**TASKS**

**1: LUHN Algorithm**

This program implements the **Luhn Algorithm**, which is used to check whether a given number (like a credit card number) is **valid or invalid**.

**Code Explaination**:

1. **Function Definition**
   * def luhn\_check(number: str) -> bool:  
     A function is created that takes a number (as string) and returns True if valid, otherwise False.
2. **Reversing Digits**
   * [int(d) for d in number] [::-1]  
     Converts the string into digits, changes them into integers, and reverses the list.  
     Reason: LUHN Algorithm always starts checking from the **rightmost digit**.
3. **Loop with Index**
   * for i, digit in enumerate(digits):  
     Iterates through each digit with its index. The index helps decide which digits will be doubled.
4. **Condition for Every Second Digit**
   * if i % 2 == 1:  
     Every second digit (odd index in reversed list) is doubled.
   * if doubled > 9: doubled -= 9  
     If doubling gives a two-digit number (>9), subtract 9 from it.
5. **Else Condition**
   * If the index is even, the digit is directly added to the total without modification.
6. **Final Validation**
   * return total % 10 == 0  
     If the total sum is divisible by 10, the number is **valid**. Otherwise, it is **invalid**.
7. **Testing the Card Number**
   * A sample number "764380765432" is tested.
   * If the function returns True, the program prints **VALID**.
   * Otherwise, it prints **INVALID**.

**2: Remove Punctuation from string**

This program removes all **punctuation marks** from a given string using Python’s (string) module.

**Code Explaination**:

1. **Importing String Module**

* import string  
  The string module is imported because it provides a predefined constant string. punctuation, which contains all common punctuation characters like !, ?, ., ,, (), etc.

2. **Function Definition**

* def remove\_punctuations(text: str) -> str:  
  A function is created that takes a string as input and returns a new string with all punctuation removed.

3. **Logic for Removing Punctuation**

* return ''. join(ch for ch in text if ch not in string.punctuation)  
  This line checks each character (ch) in the input text.
* If the character is **not** in string.punctuation, it is kept.
* Otherwise, it is skipped.
* Finally, all the kept characters are joined together to form a cleaned string.

**3: Remove Punctuation from string**

This program takes a sentence as input, splits it into words, sorts them in **alphabetical order** (ignoring case), and then joins them back into a sorted sentence.

**Code Explaination:**

1. **Function Definition**

* def sort \_sentence\_ alphabetically (sentence: str) -> str:  
  A function is created that accepts a sentence (string) and returns a new string with words sorted alphabetically.

2. **Splitting into Words**

* words = sentence.split()  
  The sentence is split into a list of words using spaces as separators.

3. **Sorting Words**

* words. sort(key= str.lower)  
  The list of words is sorted in **alphabetical order**.
* key=str. lower ensures that sorting is **case-insensitive**, meaning "Python" and "python" are treated the same.

4. **Joining Back into Sentence**

* return ' '.join(words)  
  After sorting, the words are joined back into a single string with spaces in between.

5. **Example Usage**

* sentence = "Python is a very powerful and simple language"  
  A sample sentence is provided.
* After sorting → a and is language powerful Python simple very.
* Both the **original sentence** and the **sorted sentence** are printed.