



IBM Developer SKILLS NETWORK

String Operations

Estimated time needed: 15 minutes

Objectives

After completing this lab you will be able to:

- Work with Strings
- Perform operations on String
- Manipulate Strings using indexing and escape sequences

Table of Contents

- What are Strings?
- Indexing
 - Negative Indexing
 - Slicing
 - Stride
 - Concatenate Strings
- Escape Sequences
- String Operations
- Quiz on Strings

What are Strings?

The following example shows a string contained within 2 quotation marks:

```
In [1]: # Use quotation marks for defining string
```

```
"Michael Jackson"
```

```
Out[1]: 'Michael Jackson'
```

We can also use single quotation marks:

```
In [2]: # Use single quotation marks for defining string
```

```
'Michael Jackson'
```

```
Out[2]: 'Michael Jackson'
```

A string can be a combination of spaces and digits:

```
In [3]: # Digits and spaces in string
```

```
'1 2 3 4 5 6 '
```

```
Out[3]: '1 2 3 4 5 6 '
```

A string can also be a combination of special characters :

```
In [4]: # Special characters in string
```

```
'@#_!$%^&*'
```

```
Out[4]: '@#_!$%^&*'
```

We can print our string using the print statement:

```
In [5]: # Print the string
```

```
print("hello!")
```

```
hello!
```

We can bind or assign a string to another variable:

```
In [6]: # Assign string to variable
```

```
name = "Michael Jackson"
```

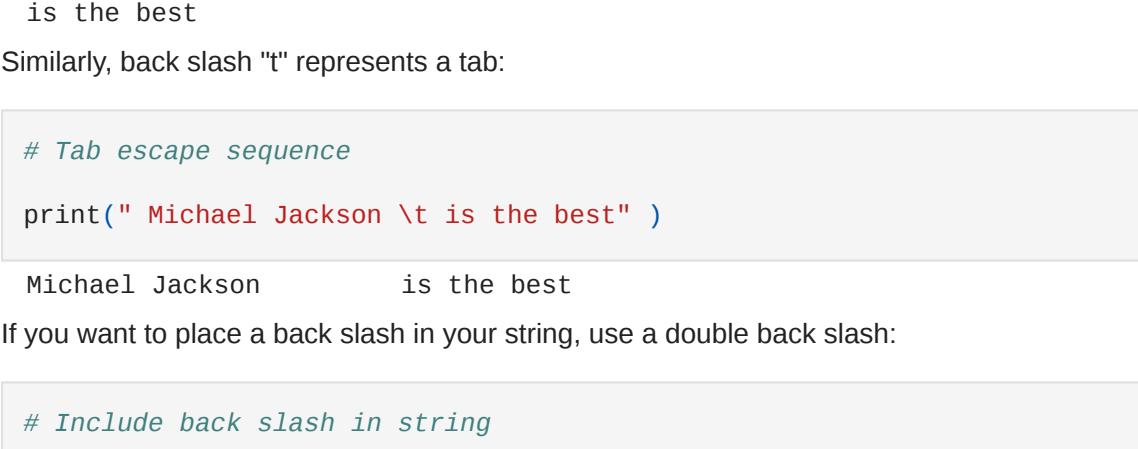
```
name
```

```
Out[6]: 'Michael Jackson'
```

Indexing

It is helpful to think of a string as an ordered sequence. Each element in the sequence can be accessed using an index represented by the array of numbers:

Name= "Michael Jackson"



The first index can be accessed as follows:

[Tip]: Because indexing starts at 0, it means the first index is on the index 0.

```
In [7]: # Print the first element in the string
```

```
print(name[0])
```

```
M
```

We can access index 6:

```
In [10]: # Print the element on index 6 in the string
```

```
print(name[6])
```

```
l
```

Moreover we can access the 13th index:

```
In [11]: # Print the element on the 13th index in the string
```

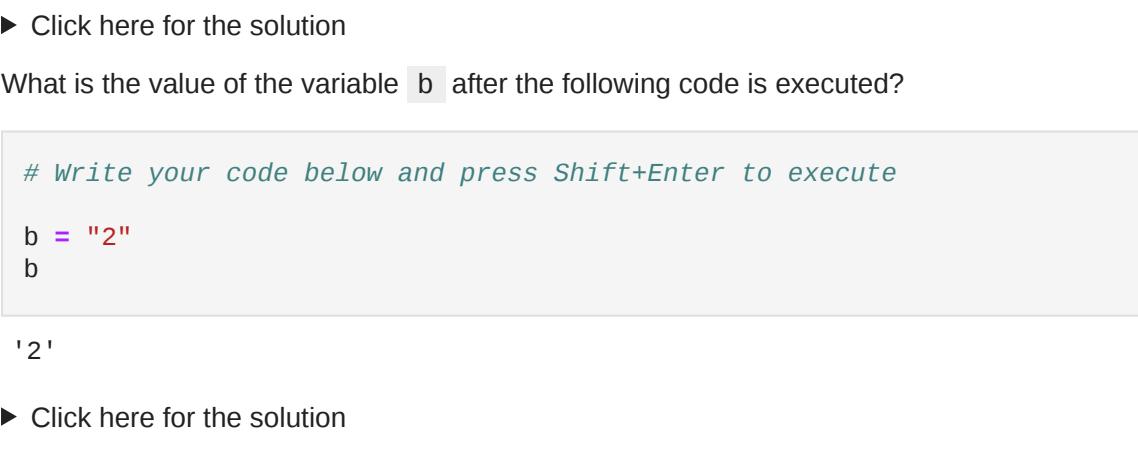
```
print(name[13])
```

```
o
```

Negative Indexing

We can also use negative indexing with strings:

Name= "Michael Jackson"



Negative index can help us to count the element from the end of the string.

