

# CS23336-Introduction to Python Programming

**Started on** Monday, 11 November 2024, 11:27 AM

**State** Finished

**Completed on** Monday, 11 November 2024, 12:44 PM

**Time taken** 1 hour 17 mins

**Marks** 10.00/10.00

**Grade** **100.00** out of 100.00

## Question 1

Correct  
Mark 1.00 out of 1.00  
 Flag question

### Question text

The problem is that we want to reverse a array in O(N) linear time complexity and we want the algorithm to be in-place as well!  
For example: input is [1,2,3,4,5] then the output is [5,4,3,2,1]  
Input  
5  
1 2 3 4 5  
Output  
5 4 3 2 1

For example:

Input	Result
5 1 2 3 4 5	5 4 3 2 1

Answer:(penalty regime: 0 %)

```
1 a=int(input())
2 l=list(map(int,input().split()))
3 for i in range(len(l)//2):
4     l[i],l[(i+1)*-1]=l[(i+1)*-1],l[i]
5 print(*l)
```


### Feedback

Input	Expected	Got
5 1 2 3 4 5	5 4 3 2 1	5 4 3 2 1
10 0 2 4 6 8 1 3 5 7 9	9 7 5 3 1 8 6 4 2 0	9 7 5 3 1 8 6 4 2 0

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

## Question 2

Correct  
Mark 1.00 out of 1.00  
 Flag question

### Question text

Given an integer array `nums` sorted in **non-decreasing** order, return *an array of **the squares of each number** sorted in non-decreasing order.*

#### Example 1:

**Input:** `nums = [-4,-1,0,3,10]`  
**Output:** `[0,1,9,16,100]`  
**Explanation:** After squaring, the array becomes `[16,1,0,9,100]`. After sorting, it becomes `[0,1,9,16,100]`.

#### Example 2:

**Input:** `nums = [-7,-3,2,3,11]`  
**Output:** `[4,9,9,49,121]`

### Constraints:

- `1 <= nums.length <= 104`
- `-104 <= nums[i] <= 104`
- `nums` is sorted in **non-decreasing** order.

For example:

Test	Result
<code>print(sortedSquares([-4,-1,0,3,10]))</code>	<code>[0, 1, 9, 16, 100]</code>

Answer:(penalty regime: 0 %)

Reset answer

```
1 def sortedSquares(l: list[int]) -> list[int]:
2     l=list(map(lambda x: x**2,l))
3     for i in range(len(l)-1):
4         for j in range(len(l)-1):
5             if l[j]>l[j+1]:
6                 l[j],l[j+1]=l[j+1],l[j]
7     return l
```

### Feedback

Test	Expected	Got
<code>print(sortedSquares([-4,-1,0,3,10]))</code>	<code>[0, 1, 9, 16, 100]</code>	<code>[0, 1, 9, 16, 100]</code>


```
print(sortedSquares([-7,-3,2,3,11])) [4, 9, 9, 49, 121] [4, 9, 9, 49, 121]
```

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

### Question 3

Correct  
Mark 1.00 out of 1.00

 Flag question

#### Question text

Given an array of integers `nums`, sort the array in **increasing** order based on the frequency of the values. If multiple values have the same frequency, sort them in **decreasing** order.

Print the *sorted array*.

#### Example 1:

**Input:**  
6  
1 1 2 2 2 3

**Output:**  
3 1 1 2 2 2

**Explanation:** '3' has a frequency of 1, '1' has a frequency of 2, and '2' has a frequency of 3.

#### Example 2:

**Input:**  
5  
2 3 1 3 2

**Output:**  
1 3 3 2 2

**Explanation:** '2' and '3' both have a frequency of 2, so they are sorted in decreasing order.

#### Example 3:

**Input:**  
9  
-1 1 -6 4 5 -6 1 4 1

**Output:**  
5 -1 4 4 -6 -6 1 1 1

#### Constraints:

- 1 <= nums.length <= 100
- -100 <= nums[i] <= 100

For example:

Input	Result
6 1 1 2 2 2 3	3 1 1 2 2 2
5 2 3 1 3 2	1 3 3 2 2

Answer:(penalty regime: 0 %)

```
1 n=int(input())
2 l=list(map(int,input().split()))
3 l.sort(reverse=True)
4 lf=[]
5 for i in range(len(l)):
6     lf.append(l.count(l[i]))
7 for i in range(len(lf)-1):
8     for j in range(len(lf)-1):
9         if lf[j]>lf[j+1]:
10             l[j],l[j+1]=l[j+1],l[j]
```

```
11         lf[j],lf[j+1]=lf[j+1],lf[j]
12     print(*l)
```


