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.

   a. Input he number through int(input)

   b. If number > 0 -> Print amount positive

   c. Elif <0 Print Amount Negative

   d. Else Print Zero

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.
   a. Input the number
   b. Convert the number into digits by type casting using str, while iterating through a for loop
   c. Convert the value to number while calculating the sum
   d. Return the sum

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.
   a. **Input the number**
   b. **Convert the number into string and reverse the number using reverse indexing**
   c. **Return the number back to integer**

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.
   a. **Input the number**
   b. **Write a function to determine if the number is prime**
      i. **If the number is divisible only by itself, then declare it as prime number, isPrime = true, else false**

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.
   a. **Recursion –** Procedure where the function is called by itself to solve a problem
   b. **Enter the number, pass the number as an argument inside the function, multiply the number with previous number(n-1)**

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.
   a. A number is called an armstrong number, if the sum of the digits to the powered value of the no of digits is equal to the same number
      i. For example 153 is armstrong number, where 1**3 + 5**3 + 3**3 = 1 + 125 + 27 = 153
      ii. Check the number of digits and store in a variable. Use the number to be powered with each digit of the number and sum it.
      iii. Return the value as true if the actual number and sum of the nth power of the digits are the same.
      iv.

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.
   a. We can try indexing with this.
   b. Create a variable strongPassword and try negative indexing to replace the 1st with last “password[-1]+password[1:-1]+password[0]

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.

    a. Can use bin() function to convert the decimal to binary
    b. Do splicing and remove the 1st 2 digits to get the actual binary number.

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.
    a. Input a sentence/variable that has a sentence
    b. Create a separate function to identify the word length
    c. Pass the sentence as a parameter
    d. Using split, let us break down the sentence into a list with multiple words
    e. Use for loop and measure the length of the word using len function

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.

    a. Remove all spaces from both words and convert them to lowercase.

    b. If the strings have different lengths, return false

    c. Sort the characters in both clean strings[sorted function].

    d. Compare the sorted versions:

    e. If they are the same, the strings are anagrams.