**Task 1.**

Implement the function to find the combination of 4 digits in a row which gives the max multiplication. If object is not a string or there are no combinations found - return nil. If combination is found - return it's multiplication result.

E.g.

max\_multiplication('abc12345def') => 120  # 2\*3\*4\*5

max\_multiplication('a1b2c3d4e') => nil

**# Solution 1:**

#I have written the code in such a manner that, it will prompt the user to give input.

#Based on the input it will execute.

# Actually it will basically search for max consecutive 4 digits and do the operation.

def max\_multiplication(input\_string):

consecutive\_digits = "" # To store consecutive digits

result = 1 # Initialize the result to 1

stored\_numbers = [] # To store the individual numbers contributing to the product

for char in input\_string:

if char.isdigit():

consecutive\_digits += char

else:

if len(consecutive\_digits) >3:

# Sort the consecutive digits and keep the highest four

sorted\_digits = sorted(consecutive\_digits, reverse=True)

highest\_four = sorted\_digits[:4]

product = 1

for digit in highest\_four:

int\_digit = int(digit)

product \*= int\_digit

stored\_numbers.append(int\_digit) # Store individual number

result \*= product

consecutive\_digits = "" # Reset if a non-digit character is encountered

if result == 1:

return "nil", [] # Return "nil" and an empty list if no consecutive digits are found

else:

return result, stored\_numbers

# Get input from the user

user\_input = input("Enter a string: ")

result, stored\_numbers = max\_multiplication(user\_input+" ")

if result != "nil":

numbers\_str = ' \* '.join(str(num) for num in stored\_numbers)

print(f"Result is: {result} # {numbers\_str} = {result}")

else:

print("nil.")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Task 2.**

Implement the function to sort array of numbers by amount of '1' in its binary representation. Numbers with identical amount of '1's should be ordered by decimal representation.

E.g.

# 3 = 11, 7 = 111, 8 = 1000, 9 = 1001.

sort([3,7,8,9]) => [8,3,9,7]  # 1000, 11, 1001, 111

**#Solution 2:**

# Function to convert a decimal number to binary

def decimal\_to\_binary(decimal\_number):

binary\_representation = ""

if decimal\_number == 0:

binary\_representation = "0"

else:

while decimal\_number > 0:

remainder = decimal\_number % 2

binary\_representation = str(remainder) + binary\_representation

decimal\_number = decimal\_number // 2

return binary\_representation

# Function to get the count of 1s in a binary number

def count\_ones(binary\_string):

return binary\_string.count('1')

# Input the size of the array

array\_size = int(input("Enter the size of the array: "))

# Initialize an empty list to store the binary representations

binary\_array = []

# Input the decimal numbers and convert them to binary

for i in range(array\_size):

decimal\_number = int(input(f"Enter decimal number {i + 1}: "))

binary\_representation = decimal\_to\_binary(decimal\_number)

binary\_array.append(binary\_representation)

# Create a dictionary to store the counts

counts = {}

# Loop through the binary numbers and count '1's

for binary\_number in binary\_array:

count = count\_ones(binary\_number)

counts[binary\_number] = count

# Sort the binary numbers based on the counts and then the decimal values

def custom\_sort\_key(item):

binary\_string, decimal\_number = item

return (counts[binary\_string], decimal\_number)

sorted\_binary\_numbers = sorted(zip(binary\_array, [int(binary, 2) for binary in binary\_array]), key=custom\_sort\_key)

# Print the sorted binary numbers in decimal format

for \_, decimal\_number in sorted\_binary\_numbers:

print(decimal\_number)