1. Eventually even -> take a number from user and print even if the sum of its eventual single digit sum is even

   1851 = 1+ 8 + 5 +1 = 15
   15 is not single digit hence
   1 + 5 = 6
   6 is single digit and 6 is even hence number is eventually even

   190 = 1 + 9 + 0 = 10
   10 = 1 + 0 = 1
   1 is not even

   Def even_odd()
   Def sum_digit()
   Def check_single()
   Def eventual_even()

2. Pointer movement

   Take the number as input from the user, and also take a threshold value. You have two pointers (i,j), one at extreme end of the number. At each iteration step, check whether the digit at the ith index is greater than the threshold.
   If the digit at the ith index is greater than the threshold, then move i to the right and j to the left. Else move only i to the right. The iteration continues until i run out of digits. During this, keep a record of the number of times j moved and print that

   For example, with threshold t=7 and Number = 123
   I =0, j=2
   1<7 so move I = 1
   I = 1, j =2
   2 < 7 so again move I = 2
   I = 2, j=2
   3<7 so again I moves I = 3, j=2 > loop breaks, j shifted 0 times.

   1994
   i=0, j=3
   1<7 so move I = 1
   i=1, j=3
   9>7 so move i and j
   I = 2 and j = 2
   9>7 so move i and j
   I = 3, j=1
   4<7 so move only i
   I =4 and j =1 —> loop breaks j shifted 2 times.

3. Write a program to computer ${}^{n}C_{r}$ without using the built-in factorial and combination function. Your program should have the following functions:

   ```
   def computeFactorial():
        #do something
   def computeCombination():
        #do something.
   ```

4. Write a program to count the number of digits in a given number using recursion.

   1 → 1
   15 → 2
   345→ 3
   1879965 -> 7

5. Write a program to print the Hailstone series of a number. Attempt this with and without recursion. See what the base conditions will be.
   Details about hailstone series can be found here: https://chortle.ccsu.edu/java5/Notes/chap73/ch73_14.html and https://www.geeksforgeeks.org/hailstone-numbers/

6. Write a program to print the Fibonacci series of a number. Attempt this with and without recursion. See what the base conditions will be.
   Details about the Fibonacci series can be found here: https://www.cuemath.com/numbers/fibonacci-series/ and https://www.geeksforgeeks.org/python-program-to-print-the-fibonacci-sequence/

7. Write a program to recursively print the square of the positive number>0 based on the formulation:

   Square(1) = 1

   Square(N) = square(N-1) + 2N -1