Feedback

Input	Expected	Got
6 1 1 2 2 2 3	3 1 1 2 2 2	3 1 1 2 2 2
5 2 3 1 3 2	1 3 3 2 2	1 3 3 2 2
9 -1 1 -6 4 5 -6 1 4 1	5 -1 4 4 -6 -6 1 1 15 -1 4 4 -6 -6 1 1 1	

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

Question 4

Correct  
Mark 1.00 out of 1.00  
 Flag question

Question text

Given an integer array `nums`, return *an integer array counts where counts[i] is the number of smaller elements to the right of nums[i]*.

Example 1:

**Input:** `nums = [5,2,6,1]`  
**Output:** `[2,1,1,0]`  
**Explanation:**  
To the right of 5 there are 2 smaller elements (2 and 1).  
To the right of 2 there is only 1 smaller element (1).  
To the right of 6 there is 1 smaller element (1).  
To the right of 1 there is 0 smaller element.

Example 2:

**Input:** `nums = [-1]`  
**Output:** `[0]`

Example 3:

**Input:** `nums = [-1,-1]`  
**Output:** `[0,0]`

Constraints:

- `1 <= nums.length <= 105`
- `-104 <= nums[i] <= 104`

For example:

Test	Result
<code>print(countSmaller([5,2,6,1]))</code>	<code>[2, 1, 1, 0]</code>

```
print(countSmaller([-1])) [0]
```

Answer:(penalty regime: 0 %)

Reset answer

```
1 def countSmaller(nums:list[int]) -> list[int]:
2     cl=[]
3     for i in range(len(nums)):
4         right=nums[i:]
5         count=0
6         for j in right:
7             if nums[i]>j:
8                 count+=1
9         cl.append(count)
10    return cl
11
```


Feedback

Test	Expected	Got
print(countSmaller([5,2,6,1]))	[2, 1, 1, 0]	[2, 1, 1, 0]
print(countSmaller([50,20,60,10]))	[2, 1, 1, 0]	[2, 1, 1, 0]
print(countSmaller([-1]))	[0]	[0]
print(countSmaller([-1, -1]))	[0, 0]	[0, 0]

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

Question 5

Correct  
Mark 1.00 out of 1.00  
 Flag question

Question text

Objective:

Develop a Python program to find the k-th maximum value in a given list of integers. The program should handle various edge cases, including lists with duplicate values, empty lists, and invalid values of k. The k-th maximum value refers to the k-th largest distinct element in the list.

Background:

Finding the k-th maximum value in a list is a common problem in computer science, often encountered in fields like data analysis, competitive programming, and software development. This problem requires an understanding of sorting algorithms, data structures, and efficient problem-solving techniques. By solving this problem, one gains insights into how to handle large datasets and optimize performance in practical applications.

Problem Description:

Given a list of integers, the task is to determine the k-th maximum value in the list. The program should meet the following requirements:

- 1. **Input:**

- A list of integers, which can contain both positive and negative values.
- An integer k, representing the position of the maximum value to find.

## 2. Output:

- The k-th maximum value in the list.
- If k is greater than the number of distinct elements in the list or if the list is empty, the program should return an appropriate message indicating the error.

## Constraints:

- The list may contain duplicate values.
- The value of k should be a positive integer.
- The list may contain up to  $10^6$  elements, and each element can be as large as  $10^9$  in magnitude.

## Examples:

Consider the following examples for better understanding:

### 1. Example 1:

- **Input:** list = [3, 1, 5, 4, 2], k = 2
- **Output:** 4
- **Explanation:** The distinct elements in the list are [1, 2, 3, 4, 5]. The 2nd maximum value is 4.

### 2. Example 2:

- **Input:** list = [7, 7, 7, 7, 7], k = 1
- **Output:** 7
- **Explanation:** The distinct elements in the list are [7]. The 1st maximum value is 7.

### 3. Example 3:

- **Input:** list = [2, 1, 2, 1, 2], k = 3
- **Output:** -1
- **Explanation:** The distinct elements in the list are [1, 2]. There is no 3rd maximum value.

