Great Learning
POWER AHEAD

Python

1. Problem

Given the participants' score sheet for your University Sports Day, you are required to find the runner-up score. You are given n scores. Store them in a list and find the score of the runner-up.

Input Format

The first line contains ***n***. The second line contains an array ***A[]*** of n integers each separated by a space.

Constraints

- $2 \leq n \leq 10$
- $-100 \leq A[i] \leq 100$

Output Format

Print the runner-up score.

Sample Input

```
5
2 3 6 6 5
```

Sample Output

```
5
```

Solution:

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    print(sorted(list(set(arr)))[-2])
```

2. Problem

The provided code stub reads and integer, ***n***, from STDIN. For all non-negative integers $i < n$, print i^2 .

Example

$n = 3$

The list of non-negative integers that are less than $n = 3$ is [0, 1, 2]. Print the square of each number on a separate line.

```
0
1
4
```

Input Format

The first and the only line contains the integer, ***n***.

Constraints

$1 \leq n \leq 20$

Output Format

Print ***n*** lines, one corresponding to each ***i***.

Sample Input

```
5
```

Sample Output

```
0
1
4
9
16
```

Solution:

```
if __name__ == '__main__':
    n = int(input())
    for i in range(0, n):
        print(i ** 2)
```

3. Problem

An extra day is added to the calendar almost every four years as February 29, and the day is called a leap day. It corrects the calendar for the fact that our planet takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:

- The year can be evenly divided by 4, is a leap year, unless:
- The year can be evenly divided by 100, it is NOT a leap year, unless:
- The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.

Task

Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False.

Note that the code stub provided reads from STDIN and passes arguments to the is_leap function. It is only necessary to complete the is_leap function.

Input Format

Read ***year***, the year to test.

Constraints

1900 <= *year* <= 10^5

Output Format

The function must return a Boolean value (True/False). Output is handled by the provided code stub.

Sample Input

```
1990
```

Sample Output

```
False
```

Solution:

```
def is_leap(year):
    leap = False

    # Write your logic here
    if (year % 400 == 0):
        return True
    if (year % 100 == 0):
        return leap
    if (year % 4 == 0):
        return True
    else:
        return False

    return leap

year = int(input())
print(is_leap(year))
```

4. Problem

Let's learn about list comprehensions! You are given three integers ***x***, ***y*** and ***z*** representing the dimensions of a cuboid along with an integer ***n***. Print a list of all possible coordinates given by (***i***, ***j***, ***k***) on a 3D grid where the sum of ***i*** + ***j*** + ***k*** is not equal to ***n***. Here, **0** <= ***i*** <= ***x*****; 0** <= ***j*** <= ***y*****; 0** <= ***k*** <= ***z***. Please use list comprehensions rather than multiple loops, as a learning exercise.

Example

x = 1
y = 1
z = 2
n = 3

All permutations of [***i***, ***j***, ***k***] are:
[[0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 1, 0], [0, 1, 1], [0, 1, 2], [1, 0, 0], [1, 0, 1], [1, 0, 2], [1, 1, 0], [1, 1, 1], [1, 1, 2]**]**.

Print an array of the elements that do not sum to ***n = 3***.
[[0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 2]**]**

Input Format

Four integers ***x***, ***y***, ***z*** and ***n***, each on a separate line.

Constraints

Print the list in lexicographic increasing order.

Sample Input

```
1
1
1
2
```

Sample Output

```
[[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1]]
```

Solution:

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    output = [];
    abc = [];
    for X in range(x+1):
        for Y in range(y+1):
            for Z in range(z+1):
                if X+Y+Z != n:
                    abc = [X,Y,Z];
                    output.append(abc);
print(output);
```

5. Problem

In this challenge, the user enters a string and a substring. You have to print the number of times that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

NOTE: String letters are case-sensitive.

Input Format

The first line of input contains the original string. The next line contains the substring.

Constraints

1 <= len(string) <= 200
Each character in the string is an ascii character.

Output Format

Output the integer number indicating the total number of occurrences of the substring in the original string.

Sample Input

```
ABCDCDC
CDC
```

Sample Output

```
2
```

Solution:

```
def count_substring(string, sub_string):
    count=0
    for i in range(0, len(string)-len(sub_string)+1):
        if string[i] == sub_string[0]:
            flag=1
            for j in range (0, len(sub_string)):
                if string[i+j] != sub_string[j]:
                    flag=0
                    break
            if flag==1:
                count += 1
    return count
```

```
if __name__ == '__main__':
    string = input().strip()
    sub_string = input().strip()

    count = count_substring(string, sub_string)
    print(count)
```