**String Methods in Python**

String methods are built-in functions in Python that allow you to perform various operations on strings. Here’s a simple explanation of some common string methods along with their syntax and examples.

1. **upper()**
   * **Definition**: Converts all characters in a string to uppercase.
   * **Syntax**: string.upper()
   * **Example**:

text = "hello"

print(text.upper()) # Output: HELLO

str = "Hi all"

print(str.upper())

output: HI ALL

NOTE: upper()method takes no arguments

1. **lower()**
   * **Definition**: Converts all characters in a string to lowercase.
   * **Syntax**: string.lower()
   * **Example**:

text = "HELLO"

print(text.lower()) # Output: hello

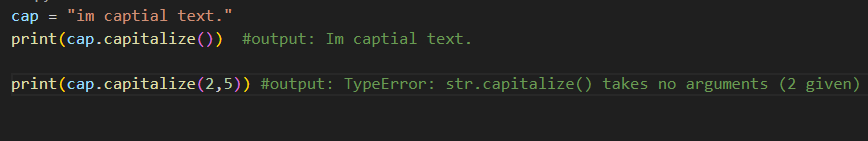
lw = " Hi ALl. Ilam LowErcAsE."

print(lw.lower())  # Output: hi all. ilam lowercase.

1. **capitalize()**
   * **Definition**: Capitalizes the first letter of the string.
   * **Syntax**: string.capitalize()
   * **Example**:

text = "hello"

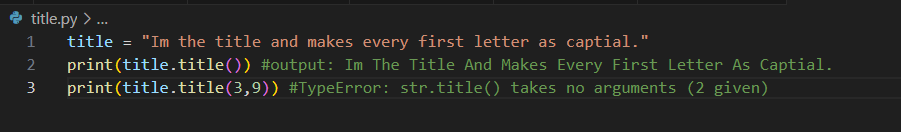
print(text.capitalize()) # Output: Hello



1. **title()**
   * **Definition**: Capitalizes the first letter of every word in the string.
   * **Syntax**: string.title()
   * **Example**:

text = "hello world"

print(text.title()) # Output: Hello World



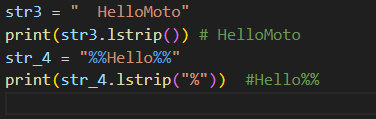
1. **strip()**
   * **Definition**: Removes leading and trailing whitespaces from the string.
   * **Syntax**: string.strip([chars])
   * **Example**:

text = " hello "

print(text.strip()) # Output: hello

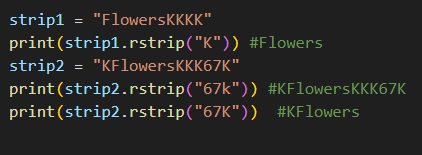
1. lstrip(): Removes characters only from the left (start) of the string.

Ex:



1. rstrip():Removes characters only from the right (end) of the string.

Ex:

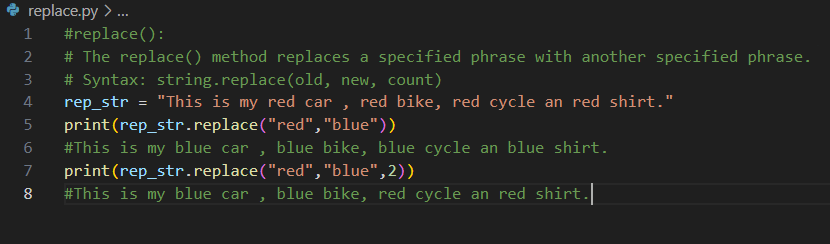


1. **replace()**
   * **Definition**: Replaces a substring with another substring in the string.
   * **Syntax**: string.replace(old, new)
   * **Example**:

text = "hello world"

print(text.replace("world", "there")) # Output: hello there

ex:



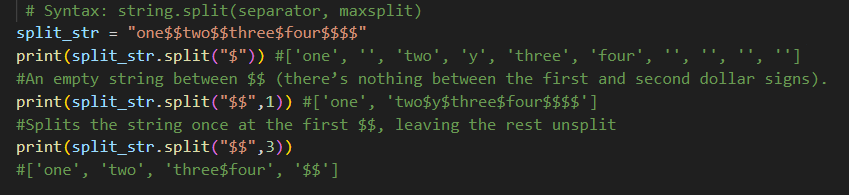
1. **split()**
   * **Definition**: Splits the string into a **list** of substrings based on a delimiter.
   * **Syntax**: string.split(delimiter)
   * **Example**:

python

Copy code

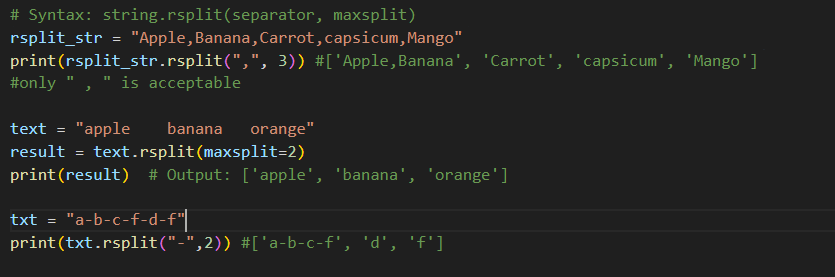
text = "hello world"

print(text.split(" ")) # Output: ['hello', 'world']



1. rsplit(): split() splits from the **left** and rsplit() splits from the **right**.

