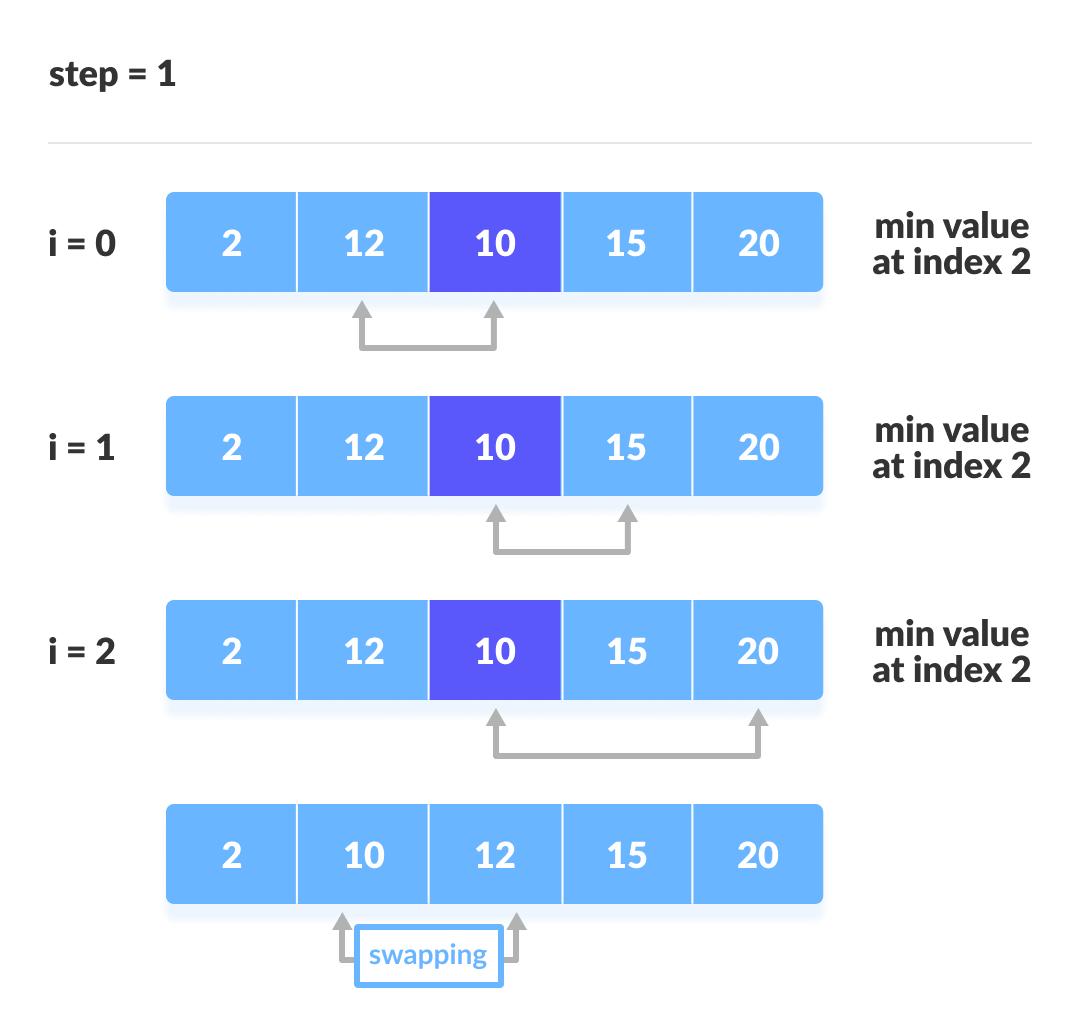
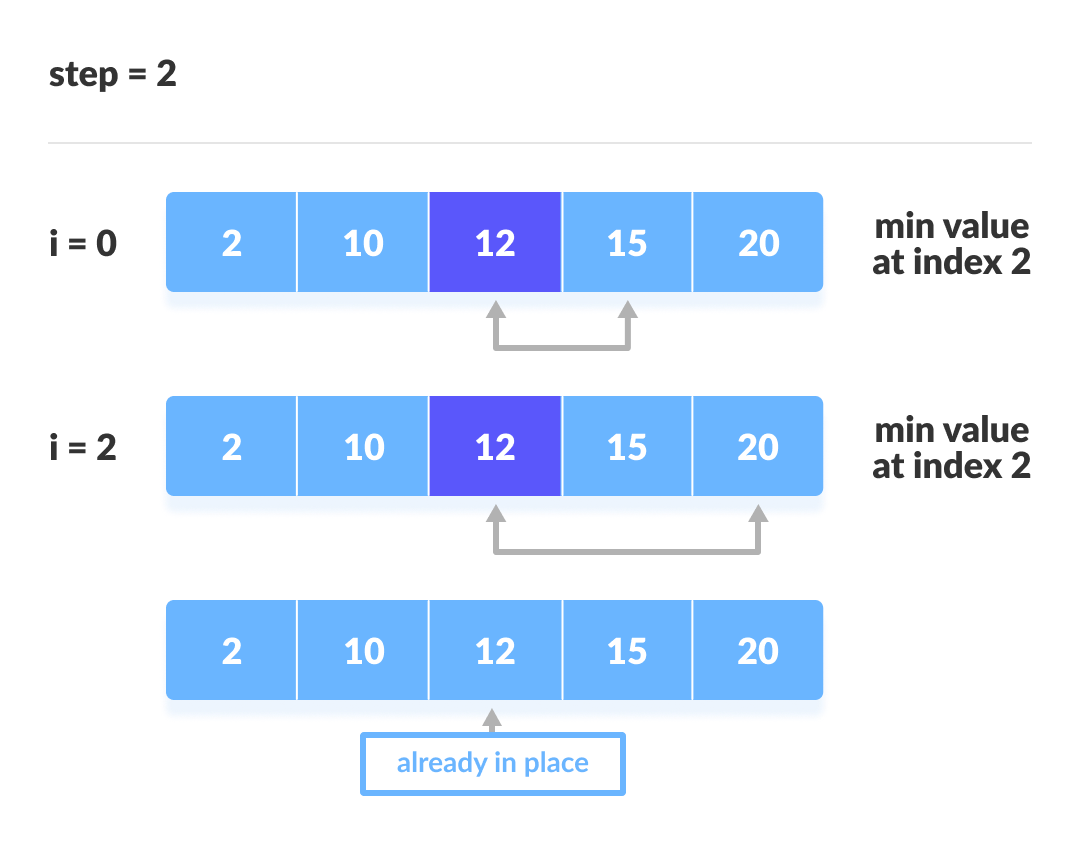
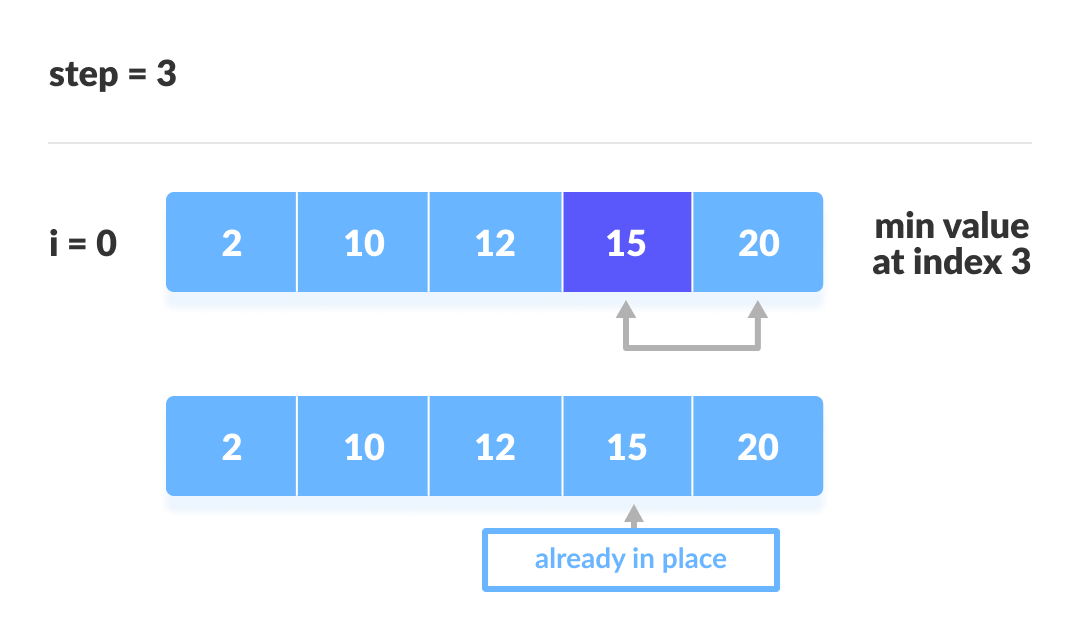
1. Create a function bubble sort that takes a list as argument.  
2. Inside the function create a loop with a loop variable i that counts from the length of the list – 1 to 1.  
3. Create an inner loop with a loop variable that counts from 0 up to i – 1.  
4. Inside the inner loop, if the elements at indexes j and j + 1 are out of order, then swap them.  
5. If in one iteration of the inner loop there were no swaps, then the list is sorted and one can return prematurely.

