String Replace

The string.replace() method in Python replaces all occurrences of a specified substring with a new value. It does not replace just the first occurrence; rather, it replaces all instances found within the string.

Here is the syntax for the replace() function in Python:

----- string.replace(old, new[, count])

old: The substring to be replaced.

new: The new substring that will replace the occurrences of old.

count (optional): The maximum number of occurrences to replace. If not specified, all occurrences will be replaced.

If count is provided, the replace() method will replace the first count occurrences of the old substring with the new substring. If count is not provided, all occurrences of the old substring in the string will be replaced with the new substring.

Here's an example to illustrate this behavior:

As you can see, all occurrences of "hello" in the string have been replaced with "hi" using the replace() method.

Questions:

1: Given the following string, remove all the spaces using the replace() function.

input_string = "Hello, how are you doing?"

Write code here to remove spaces from the string

Output should be: "Hello,howareyoudoing?"

Answer:

input_string = "Hello, how are you doing?"

output_string = input_string.replace(" ", "")

```
print(output_string)

# Output: "Hello,howareyoudoing?"
```

2: Given a string with multiple occurrences of the word "book," write a Python program to replace only the first two occurrences with "notebook."

```
input_string = "I have a book, a pen, and another book in my bag and book."

# Output should be: "I have a notebook, a pen, and another notebook in my bag and book."
```

Answer:

```
input_string = "I have a book, a pen, and another book in my bag and book."

output_string = input_string.replace("book", "notebook", 2)

print(output_string)

# Output: "I have a notebook, a pen, and another notebook in my bag and book.""
```

String len()

The len() function in Python is used to determine the length of a sequence, such as a string, list, tuple, or other iterable. It returns the number of elements present in the given sequence.

1. What happens when you pass an empty string to the len() function?

Answer :

When you pass an empty string to the len() function, it will return 0 since the length of an empty string is zero.

Example:

```
empty_string = ""
length_of_empty_string = len(empty_string)
print(length_of_empty_string)  # Output: 0
```

2. How len() works in case of multi-line string?

Answer:

len() function works the same way for multi-line strings as it does for single-line strings. It returns the number of characters in the string, including newline characters (\n) for each line break.

Example:

```
multi_line_string =  """
This is a multi-line string
that spans across
multiple lines.
"""

length = len(multi_line_string)
```

print(length)

Output:

63

In the example above, the len() function is used to find the length of the multi_line_string, which consists of three lines. The total number of characters, including spaces and newline characters, is 63.

Here's the breakdown of the characters in the multi_line_string:

"\n" (newline character)

"This is a multi-line string" (27 characters, including spaces)

"\n" (newline character)

"that spans across" (17 characters, including spaces)

"\n" (newline character)

"multiple lines." (16 characters, including spaces)

String

1: Can you concatenate strings and variables in Python? If yes, how?

Answer:

Yes, you can concatenate strings and variables in Python using the + operator.

Example:

name = "John"

greeting = "Hello, " + name + "!"

print(greeting) # Output: "Hello, John!"

2: How do you format strings in Python to include variables or expressions?

Answer:

Python provides multiple ways to format strings. One common method is by using f-strings (formatted string literals) where you prefix the string with f or F and then include the variables or expressions inside curly braces {}.

Example:

name = "Alice"

age = 30

message = f"My name is {name} and I am {age} years old."

print(message) # Output: "My name is Alice and I am 30 years old."

3: Input- abcd
Output- A_b_c_d

Answer:

input_string = "abcd"

input_string = input_string.capitalize()

 # Convert each letter to uppercase and join them with underscores

output_string = "_".join(input_string)

print(output_string)

4. How can you limit the number of splits using the maxsplit parameter?

Answer:

You can use the maxsplit parameter to limit the number of splits performed by the split() method. It specifies the maximum number of splits to make, and the remaining part of the string will be considered as a single element in the resulting list.

Example:

my_string = "one two three four five"

limited_splits = my_string.split(" ", 2)

print(limited_splits) # Output: ['one', 'two', 'three four five']

Explanation:

In the given example, we are using split(" ", 2), which means we want to perform a maximum of two splits based on the space character. The output of this example is indeed ['one', 'two', 'three four five'], which has three elements in the resulting list.

Here's how the splitting works:

The first space character is found after "one". So, "one" becomes the first element of the resulting list.

The second space character is found after "two". So, "two" becomes the second element of the resulting list.

Since we specified maxsplit=2, there is only one more split allowed. The third space character is found after "three". However, since we reached the maximum number of splits, the remaining part of the string ("three four five") is considered as a single element and becomes the third element of the resulting list.

As a result, the output is ['one', 'two', 'three four five'] with three elements in the list. The maxsplit parameter controls the maximum number of splits, not the total number of elements in the resulting list.