

Digital Career Institute

Python Course - Texts in Python



Goal of the Submodule

The goal of this submodule is to help the learners work with texts in Python. By the end of this submodule, the learners should be able to understand:

- Different types of texts used in Python.
- The easy to handle and work with text including text concatenation and modification.
- How to use the String method.
- Learning the concepts of regular expressions and patterns in Python.

Topics

- **Introduction to Texts in Python**
- **Difference between characters and strings**
- **Working with Strings:**
 - Length and concat methods
 - Concat
- **Modifying Strings:**
 - Using a variety of methods including the strip, upper, lower, split and other methods
- **Using the String methods:**
 - Using a variety of methods including find, replace, split and other methods
- **Using regular Expressions with Python**
 - Introduction to basic Python Regular Expressions to search and replace text

Term	Definition
Char	A char is a single character, that is a letter, a digit, a punctuation mark, a tab, a space or something similar.
String	String is a sequence of characters, for e.g. "Hello" is a string of 5 characters.
Built-in method	A ready made functions to be used in Python

Introduction to Python Strings

A **String** is a Python data type to store textual data. **String** characteristics include:

- It is a collection of characters stored in an array format
- We use double quotes to create a **String**
 - If we want to span a String in multiple lines we can use the triple quote, this is ideal for long text.
 - Special characters like Tabs or Newlines can also be used within the triple quotes.
- We can use the backslash character (\) to escape quotes in **Strings**
- **Strings** are arrays of characters, thus we can slice them using the brackets and the character index locations

Difference between characters and Strings

Difference Between Characters and Strings

Digital Career Institute

Character	String
A character is a single letter, number, punctuation mark or symbol.	A string is a one-dimensional array of characters terminated by a null character.
Character is an element.	A string is a set of characters.
Single or double quotes are used to represent a character.	Single or double quotes are used to represent a string.
Character refers to a single letter, number, space.	String refers to a set of characters.

How to create a String?

String characteristics

**A string is a series of characters.
To create a String we will need to use single or double quotes**

**Strings are immutable data, this means that once created we cannot change it,
but we can reinitialize it!**

**When a string reference is reinitialized with a new value, it is creating a new
object rather than overwriting the previous value.**

Python String Methods

Method	Description
<code>string.capitalize()</code>	This method is used to capitalize a text (convert the first character to uppercase)
<code>string.upper()</code>	This method is used to transform a text into an upper (uppercase) text
<code>string.lower()</code>	This method is used to transform a text into a lower (lowercase) text

⇒ The methods do not accept any arguments, but they can be used using the dot operator

Python String Methods

Method	Description
<code>string.isalpha()</code>	This method is used to check if all the characters in the text are letters
<code>string.isdecimal()</code>	This method is used to check if all the characters in the text are decimals
<code>string.isnumeric()</code>	This method is used to check if all the characters in the text are numbers

⇒ The methods do not accept any arguments, but they can be used using the dot operator

Python String Methods

Method	Description
<code>string.find(value, start, end)</code>	This method is used to find a text inside the String. This is ideal when we have a phrase. The <code>value</code> is required, while the <code>start/end</code> are optional and denote the index of the String positions.
<code>string.index(value, start, end)</code>	This method is used to finds the first occurrence of the specified value and return its index. This works similar to find, but it returns an exception if the value is not found, so we need to handle it with a catch statement.

Python String Methods

Method	Description
<code>string.split(separator, maxsplit)</code>	This method is used to split a string in a list of words. The <code>separator</code> defines the base String to split with (e.g. dash), the whitespace is default. The <code>maxsplit</code> refers to how many splits to do and it is optional.
<code>string.replace(oldvalue, newvalue, count)</code>	This method is used to replace a given String with a another String. The <code>oldvalue</code> and <code>newvalue</code> are required. The <code>count</code> is optional and specifies how many occurrences of the given value we want to replace.

Python String Creation

- Python Strings can be created (a process that is also known as casting) from a different data type, including integers, floats and booleans. For this reason, we can use the in-built `str()` method.

Method	Description
<code>str(string, encoding='utf-8', errors='strict')</code>	<p>This method contracts a string version of the given object.</p> <ul style="list-style-type: none">• The <code>string</code> is the object to be returned as String, typically a number e.g. for concatenation.• The <code>encoding</code> refers to the encoding of the object. Defaults of UTF-8 when not provided.• The <code>errors</code> is the response when decoding fails. Defaults is 'strict'.

