String Methods

Table of contents

01

Case Conversion

Case Conversion

01 — `lower()` — Converts all characters to **lower case**

02 — `upper()` — Converts all characters to **upper case**

03 — `capitalize()` — Converts the **first** character to **upper case**.

04 — `title()` — Capitalize the **first** letter of **each word**

05 — `swapcase()` — **Inverts** case for all letters in string.

Case Conversion – lower()

Python	Output
txt = “HELLO, world." print (txt.lower())	hello, world.

Case Conversion – upper()

Python	Output

`txt = "hello, world."`
 	
`print (txt.upper())`
	`HELLO, WORLD.`

Case Conversion – capitalize()

Python	Output

`txt = "hello, world."`
 	
`print (txt.capitalize())`
	`Hello, world.`

Case Conversion – title()

Python	Output

`txt = "hello, world."`
 	
`print (txt.title())`
	`Hello, World.`

Case Conversion – swapcase()

Python	Output
txt = “Hello, World." print (txt.swapcase())	hELLO, wORLD.

02

Searching & Checking

Searching & Checking

01 — `find()` — Find the **position** of a substring

02 — `startswith()` — Check if a **string starts with** a given substring

03 — `endswith()` — Check if a **string ends with** a given substring.

04 — `replace()` — **Replace** a substring with another

Searching & Checking

05 — `rfind()` — Finds the **last occurrence** of the specified value and **returns -1** if the value is not found.

06 — `index()` — Check if a **string starts with** a given substring

07 — `rindex()` — Finds the **last occurrence** of the specified value but **raises an exception** if the value is not found.

Searching & Checking – find()

- Finds the **first occurrence** of the specified value.
- Returns **-1** if the value is not found.

Python	Output

`text = "Hello World"`
 	
`print(text.find("World"))`
`print(text.find("Python"))`
	
	6
	-1

Searching & Checking – startswith()

- Returns **True** if the string starts with the specified value, otherwise **False**.

Python	Output

`text = "Python is fun“`
 	
`print(text.startswith("Python"))`
`print(text.startswith("Java"))`
	
	`True`
	`False`

Searching & Checking – endswith()

- Returns **True** if the string ends with the specified value, otherwise **False**.

Python	Output

`text = "hello.txt“`
 	
`print(text.endswith(".txt"))`
`print(text.endswith(".jpg"))`
	
	`True`
	`False`

Searching & Checking – replace()

- Replaces a **specified phrase** with another **specified phrase**.

Python	Output

`text = "Hello World“`
 	
`print(text.replace("World", "Python"))`
	`Hello Python`

Searching & Checking – rfind()

- The **rfind**() method finds the **last occurrence** of the specified value.
- The **rfind**() method returns **-1** if the value is **not** found.

Python	Output

`text = "Hello World“`
 	
`print(text.rfind("o"))`
`print(text.rfind("o", 1, 5))`
	
	8
	4

Searching & Checking – index()

- Finds the **first** occurrence of the specified value.
- Raises an **exception** if the value is **not** found.

Python	Output

`text = "Hello World“`
 	
`print(text.index("World"))`
`print(text.index("ro"))`
	
	6
	ERROR!

Searching & Checking – rindex()

- Finds the **last** occurrence of the specified value.
- Raises an **exception** if the value is **not** found.

Python	Output

`text = "Hello World“`
 	
`print(text.rindex("o"))`
`print(text.rindex("ro"))`
	
	7
	ERROR!

03

Splitting & Joining

Splitting & Joining

01 — `split()` — Split a **string** into a **list**.

02 — `rsplit()` — starts **splitting** from the **right**.

03 — `partition()` — Splits the string into **three** parts at the **first occurrence** of a separator.

04 — `splitlines()` — Splits the string at **line breaks** and returns a list.

05 — `join()` — **Inverts** case for all letters in string.

Splitting & Joining – split()

- Splits a string into a list.
- **sep** - This is any delimeter, by default it is space.
- **maxsplit** - this is number of lines minus one

Python	Output

`text = "apple,banana,cherry"`
`print(text.split(","))`
`print(text.split(",", 1))`
	`['apple', 'banana', 'cherry']`
	`['apple', 'banana,cherry']`

Splitting & Joining – rsplit()

- The **rsplit**() method splits a string into a list, starting from the right.
- **sep** - This is any delimeter, by default it is space.
- **maxsplit** - this is number of lines minus one

Python	Output

`text = "apple,banana,cherry"`
`print(text.rsplit(","))`
`print(text.rsplit(",", 1))`
	`['apple', 'banana', 'cherry']`
	`['apple,banana', 'cherry']`

Splitting & Joining – partition()

- partition() splits string into 3 parts
- **Before Match | Match | After Match**

Python	Output
`text = “Hi Hello World Python"` `print(text.partition("World"))`	`('Hi Hello ', 'World', ' Python')`

Splitting & Joining – splitlines()

- **Splits** a string into a list
- Splitting is done at **line breaks**.

Python	Output

`text = "Hello World\n Python"`
`print(text.splitlines())`
`print(text.splitlines(True))`
	`['Hello World', ' Python']`
	`['Hello World\n', ' Python']`

Splitting & Joining – join()

- The **join**() method takes all items in an **iterable** and joins them into **one string**.
- **Syntax** `separator.join(iterable)`

separator: The string placed between elements of the iterable. (‘ ‘, , , -)

iterable: A sequence of strings (e.g., list, tuple etc) to join together.

Splitting & Joining – join()

- The **join**() method takes all items in an **iterable** and joins them into **one string**.

