**Artificial Intelligent (Lab)**

**Task # 04**

**A purple circle with dots and lines

AI-generated content may be incorrect.**

**Submitted To: Sir Rasikh Ali**

**Submitted By: Shumaila Maryam**

**Roll no: SU92-BSDSM-F24-062**

**Submitted On: 18-September-25**

**Section: BSDS-3A**

**Department of Software Engineering, Superior University, Lahore**

**Question # 01:**

**Removing Punctuation from Text**

**Introduction:**

This small Python program removes punctuation from any text. It goes through each character in the text and removes symbols like [“! \"#$%&'()\*+,-./:;<=>?@[\\]^\_`{|}~"]. The result is a clean version of the text without extra symbols.

**Why I Made This:**

The goal of this project was to:

* Learn how to clean text using Python.
* Practice using classes and object-oriented programming (OOP).
* Understand how to check and remove unwanted characters step by step.
* See how small programs can make text ready for further use, like in chat apps or data analysis.

**How It Works:**

1. **Starting Text**  
   The program starts with a given text:
2. **Memory (Internal Storage)**  
   The program keeps two things:
   * The original text (self.text)
   * A list of all punctuation symbols (self.check\_char)
3. **Checking Each Character**  
   The program goes through each character in the text:
   * If the character is not a punctuation, it is added to a new string.
   * If it is a punctuation, it is ignored.

**Summary:**

This program shows a simple way to clean text by removing punctuation. Using a class makes the program organized and easy to reuse. It’s useful for preparing text for analysis, NLP, or just for making it cleaner and easier to read.

**Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Question # 02:**

**Sorting Text by Characters and Words**

**Introduction:**

This Python program sorts text in two ways:

1. By characters: It rearranges all characters in the text alphabetically.
2. By words: It rearranges the words in the text alphabetically.

This is useful for understanding how strings and words can be manipulated in Python, and it shows how sorting works for both letters and words.

**Why I Made This:**

1. To learn how to work with strings and lists in Python.
2. To practice nested loops and sorting logic.
3. To understand the difference between sorting characters and sorting words.
4. To make a simple, human-friendly text sorting program.

**How It Works:**

**Step 1: Sorting Characters**

1. The text is converted into a list of characters.
2. Nested loops compare each character with others using its ASCII value (ord()).
3. Characters are swapped if they are out of order.
4. The sorted list of characters is joined back into a string.
5. **Step 2 : Sorting Words**
6. The text is split into a list of words using. split ().
7. Nested loops compare words alphabetically using .lower() to ignore case.
8. Words are swapped if they are out of order.
9. The sorted words are joined back into a string with spaces.

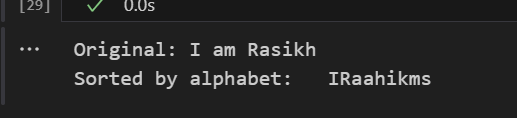
**Summary:**

**This program demonstrates:**

1. How to sort characters in a string.
2. How to sort words in a string.
3. The difference between letter-level sorting and word-level sorting.
4. A simple way to manipulate strings using loops, conditionals, and lists.

It is a good example for learning string manipulation, sorting algorithms, and Python programming basics

**Output:**

****

**A black screen with white text

AI-generated content may be incorrect.**

**Question # 03:**

**Credit Card Validation Using Luhn Algorithm**

**Introduction:**

This Python program uses the Luhn Algorithm to check if a credit card number is valid. Instead of just blindly checking the digits, it performs step-by-step operations like reversing, doubling certain digits, and summing them up to determine validity.

The program also shows the check digit as x and keeps track of all changes in a readable way.

**Why I Made This:**

1. To learn how the Luhn Algorithm works for card validation.
2. To practice object-oriented programming (OOP) in Python.
3. To understand step-by-step manipulation of numbers in a list.
4. To create a program that prints each stage of the validation clearly.

**How It Works**

**Step 1: Save the Check Digit**

1. The last digit of the card number is saved as x.
2. This digit is used later to calculate the final sum.

**Step 2: Reverse the Digits**

1. All digits except the last one are reversed.
2. This helps in proper doubling of every second digit from the right side.

**Step 3: Double Even-Indexed Digits**

1. Starting from the left (after reversal), every second digit is doubled.
2. If the doubled value is greater than 9, 9 is subtracted from it.

**Step 4: Sum and Validate**

* All digits including the check digit x are summed.
* If the total sum is divisible by 10, the card is VALID.
* Otherwise, it is INVALID.

**Output:**

**A black background with white letters

AI-generated content may be incorrect.**