TOPSTechnologies

String Manipulation

Presented for :

TOPs Technologies

Presented by :

Sumit B Yadav

Que 1

Accessing and manipulating strings in Python involves using various string methods and operations..

Accessing Strings

Strings in Python are sequences of characters. You can access individual characters or substrings using indexing and slicing.

1. *Indexing :*

String Indexing allows us to access individual characters in a string. Python treats strings like lists where each character is assigned an index, starting from 0. We can access characters in a String in Two ways :

1. *Accessing Characters by Positive Index Number*
2. *Accessing Characters by Negative Index Number*

Accessing by Positive Index Number

In this type of Indexing we pass a Positive index (which we want to access) in square brackets. The index number starts from index number 0 (which denotes the first character of a string)

Accessing by Negative Index Number

In this type of Indexing, we pass the Negative index(which we want to access) in square brackets. Here the index number starts from index number -1 (which denotes the last character of a string).

Formatting Strings :

You can format strings using f-strings (formatted string literals), the format() method, or the % operator.

2. Slicing :

String Slicing allows us to extract a part of the string. We can specify a start index, end index, and step size.

The general format for slicing is:

Syntax - *string[start : end : step]*

- start : We provide the starting index.
- end : We provide the end index(this is not included in substring).
- step : It is an optional argument that determines the increment between each index for slicing.

String Methods

Python provides many built-in methods for string manipulation. Here are some of the most commonly used ones:

1. Upper and Lower case -

```
print(my_string.upper())  # Output: 'HELLO, WORLD!'
print(my_string.lower())  # Output: 'hello, world!'
```

- *Replace -*

print(my_string.replace("World", "Python")) # Output: 'Hello, Python!'

- *Split and Join -*

print(my_string.split()) # Output: ['Hello,', 'World!']
print(" ".join(['Hello,', 'World!'])) # Output: 'Hello, World!'

- *Find and Count -*

print(my_string.find("World")) # Output: 7
print(my_string.count("o")) # Output: 2

- *Starts with and Ends with -*

print(my_string.startswith("Hello")) # Output: True
print(my_string.endswith("!")) # Output: True

Que. 2

Concatenation

String concatenation is the operation of joining two or more strings end-to-end

code -

```
string1 = "Hello"
string2 = "World"
concatenated = string1 + ", " + string2 + "!"
print(concatenated)  # Output: 'Hello, World!'
```

Repetition

String repetition involves repeating a string a specified number of times.

code -

```
repeated = "Hello" * 3
print(repeated)  # Output: 'HelloHelloHello'
```

upper()

Converts all characters in the string to uppercase.

code -

```
my_string = "Hello, World!"
print(my_string.upper())  # Output: 'HELLO, WORLD!'
```

lower()

Converts all characters in the string to lowercase.

code -

```
print(my_string.lower())  # Output: 'hello, world!'
```

capitalize()

Capitalizes the first character of the string and converts the rest to lowercase.

code -

```
print(my_string.capitalize())  # Output: 'Hello, world!'
```

title()

Converts the first character of each word to uppercase and the remaining characters to lowercase.

python

code -

```
print(my_string.title())  # Output: 'Hello, World!'
```

strip()

Removes any leading (spaces at the beginning) and trailing (spaces at the end) characters (space is the default).

code -

```
my_string = "   Hello, World!   "
print(my_string.strip())  # Output: 'Hello, World!'
```

replace()

Replaces a specified phrase with another specified phrase.

python

code -

```
print(my_string.replace("World", "Python"))  # Output: 'Hello, Python!'
```

split()

Splits the string into a list where each word is a list item.

code -

```
my_string = "Hello, World!"
print(my_string.split())  # Output: ['Hello,', 'World!']
```

join()

Joins the elements of an iterable (e.g., list) into a single string, with a specified separator.

code -

```
my_list = ['Hello', 'World']
print(", ".join(my_list))  # Output: 'Hello, World'
```

Que. 3

String Slicing -

String slicing allows you to extract a portion of a string by specifying a start, end, and step index. It's a very powerful technique for working with strings

Syntax of slicing -

string[start:end:step]

- *start: The index where the slice begins (inclusive).*
- *end: The index where the slice ends (exclusive).*
- *step: The interval between each index.*