PY0101EN-1-2-Strings

March 5, 2022

1 String Operations

Estimated time needed: **15** minutes

1.1 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

```
<ul>
    <li>
        <a href="https://#strings">What are Strings?</a>
    </li>
    <li>
        <a href="https://#index">Indexing</a>
        <ul>
            <li><a href="https://neg/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_conte
            <li><a href="https://slice/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_co
            <li><a href="https://stride/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_c
            <li><a href="https://concat/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_c
        </ul>
    </li>
    <li>
        <a href="https://#escape">Escape Sequences</a>
    </li>
    <li>
        <a href="https://#operations">String Operations</a>
    </li>
    <li>
        <a href="https://#quiz">Quiz on Strings</a>
    </li>
</ul>
```

What are Strings?

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

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

"Michael Jackson"
```

We can also use single quotation marks:

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

'Michael Jackson'
```

A string can be a combination of spaces and digits:

[ ]:
```
# Digitals and spaces in string

'1 2 3 4 5 6 '
```

A string can also be a combination of special characters :

[ ]:
```
# Special characters in string

'@#2_#]&*^%$'
```

We can print our string using the print statement:

[ ]:
```
# Print the string

print("hello!")
```

We can bind or assign a string to another variable:

[ ]:
```
# Assign string to variable

name = "Michael Jackson"
name
```

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:

The first index can be accessed as follows:

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

print(name[0])
```

We can access index 6:

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

```

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

Moreover, we can access the 13th index:

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

print(name[13])
```

Negative Indexing

We can also use negative indexing with strings:

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

The last element is given by the index -1:

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

print(name[-1])
```

The first element can be obtained by index -15:

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

print(name[-15])
```

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

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

len("Michael Jackson")
```

Slicing

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

[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)

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

name[0:4]
```

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

name[8:12]
```

Stride

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

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

name[::2]
```

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

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

name[0:5:2]
```

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:

[ ]:
```
# Concatenate two strings

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

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:

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

3 * "Michael 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".

[ ]:
```
# Concatenate strings

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

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:

[ ]:
```
# New line escape sequence

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

Similarly, back slash "t" represents a tab:

[ ]:
```
# Tab escape sequence

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

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

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

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

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

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

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

String Operations

There are many string operation methods in Python that can be used to manipulate the data. We are going to use some basic string operations on the data.

Let's try with the method upper; this method converts lower case characters to upper case characters:

[ ]:
```
# Convert all the characters in string to upper case

a = "Thriller is the sixth studio album"
print("before upper:", a)
b = a.upper()
print("After upper:", b)
```

The method replace replaces a segment of the string, i.e. a substring with a new string. We input the part of the string we would like to change. The second argument is what we would like to exchange the segment with, and the result is a new string with the segment changed:

[ ]:
```
# Replace the old substring with the new target substring is the segment has␣
 ↪been found in the string

a = "Michael Jackson is the best"
b = a.replace('Michael', 'Janet')
b
```

The method find finds a sub-string. The argument is the substring you would like to find, and the output is the first index of the sequence. We can find the sub-string jack or el.

[ ]:
```
# Find the substring in the string. Only the index of the first elment of␣
 ↪substring in string will be the output

name = "Michael Jackson"
name.find('el')
```

[ ]:
```
# Find the substring in the string.

name.find('Jack')
```

If the sub-string is not in the string then the output is a negative one. For example, the string 'Jasdfasdasdf' is not a substring:

[ ]:
```
# If cannot find the substring in the string

name.find('Jasdfasdasdf')
```

Quiz on Strings

What is the value of the variable a after the following code is executed?

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

a = "1"
```

Click here for the solution

"1"

What is the value of the variable b after the following code is executed?

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

b = "2"
```

Click here for the solution

"2"

What is the value of the variable c after the following code is executed?

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

c = a + b
```

Click here for the solution

"12"

Consider the variable d use slicing to print out the first three elements:

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

d = "ABCDEFG"
```

Click here for the solution

```
print(d[:3])
```

```
# or

print(d[0:3])
```

Use a stride value of 2 to print out every second character of the string e:

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

e = 'clocrkr1e1c1t'
```

Click here for the solution

```
print(e[::2])
```

Print out a backslash:

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

Click here for the solution

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

or

print(r"\ ")
```

Convert the variable f to uppercase:

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

f = "You are wrong"
```

Click here for the solution

```
f.upper()
```

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

```
[ ]: # 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 \
Its fleece was white as snow And everywhere that Mary went Mary went, Mary went␣
 ↪\
Everywhere that Mary went The lamb was sure to go"
```

Click here for the solution

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

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

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

Click here for the solution

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

The last exercise!

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

1.2 Author

Joseph Santarcangelo

1.3 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-01-10	2.1	Malika	Removed the readme for GitShare
2020-11-11	2.1	Aije	Updated variable names to lowercase
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab

##