Practice Programs

1. Write a Python program to count the number of characters (character frequency) in a string.
   Sample String : google.com'
   Expected Result : {'o': 3, 'g': 2, '.': 1, 'e': 1, 'l': 1, 'm': 1, 'c': 1}
2. Write a Python program to count the occurrences of each word in a given sentence.
3. Write a Python function that takes a list of words and returns the length of the longest one.
4. Write a Python program to remove the n^{th} index character from a nonempty string.
5. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string.
   Sample String : 'hello'
   Expected Result : 'helo'
   Sample String : 'hi'
   Expected Result : 'hi'
   Sample String : 'u'
   Expected Result : Empty String
6. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself.
   Sample String : 'restart'
   Expected Result : 'resta$t'
7. Write a Python program to add 'ing' at the end of a given string (length should be at least 3). If the given string already ends with 'ing' then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.
   Sample String : 'abc'
   Expected Result : 'abcing'
   Sample String : 'string'
   Expected Result : 'stringly'
8. Write a Python program to find the first appearance of the substring 'not' and 'poor' from a given string, if 'bad' follows the 'poor', replace the whole 'not'...'poor' substring with 'good'. Return the resulting string.
   Sample String : 'The lyrics is not that poor!'
   Expected Result : 'The lyrics is good!'
9. Write a Python program that accepts a comma separated sequence of words as input and prints the unique words in sorted form (alphanumerically).
   Sample Words : red, white, black, red, green, black
   Expected Result : black, green, red, white
10. Write a Python function to create the HTML string with tags around the word(s).
    Sample function and result :
    add_tags('i', 'Python') -> 'Python'
    add_tags('b', 'Python Tutorial') -> 'Python Tutorial '
11. Write a Python program to create a Caesar encryption. In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example,

with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

12. Write a Python program to print a floating number upto 2 decimal places with a sign.
13. Write a Python program to print integers with '*' on the right of specified width.
14. Write a Python program to swap comma and dot in a string.
    Sample string: "32.054,23"
    Expected Output: "32,054.23"
15. Write a Python program to split a string on the last occurrence of the delimiter.
16. Write a Python program for the game of Hangman. Have a predefined set of words in your program. For every new game, your program must choose a word from the predefined set randomly.
    Note: If you find this easy, try adding hints for each word.
17. Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. The passwords should be random, generating a new password every time the user asks for a new password. Ask the user how strong they want their password to be. For weak passwords, pick a word or two from a list.
18. Write a password strength validator in Python. Use the conditions from the above password generator question.
19. You are given a string, and you have to validate whether it's a valid Roman numeral. If it is valid, print *True*. Otherwise, print *False*. Try to create a regular expression for a valid Roman numeral.
20. A company decides to create a unique identification number (UID) for each of its employees. The company has assigned you the task of validating all the randomly generated UIDs.
    A valid UID must follow the rules below:
    - It must contain at least 2 uppercase English alphabet characters.
    - It must contain at least 3 digits (0-9).
    - It should only contain alphanumeric characters (a-z,A-Z,0-9).
    - No character should repeat.
    - There must be exactly 10 characters in a valid UID.
21. You and Fredrick are good friends. Yesterday, Fredrick received credit cards from ABCD Bank. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!
    A valid credit card from ABCD Bank has the following characteristics:
    - It must start with a 4, 5 or 6.
    - It must contain exactly 16 digits.
    - It must only consist of digits (0-9).
    - It may have digits in groups of 4, separated by one hyphen "-".
    - It must NOT use any other separator like ' ' , '_', etc.
    - It must NOT have 4 or more consecutive repeated digits.