ASSIGNMENT - 06

1. Write a function that takes a string as a parameter and returns a string with every successive repetitive character replaced with a star (*). For example, 'balloon' is returned as 'bal*o*n'.

Ans :
```
def replace_repetitive_chars(input_str):
    result = ""
    prev_char = ""
    for char in input_str:
        if char == prev_char:
            result += "*"
        else:
            result += char
        prev_char = char
    return result

original_string = "balloon"
modified_string = replace_repetitive_chars
        (original_string)

print(modified_string)

Output : bal*o*n
```

2. Write a function that takes two strings and returns True if they are anagrams and False otherwise. A pair of strings is anagrams if the letters in one word can be arranged to form the second one.

Ans :
```
def are_anagrams(str1, str2):
    str1 = str1.replace(" ","").lower()
    str2 = str2.replace(" ","").lower()

    return sorted(str1) == sorted(str2)

string1 = "listen"
string2 = "silent"
```

```
if are_anagrams (string1 , string2 ) :
        print (f " { string1 } and {string2 } are anagrams.")
else :
        print(f " { string 1 } and { string2 } are not anagrams.")
```

Output : listen and silent are anagrams .

3 . Write a function that takes a sentence as an input parameter and displays the number of words in the sentence .

Ans :
```
def count_words (sentence ) :
        words = sentence . split()
        return len (words )

input_sentence = " This is a sample sentence."
word_count = count_words (input_sentence )
print(f " The number of words in the sentence is :
        {word_count } " )
```

Output : The number of words in the sentence is

4 . Write a function that takes a sentence as an input parameter and replaces the first letter of every word wi the corresponding uppercase letter and rest of the letters in the word by corresponding letters in lowercase witho using builtin function .

Ans :
```
def capitalize_first_letter (sentence ) :
        words = sentence . split ()
        result = [ ]
        for word. in words :
            modified_word = word [0] . upper () + word[
                lower()
            result . append ( modified_word )
        modified_sentence = ' ' .join (result )
        return modified_sentence
input_sentence = " this is a sample sentence ."
```

```
modified_sentence = capitalize_first_letter (input_sentence)
print (f"Modified sentence : {modified_sentence}")
```

Output : Modified sentence : This Is A Sample Sentence .

5. Write a function that takes a string as an input and determines the count of the number of words using regular expression.

Ans :
```
import re

def count_words_regex (input_str) :
    words = re.findall (r'\b\w+\b', input_str)
    return len (words)

input_string = "This is a sample string with words."
word_count = count_words_regex (input_string)
print (f"The number of words in the string is :{word_count}")
```

Output : The number of words in the string is : 8

6. What will be the output on executing each of the statements, following the assignment statement :

address = 'B-6 , Lodhi road , Delhi'

a. len (address) : This will give the length of the string.

b. address [17:-1] ; This will extract a substring starting from the 18th character (index 17) up to the second-to-last character.

c. address-len (address):len (address)] : This will give the entire Starting because the start index is -length , which is equivalent to 0 .

d. address [:-12] + address [-12:] : This will concantenate two substrings - the first one goes up to 12 th character from the end, and the second one starts from the 12 th character from the end.

e. address .find ('delhi') : This will return -1 because 'delhi' (in lowercase) is not found in the given string.

f. address . swapcase () ; This will swap the case of each character in the string .

g. address.split(',') : This will split the string into a list of substrings using ',' as the delimiter.

h. address.isalpha() : This will return False because the string contains non-alphabetic characters and spaces.