For example:

Input	Result
5 3 1 5 4 2 2	4
6 7 7 7 7 7 7 1	7
10 2 1 2 1 2 1 2 1 2 1 -1 3	

Answer:(penalty regime: 0 %)

```

1 n=int(input())
2 l=list(set(list(map(int,input().split()))))
3 k=int(input())
4 if k<=len(l):
5     print(l[k*(-1)])
6 else:
7     print(-1)

```


Feedback

Input	Expected	Got
5 3 1 5 4 2 2	4	4
6 7 7 7 7 7 1	7	7
10 2 1 2 1 2 1 2 1 2 1 3	-1	

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

Question 6

Correct  
Mark 1.00 out of 1.00  
 Flag question

Question text

To find the frequency of numbers in a list and display in sorted order.

Constraints:

1<=n, arr[i]<=100

Input:

1 68 79 4 90 68 1 4 5

output:

1 2  
4 2  
5 1  
68 2  
79 1  
90 1

For example:

Input	Result
3 2	
4 3 5 3 4 5 4 2	
5 2	

Answer:(penalty regime: 0 %)

```
1 n=list(map(int,input().split()))
2 for i in set(n):
3     print(i,n.count(i))
```



Feedback

Input	Expected Got	
4 3 5 3 4 5	3 2	3 2
	4 2	4 2
	5 2	5 2
12 4 4 4 2 3 5	2 1	2 1
	3 1	3 1
	4 3	4 3
	5 1	5 1
	12 1	12 1
5 4 5 4 6 5 7 3 5	3 1	3 1
	4 2	4 2
	5 3	5 3
	6 1	6 1
	7 1	7 1

Passed all tests!


Correct

Marks for this submission: 1.00/1.00.

Question 7

Correct

Mark 1.00 out of 1.00

 Flag question

Question text

Given an listof integers, sort the array in ascending order using the *Bubble Sort* algorithm above. Once sorted, print the following three lines:

1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
2. First Element: firstElement, the *first* element in the sorted list.
3. Last Element: lastElement, the *last* element in the sorted list.

For example, given a worst-case but small array to sort: a=[6,4,1]. It took 3 swaps to sort the array. Output would be

Array is sorted in 3 swaps.

First Element: 1

Last Element: 6

Input Format

The first line contains an integer,n , the size of the list a .

The second line contains n, space-separated integers a[i].

Constraints

- 2<=n<=600
- 1<=a[i]<=2x10<sup>6</sup>.

Output Format

You must print the following three lines of output:



- 1. List is sorted in numSwaps swaps., where numSwaps is the number of swaps that took place.
- 2. First Element: firstElement, the *first* element in the sorted list.
- 3. Last Element: lastElement, the *last* element in the sorted list.

Sample Input 0

3  
1 2 3

Sample Output 0

List is sorted in 0 swaps.  
First Element: 1  
Last Element: 3

For example:

Input	Result
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Answer:(penalty regime: 0 %)

```
1 n=int(input())
2 l=list(map(int,input().split()))
3 swap=0
4 for i in range(len(l)-1):
5     for j in range(len(l)-1):
6         if l[j]>l[j+1]:
7             t=l[j]
8             l[j]=l[j+1]
9             l[j+1]=t
10            swap+=1
11 print(f"List is sorted in {swap} swaps.\nFirst Element: {l[0]}\nLast Element: {l[-1]}")
```


Feedback

Input	Expected	Got
3 3 2 1	List is sorted in 3 swaps. First Element: 1 Last Element: 3	List is sorted in 3 swaps. First Element: 1 Last Element: 3
5 1 9 2 8 4	List is sorted in 4 swaps. First Element: 1 Last Element: 9	List is sorted in 4 swaps. First Element: 1 Last Element: 9

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

## Question 8

Correct  
Mark 1.00 out of 1.00  
 Flag question

### Question text

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read an list of numbers. You need to arrange the elements in ascending order and print the result. The sorting should be done using bubble sort.

**Input Format:** The first line reads the number of elements in the array. The second line reads the array elements one by one.

**Output Format:** The output should be a sorted list.

For example:

Input	Result
6 3 4 8 7 1 2	1 2 3 4 7 8
5 4 5 2 3 1	1 2 3 4 5

Answer:(penalty regime: 0 %)

```
1 def bubble_sort(l):
2     for i in range(len(l)-1):
3         for j in range(len(l)-1):
4             if l[j]>l[j+1]:
5                 l[j],l[j+1]=l[j+1],l[j]
6 n=int(input())
7 lists=list(map(int,input().split()))
8 bubble_sort(lists)
9 print(*lists)
```


### Feedback

Input	Expected	Got
6 3 4 8 7 1 2	1 2 3 4 7 8	1 2 3 4 7 8
6 9 18 1 3 4 6	1 3 4 6 9 18	1 3 4 6 9 18
5 4 5 2 3 1	1 2 3 4 5	1 2 3 4 5

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

## Question 9

Correct  
Mark 1.00 out of 1.00  
 Flag question

### Question text

Given an array of integers `arr`, replace each element with its rank.

