STRINGS METHODS :-

PRESENTED BY :-KARAN V DAYAL

ASSISTANT PROFESSOR

CSE DEPARTMENT

Methods

Python has a set of built-in reusable utilities. They simplify the most commonly performed operations are:

String Methods

- isdigit()
- strip()
- lower()
- upper()
- startswith()
- endswith()
- replace()
- and more...

isdigit():-

Isdigit

Syntax:

str_var.isdigit()

Gives True if all the characters are digits. Otherwise, False

Code example:-

```
is_digit = "4748".isdigit()
print(is_digit)
```

output:-

True

Strip():-

Strip

Syntax:

str_var.strip()

Removes all the leading and trailing spaces from a string.

Code examples:-

```
mobile = " 9876543210 "
mobile = mobile.strip()
print(mobile)
```

Strip specific characters:-

Syntax:

str_var.strip(chars)

We can also specify characters that need to be removed.

Code examples:-

```
 name = "Ravi."
name = name.strip(".")
print(name)
```

Output:-Ravi

Strip-Mutliple characters:-

Removes all spaces, comma(,) and full stop(.) that lead or trail the string.

Code example:-

```
 name = ", .. ,, ravi ,, .. ."
name = name.strip(" ,.")
print(name)
 output:- ravi
```

Replace():-

Syntax:

str_var.replace(old,new)

Gives a new string after replacing all the occurrences of the old substring with the new substring.

Code Example:-

```
sentence = "teh cat and teh dog"
sentence = sentence.replace("teh","the")
print(sentence)
```

Output:- the cat and the dog

Startswith():-

Syntax:

str_var.startswith(value)

Gives True if the string starts with the specified value. Otherwise, False

Code example:-

```
url = "https://onthegomodel.com"
is_secure_url = url.startswith("https://")
print(is_secure_url)
```

Output:- True

EndsWith():-

Syntax:

str_var.endswith(value)

Gives True if the string ends with the specified value. Otherwise, False

Code example:-

```
gmail_id = "rahul123@gmail.com"
is_gmail = gmail_id.endswith("@gmail.com")
print(is_gmail)
```

Ouput: True

Upper():-

Syntax:

str_var.upper()

Gives a new string by converting each character of the given string to uppercase.

Code Example:-

```
name = "ravi"
print(name.upper())
```

Output:-RAVI

Lower():-

Syntax:

str_var.lower()

Gives a new string by converting each character of the given string to lowercase.

Code example:-

```
name = "RAVI"
print(name.lower())
```

Output:- ravi

Note:-

The upper()

and lower()

methods in Python work only on **alphabetic characters** and on **special characters** (like punctuation marks, symbols, digits, etc.), these methods do **not have any effect**.

Code Examples:-

```
s = "Hello, John! 123“
print(s.upper())  # Output: "HELLO, JOHN! 123“
print(s.lower())  # Output: "hello, john! 123"
```

More on Strings:-

- Classification Methods
  - isalpha()
  - isdecimal()
  - islower()
  - isupper()
  - isalnum()
- Case Conversion Methods
  - capitalize()
  - title()
  - swapcase()
- Counting and Searching Methods
  - count()
  - index()
  - rindex()
  - find()
  - rfind()

Classification Methods:-

1. Classification Methods

These methods are used to check the characteristics of individual characters in a string.

Isalpha:-

1.1 Isalpha

Syntax:

str_var.isalpha()

Gives True if all the characters are alphabets. Otherwise, False

Code Example:-

Example1:- is_alpha = "Rahul".isalpha()

print(is_alpha)

Output:-True

Example 2:-

is_alpha = "Rahul@123".isalpha()

print(is_alpha)

Output:- False

1.2 Isdecimal:-

Syntax:

str_var.isdecimal()

Gives True if all the characters are decimals. Otherwise, False

Example 1:-

```
is_decimal = "9876543210".isdecimal()
print(is_decimal)
Output: True
```

Example 2:-

```
is_decimal = "123.R".isdecimal()
print(is_decimal)
Output:- False
```

1.3 Islower

Syntax:

str_var.islower()

Gives True if all letters in the string are in lowercase. Otherwise, False.

