Question 1: Count the Characters

Imagine you are developing a text analysis tool for a messaging app. One of the features you want to implement is the ability to count how often each character (letters, digits, spaces, and punctuation marks) appears in a given message.

Your task is to write a simple Python program that takes a message from the user and counts the frequency of each character, including letters, digits, spaces, and special characters.

Note : The program should take a user's message as input and output a dictionary with characters as keys and their respective frequencies as values.

Example 1:

Input:

google.com

Output:

{'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}

Explanation:

In the string "google.com", 'g' appears 2 times, 'o' 3 times, 'l' 1 time, 'e' 1 time, '.' 1 time, 'c' 1 time, and 'm' 1 time.

Example 2:

Input:

Data Scientist

Output:

{'d': 1, 'a': 2, 't': 3, ' ': 1, 's': 2, 'c': 1, 'i': 2, 'e': 1, 'n': 1}

Explanation:

In the string "Data Scientist", 'D' appears 1 time, 'a' 2 times, 't' 3 times, space ' ' appears 1 time, 'S' 1 time, 'c' 1 time, 'i' 2 times, 'e' 1 time, and 'n' 1 time.

Question 2: Race for the Armstrong Award

In a maths competition, the Armstrong award would be presented to the one who would first tell the Armstrong number among all given numbers. Thus, write a program for Sam in Python to help him won the award.

What is Armstrong Number?

It is a number which is equal to the sum of cube of its digits. For eg: 153, 370 etc.

Let's take 153 for an example

First calculate the cube of its each digits

1^3 = (1 * 1 * 1) = 1

5^3 = (5 * 5 * 5) = 125

3^3= (3 * 3 * 3) = 27

Now add the cube

1+125+27 = 153

It means 153 is an Armstrong Number.

Example 1:

Input:

153

Output:

Number is Armstrong

Explanation:

$1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$

Example 2:

Input:

450

Output:

Number is not Armstrong

Explanation:

$4^3 + 5^3 + 0^3 = 64 + 125 + 0 = 189$ (not equal to 450)

Example 3:

Input:

9474

Output:

Number is Armstrong

Explanation:

$9^4 + 4^4 + 7^4 + 4^4 = 6561 + 256 + 2401 + 256 = 9474$

Question 3: Identify Words with Prime Lengths

Imagine you're helping a teacher grade a language test. In one of the tasks, students were asked to identify and list all the words from a passage that have a length equal to a prime number. Your task is to write a Python program that does this automatically.

The program will take a sentence or passage as input and output all the words whose length is a prime number.

Note: A prime number is a number greater than 1 that has no divisors other than 1 and itself.

Example 1:

Input:

The quick brown fox jumps over the lazy dog

Output:

The quick brown fox jumps the

Explanation:

Word Lengths "The" → 3 letters (prime) "quick" → 5 letters (prime) "brown" → 5 letters (prime) "fox" → 3 letters (prime) "jumps" → 5 letters (prime) "over" → 4 letters (not prime) "the" → 3 letters (prime) "lazy" → 4 letters (not prime) "dog" → 3 letters (prime)

Prime-length words: The quick brown fox jumps the

Example 2:

Input:

Welcome to the wonderful world of coding

Output:

Welcome to the world of

Explanation:

Word Lengths "Welcome" → 7 letters (prime) "to" → 2 letters (prime) "the" → 3 letters (prime) "wonderful" → 9 letters (not prime) "world" → 5 letters (prime) "of" → 2 letters (prime) "coding" → 6 letters (not prime)

Prime-length words: "Welcome", "to", "the", "world", "of" all have prime lengths.

Example 3:

Input:

Hello, world! Python is amazing and fun

Output:

Hello world is amazing and fun

Explanation:

Word lengths (ignoring punctuation): "Hello" → 5 letters (prime) "world" → 5 letters (prime) "Python" → 6 letters (not prime) "is" → 2 letters (prime) "amazing" → 7 letters (prime) "and" → 3 letters (prime) "fun" → 3 letters (prime)

Prime-length words: "Hello", "world", "is", "amazing", "and", "fun" all have prime lengths.