7. Examine the following string :

greeting = 'Good Morning, Have a Good Day !!'

what will be the output for the following function calls ;

a. greeting.count('Good') : This will count the occurrences of the substring 'Good' in the given string.

b. greeting.find('a') : This will returns the index of the first occurence of 'a' in the string.

c. greeting.rfind('a') : This will return the index of the last occurence of 'a' in the string.

d. greeting.capitalize() : This will capitalize the first letter of the string.

e. greeting.lower() : This will convert the entire string to lowercase.

f. greeting.upper() : This will convert the entire string to uppercase.

g. greeting.swapcase() : This will swap the case of each character in the string.

h. greeting.istitle() : This will return True if the string is a titlecased string, False otherwise.

i. greeting.replace('Good', 'sweet') : This will replace all occurences of 'Good' with 'Sweet' in the string.

j. greeting.strip() : This will remove leading and trailing white-spaces from the string.

k. greeting.split() : This will split the string into a list of words.

l. greeting . partition (',') : This will split the string into three parts based on the first occurrence of ','.

m. greeting . startswith ('good') : This will return False because it's case - sensitive, and 'Good' doesn't start with 'good'.

n. greeting . endswith ('!!!') : This will return True because the string ends with '!!'.

8. Determine the patterns extracted by the following regular expressions.

1. string 1 = 'Python Programming Language'

1. match 1 = re . search ('.....m?' , string 1) : This matches any six characters followed by an optional 'm'.
print (match 1 . group())
-The result will be 'Program'.

2. match 2 = re . search ('.....m{1,2}', string 1) : This matches any six characters followed by one or two 'm'.
print (match2. group())
The result will be "Programm".

3. match 3 = re . search ('.* Language$', string 1) : This matches any characters before the word "Language$" at the end of the string.
print (match3 . group())
The result will be "Python Programming Language".

4. match4 = re. search ('\w* \s\w*', string 1) : This matches any characters before words separated by a space.
print (match4 . group())
The result will be "Python Programming".

5. match5 = re . search ('.*', string 1) : This matches any characters (zero or more).
print(match5 . group())
The result will be the entire string "Python Programming Language".

2. string2 = 'car Number DL5645'

1. match1 = re.search('\w\w ?\d{1,4}', string2) : This matches one
two word characters followed by one to four digits.
print(match1.group())
The result will be "DL5645".

2. match2 = re.search('.*5', string2) : this matches any character
followed by '5'.
print(match2.group())
The result will be "Car Number DL5645".

3. match3 = re.search('.*5?', string2) : This matches any character
followed by an optional '5'.
print(match3.group())
The ~~match~~ result will be "Car Number DL5645".

4. match4 = re.search('\d{3}', string2) : This matches exactly
three digits.
print(match4.group())
The result will be "564".

5. match5 = re.search('^C.*5$', string2) : This matches a
string starting with space followed by 'C' and ending with '5
print(match5.group())
The result will be "Car Number DL5645".

3. string3 = 'cdcccdcddd343344aabb'

1. match1 = re.search('(c|d)*\d*(a|b)*', string3) : This match
zero or more 'c' or 'd', followed by zero or more digits, follo
by zero or more 'a' or 'b'.
print(match1.group())
The result will be the entire string "cdcccdcddd343344aabb".

2. match2 = re.search('(cd)*d', string3) : This matches zero or
more occurrences of "cd" followed by 'd'.
print(match2.group())
The result will be "cdd".

3. match3 = re.search('(cd|cd)*(3|4)*(aa|bb)', string3):
This matches zero or more occurrences of "cc" or "cd",
followed by zero or more occurrences of '3' or '4', followed by
zero or more occurrences of "aa" or "bb".

print(match3.group())

The result will be "cccdcddd343344aabb".

4. match4 = re.search('(cc|cd|dd)* (3|4) * (aa|bb)', string3) : This matches zero or more occurrences of "cc", "cd", or "dd", followed by zero or more occurrences of '3' or '4', followed by zero or more occurrences of "aa" or "bb".

print(match4.group())

The result will be "cccdcddd343344aabb".

5. match5 = re.search('(cc|cd|dd)*(3|4)* (aa|bb)*', string3) : This matches zero or more occurrences of "cc", "cd", or "dd", followed by zero or more occurrences of '3' or '4', followed by zero or more occurrences of "aa" or "bb".

print(match5.group())

The result will be "cccdcddd343344aabb".

