Python String

Summary: in this tutorial, you'll learn about Python string and its basic operations.

Introduction to Python string

A string is a series of characters. In Python, anything inside quotes is a string. And you can use either single or double quotes. For example:

```
message = 'This is a string in Python'
message = "This is also a string"
```

If a string contains a single quote, you should place it in double-quotes like this:

```
message = "It's a string"
```

And when a string contains double quotes, you can use the single quotes:

```
message = '"Beautiful is better than ugly.". Said Tim Peters'
```

To escape the quotes, you use the backslash (`\`). For example:

```
message = 'It\'s also a valid string'
```

The Python interpreter will treat the backslash character (\) special. If you don't want it to do so, you can use raw strings by adding the letter `r` before the first quote. For example:

```
message = r'C:\python\bin'
```

Creating multiline strings

To span a string multiple lines, you use triple-quotes """..."""" or '''...'''. For example:

```
help_message = '''
Usage: mysql command
    -h hostname
    -d database name
    -u username
    -p password
'''

print(help_message)
```

It'll output the following if you execute the program:

```
Usage: mysql command
    -h hostname
    -d database name
    -u username
    -p password
```

Using variables in Python strings with the f-strings

Sometimes, you want to use the values of variables (https://www.pythontutorial.net/python-basics/python-variables/) in a string.

For example, you may want to use the value of the `name` variable inside the `message` string variable:

```
name = 'John'
message = 'Hi'
```

To do it, you place the letter `f` before the opening quotation mark and put the brace around the variable name:

```
name = 'John'
message = f'Hi {name}'
print(message)
```

Python will replace the `{name}` by the value of the `name` variable. The code will show the following on the screen:

```
Hi John
```

The `message` is a format string, or f-string in short. Python introduced the f-string in version 3.6.

Concatenating Python strings

When you place the string literals next to each other, Python automatically concatenates (https://www.pythontutorial.net/python-string-methods/python-string-concatenation/) them into one string. For example:

```
greeting = 'Good ' 'Morning!'
print(greeting)
```

Output:

```
Good Morning!
```

To concatenate two string variables (https://www.pythontutorial.net/python-basics/python-string-concatenation/) , you use the operator `+` :

```
greeting = 'Good '
time = 'Afternoon'

greeting = greeting + time + '!'
print(greeting)
```

Output:

```
Good Afternoon!
```

Accessing string elements

Since a string is a sequence (https://www.pythontutorial.net/advanced-python/python-sequences/) of characters, you can access its elements using an index. The first character in the string has an index of zero.

The following example shows how to access elements using an index:

```
str = "Python String"
print(str[0]) # P
print(str[1]) # y
```

How it works:

- First, create a variable that holds a string `"Python String"` .
- Then, access the first and second characters of the string by using the square brackets `[]` and indexes.

If you use a negative index, Python returns the character starting from the end of the string. For example:

```
str = "Python String"
print(str[-1])  # g
print(str[-2])  # n
```

The following illustrates the indexes of the string `"Python String"` :

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+
| P | y | t | h | o | n |   | S | t | r | i | n | g |
+---+---+---+---+---+---+---+---+---+---+---+---+---+
  0   1   2   3   4   5   6   7   8   9   10  11  12
-13  -12  -11  -10 -9  -8  -7  -6  -5  -4  -3  -2  -1
```

Getting the length of a string

To get the length of a string, you use the `len()` function. For example:

```
str = "Python String"
str_len = len(str)
print(str_len)
```

Output:

```
13
```

Slicing strings

Slicing (https://www.pythontutorial.net/advanced-python/python-slicing/) allows you to get a substring from a string. For example:

```
str = "Python String"
print(str[0:2])
```

Output:

```
Py
```

The `str[0:2]` returns a substring that includes the character from the index 0 (included) to 2 (excluded).

The syntax for slicing is as follows:

```
string[start:end]
```

The substring always includes the character at the `start` and excludes the string at the `end` .

The `start` and `end` are optional. If you omit the `start` , it defaults to zero. If you omit the `end` , it defaults to the string's length.

Python strings are immutable

Python strings are immutable (https://www.pythontutorial.net/advanced-python/python-mutable-and-immutable/) . It means that you cannot change the string. For example, you'll get an error if you update one or more characters in a string:

```
str = "Python String"
str[0] = 'J'
```

Error:

```
Traceback (most recent call last):
  File "app.py", line 2, in <module>
```

```
    str[0] = 'J'
TypeError: 'str' object does not support item assignment</module>
```

When want to modify a string, you need to create a new one from the existing string. For example:

```
str = "Python String"
new_str = 'J' + str[1:]
print(new_str)
```

Output:

```
Jython String
```

Summary

- In Python, a string is a series of characters. Also, Python strings are immutable.
- Use quotes, either single-quotes or double-quotes to create string literals.
- Use the backslash character `\` to escape quotes in strings
- Use raw strings `r'...'` to escape the backslash character.
- Use f-strings to insert substitute variables in literal strings.
- Place literal strings next to each other to concatenate them. And use the + operator to concatenate string variables.
- Use the `len()` function to get the size of a string.
- Use the `str[n]` to access the character at the position n of the string `str` .
- Use slicing to extract a substring from a string.