Rajalakshmi Engineering College

Name: SONASREE RP
Email: 240701521@rajalakshmi.edu.in
Roll no: 240701521
Phone: 7305340666
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 3_PAH

Attempt : 1
Total Mark : 60
Marks Obtained : 60

Section 1 : Coding

1. Problem Statement

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to analyze input strings to categorize and count the characters into four categories: uppercase letters, lowercase letters, digits, and special characters. The company needs this tool to process log files and identify potential security threats.

Input Format

The input consists of the log entry provided as a single string.

Output Format

The output consists of four lines:

The first line contains an integer representing the count of uppercase letters in the format "Uppercase letters: {uppercase count}".

The second line contains an integer representing the count of lowercase letters in the format "Lowercase letters: {lowercase count}".

The third line contains an integer representing the count of digits in the format "Digits: {digits count}".

The fourth line contains an integer representing the count of special characters in the format "Special characters: {special characters count}".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: Hello123

Output: Uppercase letters: 1
Lowercase letters: 4
Digits: 3
Special characters: 0

Answer

```
# You are using Python
def analyze_log_entry(log_entry):

    uppercase_count = 0
    lowercase_count = 0
    digits_count = 0
    special_count = 0

    for char in log_entry:
        if 'A' <= char <= 'Z':
            uppercase_count += 1
        elif 'a' <= char <= 'z':
            lowercase_count += 1
        elif '0' <= char <= '9':
            digits_count += 1
        else:
            special_count += 1

    return uppercase_count, lowercase_count, digits_count, special_count
```

```
if __name__ == "__main__":
    log_entry = input()
    uppercase, lowercase, digits, special = analyze_log_entry(log_entry)
    print(f"Uppercase letters: {uppercase}")
    print(f"Lowercase letters: {lowercase}")
    print(f"Digits: {digits}")
    print(f"Special characters: {special}")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Accept an unsorted list of length n with both positive and negative integers, including 0. The task is to find the smallest positive number missing from the array. Assume the n value is always greater than zero.

Input Format

The first line consists of n, which means the number of elements in the array.

The second line consists of the values in the list as space-separated integers.

Output Format

The output displays the smallest positive number, which is missing from the array.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
-5 2 0 -1 -10 2
Output: 1

Answer

```
# You are using Python
def find_smallest_missing_positive(nums):
```

```
    seen = set()
    for num in nums:
        if num > 0:
            seen.add(num)

    i = 1
    while True:
        if i not in seen:
            return i
        i += 1

if __name__ == "__main__":
    n = int(input())
    nums_str = input().split()
    nums = [int(x) for x in nums_str]

    result = find_smallest_missing_positive(nums)
    print(result)
```

Status : Correct **Marks : 10/10**

3. Problem Statement

You are tasked with writing a program that takes n integers as input from the user and stores them in a list. After this, you need to transform the list according to the following rules:

The element at index 0 should be replaced with 0.For elements at even indices (excluding index 0), replace the element with its cube.For elements at odd indices, replace the element with its square.

Additionally, you should sort the list in ascending order before applying these transformations.

Input Format

The first line of input represents the size of the list, N.

The elements of the list are represented by the next N lines.

Output Format

The first line of output displays "Original List: " followed by the original list.

The second line displays "Replaced List: " followed by the replacement list as per the given condition.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
5
1
2
3
4
Output: Original List: [1, 2, 3, 4, 5]
Replaced List: [0, 4, 27, 16, 125]

Answer

```
# You are using Python
def transform_list():

    n = int(input())
    original_list = []
    for _ in range(n):
        num = int(input())
        original_list.append(num)

    sorted_list = sorted(original_list)
    replaced_list = [0] * n

    for i in range(n):
        if i == 0:
            replaced_list[i] = 0
        elif i % 2 == 0:
            replaced_list[i] = sorted_list[i] ** 3
        else:
            replaced_list[i] = sorted_list[i] ** 2

    print("Original List:", sorted_list)
```

```
    print("Replaced List:", replaced_list)

if __name__ == "__main__":
    transform_list()
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Neha is learning string operations in Python and wants to practice using built-in functions. She is given a string A, and her task is to:

Find the length of the string using a built-in function.Copy the content of A into another string B using built-in functionality.

Help Neha implement a program that efficiently performs these operations.

Input Format

The input consists of a single line containing the string A (without spaces).

Output Format

The first line of output prints the length of the given string.

The second line prints the copied string without an extra newline at the end.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: technology-23
Output: Length of the string: 13
Copied string: technology-23

Answer

```
# You are using Python
def string_operations():

    string_a = input()
```

```
    length_a = len(string_a)
    string_b = string_a[:]

    print(f"Length of the string: {length_a}")
    print(f"Copied string: {string_b}", end="")

if __name__ == "__main__":
    string_operations()
```

Status : Correct **Marks : 10/10**

5. Problem Statement

Gowri was doing her homework. She needed to write a paragraph about modern history. During that time, she noticed that some words were repeated repeatedly. She started counting the number of times a particular word was repeated.

Your task is to help Gowri to write a program to get a string from the user. Count the number of times a word is repeated in the string.

Note: Case-sensitive

Input Format

The first line of input consists of a string, str1.

The second line consists of a single word that needs to be counted, str2.

Output Format

The output displays the number of times the given word is in the string.

If the second string str2 is not present in the first string str1, it prints 0.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: I felt happy because I saw the others were happy and because I knew I should feel happy
happy
Output: 3

Answer

```
# You are using Python
def count_word_occurrence():

    str1 = input()
    str2 = input()

    count = str1.count(str2)
    print(count)

if __name__ == "__main__":
    count_word_occurrence()
```

Status : Correct ***Marks : 10/10***

6. Problem Statement

Kyara is analyzing a series of measurements taken over time. She needs to identify all the "peaks" in this list of integers.

A peak is defined as an element that is greater than its immediate neighbors. Boundary elements are considered peaks if they are greater than their single neighbor.

Your task is to find and list all such peaks using list comprehension.

Example

Input

1 3 2 4 1 5 7 6 10 2 8

Output

Peaks: [3, 4, 7, 10, 8]

Explanation

3 is a peak because it's greater than 1 and 2.

4 is a peak because it's greater than 2 and 1.

7 is a peak because it's greater than 5 and 6.

10 is a peak because it's greater than 6 and 2.

8 is a peak because it is an boundary element and it is greater than 2.

Input Format

The input consists of several integers separated by spaces, representing the measurements.

Output Format

The output displays "Peaks: " followed by a list of integers, representing the peak elements in the list.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1 3 2 4 1 5 7 6 10 2 8

Output: Peaks: [3, 4, 7, 10, 8]

Answer

```
# You are using Python
def find_peaks():

    measurements_str = input()
    measurements = [int(x) for x in measurements_str.split()]
    n = len(measurements)

    peaks = [
        measurements[i]
        for i in range(n)
        if (i == 0 and n > 1 and measurements[i] > measurements[i + 1]) or
           (i == 0 and n == 1) or
```

```
        (i == n - 1 and n > 1 and measurements[i] > measurements[i - 1]) or
        (0 < i < n - 1 and measurements[i] > measurements[i - 1] and
measurements[i] > measurements[i + 1])
    ]

    print("Peaks:", peaks)

if __name__ == "__main__":
    find_peaks()
```

Status : Correct ***Marks : 10/10***