- **Syntax** `separator.join(iterable)`

Python	Output

`text = "Hello, World, Python"`
`print("".join(text))`
`print(" ".join(text))`
	`Hello, World, Python`
	
	`H e l l o ,  W o r l d ,  P y t h o n`

04

String Validation

String Validation

01 — `isalnum()` — Check if the string is **alphanumeric**.

02 — `isalpha()` — Check if the string contains **only alphabets**.

03 — `isdigit()` — Check if the string contains **only digits**.

04 — `isspace()` — Check if the string contains **only spaces**.

String Validation

05 — `islower()` — Check if the string is **all lowercase**.

06 — `isupper()` — Check if the string is **all uppercase**.

07 — `istitle()` — Check if the string is **all title case**.

String Validation – isalnum()

- The **isalnum**() method returns True if all the characters are alphanumeric, meaning **alphabet letter (a-z)** and **numbers (0-9)**

- Characters that are not alphanumeric: !#%&? etc.

Python	Output
`print("Python123".isalnum())` `print("Python@123".isalnum())`	`True` `False`

String Validation – isalpha()

- Returns **True** if all the characters are **alphabet letters (a-z)** otherwise **False**.

Python	Output
`print("Python".isalpha())` `print("Python3".isalpha())`	`True` `False`

String Validation – isdigit()

- Returns **True** if all the characters are **digits (0-9)**, otherwise **False**.

Python	Output
`print("12345".isdigit())` `print("123abc".isdigit())`	`True` `False`

String Validation – isspace()

- Returns **True** if all the characters in a string are whitespaces, otherwise **False**.

Python	Output
`print("   ".isspace())` `print(" Hello ".isspace())`	`True` `False`

String Validation – islower()

- Returns **True** if all the characters are in **lower case**, otherwise **False**.

Python	Output
`print("hello".islower())` `print("HEllo".islower())`	`True` `False`

String Validation – isupper()

- Returns **True** if all the characters are in **upper case**, otherwise **False**.

Python	Output
`print("HELLO".isupper())` `print("Hello".isupper())`	`True` `False`

String Validation – istitle()

- Returns **True** if all words in a text are capitalized, otherwise False.

Python	Output
`print("Hello".istitle())` `print("HELLO".istitle())`	`True` `False`

05

String Formatting

String Formatting

01 — `strip()` — Removes any **leading**, and **trailing whitespaces**.

02 — `lstrip()` — Removes any **leading** space/characters

03 — `rstrip()` — Removes any **trailing** space/characters

04 — `center()` — Returns a **centered** string

String Formatting

05 — `ljust()` — Returns a **left justified** version of the string

06 — `rjust()` — Returns a **right justified** version of the string

07 — `format()` — Returns the **formatted** string.

08 — `count()` — Returns the **number of times** a specified value **occurs** in a string

String Formatting – strip()

- Removes any **leading, and trailing whitespaces**.
- **Leading**: At the start of the string
- **Trailing**: At the end of the string
- Can specify **characters** to remove
- **Default**: Removes all whitespace

Python	Output
`print("##hello##".strip('#'))` `print("  hello  ".strip())`	`hello` `hello`

String Formatting – lstrip()

- Removes any **leading whitespaces**.
- **Leading**: At the start of the string
- Can specify **characters** to remove
- **Default**: Removes all whitespace

Python	Output
`print("#@#hello##".lstrip("#@"))` `print("  hello  ".lstrip())`	`hello##` `hello••`

String Formatting – rstrip()

- Removes any **trailing whitespaces**.
- **Trailing**: At the end of the string
- Can specify **characters** to remove
- **Default**: Removes all whitespace

Python	Output
`print("##hello##".rstrip('#'))` `print("  hello  ".rstrip())`	`#@#hello` `  hello`

String Formatting – center()

- **Center align** the string, using a **specified character** (space is default) as the fill character.

Python	Output

`txt = "hello"`
`print(txt.center(10))`
`print(txt.center(10, "_"))`
	`  hello`
	`__banana__`

String Formatting – ljust()

- **Left align** the string, using a **specified character** (space is default) as the fill character.

Python	Output

`txt = "hello"`
`print(txt.ljust(10))`
`print(txt.ljust(10, "_"))`
	`hello`
	`hello_____`

String Formatting – rjust()

- **Right align** the string, using a **specified character** (space is default) as the fill character.

Python	Output

`txt = "hello"`
`print(txt.rjust(10))`
`print(txt.rjust(10, "_"))`
	`     hello`
	`_____hello`

String Formatting – format()

- **Formats** values & **inserts** into {} placeholders
- Supports **multiple** values
- Can use **indexed** or **named** placeholders

Python

```
name = “Ravi"
age = 25
print(“I am {} and I am {} years old.".format(name, age))
```

Output

```
I am Ravi and I am 25 years old.
```

String Formatting – count()

- Returns the **number of times** a specified value **appears** in the string.

Python	Output

`text = "hello hello hello”`
`txt = "first hello, second hello"`
 	
`print(text.count("hello"))`
`print(txt.count("hello", 10, 30))`
	
	3
	1

Summary Table

Category	Methods			
Case Conversion	lower()	upper()	capitalize()	title()
	swapcase()			
Searching & Checking	find()	startswith()	endswith()	replace()
	rfind()	index()	rindex()	
Splitting & Joining	split()	rsplit()	join()	
	partition()	splitlines()		
Validation	isalnum()	isalpha()	isdigit(),	isspace()
	islower()	isupper()	istitle()	
Formatting	Strip()	lstrip()	rstrip()	center()
	ljust()	rjust()	format()	count()

“If you focus on these **32 methods**, you will handle almost all string operations **99.9%** of the time.”

String Methods

Download Link in **Description**

WATCH

Level up your coding with each episode in this focused Python series.

Next Video!

Practice Set Solution