06 - Strings in Python

Ex. No. : 6.1 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Count Frequency

Complete the program to count frequency of each element of an array. Frequency of a particular element will be printed once.

Sample Test Cases

Test Case 1

Input

7
23
45
23
56
45
23
40

Output

23 occurs 3 times
45 occurs 2 times
56 occurs 1 times
40 occurs 1 times

```
def count_frequency(arr):
    freq_dict = {}
    for num in arr:
        if num in freq_dict:
            freq_dict[num] += 1
        else:
            freq_dict[num] = 1
    return freq_dict
```

```
arr = []
    for _ in range(n):
        arr.append(int(input()))

frequency_dict = count_frequency(arr)


    for key, value in frequency_dict.items():
        print(f"{key} occurs {value} times")
```

	Input	Expected	Got	
	7 23 45 23 56 45 23 40	23 occurs 3 times 45 occurs 2 times 56 occurs 1 times 40 occurs 1 times	23 occurs 3 times 45 occurs 2 times 56 occurs 1 times 40 occurs 1 times	

Ex. No. : 6.2 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Non-duplicate elements

Program to print all the distinct elements in an array. Distinct elements are nothing but the unique (non-duplicate) elements present in the given array.
Input Format:
First line take an Integer input from stdin which is array length n.
Second line take n Integers which is inputs of array.
Output Format:
Print the Distinct Elements in Array in single line which is space Separated
Example Input:
5
1
2
2
3
4
Output:
1 2 3 4
Example Input:
6
1
1
2
2
3
3
Output:
1 2 3

For example:

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

```
n = int(input())
arr = [int(input()) for _ in range(n)]
distinct_elements = set()
for num in arr:
    distinct_elements.add(num)
print(" ".join(map(str, distinct_elements)))
```

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

Ex. No. : 6.3 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Merged_array

Output is a merged array without duplicates.

Input Format

N1 - no of elements in array 1
Array elements for array 1
N2 - no of elements in array 2
Array elements for array2

Output Format

Display the merged array

Sample Input 1

5
1
2
3
6
9
4
2
4
5
10

Sample Output 1

1 2 3 4 5 6 9 10

```
 def
merge_arrays(arr1,
arr2):

   set1 = set(arr1)
   set2 = set(arr2)



   merged_array =
sorted(set1.union(se
t2))

   return
merged_array

def main():
   try:

      n1 = int(input())
      arr1 =
[int(input()) for _ in
range(n1)]



      n2 = int(input())
      arr2 =
[int(input()) for _ in
range(n2)]



      merged =
merge_arrays(arr1,
arr2)
```

```
        print(end="")
        for num in merged:
            print(num, end=" ")

    except ValueError:
        print()

if__name___== "_main_":
    main()
```

Input	Expected	Got	
	5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10	1 2 3 4 5 6 9 10
	7 4 7 8 10 12 30 35 9 1 3 4 5 7 8 11 13 22	1 3 4 5 7 8 10 11 12 13 22 30 35	1 3 4 5 7 8 10 11 12 13 22 30 35

Ex. No. : 6.4 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Sorted array

Consider a program to insert an element / item in the sorted array. Complete the logic by filling up required code in editable section. Consider an array of size 10. The eleventh item is the data is to be inserted.

Sample Test Cases

Test Case 1

Input

1
3
4
5
6
7
8
9
10
11
2

Output

ITEM to be inserted:2
After insertion array is:
1
2
3
4
5
6
7
8
9
10
11

```
def insert_into_sorted_array(arr, n, item):
    arr.append(0)
    i = n - 1
    while i >= 0 and arr[i] > item:
        arr[i+1] = arr[i]
        i -= 1
    arr[i+1] = item
    return arr
n = 10
arr = [int(input()) for _ in range(n)]
item = int(input())
arr = insert_into_sorted_array(arr, n, item)
print(f"ITEM to be inserted:{item}")
print("After insertion array is:")
for element in arr:
    print(element)
```

	Input	Expected	Got	
	1 3 4 5 6 7 8 9 10 11 2	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	ITEM to be inserted:2 After insertion array is: 1 2 3 4 5 6 7 8 9 10 11	
	11 22 33 55 66 77 88 99 110 120 44	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	ITEM to be inserted:44 After insertion array is: 11 22 33 44 55 66 77 88 99 110 120	

Ex. No. : 6.5 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Deleting element in list

Write a Python program to check if a given list is strictly increasing or not. Moreover, If removing only one element from the list results in a strictly increasing list, we still consider the list true
Input:
n : Number of elements
List1: List of values
Output
Print "True" if list is strictly increasing or decreasing else print "False"
Sample Test Case
Input
7
1
2
3
0
4
5
6
Output
True

```
n = int(input())
List1 = list(map(int, input().split()))

def is_strictly_increasing(n, List1):
    # Remove one element from the list and check if the remaining elements are strictly increasing
    for i in range(n):
        new_list = List1[:i] + List1[i+1:]
        if is_strictly_increasing_helper(new_list):
            return True

    # Check if the original list is strictly increasing
    return is_strictly_increasing_helper(List1)

def is_strictly_increasing_helper(List1):
    for i in range(len(List1) - 1):
        if List1[i] >= List1[i+1]:
            return False
    return True
```

```
# Check if the list is strictly decreasing
if List1 == sorted(List1, reverse=True):
    print("True")
elif is_strictly_increasing(n, List1):
    print("True")
else:
    print("False")
```

	Input	Expected	Got	
	7 1 2 3 0 4 5 6	True	True	
	4 2 1 0 -1	True	True	

Ex. No. : 6.6 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Repeated integers

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Input Format

1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

Output format

Print 1 if such a pair exists and 0 if it doesn't.

