**Q.6 :** The included code stub will read an integer, n, from STDIN.

Without using any string methods, try to print the following:

123…n

Note that "…" represents the consecutive values in between.

**Example**

**n = 5**  
Print the string 12345.

**Input Format**

The first line contains an integer n.

**Constraints**

**1 <= n <= 150**

**Output Format**

Print the list of integers from 1 through n as a string, without spaces.

**Answer :** if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

for i in range(1, n + 1):

    print(i, end = "")

**Output :** Input (stdin)

* **3**

Your Output (stdout)

* **123**

Expected Output

* **123**

**Q.7 :** 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 . 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.

**Answer :** if \_\_name\_\_ == '\_\_main\_\_':

    x = int(input())

    y = int(input())

    z = int(input())

    n = int(input())

    print(list([i, j, k]

        for i in range(x+1)

        for j in range(y+1)

        for k in range(z+1)

        if i + j + k != n))

**Output :** Input (stdin)

* **1**
* **1**
* **1**
* **2**

Your Output (stdout)

* **[[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1]]**

Expected Output

* **[[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1]]**

**Q.8 :** 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 <= n <= 10
* -100 <= A[ i ] <= 100

**Output Format**

Print the runner-up score.

**Answer :** if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    scores = list(map(int, input().split()))

    runner\_up = sorted(set(scores), reverse = True)[1]

    print(runner\_up)

**Output :** Input (stdin)

* **5**
* **2 3 6 6 5**

Your Output (stdout)

* **5**

Expected Output

* **5**

**Q.9 :** Given the names and grades for each student in a class of N students, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.

**Note:** If there are multiple students with the second lowest grade, order their names alphabetically and print each name on a new line.

**Example**  
records = [[“chi”, 20.0], [“beta”, 50.0], [“alpha”, 50.0]]

The ordered list of scores is [20.0, 50.0], so the second lowest score is 50.0. There are two students with that score: [“beta”, “alpha”]. Ordered alphabetically, the names are printed as:

alpha

beta

**Input Format**

The first line contains an integer, N, the number of students.  
The 2N subsequent lines describe each student over 2 lines.  
- The first line contains a student's name.  
- The second line contains their grade.

**Constraints**

* 2 <= N <= 5
* There will always be one or more students having the second lowest grade.

**Output Format**

Print the name(s) of any student(s) having the second lowest grade in. If there are multiple students, order their names alphabetically and print each one on a new line.

**Answer :** if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    students = []

    for \_ in range(n):

        name = input().strip()

        grade = float(input().strip())

        students.append([name, grade])

    students.sort()

    grades = sorted(set([student[1] for student in students]))

    second\_lowest\_grade = grades[1]

    second\_lowest\_students = []

    for student in students:

        if student[1] == second\_lowest\_grade:

            second\_lowest\_students.append(student[0])

    second\_lowest\_students.sort()

    for name in second\_lowest\_students:

        print(name)

**Output :** Input (stdin)

* **5**
* **Harry**
* **37.21**
* **Berry**
* **37.21**
* **Tina**
* **37.2**
* **Akriti**
* **41**
* **Harsh**
* **39**

Your Output (stdout)

* **Berry**
* **Harry**

Expected Output

* **Berry**
* **Harry**

**Q.10 :** The provided code stub will read in a dictionary containing key/value pairs of name:[marks] for a list of students. Print the average of the marks array for the student name provided, showing 2 places after the decimal.

**Example**  
marks key : value pairs are  
‘alpha’ : [20, 30, 40]  
‘beta’ : [30, 50 70]  
query\_name = ‘beta’

The **query\_name** is 'beta'. beta's average score is (30 + 50 + 70) / 3 = 50.0.

**Input Format**

The first line contains the integer , the number of students' records. The next n lines contain the names and marks obtained by a student, each value separated by a space. The final line contains **query\_name**, the name of a student to query.

**Constraints**

* 2 <= n <= 10
* 0 <= marks[i] <= 100
* Length of marks arrays = 3

**Output Format**

Print one line: The average of the marks obtained by the particular student correct to 2 decimal places.

**Answer :** if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    student\_marks = {}

    for \_ in range(n):

        name, \*line = input().split()

        scores = list(map(float, line))

        student\_marks[name] = scores

    query\_name = input()

    average\_marks = sum(student\_marks[query\_name]) / len(student\_marks[query\_name])

    print(f"{average\_marks:.2f}")

**Output :** Input (stdin)

* **3**
* **Krishna 67 68 69**
* **Arjun 70 98 63**
* **Malika 52 56 60**
* **Malika**

Your Output (stdout)

* **56.00**

Expected Output

* **56.00**