At the core of the lesson

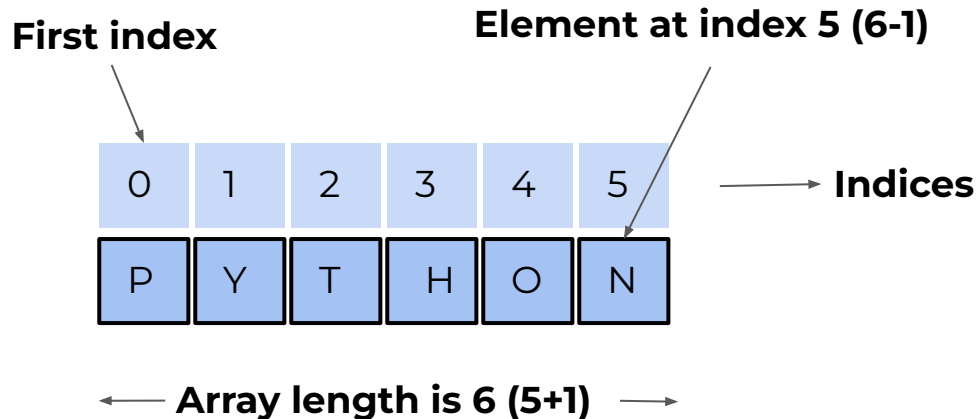
Lessons Learned:

- We know what is String and why it is immutable
- We know how to create a String and what is the difference between Strings and Characters in Python
- We know how to use different String manipulation methods including:
 - `capitalize()`
 - `upper()`
 - `lower()`
 - `isalpha()`
 - `isdecimal()`
 - `isnumeric()`
 - `find()`
 - `index()`
 - `split()`
 - `replace()`

Working with Strings

Strings are Arrays of Characters

- **len()**: is an in-built function of Python that returns the **length** of the String, or the size of the array of characters
 - An array in Python starts always from index 0
 - The length of the array equals to the last index value plus one
 - We can also say that the last index of the array equals to the length of the array minus one



Counting characters

- **count()**: is an in-built function of Python that returns the **number of time** a value appears in the String

String



```
str = "Hello World"
```



Method



```
str.count("l")
```



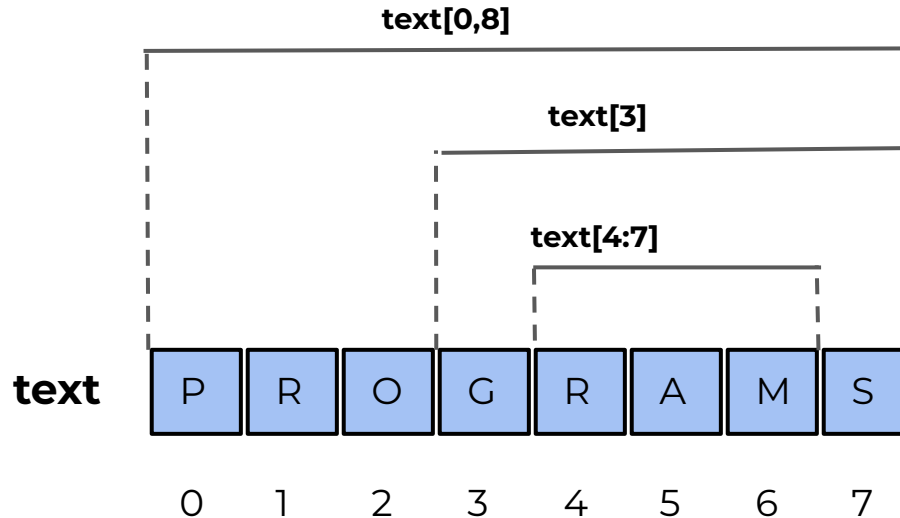
Result



```
"3"
```

Extracting Substrings: Slicing Strings

- **Substring Method:** A part of string is called substring. In other words, substring is a subset of another string. In case of substring **startIndex** is inclusive and **endIndex** is exclusive.



Strings Slicing options

- **You can slice Strings with different options**

1. `string[x:y]`

Extract a slice of characters starting from the `x` index value until the `y` index value.

2. `string[x:]`

Extract the last slice of characters starting from the `x` index value.