The rank represents how large the element is. The rank has the following rules:

- Rank is an integer starting from 1.
- The larger the element, the larger the rank. If two elements are equal, their rank must be the same.
- Rank should be as small as possible.

#### Example 1:

**Input:** `arr = [40,10,20,30]`  
**Output:** `[4,1,2,3]`  
**Explanation:** 40 is the largest element. 10 is the smallest. 20 is the second smallest. 30 is the third smallest.

#### Example 2:

**Input:** `arr = [100,100,100]`  
**Output:** `[1,1,1]`  
**Explanation:** Same elements share the same rank.

#### Example 3:

**Input:** `arr = [37,12,28,9,100,56,80,5,12]`  
**Output:** `[5,3,4,2,8,6,7,1,3]`

### Constraints:

- $0 \leq \text{arr.length} \leq 10^5$
- $-10^9 \leq \text{arr}[i] \leq 10^9$

For example:

Test	Result
<code>print(arrayRankTransform([40,10,20,30]))</code>	<code>[4, 1, 2, 3]</code>

Answer:(penalty regime: 0 %)

Reset answer

```
1 def bsort(l):
2     for i in range(len(l)-1):
3         for j in range(len(l)-1):
4             if l[j]>l[j+1]:
5                 l[j],l[j+1]=l[j+1],l[j]
6     return l
7
8
9
10 def arrayRankTransform(arr: list[int]) -> list[int]:
11     arrsort=bsort(list(set(arr)))
12     rank=[]
13     for i in arr:
14         rank.append(arrsort.index(i)+1)
15     return rank
16
```

### Feedback

Test	Expected	Got
------	----------	-----

```
print(arrayRankTransform([40,10,20,30]))           [4, 1, 2, 3]           [4, 1, 2, 3]


print(arrayRankTransform([100,100,100]))           [1, 1, 1]              [1, 1, 1]

print(arrayRankTransform([37,12,28,9,100,56,80,5,12])) [5, 3, 4, 2, 8, 6, 7, 1, 3] [5, 3, 4, 2, 8, 6, 7, 1, 3]
```

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

## Question 10

Correct  
Mark 1.00 out of 1.00  
 Flag question

### Question text

Write a Python program to sort a list of elements using the merge sort algorithm.  
For example:

Input	Result
5 6 5 4 3 8	3 4 5 6 8

Answer:(penalty regime: 0 %)

```
1 def mergesort(arr):
2     if len(arr)>1:
3         mid=len(arr)//2
4         left=arr[:mid]
5         right=arr[mid:]
6         mergesort(left)
7         mergesort(right)
8         i=j=k=0
9         while i<len(left) and j<len(right):
10            if left[i]<right[j]:
11                arr[k]=left[i]
12                i+=1
13            else:
14                arr[k]=right[j]
15                j+=1
16                k+=1
17        while i<len(left):
18            arr[k]=left[i]
19            i+=1
20            k+=1
21        while j<len(right):
22            arr[k]=right[j]
23            j+=1
24            k+=1
25    return arr
26 n=int(input())
27 arr=list(map(int,input().split()))
28 print(*mergesort(arr))
```

### Feedback

Input	Expected	Got
5 6 5 4 3 8	3 4 5 6 8	3 4 5 6 8
9 14 46 43 27 57 41 45 21 70	14 21 27 41 43 45 46 57 70	14 21 27 41 43 45 46 57 70
4 86 43 23 49	23 43 49 86	23 43 49 86

Passed all tests!

Correct  
Marks for this submission: 1.00/1.00.

Finish review

[Skip Quiz navigation](#)

**Quiz navigation**

[Question 1 This page](#) [Question 2 This page](#) [Question 3 This page](#) [Question 4 This page](#) [Question 5 This page](#) [Question 6 This page](#) [Question 7 This page](#) [Question 8 This page](#) [Question 9 This page](#) [Question 10 This page](#)

[Show one page at a time](#) Finish review