**1. Number Pyramid**

Write a function that receives an integer **n** and prints a pyramid of numbers, as in the examples:

**Input → Output**

7 1

2 3

4 5 6

7

12 1

2 3

4 5 6

7 8 9 10

11 12

15 1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

**Hints:**

1. Use two nested **for** loops:
   * The outer loop defines how many rows to print.
   * The inner loop defines how many numbers to print on the current row.
2. Keep a counter for the current number.
3. Use a variable to track whether you should exit both loops when you reach **n**.
4. Print the numbers row by row.

**2. Equal Sums Even Odd Position**

Write a function that receives two six-digit integers in the range **[100000…300000]**.  
The first number will always be smaller than the second one.

Print all numbers in the range for which the **sum of digits at even positions** equals the **sum of digits at odd positions**.  
If no numbers meet the condition → print nothing.

**Example Input → Output**

100000, 100050 → 100001 100012 100023 100034 100045

123456, 124000 → 123464 123475 123486 123497 … 123992

299900, 300000 → 299970 299981 299992

100115, 100120 → (no output)

**Hints:**

1. Use a **for loop** to iterate through all numbers in the range.
2. Convert each number to a string to access its digits.
3. Use an additional **for loop** to go through each digit.
4. Keep two sums: one for even positions, one for odd positions.
5. Compare them → if equal → print the number.

**3. Sums Prime and Non-Prime Numbers**

Write a function that reads integers (range: **[-2,147,483,648 … 2,147,483,647]**) until the command "stop".

* Find the **sum of all prime numbers** and the **sum of all non-prime numbers**.
* Negative numbers are not prime → print "Number is negative." and skip them.

**Output format:**

Sum of all prime numbers is: {sum of primes}

Sum of all non prime numbers is: {sum of non-primes}

**Example Input → Output**

["3", "9", "0", "7", "19", "4", "stop"]

→

Sum of all prime numbers is: 29

Sum of all non prime numbers is: 13

**4. Train the Trainers**

You are helping a jury evaluate presentations.

* First input: **n** → number of jury members [1…20].
* Then → presentation name.
* Then → n grades per presentation [2.00…6.00].

After each presentation print:

{presentation name} - {average grade}.

When "Finish" is entered → print final result:

Student's final assessment is {average of all grades}.

**All grades formatted to 2 decimal places.**

**5. Special Numbers**

Write a program that reads an integer **N** [1…600000] and generates all **“special numbers”** in the range **[1111…9999]**.

A number is **special** if:

* **N is divisible by each of its digits without remainder**.

**Example Input → Output**

N = 3 → 1111 1113 1131 1133 1311 … 3333

N = 11 → 1111

N = 16 → many numbers (e.g., 2418 because 16 % 2=0, 16 % 4=0, 16 % 1=0, 16 % 8=0)

**6. Cinema Tickets**

Write a program that calculates the percentage of tickets sold by type:

* student
* standard
* kid

Also calculate how full the cinema hall is for each projection.

**Input:**

* First line: movie name (or "Finish" to end).
* Second line: free seats (1…100).
* Then → tickets ("student", "standard", "kid") until seats are full or "End".

**Output:**

* After each movie:
* {movie name} - {percentage full}% full.
* After "Finish":
* Total tickets: {total}
* {percentage}% student tickets.
* {percentage}% standard tickets.
* {percentage}% kids tickets.

**Example Input → Output**

["Taxi","10","standard","kid","student","student","standard","standard","End",

"Scary Movie","6","student","student","student","student","student","student",

"Finish"]

→

Taxi - 60.00% full.

Scary Movie - 100.00% full.

Total tickets: 12

66.67% student tickets.

25.00% standard tickets.

8.33% kids tickets.