**1.8. Sort an array of integers using the bubble sort technique. Analyze its time complexity using Big-O notation. Write the code**

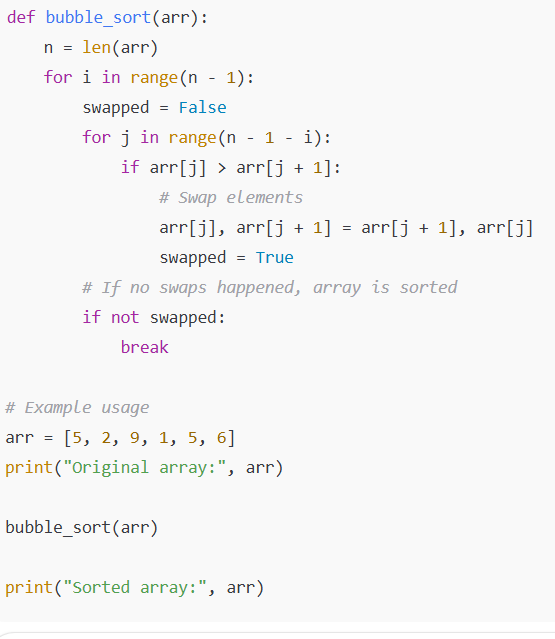
**Aim:**

To write a program that sorts an array of integers using the **Bubble Sort** technique and to analyse its **time complexity** using **Big-O notation**.

**Algorithm:**

1. Start with the first element of the array.
2. Compare the current element with the next element.
3. If the current element is greater than the next element, swap them.
4. Move to the next element and repeat steps 2 and 3 until the end of the array is reached.
5. After each pass, the largest element will be at the end.
6. Repeat steps 1 to 5 for the remaining unsorted part of the array (reduce the range by 1 each time).
7. Stop when no swaps are needed during a pass (the array is sorted).

**Code:**



**Input:**

****

**Output:  
**

**Performance analysis:**

**Time Complexity of Bubble Sort**

* **Best Case (Already Sorted):**
  + **The algorithm only needs one pass through the array to check if it is sorted.**
  + **Time Complexity: O(n)*O*(*n*)**
* **Average Case:**
  + **The algorithm needs to perform multiple passes, comparing and swapping elements.**
  + **Time Complexity: O(n2)*O*(*n*2)**
* **Worst Case (Reversed Order):**
  + **The algorithm performs the maximum number of comparisons and swaps.**
  + **Time Complexity: O(n2)*O*(*n*2)**

**Space Complexity of Bubble Sort**

* **Bubble Sort sorts the array in-place, meaning it doesn’t require extra space proportional to input size.**
* **Only a few extra variables are used for swapping and looping.**
* **Space Complexity: O(1)*O*(1)**