**Bubble sort :**

There are other methods as well to sort a list. However, we will try to sort a list using bubble sort. We will find out why it is special. Before going to look at bubble sort, we should know that in almost all sorting techniques we have on thing common which is called ***‘swapping’*** . below is how we perform swapping with the help of 3rd variable. We will use the same method in bubble sort.

Diagram, polygon

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with medium confidence

Diagram

Description automatically generated

After the first iteration we got the biggest value 8 at the end.

A picture containing text, electronics, keyboard, calculator

Description automatically generated

After 5 iterations we got the list sorted as shown below.

A screenshot of a computer

Description automatically generated with low confidence

For the first iteration we will make sure the biggest element comes to the end as shown below.

A screenshot of a computer

Description automatically generated with low confidence

The outer loop will be for doing the same thing repeatedly as shown below.

A picture containing text, electronics, calculator

Description automatically generated

Let’s do it with code using a list.

def sort(nums):

    for i in range(len(nums)-1, 0, -1):#outer loop and len(nums)-1, we have to reach till 0 which is first index value, -1 to reduce the values one by one

        for j in range(i): #inner loop for the first iteration we have to go till end(8), we have to start with zero and reach till 5. In the next iteration we have to go from 0 to 4, next 0 to 3...which will place the maximum values at the end.

            if nums[j]>nums[j+1]:#in this step we will find to swap or not to swap. If the first value is greater than 2nd value - swap.

                temp = nums[j]

                nums[j] = nums[j+1]

                nums[j+1] = temp

#starting steps

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

sort(nums)

print(nums)

**Selection sort using python:**

We know that in bubble sort we are performing multiple swaps in each iteration. Swapping consumes lot of processing power and memory, we don’t want to do that.

Hence in selection sort we go from starting to end and find either minimum value or maximum value depending upon different implementations. In this example we will go for min value, which means we will go from start to end and we will find the min value. So, in each iteration you will do only one thing which is to find the min value.

Let’s take the same values as we have seen for bubble sort. We will go from start to end to find the minimum value.

Icon

Description automatically generated

Frist we assume that min is the first value which is 5 and then for every iteration we will start comparing with the other values.

Initially 5 is the min value and then we will compare it with 3 and the min value will change to 3 as we know that 5>3.

Graphical user interface, application

Description automatically generated with medium confidence

New min value is 3 and it will compare with the next value 8 and so on till 2. So finally at the end of first iteration 3>2 so it will be 2 which is the min value.

Icon

Description automatically generated

Now we know 2 is min value and what we will do is swap(5,2) as shown below. What we are creating is a sorted array 2 which is the min value and unsorted array which are remaining values.

Diagram

Description automatically generated with medium confidence

Now from the remaining elements of the unsorted array, we will perform the same steps. Start from the first value and go till the end.

Text, icon

Description automatically generated

A picture containing icon

Description automatically generated

Same steps will be performed and swap the values. Now 2,3 becomes sorted array and the remaining values become unsorted array.

Diagram

Description automatically generated

Final result

A picture containing text, electronics, calculator, keyboard

Description automatically generated

Now let’s write it in code as shown below.

We need 2 loops again as we need multiple iterations

def sort(nums):

    for i in range(5):#first loop we have to reach till 4

        minpos = i #when i start with 0 it is 0 and it will reach till 4

        for j in range(i,6): #nested loop, after every iteration we are creating sorted array and we have to reduce our unsorted array. start with i and end with 6 as we have 6 values

            if nums[j]<nums[minpos]:#only thing is we have to find the min value, we defined min value i starting with 5 and we will find the min value and swap accordingly

                minpos = j#the moment you find the value which is less that minpos we will change the position

        temp = nums[i]     # we are taking 3rd variable temp for swapping and doing it for i because we need a fixed value after outer loop iteration. If we do it for j it keeps on swapping and never ends.

        nums[i] = nums[minpos] #this will result in sorted array

        nums[minpos] = temp

#starting steps

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

sort(nums)

print(nums)

now if we want to see how it looks like after every iteration, add print(nums) after the swapping as shown below.

Timeline

Description automatically generated

Observe the output and understand how the values are getting swapped.