* Python program to implement binary search without recursion.

1. Create a function binary search that takes a list and key as arguments.  
2. The variable start is set to 0 and end is set to the length of the list.  
3. The variable start keeps track of the first element in the part of the list being searched while end keeps track of the element one after the end of the part being searched.  
4. A while loop is created that iterates as long as start is less than end.  
5. mid is calculated as the floor of the average of start and end.  
6. If the element at index mid is less than key, start is set to mid + 1 and if it is more than key, end is set to mid. Otherwise, mid is returned as the index of the found element.  
7. If no such item is found, -1 is returned.

* Python program to implement selection sort.

Selection sort is [a sorting algorithm](https://www.programiz.com/dsa/sorting-algorithm) that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

1. Set the first element as minimum.Select first element as minimum
2. Compare minimum with the second element. If the second element is smaller than minimum, assign the second element as minimum.  
     
   Compare minimum with the third element. Again, if the third element is smaller, then assign minimum to the third element otherwise do nothing. The process goes on until the last element.Compare minimum with the remaining elements
3. After each iteration, minimum is placed in the front of the unsorted list.Swap the first with minimum
4. For each iteration, indexing starts from the first unsorted element. Step 1 to 3 are repeated until all the elements are placed at their correct positions.The first iterationThe second iterationThe third iterationThe fourth iteration

Selection Sort Algorithm

selectionSort(array, size)

repeat (size - 1) times

set the first unsorted element as the minimum

for each of the unsorted elements

if element < currentMinimum

set element as new minimum

swap minimum with first unsorted position

end selectionSort

1. Steps for experiment/practical/Code:

* Python program to implement linear search.

def linear\_Search(list1, n, key):

# Searching list1 sequentially

for i in range(0, n):

if (list1[i] == key):

return i

return -1

list1 = [1 ,3, 5, 4, 7, 9]

key = 4

n = len(list1)

res = linear\_Search(list1, n, key)

if(res == -1):

print("Element not found")

else:

print("Element found at index: ", res)

* Python program to implement bubble sort.

# Creating a bubble sort function

def bubble\_sort(list1):

# Outer loop for traverse the entire list

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

for j in range(len(list1)-1):

if(list1[j]>list1[j+1]):

temp = list1[j]

list1[j] = list1[j+1]

list1[j+1] = temp

return list1

list1 = [5, 3, 8, 6, 7, 2]

print("The unsorted list is: ", list1)

# Calling the bubble sort function

print("The sorted list is: ", bubble\_sort(list1))

* Python program to implement binary search without recursion.

def binary\_search(alist, key):

"""Search key in alist[start... end - 1]."""

start = 0

end = len(alist)

while start < end:

mid = (start + end)//2

if alist[mid] > key:

end = mid

elif alist[mid] < key:

start = mid + 1

else:

return mid

return -1

alist = input('Enter the sorted list of numbers: ')

alist = alist.split()

alist = [int(x) for x in alist]

key = int(input('The number to search for: '))

index = binary\_search(alist, key)

if index < 0:

print('{} was not found.'.format(key))

else:

print('{} was found at index {}.'.format(key, index))

