L11 : Strings Practice Questions

In python string is sequence of characters that is stored in machine language using unicode encoding technique whereas in c++ / java ASCII encoding is used.
Python does the auto optimization for smaller strings, meaning no extra space will be allocated to the same string (ex : a = ‘Parikh’, b = ‘Parikh’) then id(a) will be the same as id(b) means both will point to the same memory where ‘Parikh’ is stored.

We can enclose string in single quote(‘ ’) , double quote (“ ”), Triple quote (‘’’ ‘’’): multiline string allowed only in triple quote.

1-Tut : Predict the output

Send Feedback
What will be the output of the following code ?

```
s = “abcd”

print(s[-2])
```

Options

d
c
b
None of these

Correct Answer : B

How Strings are Stored

2-Tut : Predict the output

Send Feedback
What will be the output of the following code?

```
s="abcd"
s[0]='c'

print(s)
```

Note : in Python Strings are immutable means change in string is not allowed whereas in c++ we can make changes in the string.

Options

cbcd
abcd
Error
None of these

Correct Answer : C

3-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
s =”abcd”
a = “abcd”
if id(s) == id(a):
   print(“They are same”)
else:
   print(“They are not same”)
```

Options

They are same
They are not same
Error
None of the above

Correct Answer : A

Concatenation of Strings

4-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
s = "abcd"
b = s + "ef"
print(s)
```

Options

abcdef
abcd
Error
None of the above

Correct Answer : B

5-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
s = "abcd"
b = s + 2

print(b)
```

Options

abcd2
abcd
Error
None of the above

Correct Answer : C

Slicing of Strings

6-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
s = "abcdef"

print (s[2:])
```

Options

bcdef
cdef
abcdef
None of the above

Correct Answer : B

7-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
s = "abcdef"

print (s[4:2:-1])
```

Options

edc
ed
bcd
None of the above

Correct Answer : B

Iterating on strings, Comparison Operator on strings

8-Tut : Check Palindrome

Send Feedback

Given a string, determine if it is a palindrome, considering only alphanumeric characters.

Palindrome

A palindrome is a word, number, phrase, or other sequences of characters which read the same backwards and forwards.

Example:

If the input string happens to be, "malayalam" then as we see that this word can be read the same as forward and backwards, it is said to be a valid palindrome.
The expected output for this example will print, 'true'.
From that being said, you are required to return a boolean value from the function that has been asked to implement.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false'.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1 :

abcdcba

Sample Output 1 :

true

Sample Input 2:

coding

Sample Output 2:

false

```
def isPalindrome(string) :
        # Your code goes here
    i,j = 0,len(string)-1
    ans = True
    while(i <= j):
        if(string[i] != string[j]):
            ans = False
            break
        i += 1
        j -= 1
    return ans
```

9-Tut : Predict the output

Send Feedback
What will be the output of the following code?

```
a = "abcdef" == "abcd"
```

```
print(a)
```

Options

False
True
Error
None of the above

Correct Answer : A

10-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
a = "abcdef" >= "abcd"

print(a)
```

Options

False
True
Error
None of the above

Correct Answer : B

11-Tut : Predict the output

Send Feedback

What will be the output of the following code?

```
a = "abce" >= "abcdef"

print(a)
```

Options

False
True
Error
None of the above

Correct Answer : B

Some Commonly used functions of strings :

1. List = String.split (“delimiter”, k): Default value : space, default k = max possible split based on delimiter , split function returns the list
2. String = string.replace(“to be replaced”, “to by Replaced”,k) : default k = 1(how many occurrence to be replaced)

3. Index = string.find(“str to be find”, start , end) : default : start = 0, end = len-1+1 = len of original string in which we will perform find operation it returns the first starting index of char if string is available otherwise return -1
4. String = str.lower(), string = str.upper(),
5. ans = str.startswith(“string”, startindex, endindex)

12-Ass : Check Permutation

Send Feedback

For a given two strings, 'str1' and 'str2', check whether they are a permutation of each other or not.

Permutations of each other

Two strings are said to be a permutation of each other when either of the string's characters can be rearranged so that it becomes identical to the other one.

Example:
str1= "sinrtg"
str2 = "string"

The character of the first string(str1) can be rearranged to form str2 and hence we can say that the given strings are a permutation of each other.

Input Format:

The first line of input contains a string without any leading and trailing spaces, representing the first string 'str1'.
The second line of input contains a string without any leading and trailing spaces, representing the second string 'str2'.

Note:

All the characters in the input strings would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false', denoting whether the two strings are a permutation of each other or not.
You are not required to print anything. It has already been taken care of. Just implement the function.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

abcde
baedc

Sample Output 1:

true

Sample Input 2:

abc
cbd

Sample Output 2:

false

```
def isPermutation(string1, string2) :
        #Your code goes here
    n1 = len(string1)
    n2 = len(string2)
    if (n1 != n2):
        return False
    else:
        ans = True
        for char in string1:
            if char not in string2:
                ans = False
                break
        return ans
```

Note : Some test case will fail by above code ex : (string1 = aa, string2 = ab) one optimisation is as below

```
def isPermutation(string1, string2) :
        #Your code goes here
    n1 = len(string1)
    n2 = len(string2)
    if (n1 != n2):
        return False
    else:
        ans = True
        for char in string1:
            if char not in string2:
                ans = False
                break
        ans1 = True
        for char in string2:
            if char not in string1:
                ans1 = False
                break
        return ans and ans1
```

Note : still some cases will not pass ex: (aba, bab) expected o/p should be false but it gives true

Idea : other than length, Frequency of each character should also be same

```
def isPermutation(string1, string2) :
        #Your code goes here
    n1 = len(string1)
    n2 = len(string2)
    if (n1 != n2):
        return False
    else:
        arr = [0] * 26
        for char in string1:
```

```
            arr[ord(char)-ord('a')] += 1
        for char in string2:
            arr[ord(char)-ord('a')] -= 1
        for i in arr:
            if i != 0:
                return False
        else:
            return True
```

13-Ass : Remove Consecutive Duplicates

Send Feedback

For a given string(str), remove all the consecutive duplicate characters.

