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.

**Answer:** Analysis Transaction Amount Categories. ***Positive Amount***: When the amount is greater than zero (e.g., +100), it's considered a credit or deposit. ***Negative Amount***: When the amount is less than zero (e.g., -50), it's considered a debit or withdrawal. ***Zero Amount***: When the amount is exactly zero, there's no change or transaction.

**Code:** Amounts = (4555, -3300, 0, 400, 3000, -20, 0, 3000, -400, 0)

Positive=[]

Negative=[]

Zero=[]

for amt in Amounts:

if amt>0:

print("Positive Amount=",amt)

Positive.append(amt)

elif amt<0:

print("Negative Amount=",amt)

Negative.append(amt)

else:

print("Zero Amount=",amt)

Zero.append(amt)

1. **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.

**Answer:** Convert the number to a string to access individual digits. Iterate through each character (digit) in the string. Convert each character back to an integer and add it to the sum.

**Code:** number = input("Enter the passcode: ")

sum\_of\_digit = 0

for digit in number:

sum\_of\_digit += int(digit)

print(sum\_of\_digit)

1. **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.

**Answer:** Converts the number to a string. Reverses the string using slicing. Converts the reversed string back to an integer.

**Code**: def Rev\_number(n):

Rev\_number=int(str(n)[::-1])

return Rev\_number

print(Rev\_number(23456))

1. **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.

**Answer:** Start with a number we want to check — let’s call it n. If n is less than or equal to 1, it is not a prime number. Now, check every number from 2 up to n - 1. If any of them divides n exactly (no remainder), then n is not a prime number. If none of them divide n exactly, then n is a prime number.

**Code**: def is\_prime(n):

if n <= 1:

return False

for i in range(2, n):

if n % i == 0:

return False

return True

num = 13

if is\_prime(num):

print(num, "is a prime number")

else:

print(num, "is not a prime number")

1. **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.

**Answer:** Start with a number (n) we want the factorial of. If n is 0 or 1, then the factorial is 1. This is the base case (Because 0! = 1! = 1 by definition). Otherwise, multiply n by the factorial of n - 1. This is the recursive step.

**Code:** def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

print(factorial(6))

1. **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.

**Answer:** Take the number n. Count how many digits it has. For each digit: Raise it to the power of the number of digits. Add the result to a total sum. If the total sum equals the original number → Armstrong, Else → Not Armstrong.

**Code:** def is\_armstrong(n):

num\_str = str(n)

num\_digits = len(num\_str)

total = 0

for digit in num\_str:

total += int(digit) \*\* num\_digits

return total == n

num = 1634

if is\_armstrong(num):

print(num, "is an Armstrong number")

else:

print(num, "is not an Armstrong number")

1. **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.

**Answer:** Take the input string (password). If the string has less than 2 characters, return it as-is. Otherwise: Store the first character. Store the last character. Take the middle part (excluding first and last). Combine in this order: last character + middle part + first character. Return the new string.

**Code:** def swap\_first\_last(password):

if len(password) < 2:

return password

return password[-1] + password[1:-1] + password[0]

print(swap\_first\_last("passw"))

1. **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.

**Answer:** Read the input decimal number. Use the built-in bin() function to convert it to binary. Remove the '0b' prefix from the result. Return the binary digits as a string. Print the result.

**Code:** def decimal\_to\_binary(n):

return bin(n)[2:]

print(decimal\_to\_binary(34))

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

**Answer:** Take a sentence as input. Split the sentence into words.Compare the lengths of all the words. Identify the word with the maximum number of characters. Return or print that word.

**Code:** def find\_longest\_word(sentence):

words=sentence.split()

longest=max(words,key=len)

return longest

sentence="This logic can be readily incorporated into a text processing tool for summarizing articles"

print("Longest word:",find\_longest\_word(sentence))

1. **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.

**Answer:** Take two strings as input. Sort the characters of both strings. Compare the sorted lists. Return the result (True or False).

**Code:** def anagrams(str1,str2):

return sorted(str1) == sorted(str2)

print(anagrams("hello","world"))

print(anagrams("listen","silent"))

**Ramishahope Artificial Intelligence Pvt Ltd**

**36, Old Anandas, SG Arcade, Marudhamalai Main Road, Vadavalli, Coimbatore -641041.**

**+91 6385383227 |** [**www.hopelearning.net**](http://www.hopelearning.net/) **|** [**mdaravind@hopelearning.net**](mailto:mdaravind@hopelearning.net) **| 33AAMCR3722R1ZU**