* Python program to implement selection sort.

import sys

A = [64, 25, 12, 22, 11]

# Traverse through all array elements

for i in range(len(A)):

# Find the minimum element in remaining

# unsorted array

min\_idx = i

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

if A[min\_idx] > A[j]:

min\_idx = j

# Swap the found minimum element with

# the first element

A[i], A[min\_idx] = A[min\_idx], A[i]

# Driver code to test above

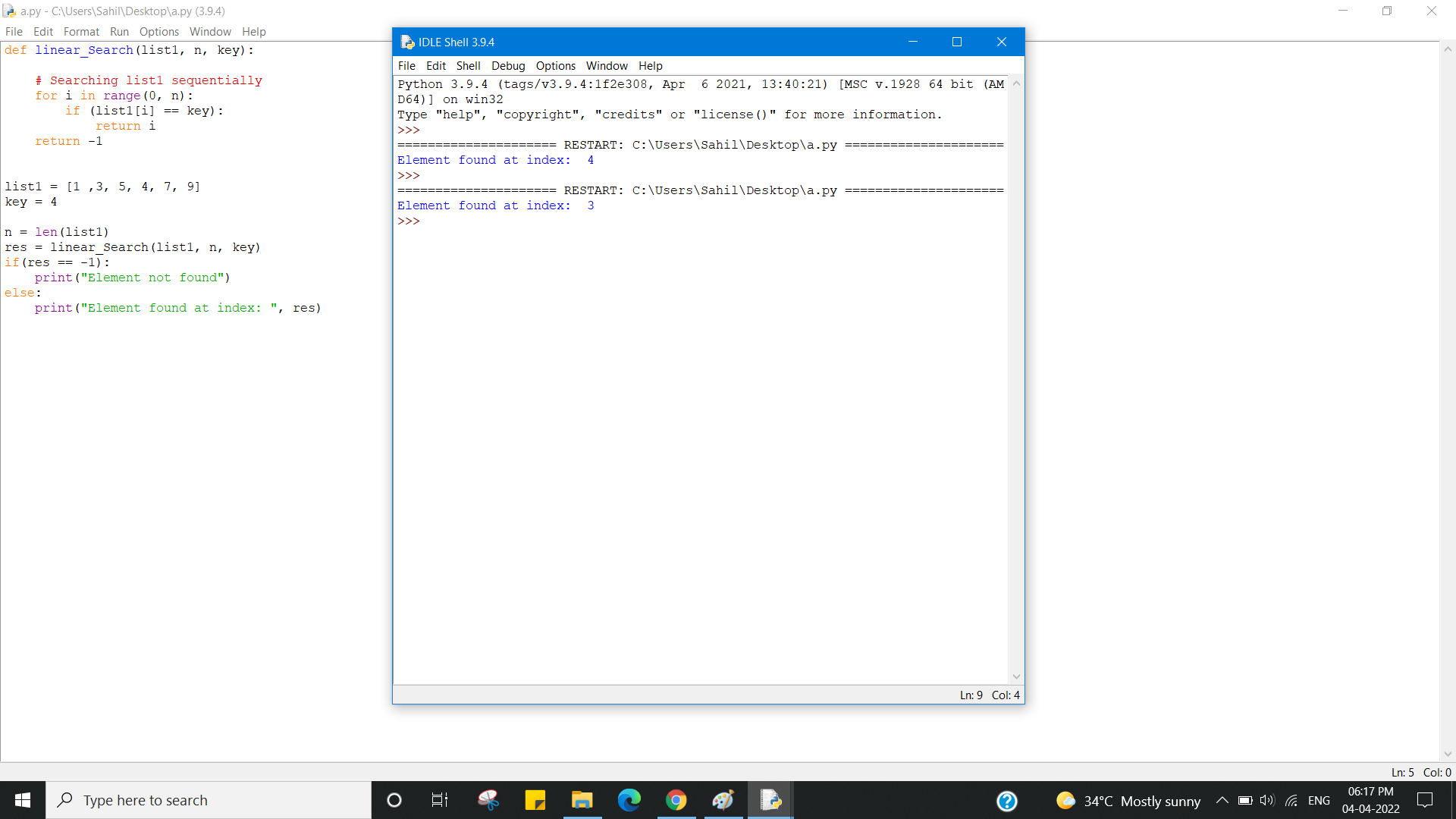
print ("Sorted array from Selection Sort: ")

for i in range(len(A)):

