1. **Scenario:** You are developing a banking application that categorizes transactions based on the amount entered.
   Write logic to determine whether the amount is positive, negative, or zero.

SOLUTION:

STEP 1: GET AMOUNT FROM USER.

STEP 2: CHECK THE CONDITION. IF(AMOUNT>0) PRINT (“POSITIVE TRANSACTION”)

STEP 3: elif AMOUNT <0 print(“NEGATIVE TRANSACTION”)

STEP 4: else print(“ZERO TRANSACTION”)

2. **Scenario:** A digital locker requires users to enter a numerical passcode. As part of a security feature, the system checks the sum of the digits of the passcode.
   Write logic to compute the sum of the digits of a given number.
   SOLUTION:
   1. READ A NUMBER FROM USER
   2. CONVERT THE USER INPUT INTO INTEGER
   3. IF USER ENTERED INPUT LIKE -123, THIS CONVERT INTO INTEGER (For eg: -123 convert into integer using abs(int(number)
   4. Initialize the variable digital_sum=0
   5. Each number adding into digital_sum
   6. Finally we get answer
3. **Scenario:** A mobile payment app uses a simple checksum validation where reversing a transaction ID helps detect fraud.
   Write logic to take a number and return its reverse.
   SOLUTION:
   1. Get user input N
   2. Set variable reverse=0
   3. Get the last digit of N using N%10
   4. Multiply reverse by 10 and **add the last digit**
   5. Output the reverse

4. **Scenario:** In a secure login system, certain features are enabled only for users with prime-numbered user IDs.
   Write logic to check if a given number is prime.
   SOLUTION:
   1. Get the number from user

2. Check that number <=1 , print this not prime
3. If number > 2 , print prime
4. Loop from 2 to the square root of the number:
   If the number is divisible by any of these values, print "Not Prime" and exit.
5. If no divisors are found, print "Prime".

5. **Scenario:** A scientist is working on permutations and needs to calculate the factorial of numbers frequently.
   Write logic to find the factorial of a given number using recursion.
   SOLUTION:
   1. Get the number from user n.
   2. If n is 0 or 1, then return 1, Because 0! = 1 and 1! = 1
   3. If n>1 then Return n × factorial(n - 1)
   4. Call the same function again with n – 1
   5. Multiply the result with n
   6.

6. **Scenario:** A unique lottery system assigns ticket numbers where only Armstrong numbers win the jackpot.
   Write logic to check whether a given number is an Armstrong number.
   SOLUTION:
   1. Armstrong number means The **sum of its digits raised to the power of the number of digits** equals the number itself. for eg: n=123, count is 3. So 1 power 3=1, 2 power 3=8, 3 power 3 =27. Then add 1+8+27=36. 123 not equal to 36 so this is not a Armstrong number
   2. Get the input from user n
   3. Then count the number of digit of n
   4. Initialize variable sum=0.
   5. Sum=sum+(digit power total count digit(n))
   6. If sum== original user input
   7. Print Armstrong number otherwise not, a Armstrong number

7. **Scenario:** A password manager needs to strengthen weak passwords by swapping the first and last characters of user-generated passwords.
   Write logic to perform this operation on a given string.
   SOLUTION:
   1. **Get the user input using string**

2. **Check length , if <2 characters , return as is**
3. Extract the first, last, middle character
4. **Rebuild the string newpassword=last+middle+first**
5. **Display the newpassword**

8. **Scenario:** A low-level networking application requires decimal numbers to be converted into binary format before transmission.
   Write logic to convert a given decimal number into its binary equivalent.
   SOLUTION:

   1. the input decimal number.
   2. Initialize an empty string for binary representation.
   3. While the number is greater than 0:
   4. Divide the number by 2 and store the remainder.
   5. Add the remainder to the binary string
   6. Update the number by dividing it by 2
   7. Reverse the binary string then print

9. **Scenario:** A text-processing tool helps summarize articles by identifying the most significant words.
   Write logic to find the longest word in a sentence.
   SOLUTION:

   1. Input the sentence
   2. Split the sentence into words
   3. Initialize a variable to track the longest word . “longest”
   4. Loop through each word
   5. If the current word is longer than the stored longest word, update the longest word
   6. Print the longest word

10 . Scenario: A plagiarism detection tool compares words from different documents and checks if they are anagrams (same characters but different order).
Write logic to check whether two given strings are anagrams.
SOLUTION:

1. Read the two input strings.
2. Remove spaces and convert both strings to lowercase.
3. Sort the characters of both strings.
4. If the sorted versions of both strings are identical, print "Anagram".
5. Else, print "Not an Anagram".

1.