Example-1:-

```
is_lower = "hello ravi!".islower()
print(is_lower)
```

Output:- True

Example 2:-

```
is_lower = "Hello Ravi!".islower()
print(is_lower)
```

Output:- False

1.4 IsUpper:-

Syntax:

str_var.isupper()

Gives True if all letters in the string are in uppercase. Otherwise, False

Code Example 1:-

```
is_upper = "HELLO RAVI!".isupper()
print(is_upper)
```

Output:- True

Example 2:-

```
is_upper = "hELLO rAVI!".isupper()
print(is_upper)
```

Output:- False

Isalnum:-

Syntax:

str_var.isalnum()

Gives True if the string is alphanumeric (a letter or a number). Otherwise, False

Example 1:-

```
is_alnum = "Rahul123".isalnum()
print(is_alnum)
```

Output:- True

Example 2:-

```
is_alnum = "Rahul".isalnum()
print(is_alnum)
```

Output:- True

Example 3:-

```
is_alnum = "Rahul@123".isalnum()
print(is_alnum)
```

Output:- False

2.CASE CONVERSION METHODS:-

These methods are used to change the case of a string.

2.1 Capitalize

Syntax:

str_var.capitalize()

Gives a new string after converting the first letter in the string to uppercase and all other letters to lowercase.

CODE EXAMPLE:-

```
capitalized = "the Planet Earth".capitalize()
print(capitalized)
```

Ouput:- The planet earth

2.2 Title

Syntax:

str_var.title()

Gives a new string after converting the first letter of every word to uppercase.

If a word contains a number or a special character, the first letter after that is converted to uppercase.

Example 1:-

```
title_case = "the planet earth".title()
print(title_case)
```

Output:- The Planet Earth

Example 2:-

```
title_case = "my_name#is john1doe and i love python".title()
print(title_case)
```

Output:- My_Name#Is John1Doe And I Lov

2.3 Swapcase:-

Syntax:

str_var.swapcase()

Gives a new string after converting the uppercase letters to lowercase and vice-versa.

Example:-

```
swapped = "mY nAME IS rAVI".swapcase()

print(swapped)
```

Output:- My Name is Ravi

3. Counting and Searching Methods

These methods are used to count the occurrences of a substring in a string and to find the position of a substring in a string.

3.1 Count:-

Syntax:

str_var.count(str, start_index, end_index)

Here, the start_index and the end_index are optional.

The count() method gives the number of times the specified string str appears in the string. It searches the complete string as default.

If start_index and end_index are provided, it searches between these indices. The end_index is not included.

Examples:-

Coding example 1:-

```
text = "Hello, world!“
letter_count = text.count("l")
print(letter_count)
```

Output:- 3

Coding example 2:-

```
text = "Hello, world!“
letter_count = text.count("l", 2, 10)
print(letter_count)
```

Output:- 2

3.2 Index:-

Syntax:

str_var.index(str, start_index, end_index)

Here, the start_index and the end_index are optional.

The index() method gives the index at the first occurrence of the specified string str.It results in an error if the specified string str is not found.

The index() method searches the complete string as default. If start_index and end_index are provided, it searches between these indices. The end_index is not included.

Examples:-

Example 1:-

```
text = "I have a spare key, if I lose my key“
word_index = text.index("key")
print(word_index)
```

Output:- 15

Example 2:-

```
text = "coo coo“
word_index = text.index("co", 3, 6)
print(word_index)
```

Output:- 4

Example:-

Example 3:-

```
text = "coo coo“

word_index = text.index("ha")

print(word_index)
```

Output:- ValueError: substring not found

3.3 rIndex:-

Syntax:

str_var.rindex(str, start_index, end_index)

Here, the start_index and the end_index are optional.

The rindex() method gives the index at the last occurrence of the specified string str.It results in an error if the specified string strisis not found.

The rindex() method searches the complete string as default.