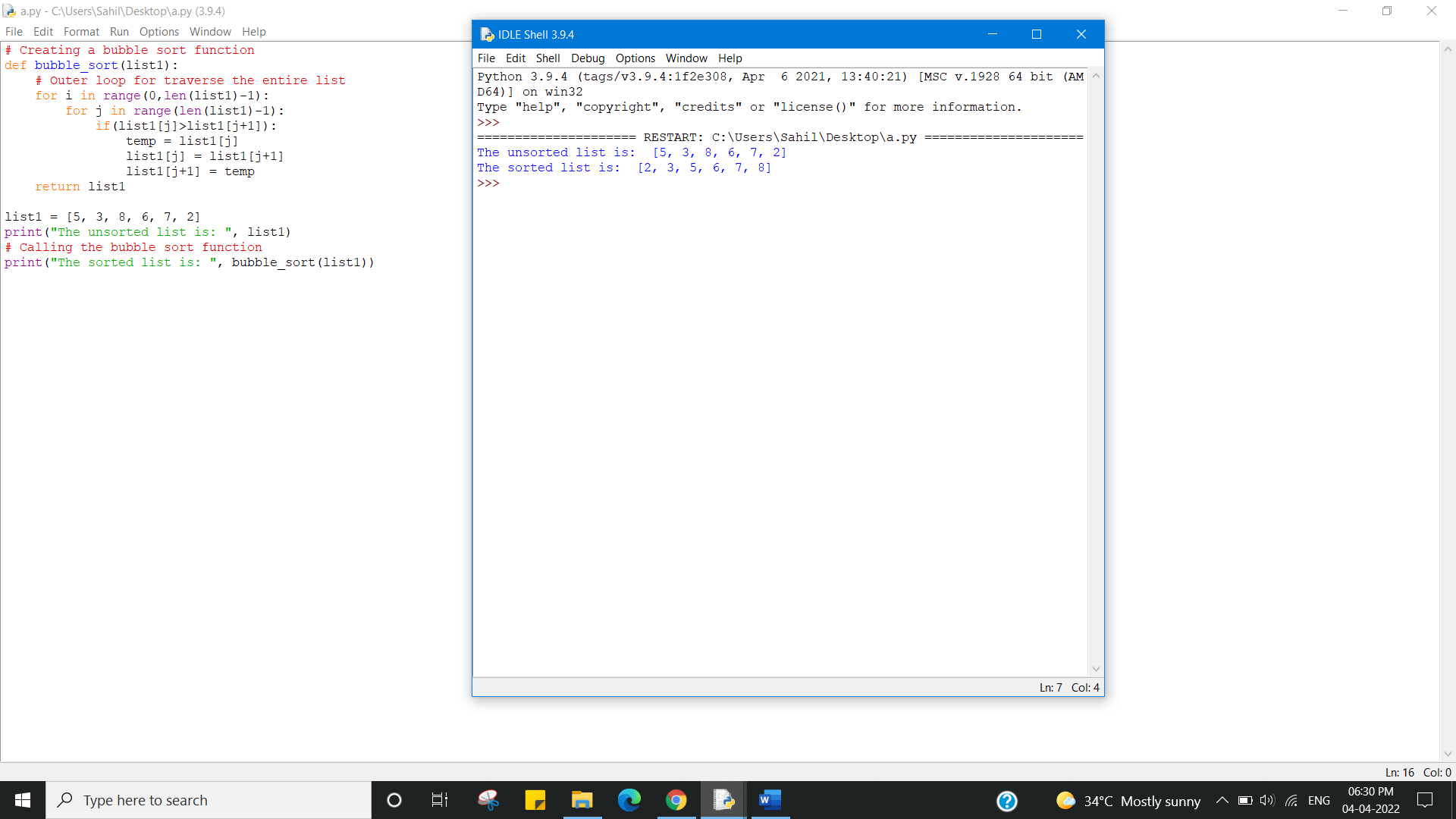
print("%d" %A[i])