1. Write a python programs to perform the following tasks:

a. Reverse a string:

```
def reverse-string(input-str):
        return input_str[::-1]
Original_string ="hello"
reversed_string = reverse-string(original-string)
print(f"Original: {original_string} \nReversed: {reversed-staing}")
```

b. Reverse a string without reversing the words. Example:

input data: 'welcome to iter'

Output: 'iter to welcome'

```
def reverse_words(input-str):
        words = input-str.split()
        reversed_words = ' '.join(reversed(words))
        return reversed_words
input_data = "welcome to iter"
Output_data = reverse_words(input_data)
print(f"Input: {input_data} \nOutput: {output-data}")
```

c. Check if a string is symmetric or asymmetric:

```
def is-symmetric(input-str):
        return input-str == input-str[:: -1]
symmetric-str = "madam"
```

```
asymmetric_str = "hello"
print(f"Symmetric : {is_symmetric(symmetric_str)}")
print(f"Asymmetric : {is_symmetric(asymmetric_str)}")
```

d. check if a string is palindrome.

```
def is_palindrome(input_str):
        return input_str == input_str[::-1]
palindrome_str = "radar"
non_palindrome_str = "hello"
print(f"Palindrome : {is_palindrome(palindrome_str)}")
print(f"Non-Palindrome : {is_palindrome(non_palindrome_str)}")
```

e. Given a string s and index, i, delete ith value from s

```
def delete_character_at_index(input_str, index):
        return. input_str[:index] + input_str[index+1:]
original_string = "example"
index_to_delete = 3
result_string = delete_character_at_index(original_string,
index_to_delete)
print(f"Original : {original_string} \nAfter deletion :
{result_string}")
```

f. Count the number of vowels and consonants in a string.

```
def count_vowels_consonants(input_str):
        vowels = "aeiouAEIOU"
        num_vowels = sum(1 for char in input_str if char in
            vowels)
        num_consonants = len(input_str) - num_vowels
        return num_vowels, num_consonants
sample_string = "hello world"
vowels, consonants = count_vowels_consonants(sample_string)
print(f"Vowels : {vowels} \nConsonants : {consonants}")
```

g. Find length of a string without using inbuilt function.

```
def find_length_without_inbuit {input_str):
        length = 0
        for _ in input_str:
            length += 1
        return length
```

```
sample_string = "hello world"
length_without_inbuilt =
find_length_without_inbuilt(sample_string)
print(f"Length without using inbuilt function :
{length_without_inbuilt}")
```

h. check if a string contains at least one digit and one alphabet.

```
def contains_digit_and_alphabet(input_str):
    has_digit = any(char.isdigit() for char in input_str)
    has_alpha = any(char.isalpha() for char in input_str)
    return has_digit and has_alpha
Sample_string1 = "abc123"
sample_string2 = "123456"
print(f"Sample 1 : {contains_digit_and_alphabet(sample_string1)}")
print(f"Sample 2 : {contains_digit_and_alphabet(sample_string2)}")
```

i. Remove duplicates from a string.

```
def remove_duplicates(input_str):
        unique_chars = ''.join(sorted(set(input_str),
key = input_str.index))
        return unique_chars
original_string = "programming"
result_string = remore_duplicates(original_string)
print(f"Original : {original_string}\nAfter removing duplicates:
{result_string}")
```

j. Count frequency of characters in a string.

```
def count_character_frequency(input_str):
    frequency = {}
    for char in input_str:
        frequency[char] = frequency.get(char, 0)+1
    return frequency
sample_string = "hello world"
char_frequency = count_character_frequency(sample_string)
print(f"Character frequency :{char_frequency}")
```

K. Find the character having maximum frequency in a string.

```
def max_frequency_character(input_str):
    frequency = count_character_frequency(input_str)
    max_char = max(frequency, key = frequency.get)
    return max_char

sample_string = "programming"
max_char = max_frequency_character(sample_string)
print(f"character with maximum frequency : {max_char}")
```