The last element is given by the index -1:

```
In [12]: # Print the last element in the string
```

```
print(name[-1])
```

```
n
```

The first element can be obtained by index -15:

```
In [13]: # Print the first element in the string
```

```
print(name[-15])
```

```
M
```

We can find the number of characters in a string by using `len`, short for length:

```
In [14]: # Find the length of string
```

```
len("Michael Jackson")
```

```
15
```

Slicing

We can obtain multiple characters from a string using slicing, we can obtain the 0 to 4th and 8th to the 12th element:

Name= "Michael Jackson"



Name[0:4] =Mich

Name[8:12] =Jack

[Tip]: When taking the slice, the first number means the index (start at 0), and the second number means the length from the index to the last element you want (start at 1).

```
In [15]: # Take the slice on variable name with only index 0 to index 3
```

```
name[0:4]
```

```
Mich
```

```
Out[15]: 'Mich'
```

```
In [16]: # Take the slice on variable name with only index 8 to index 11
```

```
name[8:12]
```

```
Jack
```

```
Out[16]: 'Jack'
```

Stride

We can also input a stride value as follows, with the '2' indicating that we are selecting every second variable:

Name= "Michael Jackson"

Name[::2] =Mcalscn

```
In [17]: # Get every second element. The elements on index 1, 3, 5 ...
```

```
name[::2]
```

```
Mcalscn
```

```
Out[17]: 'Mcalscn'
```

We can also incorporate slicing with the stride. In this case, we select the first five elements and then use the stride:

```
In [20]: # Get every second element in the range from index 0 to index 4
```

```
name[0:5:2]
```

```
Mca
```

```
Out[20]: 'Mca'
```

Concatenate Strings

We can concatenate or combine strings by using the addition symbols, and the result is a new string that is a combination of both:

```
In [21]: # Concatenate two strings
```

```
statement = name + "is the best"
```

```
statement
```

```
Out[21]: 'Michael Jacksonis the best'
```

To replicate values of a string we simply multiply the string by the number of times we would like to replicate it. In this case, the number is three. The result is a new string, and this new string consists of three copies of the original string:

```
In [22]: # Print the string for 3 times
```

```
3 * "Michael Jackson"
```

```
Michael JacksonMichael JacksonMichael Jackson
```

```
Out[22]: 'Michael JacksonMichael JacksonMichael Jackson'
```

You can create a new string by setting it to the original variable. Concatenated with a new string, the result is a new string that changes from Michael Jackson to "Michael Jackson is the best".

```
In [23]: # Concatenate strings
```

```
name = "Michael Jackson"
```

```
name = name + " is the best"
```

```
name
```

```
Out[23]: 'Michael Jackson is the best'
```

Escape Sequences

Back slashes represent the beginning of escape sequences. Escape sequences represent strings that may be difficult to input. For example, back slash "n" represents a new line. The output is given by a new line after the back slash "n" is encountered:

```
In [24]: # New line escape sequence
```

```
print(" Michael Jackson \n is the best" )
```

```
Michael Jackson
```

```
is the best
```

Similarly, back slash "t" represents a tab:

```
In [25]: # Tab escape sequence
```

```
print(" Michael Jackson \t is the best" )
```

```
Michael Jackson
```

```
                  is the best
```

If you want to place a back slash in your string, use a double back slash:

```
In [26]: # Include back slash in string
```

```
print(" Michael Jackson \\ is the best" )
```

```
Michael Jackson \ is the best
```

We can also place an "r" before the string to display the backslash:

```
In [27]: # r will tell python that string will be display as raw string
```

```
print(r" Michael Jackson \ is the best" )
```

```
Michael Jackson \ is the best
```

Print out a backslash:

```
In [47]: # Write your code below and press Shift+Enter to execute
```

```
print("\\\\")
```

```
\\\
```

► Click here for the solution

Convert the variable `f` to uppercase:

```
In [48]: # Write your code below and press Shift+Enter to execute
```

```
f = "you are wrong"
```

```
f = f.upper()
```

```
f
```

```
Out[48]: 'YOU ARE WRONG'
```

► Click here for the solution

Consider the variable `g`, and find the first index of the sub-string snow:

```
In [49]: # Write your code below and press Shift+Enter to execute
```

```
g = "mary had a little lamb Little lamb, little lamb Mary had a little lamb \n Its fleece was white as snow And everywhere that Mary went Mary went, Mary went \n Everywhere that Mary went the lamb was sure to go"
```

```
g.find("snow")
```

```
95
```

► Click here for the solution

In the variable `g`, replace the sub-string Mary with Bob:

```
In [50]: # Write your code below and press Shift+Enter to execute
```

```
g.replace("Mary", "Bob")
```

```
g
```

```
Out[50]: 'Bob had a little lamb Little lamb, little lamb Bob had a little lamb Its fleece was white as snow And everywhere that Bob went Bob went \n Everywhere that Bob went the lamb was sure to go'
```

► Click here for the solution

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python.

Author

Joseph Santarcangelo

Change Log

Date (YYYY-MM-DD) Version Changed By Change Description

2022-01-10 2.1 Malika Removed the readme for GitShare