Example

Input

1

3

1

3

5

4

Output:

1

```
def find_pair_with_difference(arr, k):
    seen = set()
    for i in range(len(arr)):
        if (arr[i] - k) in seen or (arr[i] + k) in seen:
            return 1
        seen.add(arr[i])
    return 0
t = int(input())
for _ in range(t):
    n = int(input())
    arr = [int(input()) for _ in range(n)]
    k = int(input())
    result = find_pair_with_difference(arr, k)
```

	Input	Expected	Got	
	1 3 1 3 5 4	1	1	
	1 3 1 3 5 99	0	0	

x. No. : 6.7 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Pivot element

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example

arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Constraints

- $3 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 2 \times 10^4$, where $0 \leq I < n$
- It is guaranteed that a solution always exists.

The first line contains an integer n, the size of the array arr.

Each of the next n lines contains an integer, arr[i], where $0 \leq I < n$.

Sample Case 0

Sample Input 0

4
1
2
3
3

Sample Output 0

2

```
ef find_pivot(arr):
    n = len(arr)
    total_sum = sum(arr)
    left_sum = 0

    for I in range(n):
        right_sum = total_sum – left_sum – arr[i]
        if left_sum == right_sum:
            return i
        left_sum += arr[i]

    return -1  # not found

n = int(input())
arr = [int(input()) for _ in range(n)]
```

pivot_index = find_pivot(arr)

Input	Expected	Got	
4 1 2 3 3	2	2	
3 1 2 1	1	1	

Ex. No. : 6.8 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Zip List

Write a Python program to Zip two given lists of lists.

Input:

m : row size

n: column size

list1 and list 2 : Two lists

Output

Zipped List : List which combined both list1 and list2

```
def zip_lists_user_input():
    m = int(input())
    n = int(input())
    list1 = [[int(input()) for _ in range(n)] for _ in range(m)]
    list2 = [[int(input()) for _ in range(n)] for _ in range(m)]

    zipped_list = [a + b for a, b in zip(list1, list2)]
    return zipped_list
zipped_result = zip_lists_user_input()
print(zipped_result)
```

	Input	Expected	Got	
	2 2 1 2 3 4 5 6 7 8	[[1, 2, 5, 6], [3, 4, 7, 8]]	[[1, 2, 5, 6], [3, 4, 7, 8]]	

Ex. No. : 6.9 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Factors of a number

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return 0.

Example

n = 20

p = 3

The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if p = 3, then 4 is returned. If p > 6, 0 would be returned.

Constraints

$1 \leq n \leq 10^{15}$

$1 \leq p \leq 10^{9}$

The first line contains an integer n, the number to factor.

The second line contains an integer p, the 1-based index of the factor to return.

Sample Case 0

Sample Input 0

10

3

Sample Output 0

5

```
def find_pth_factor(n, p):
    factors = []
    for i in range(1, n + 1):
        if n % i == 0:
            factors.append(i)
    factors.sort()

    if p <= len(factors):
        return factors[p - 1]
    else:
        return 0

def main():
    try:
        n = int(input())
        p = int(input())
```

```
pth_factor =
find_pth_factor(n,
p)

print(pth_factor)
   except
ValueError:
      print()

if _name_ ==
"_main_":
   main()
```

	Input	Expected	Got	
	10 3	5	5	
	10 5	0	0	
	1 1	1	1	

Ex. No. : 6.10 **Date: 3.05.2024**

Register No.: 231401063 **Name: ADITHIYAN NATARAJAN**

Index Mapping

Given two lists A and B, and B is an anagram of A. B is an anagram of A means B is made by randomizing the order of the elements in A.
We want to find an *index mapping* P, from A to B. A mapping P[i] = j means the ith element in A appears in B at index j.
These lists A and B may contain duplicates. If there are multiple answers, output any of them.
For example, given
Input
5
12 28 46 32 50
50 12 32 46 28
Output
1 4 3 2 0

```
def find_mapping(A, B):
    mapping = {}
    for i, num in enumerate(B):
        mapping[num] = i
    return [mapping[num] for num in A]

if__name__== "__main__": n
    = int(input())
    A = list(map(int, input().split()))
    B = list(map(int, input().split()))
    mapping = find_mapping(A, B)
    print(*mapping)
```

Input	Expected	Got	
	5 12 28 46 32 50 50 12 32 46 28	1 4 3 2 0	1 4 3 2 0