Today I'll Cover :

1. String
2. String Indexing and Slicing
3. Methods of String
4. Operator that can be used with string.

In the name of ALLAH,
the Most Beneficent, the Most Merciful

Mohammad Altaf Hussain

String

- It is a sequence of character.
- Enclosed within single quote (' '), double quote(" "), triple single quote (''' ''') or triple double quote (""" """).
- It is most commonly used object in any programming language.

- We can use single quote inside double quote and vice-versa.
- Single and double quote both, can be used together inside triple quote.
- We can define multi-line String literals by using triple single or double quotes.
- Escape sequence plays and important role in strings.
- To print escape sequence itself, we can use \ letter followed by escape sequence.

String Indexing

- We can access a character of string using indexing.
- +(ve) index means left to right (forward).
- -(ve) index means right to left (backward).
- Indexing start from 0 in forward direction -1 in backward direction.
- If we are trying to access characters of a string with out of range index then we will get IndexError.

String Slicing

- **We can access one or more character of a string using string slicing**

Syntax : **any_string[bIndex:eIndex:step]**

Where, **bIndex**: From where we have to consider slice(substring).

eIndex: From where to terminate the slice(substring) at endindex-1

Note :-

- If we not specify begin index then it will consider from beginning of the string.
- If we not specify end index then it will consider up to end of the string.
- The default value for step is 1.

Note :-

- if step value is +ve then it should be traverse in forward direction(left to right).
- if step value is -ve then it should be traverse in backward direction(right to left).

Operator that can be applied on string

+			Concatenation		
*			Repetition		
<	<=	==	!=	>=	>
in	not in		Membership		

1. To use + operator for Strings, compulsory both arguments should be str type
2. To use * operator for Strings, compulsory one argument should be str and other argument should be int
3. We can compare strings, comparison is based on how the letters are arranged in dictionary that we use in real life.

Mohammad Altaf Hussain

Methods of String

❖ Finding Substrings

- **find() :-** Return index of 1st occurrence of the given substring. If it is not available then we will get -1.

Syntax: any_str.find(substring, begin, end)

Where, **begin** and **end** indicates beginning index position and ending index position respectively. It is optional.

- **index() :-** It works exactly same as find() method except that if the specified substring is not available then we will get ValueError.

- **rfind() :-** It works exactly same as find() but it return the reverse index.

- **rindex() :-** It works exactly same as index() but it return the reverse index.

❖ Counting Substring in given string

- **count() :-** It is used to find the occurrence of specified substring.

Syntax: any_str.count(substring, begin, end)

Where, **begin** and **end** indicates beginning index position and ending index position respectively. It is optional.

❖ Replacement of String.

- **replace() :-** It is used to replace a string with another substring.

Syntax: any_str.replace(Oldstring, Newstring)

Here every occurrence of Oldstring will be replaced with Newstring.

Note: when we use replace() method, a new object got created but existing object won't be changed, because strings are immutable.

❖ Splitting of String.

- **split()** :- It is used to split the given string based on specified separator. Its return type is List.

Syntax: any_str.split(separator)

Here separator is a string based on which we want to split the string. It is optional.

❖ Joining of String.

- **join() :-** It is used to join a group of strings with respect to the given separator.

Syntax: separator.join(group_of_string)

Here separator is a string based on which we want to join the string.

❖ Change the case of String.

upper()	Convert all characters to uppercase.
lower()	Convert all characters to uppercase.
swapcase()	Convert all uppercase character to lowercase and vice-versa.
title()	Convert all the 1st character of each word to uppercase.
capitalize()	Convert the 1st character to uppercase only.

❖ Check starting & ending part of String.

- **startswith() :-** It is used check whether a string starts with specified string or not.

Syntax: any_stirng.startswith(string)

- **endswith() :-** It is used check whether a string ends with specified string or not.

Syntax: any_stirng.endswith(string)

❖ To check type of character present in a String.

isalnum()	Returns True if all characters are alphanumeric.
isalpha()	Returns True if all characters are alphabhet only
isdigit()	Returns True if all characters are digit only
islower()	Returns True if all characters are lowercase only
isupper()	Returns True if all characters are uppercase only
istitle()	Returns True if string is in title case.
isspace()	Returns True if all characters are space only

❖ Formatting of String.

- We can format the strings with variable values by using replacement operator {}.

:	Use a space to insert an extra space before positive numbers
:>	Right aligns the result
:<	Left aligns the result
:^	Center aligns the result
:,	a comma as a thousand separator
:_	a underscore as a thousand separator
:b	Binary format
:d	Decimal format
:o	Octal format

:x	Hexadecimal format
:c	Converts the value into the corresponding unicode character
:e	Scientific format
:.	Floating number upto given decimal

Note:

We can print string in raw format with the help of r flag.

Questions

1. Take input from the user and find out the total number of vowels, consonants, digits, space, words, letters, special characters (!@#$%^&).
2. Find out the reverse of each word of user input.

Eg. User_input = how are you

output = woh era uoy

Mohammad Altaf Hussain