Code Academy

Chapter Two

USING PYTHON OBJECTS

Sections 2.4 – Using Strings as an Object

Lecture Goals

- To learn how to use Python objects
  - String as an object

Code Academy

Strings as an object

REFERENCE:

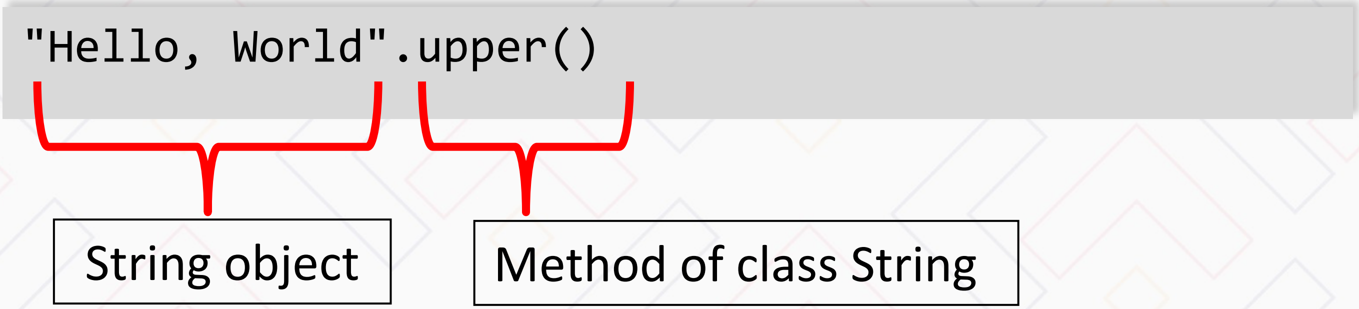
- Python for Everyone (2.4)

Objects and Methods

- In computer programming, an `object` is a software entity that represents a value with certain behavior.
  - The `value` can be `simple`, such as a `string`, or `complex`, like a `graphical window`, or `robot`, or `data file`, etc.
- The behavior of an object is given through its **`methods`**.
  - A method is a collection of programming instructions to carry out a specific task – similar to a function
- But unlike a **`function`**, which is a standalone operation, a **`method`** can only be applied to an object of the type for which it was defined (class).
  - **`Methods`** are specific to a type of object
  - **`Functions`** are general and can accept arguments of different types

Python Classes

- A class describes a set of objects with the same behavior.
  - For example, the str class describes the behavior of all strings
  - This class specifies how a string stores its characters, which methods can be used with strings, and how the methods are implemented.
  - For example, when you have a str object, you can invoke (call) the upper method:



```
"Hello, World".upper()
```

Calling methods

- Calling methods :

  object . method (**parameters**)

- You can apply the upper() method to any string, like this:

```
name = "John Smith”
# Sets uppercaseName to "JOHN SMITH”
uppercaseName = name.upper()
```