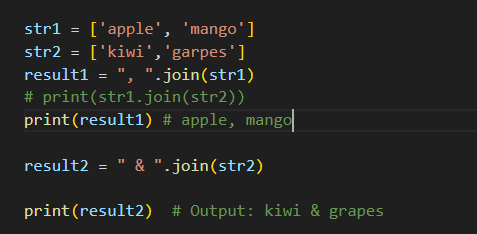
Ex:



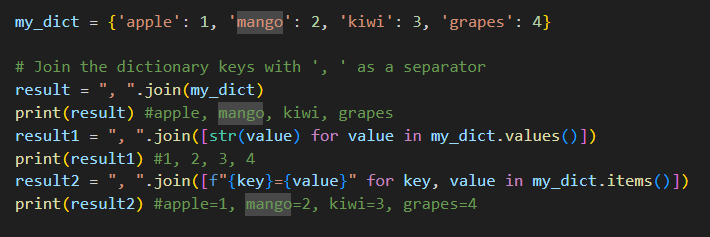
1. **join()**
   * **Definition**: Joins a list of strings into a single string, with a specified separator.
   * **Syntax**: separator.join(list)
   * **Example**:

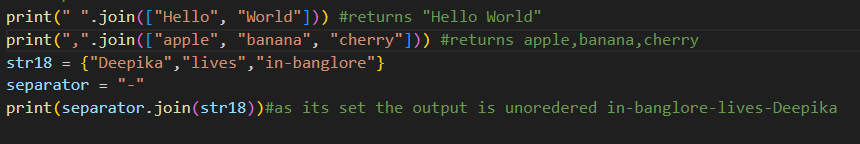
words = ["hello", "world"]

print(" ".join(words)) # Output: hello world



Ex with dict

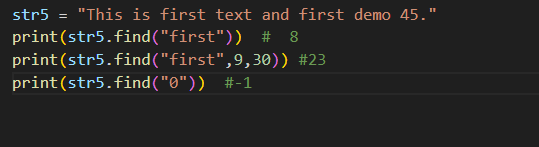




1. **find()**
   * **Definition**: Returns the index of the first occurrence of a substring, or -1 if not found.
   * **Syntax**: string.find(substring)
   * **Example**:

text = "hello world"

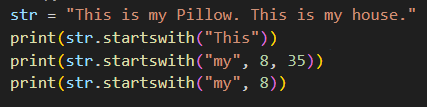
print(text.find("world")) # Output: 6



1. **startswith()**
   * **Definition**: Checks if the string starts with the specified substring.
   * **Syntax**: string.startswith(substring)
   * **Example**:

text = "hello world"

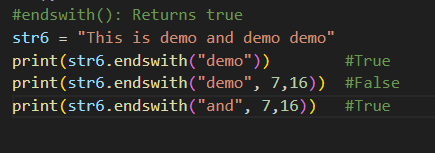
print(text.startswith("hello")) # Output: True



1. **endswith()**
   * **Definition**: Checks if the string ends with the specified substring.
   * **Syntax**: string.endswith(substring)
   * **Example**:

text = "hello world"

print(text.endswith("world")) # Output: True



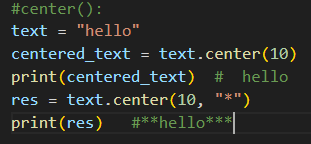
1. Centre:

The center() method in Python is used to center-align a string by padding it with a specified character (or spaces by default) to a certain length.

### ****Syntax****

string.center(width[, fillchar])

example:



16.Count():The count() method in Python is used to count the number of occurrences of a substring within a string.

### ****Syntax****

string.count(substring[, start[, end]])

* **substring**: The string or character you want to count the occurrences of.
* **start** (optional): The starting index from where the search begins. Default is the beginning of the string.
* **end** (optional): The ending index where the search ends. Default is the end of the string.

### ****Return Value****

* It returns an integer that represents how many times the substring appears in the string.

### ****Example 1: Basic Usage****

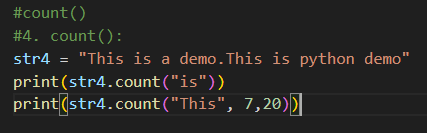
text = "hello world"

count = text.count("o")

print(count)

#### **Output:2**

Ex2:



17: index():The index() method in Python is used to find the first occurrence of a specified substring in a string. It returns the index of the first character of the substring. If the substring is not found, it raises a ValueError.

### ****Syntax****

string.index(substring[, start[, end]])

* **substring**: The string you are searching for.
* **start** (optional): The position to start the search from. By default, it starts at the beginning of the string.
* **end** (optional): The position to end the search. By default, it ends at the last character of the string.

### ****Return Value****

* Returns the index of the first occurrence of the substring.
* Raises a ValueError if the substring is not found.

### ****Example 1: Basic Usage****

text = "hello world"

index\_position = text.index("o")

print(index\_position)

#### **Output:**

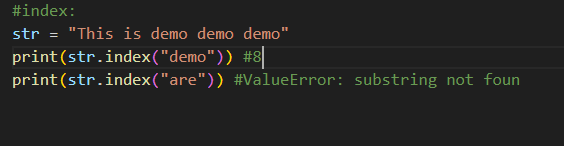
4

#index():

str8 = "This is demo demo demo"

print(str8.index("demo"))

print(str8.index("demo",15,30))



Top of Form

### ****Difference Between**** find() ****and**** index()

* The find() method is similar to index() but **does not raise an error** if the substring is not found; instead, it returns -1.
* index() raises a ValueError if the substring is not found.

18. rindex():

# The rindex() method returns the highest index of the substring if it is found in the string

str24 = "This is test. This is another test"

print(str24.rindex("test")) #30

str\_24 = "This is test. This is another test.yet another test"

print(str\_24.rindex("test"))

Bottom of Form