Example:

Input String: "aaaa"

Expected Output: "a"

Input String: "aabbbcc"

Expected Output: "abc"

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1: aabccbaa
Sample Output 1: abcba

Sample Input 2: xxyyzxx
Sample Output 2: xyzx

```
def removeConsecutiveDuplicates(string) :
        # Your code goes here
    n = len(string)
    i = 0
    string2 = ""
    while(i < n):
        x = string[i]
        j = i + 1
        while(j < n and string[j] == x):
            j += 1
        string2 += x
        i = j
    return string2
```

Method 2 :- using list operation bcz of string += k in python always creates a new string whereas in c++ strings are mutable hence no need to create another string

Method 3 : inbuilt :

```
import itertools
def removeConsecutiveDuplicates(string) :
        # Your code goes here
    return (''.join(i for i, _ in itertools.groupby(string) ) )
```

14-Ass : Reverse Each Word

Send Feedback

Aadil has been provided with a sentence in the form of a string as a function parameter. The task is to implement a function so as to print the sentence such that each word in the sentence is reversed.

Example:

Input Sentence: "Hello, I am Aadil!"

The expected output will print, ",olleH I ma !lidaA".

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. The input string represents the sentence given to Aadil.

Output Format: The only line of output prints the sentence(string) such that each word in the sentence is reversed.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1:

Welcome to Coding Ninjas

Sample Output 1:

emocleW ot gnidoC sajniN

Sample Input 2:

Always indent your code

Sample Output 2:

syawlA tnedni ruoy edoc

```
def reverseEachWord(string) :
    # Your code goes here
    w = string.split(" ")
    nw = [i[::-1] for i in w]
    ns = " ".join(nw)
    return ns
```

15-Ass : Remove character

Send Feedback

For a given string(str) and a character X, write a function to remove all the occurrences of X from the given string.

The input string will remain unchanged if the given character(X) doesn't exist in the input string.

Input Format:

The first line of input contains a string without any leading and trailing spaces.

The second line of input contains a character(X) without any leading and trailing spaces.

Output Format: The only line of output prints the updated string.

Note: You are not required to print anything explicitly. It has already been taken care of.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1:

aabccbaa

a

Sample Output 1:

bccb

Sample Input 2:

xxyyzxx

y

Sample Output 2:

Xxzxx

```
def removeAllOccurrencesOfChar(string, ch) :
        # Your code goes here
    list1 = string.split(ch)
    string2 = "".join(list1)
    return string2
```

16-Ass : Highest Occurring Character

Send Feedback

For a given string(str), find and return the highest occurring character.

Example:

Input String: "abcdeapapqarr"

Expected Output: 'a'

Since 'a' has appeared four times in the string which happens to be the highest frequency character, the answer would be 'a'.

If there are two characters in the input string with the same frequency, return the character which comes first.

Consider:

Assume all the characters in the given string to be in lowercase always.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything explicitly. It has already been taken care of.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1:

abdefgbabfba

Sample Output 1:

b

Sample Input 2:

xy

Sample Output 2:

x

```
def highestOccuringChar(string) :
        #Your code goes here
    ASCII_SIZE = 256
    ctr = [0] * ASCII_SIZE
    max = -1
    ch = ''
```

```
    for i in string:
        ctr[ord(i)] += 1
    for i in string:
        if max < ctr[ord(i)]:
            max = ctr[ord(i)]
            ch = i
    return ch
```

17-Ass : Compress the String

Send Feedback

Write a program to do basic string compression. For a character which is consecutively repeated more than once, replace consecutive duplicate occurrences with the count of repetitions.

Example:

If a string has 'x' repeated 5 times, replace this "xxxxx" with "x5".

The string is compressed only when the repeated character count is more than 1.

Note:

Consecutive count of every character in the input string is less than or equal to 9.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The output contains the string after compression printed in a single line.

Note:

You are not required to print anything. It has already been taken care of. Just implement the given function.

Constraints:

0 <= N <= 10^6

Where 'N' is the length of the input string.

Time Limit: 1 sec

Sample Input 1:

aaabbccdsa

Sample Output 1:

a3b2c2dsa

Explanation for Sample Output 1:

In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is repeated 2 times and 'd', 's' and 'a' and occuring 1 time hence no compression for last 3 characters.

Sample Input 2:

aaabbcddeeeee

Sample Output 2:

a3b2cd2e5

Explanation for Sample Output 2:

In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is occuring single time, 'd' is repeating 2 times and 'e' is repeating 25times.

```
1.  def getCompressedString(s) :
2.              # Write your code here.
3.      i=0
4.      y = ""
5.      while i < len(s):
6.          x = s[i]
7.          j = i+1
8.          c=1
9.          while j < len(s) and s[j] == x:
10.             j=j+1
11.             c=c+1
12.         if c>1:
13.             y=y+s[i]+str(c)
14.         else:
15.             y=y+s[i]
16.         i = j
17.     return y
```