Some Useful String Methods

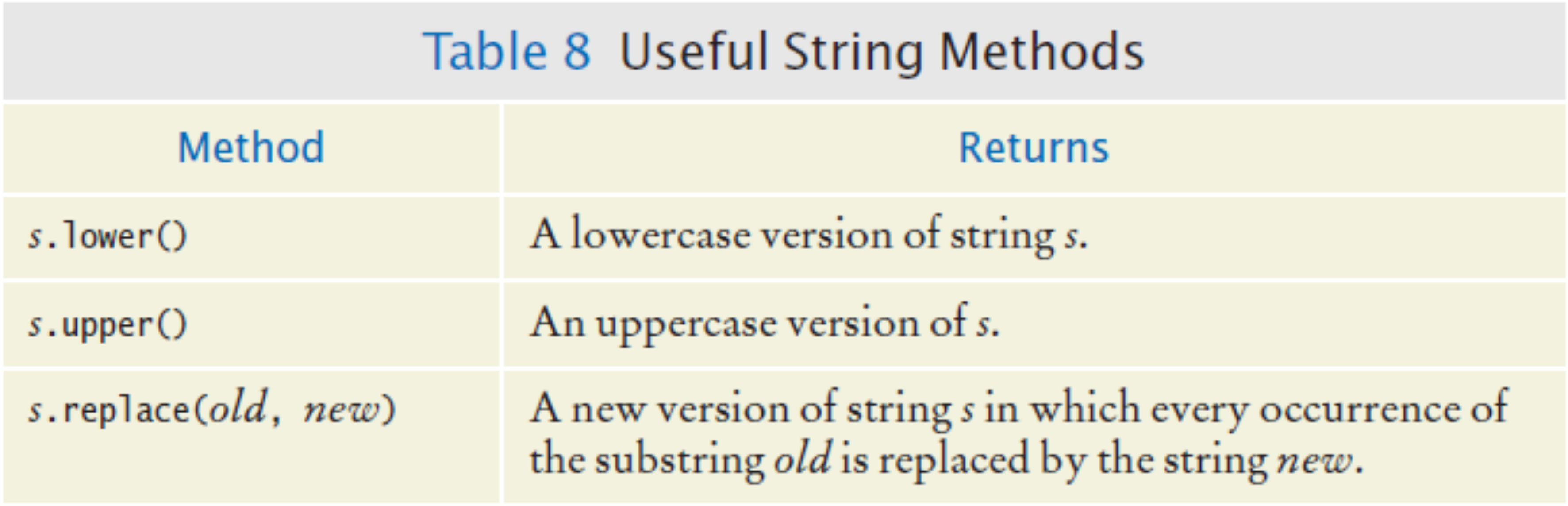


Table 8 Useful String Methods

Method	Returns
s.lower()	A lowercase version of string *s*.
s.upper()	An uppercase version of *s*.
s.replace(*old*, *new*)	A new version of string *s* in which every occurrence of the substring *old* is replaced by the string *new*.

Example: What is the result of each of the following Python statement?

```
title = "Python for everyone”
```

Statement	**Value**
`title[0]`	`"P"`
`title = title.replace("for everyone", "")`	`"Python "`
`start = title.upper()`	`"PYTHON "`
`title = title + "Program"`	`"Python Program"`

replace method

- The `replace()` method can take maximum of 3 parameters:
  - **old** - old substring you want to replace
  - **new** - new substring which would replace the old substring
  - **count** (optional) - the number of times you want to replace the old substring with the new substring
- If count is not specified, `replace()` method replaces all occurrences of the old substring with the new substring.

Example: What is the result of each of the following Python statement?

```
quote = 'Let it be, let it be, let it be'
```

Statement	Value
`result1 = quote.replace(‘let it’,’not’)`	`'Let it be, not be, not be’`
`result2 = quote.replace(‘let it’,’not’, 1)`	`'Let it be, not be, let it be’`

find method

- **The `find()` method returns the index of first occurrence of the substring (if found). If not found, it returns `-1`**
  - The `find()` method takes maximum of three parameters:
  - **The syntax of find method is**

    s.find(substring) or **s**.find(substring, start) or **s**.find(substring, start, end) :

Example: What is the result of each of the following Python statement?

```
quote = 'Let it be, let it be, let it be'
```

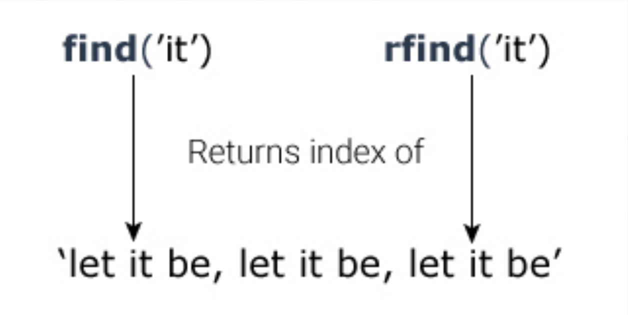
Statement	Value
`result1 = quote.find(‘let it’)`	11
`result2 = quote.find(‘small’)`	-1
`result3 = quote.find(‘let it’, 12)`	22

Code Academy

rfind method

- **The** `rfind()` returns the index of most rightmost matched substring of the string. It returns -1 if substring not found.
  - The `rfind()` method takes maximum of three parameters:
  - **The syntax of** `rfind` **method is**

    s.rfind(substring) or **s.**rfind(substring, start) or **s.**rfind(substring, start, end) :



find('it') **rfind**('it')

Returns index of

'let it be, let it be, let it be'

Example: What is the result of each of the following Python statement?

```
quote = 'Let it be, let it be, let it be'
```

Statement	Value
`result1 = quote.rfind('let it')`	22
`result2 = quote.rfind('small')`	-1
`result3 = quote.rfind('let it', 12)`	22

Code Academy

count method

- **The string** count() **method returns the number of occurrences of a substring in the given string.**
  - In simple words, count() method searches the substring in the given string and returns how many times the substring is present in it.
  - count() method only requires a single parameter for execution. However, it also has two optional parameters:

    substring - string whose count is to be found.

    start (Optional) - starting index within the string where search starts.

    end (Optional) - ending index within the string where search ends.

Example: What is the result of each of the following Python statement?

```
quote = 'Let it be, let it be, let it be'
```

Statement	**Value**
`result1 = quote.count(‘let it’)`	2
`result2 = quote.count(‘let it’,15)`	1
`result3 = quote.find(‘let it’, 15, 20)`	-1
`result4 = quote.count(‘ ‘)`	8

Summary:

- **Python** has quite a few methods that **string objects** can call to perform frequency occurring task (related to **string**)