1. Result/Output/Writing Summary:

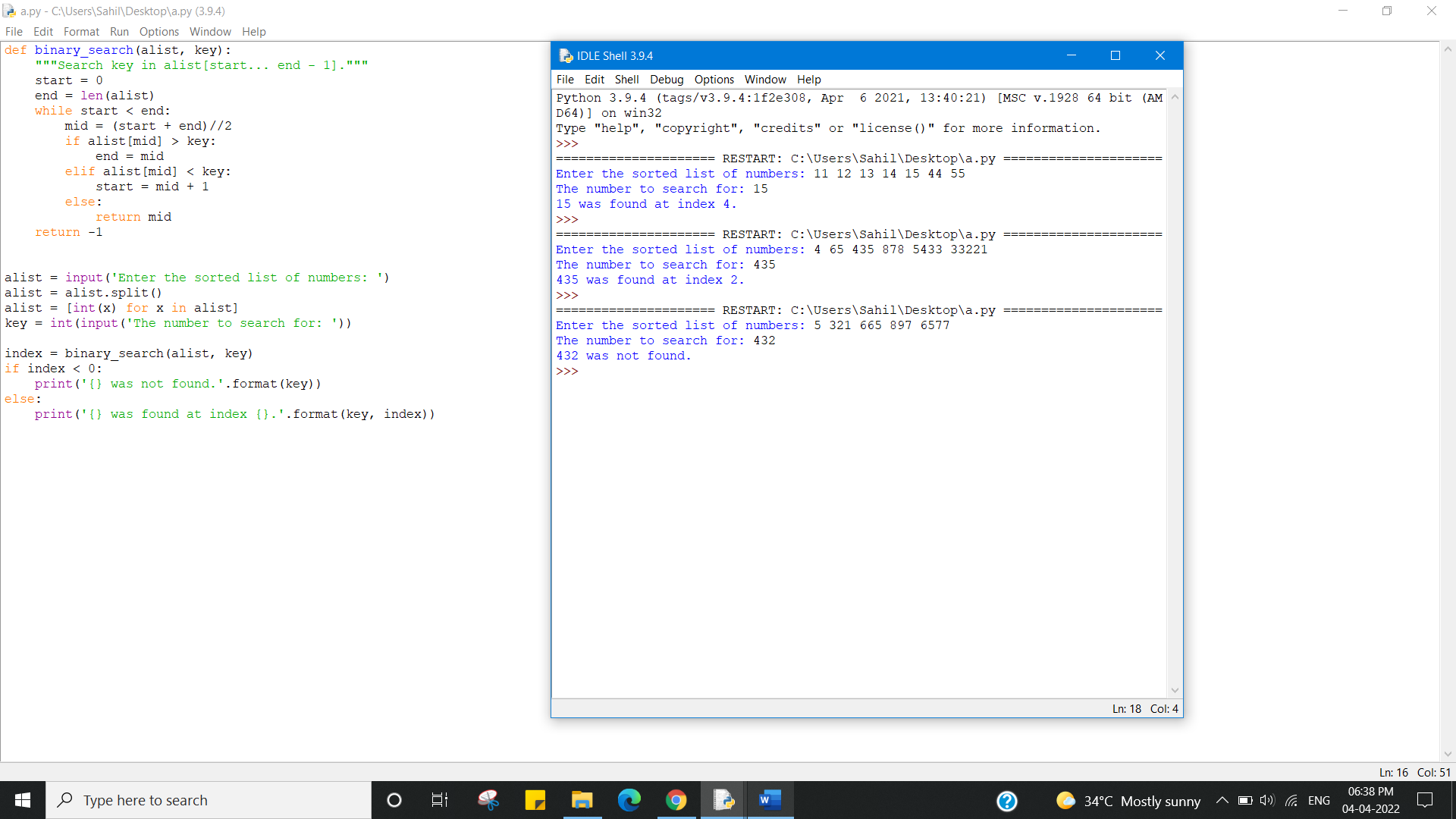
* Python program to implement linear search.