If start_index and end_index are provided, it searches between these indices. The end_index is not included.

Examples:-

Example 1:-

```
text = "I have a spare key, if I lose my key“
word_index = text.rindex("key")
print(word_index)
```

Output:- 33

Example 2:-

```
text = "coo coo coo“
word_index = text.rindex("co", 3, 10)
print(word_index)
```

Output:- 8

Example 3:-

```
text = "coo coo“
word_index = text.rindex("ha")
print(word_index)
```

Output:- ValueError: substring not found

3.4 Find

Syntax:

str_var.find(str, start_index, end_index)

Here, the start_index and the end_index are optional.

The find() method gives the index at the first occurrence of the specified string str.If the specified string str is not found, it returns -1.

The find() method searches the complete string as default. If start_index and end_index are provided, it searches between these indices. The end_index is not included.

It works similarly to the index()method. The only difference is that the index() method results in an error if the specified string is not found, while find() does not.

Coding Examples:-

Example 1:-

```
text = "I have a spare key, if I lose my key“
word_index = text.find("key")
print(word_index)
```

Output:- 15

Example 2:-

```
text = "coo coo“
word_index = text.find("co", 3, 6)
print(word_index)
```

Output:- 4

Example 3: -

```
text = "coo coo“
word_index = text.find("ha")
print(word_index)
```

Output:- -1

3.5 rfind:-

Syntax:

str_var.rfind(str, start_index, end_index)

Here, the start_index and the end_indexare optional.

The rfind() method gives the index at the last occurrence of the specified string str.If the specified string str is not found, it returns -1.The rfind() method searches the complete string as default.

If start_index and end_index are provided, it searches between these indices. The end_index is not included.

It works similarly to the rindex()method.

The only difference is that the rindex() method results in an error if the specified string is not found, while rfind()does not.

Examples:-

Example 1:-

```
text = "I have a spare key, if I lose my key“
word_index = text.rfind("key")
print(word_index)
```

Ouput:- 33

Example 2:-

```
text = "coo coo coo“
word_index = text.rfind("co", 3, 10)
print(word_index)
```

Output:- 8

Examples

Example 3:-

```
text = "coo coo“
word_index = text.rfind("ha")
print(word_index)
```

Output:- -1

lstrip() and rstrip():-

In Python, `rstrip()` and `lstrip()` are string methods used to remove unwanted characters (whitespace by default) from the end and the start of a string, respectively.

`lstrip()`

- **Purpose:** Removes leading (left) whitespace or specified characters from the beginning of the string.
- **Syntax:** `string.lstrip([chars])`
  - **chars (optional):** A string specifying a set of characters to be removed. If not provided, it defaults to whitespace (spaces, tabs, newlines).

`rstrip()`

- **Purpose:** Removes trailing (right) whitespace or specified characters from the end of the string.
- **Syntax:** `string.rstrip([chars])`
  - **chars (optional):** A string specifying a set of characters to be removed. If not provided, it defaults to whitespace.

Examples:-

>Using lstrip() to remove leading spaces:-

```
 text = "   Hello, World!"
result = text.lstrip()
print(result)  # Output: 'Hello, World!’
```

> Using rstrip() to remove trailing spaces:-

```
 text = "Hello, World!   "
result = text.rstrip()
print(result)  # Output: 'Hello, World!’
```

>Removing specific characters using lstrip():-

```
text = "###Hello, World!"
result = text.lstrip("#")
print(result)  # Output: 'Hello, World!'
```

Examples:-

Removing specific characters using rstrip():-

```
text = "Hello, World!***"
result = text.rstrip("*")
print(result)  # Output: 'Hello, World!’
```

>Using both to remove leading and trailing characters:-

```
text = ">>>Python is fun<<<"
result = text.lstrip(">").rstrip("<")
print(result)  # Output: 'Python is fun’
```

>Removing characters like newline or tabs:-

```
text = "\n\tHello, World!\t\n"
result = text.lstrip().rstrip()  # Removes newline and tab characters
print(result)  # Output: 'Hello, World!'
```