**Task num 01:**

**Credit Card Validation using a Custom Luhn Algorithm**

**Overview**

This script implements a custom version of the Luhn algorithm to validate credit card numbers. The Luhn algorithm is commonly used to validate a variety of identification numbers, such as credit card numbers, social security numbers, and more.

The custom implementation works by checking the input credit card number based on a series of calculations involving digit manipulation and checksum verification.

**Function Documentation**

**1. custom\_luhn\_check(card\_number: str) -> bool**

**Description:**

This function performs a custom validation of the given card\_number using a modified version of the Luhn algorithm.

**Parameters:**

* card\_number (str): The credit card number as a string (e.g., "1234567890123456"). This input must only contain digits, and the length must be at least 2 characters.

**Returns:**

* bool:
  + **True** if the card number is valid based on the custom Luhn algorithm.
  + **False** if the card number is invalid.

**Details of the Algorithm:**

1. **Initial Validation:**
   * The function checks if the input contains at least two digits and is composed only of numeric characters. If not, it prints an error message and returns False.
2. **Extract the Check Digit:**
   * The check digit is considered to be the last digit of the card number (e.g., for the card number 1234567890123456, the check digit is 6).
3. **Digit Processing:**
   * The card number is split into individual digits, with the check digit excluded from this list.
   * The remaining digits are reversed for processing.
   * Every second digit (starting from the rightmost digit) is doubled.
4. **Subsequent Calculations:**
   * After doubling every second digit, the function processes each digit, adjusting for any overflow and ensuring that the digits are properly transformed.
   * The processed digits are then summed up.
5. **Final Check:**
   * The total sum of the digits is checked to see if it's divisible by 10. If it is, the card number is valid, and the function returns True. Otherwise, it returns False.

**Script Documentation**

**2. User Input and Output:**

* **User Input:**
  + The script prompts the user to input a credit card number.
  + The input is stripped of any leading or trailing spaces using strip().
* **Output:**
  + After calling the custom\_luhn\_check() function, the script prints whether the entered credit card number is valid or invalid based on the custom Luhn algorithm.

## ****Conclusion:****

This script provides a simple implementation of the custom Luhn algorithm for credit card validation. It takes the card number as input, processes the digits according to the Luhn formula, and prints out whether the card number is valid or not.

**Output:**

****

**Task Num 02:**

**String Punctuation Removal Script**

**Overview**

This Python script removes all punctuation from a given string. The remove\_punctuation function uses Python's string.punctuation and str.translate() to strip out any punctuation characters from the input text.

**Function Documentation**

**1. remove\_punctuation(text: str) -> str**

**Description:**

This function removes all punctuation characters from the provided input string and returns the cleaned text.

**Parameters:**

* **text (str)**: A string that may contain punctuation characters. The function will remove any characters that belong to string.punctuation.

**Returns:**

* **str**: A new string with all punctuation characters removed.

**Details of the Implementation:**

1. **string.punctuation**: This is a predefined string constant in Python's string module that contains all punctuation characters (e.g., !"#$%&'()\*+,-./:;<=>?@[\\]^\_{|}~`).
2. **str.maketrans()**: The str.maketrans() function is used to create a translation table. In this case, it's used to map all punctuation characters to None (i.e., remove them).
3. **text.translate()**: The translate() method is used to apply the translation table to the input string, removing the punctuation characters.

**Example:**

**Input:**

remove\_punctuation("Hello, world! How's it going?")

**Output:**

"Hello world Hows it going"

**Script Documentation**

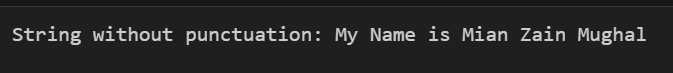
**2. User Input and Output:**

* **User Input:**
  + The script prompts the user to input a string using the input() function. The input string can contain any characters, including punctuation.
* **Output:**
  + The script prints the string after removing all punctuation characters. It uses the remove\_punctuation function to clean the input string.

## ****Conclusion****

This script provides an efficient way to clean input text by removing punctuation characters using Python's string.punctuation and str.translate(). It is a useful tool for preparing text for further processing, such as natural language processing or text analysis.

**Output:**



**Task Num 03:**

## ****Function Documentation****

### ****1.**** sort\_sentence\_alphabetically(sentence: str) -> str

#### ****Description:****

This function takes a sentence as input, splits it into words, sorts the words in alphabetical order (case-insensitive), and then returns the sorted sentence as a string.

#### ****Parameters:****

* **sentence (str)**: A string containing the sentence to be sorted.

#### ****Returns:****

* **str**: A new string where the words from the input sentence are sorted in alphabetical order.

#### ****Details of the Implementation:****

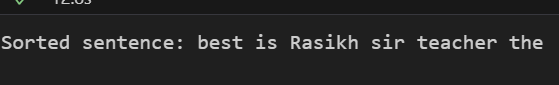
1. **sentence.split()**: Splits the input sentence into a list of words using whitespace as the delimiter.
2. **words.sort(key=str.lower)**: Sorts the list of words in alphabetical order, ignoring case. The key=str.lower ensures that the sorting is case-insensitive.
3. **' '.join(words)**: Joins the sorted list of words back into a single string, separated by spaces.

## ****Script Documentation****

### ****2. User Input and Output:****

* **User Input:**
  + The script prompts the user to input a sentence using the input() function.
* **Output:**
  + The script prints the sorted sentence, with the words arranged in alphabetical order.

**Output:**

****