Task 5

1.Immutable Data types:

Immutable data types cannot be changed after they are created

- Integer (int)
- Float (float)
- Complex (complex)
- Strings (str)
- Tuples (tuple)
- Booleans

Mutable Data types:

Mutable data types can be changed after they are created

- Lists (list)
- Dictionaries (dict)
- Sets (set)
- Byte arrays

2.Sorting:

- **Selection Sort**

  This sorting technique repeatedly finds the minimum element and sorts it in order.

```
# Selection Sort algorithm in Python
def selectionSort(array, size):

    for s in range(size):
        min_idx = s

        for i in range(s + 1, size):

            # For sorting in descending order
            # for minimum element in each loop
            if array[i] < array[min_idx]:
                min_idx = i

        # Arranging min at the correct position
        (array[s], array[min_idx]) = (array[min_idx], array[s])

# Driver code
data = [ 7, 2, 1, 6 ]
size = len(data)
selectionSort(data, size)

print('Sorted Array in Ascending Order is :')
print(data)
```

- **Insertion Sort:**

  This sorting algorithm maintains a sub-array that is always sorted.

```
# Creating a function for insertion sort algorithm
def insertion_sort(list1):

        # Outer loop to traverse on len(list1)
        for i in range(1, len(list1)):

            a = list1[i]

            # Move elements of list1[0 to i-1],
            # which are greater to one position
            # ahead of their current position
            j = i - 1

            while j >= 0 and a < list1[j]:
                list1[j + 1] = list1[j]
                j -= 1

            list1[j + 1] = a

        return list1

# Driver code
list1 = [ 7, 2, 1, 6 ]
print("The unsorted list is:", list1)
print("The sorted new list is:", insertion_sort(list1))
```

- **Quicksort**
  the Quicksort algorithm applies the divide-and-conquer principle to divide the input array into two lists, the first with small items and the second with large items. The algorithm then sorts both lists recursively until the resultant list is completely sorted.
  from random import randint

```
from random import randint

def quicksort(array):
    # If the input array contains fewer than two elements,
    # then return it as the result of the function
    if len(array) < 2:
        return array

    low, same, high = [], [], []

    # Select your `pivot` element randomly
    pivot = array[randint(0, len(array) - 1)]

    for item in array:
        if item < pivot:
            low.append(item)
        elif item == pivot:
            same.append(item)
        elif item > pivot:
            high.append(item)

    # The final result combines the sorted `low` list
    # with the `same` list and the sorted `high` list
    return quicksort(low) + same + quicksort(high)
```

3.Python Sequence Datatype

It is a type that stores more than one item at a time. Therefore, we can say that it is a collection of items. Moreover, to access these items each item has a particular index number. There are three types of sequence data types **strings, lists, and tuples.**