3. `string[:y]`

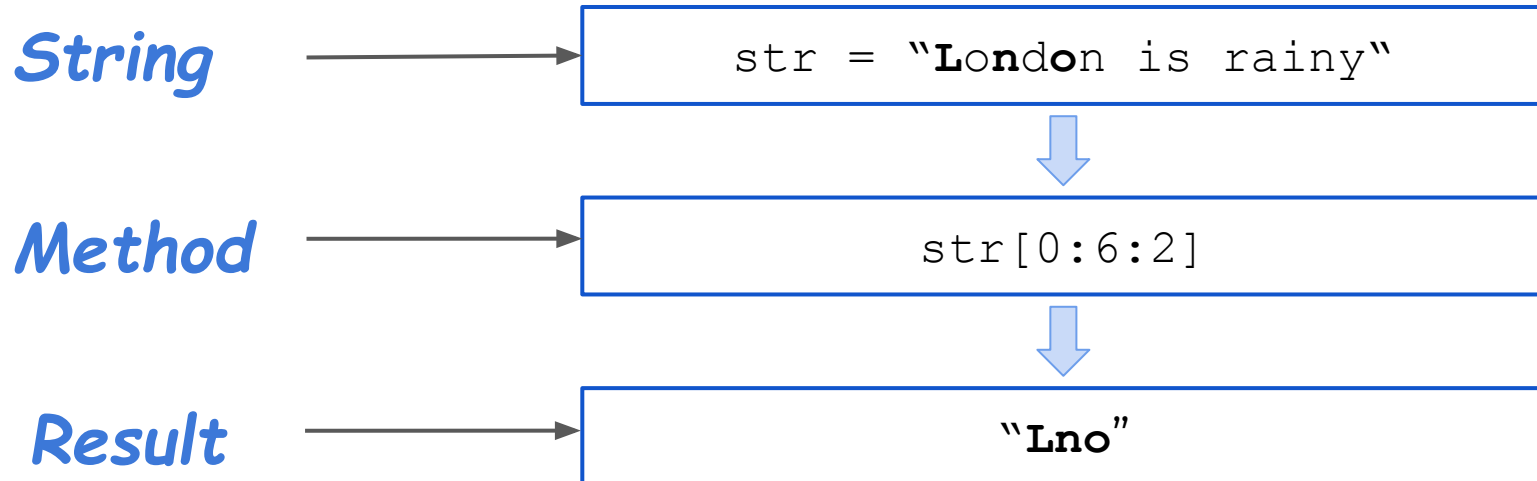
Extract the first slice of characters until the `y` index value.

Specifying Stride while Slicing Strings

- You can specify strides with different options

1. `String[x:y:stride]`

The new parameter `stride` refers to how many characters to move forward after the first character is retrieved from the string.



Python Trim (strip) Methods

- **Strip Methods:** Built-in function to remove leading and trailing whitespaces

String



```
str = "  Hello World!  "
```



Method



```
str.strip()
```



Result



```
"Hello World!"
```

- `rstrip` removes leading and trailing whitespaces from “**r**ight” side of string
- `lstrip` removes leading and trailing whitespaces from “**l**eft” side of string

Strings Concatenation

- **Using the plus operator (+):** Add a variable to another variable

String



```
str1 = "Hello"
```



String



```
str2 = "World"
```



Concatenate



```
str1 + " " + str2
```



Result




```
"Hello World!"
```


- **For Strings:**

The **plus** (+) works as a String concatenation operator.

- **For Numbers:**

The **plus** (+) works as a mathematical operator.

 **Tip:** Use the plus operator carefully! Do not try to concatenate a String with a number, in this case Python will return an error.

- **Using the multiplication operator (*):** Repeat a variable multiple times by multiplying with a number.

String



```
str = "Bye"
```



Method



```
str * 2
```



Result



```
"ByeBye"
```

String Concatenation: Formatting

- Another way of concatenating strings is using string formatting.
- String formatting can be done using the **%** operator.

Pattern



```
str = "%d. %s (%s) "
```



Format



```
str % (1, "Player", "Team")
```



Result



```
"1. Player (Team) "
```

At the core of the lesson

Python String methods:

- Using the variety of Python methods and concatenation/slicing operators we can manipulate Strings.
 - With the help of these methods, we can perform operations on string such as trimming, concatenating, converting, comparing, replacing strings etc.
- We used a variety of String methods for string manipulation.

Self Study



String Methods:

- multiline
- escaping
- Template