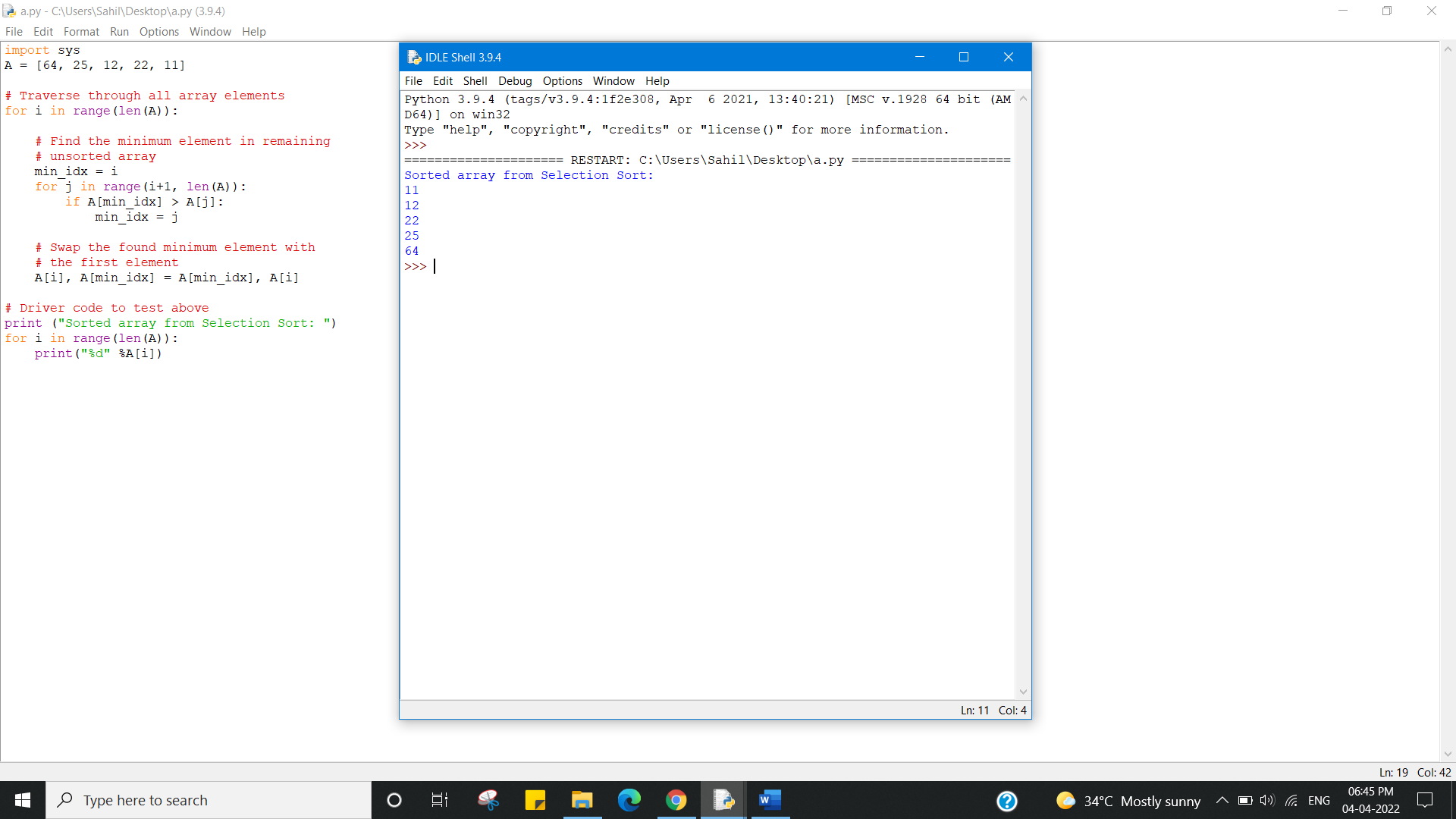
* Python program to implement bubble sort.



* Python program to implement binary search without recursion.



* Python program to implement selection sort.



Learning outcomes (What I have learnt):

1. Python program to implement linear search.
2. Python program to implement bubble sort.
3. Python program to implement binary search without recursion.
4. Python program to implement selection sort.

Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):



|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |