Understanding Strings and String Methods in Python

A COMPREHENSIVE GUIDE

What is a String in Python?

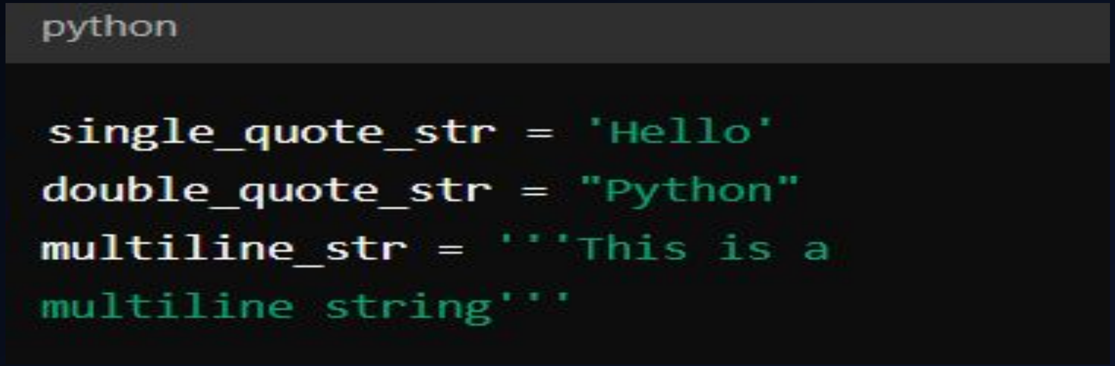
•A string is a sequence of characters enclosed within single, double, or triple quotes.

•Example: 'Hello, World!', "Python", '''Multiline String'‘’.

•Strings are immutable, meaning they cannot be changed after creation.

Creating Strings in Python

- Single quotes: 'Hello'
- Double quotes: "Python"
- Triple quotes for multiline: '''This is a multiline string'''

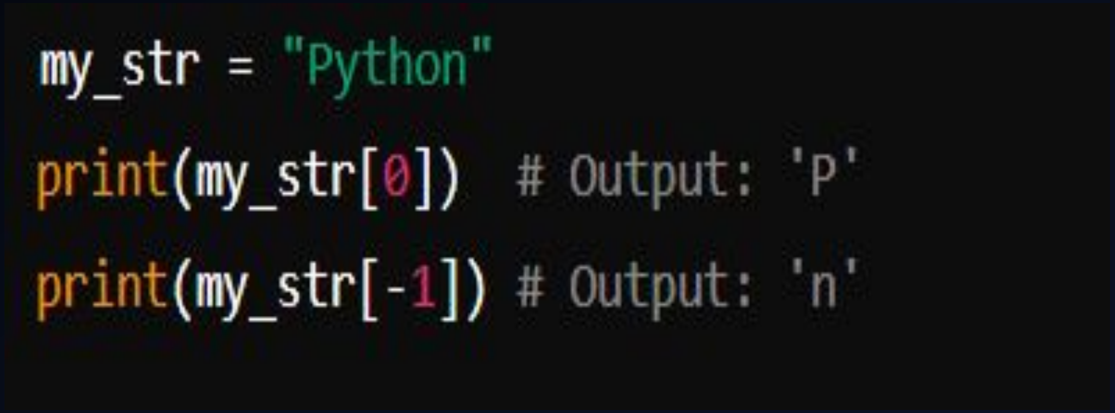


python

```python
single_quote_str = 'Hello'
double_quote_str = "Python"
multiline_str = '''This is a
multiline string'''
```

Accessing Characters in a String

- Strings are indexed, allowing access to characters using indices.
- Index starts at 0 and go in both direction positive and negative.



```
my_str = "Python"
print(my_str[0])  # Output: 'P'
print(my_str[-1]) # Output: 'n'
```

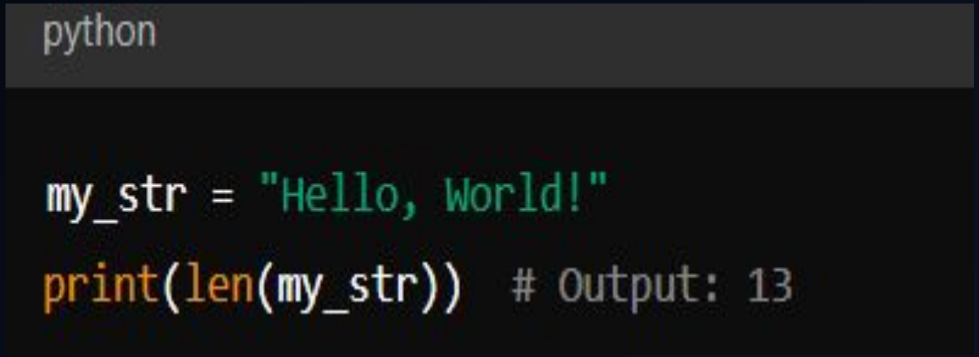
Common String Methods in Python

- Introduce key methods: len(), lower(), upper(), strip(), split(), replace(), find(), join(), startswith(), endswith(), count().

- Mention that these methods return new strings and do not modify the original string.

len() Method

- Returns the length of a string.

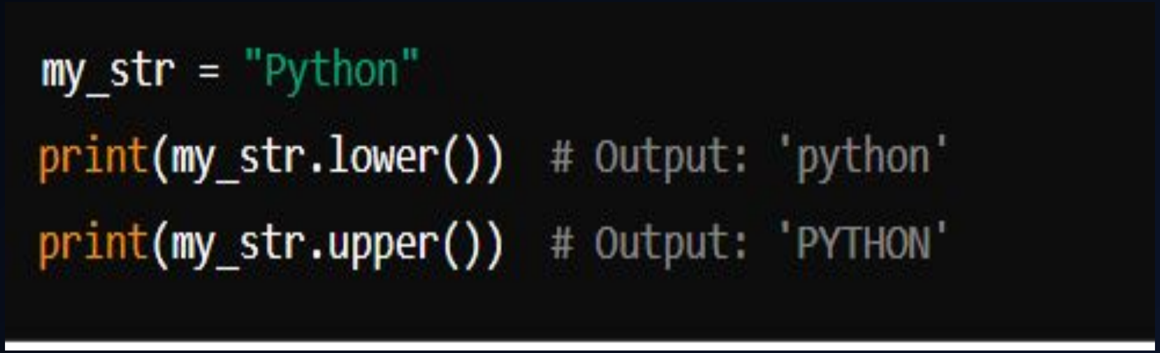


python

```
my_str = "Hello, World!"
print(len(my_str))  # Output: 13
```

lower() and upper() Methods

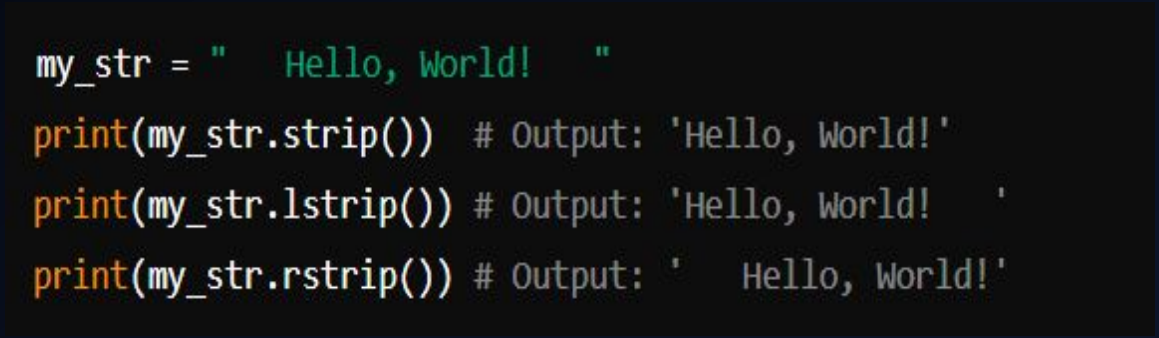
- lower(): Converts all characters to lowercase.
- upper(): Converts all characters to uppercase.



```
my_str = "Python"
print(my_str.lower())  # Output: 'python'
print(my_str.upper())  # Output: 'PYTHON'
```

strip(), lstrip(), rstrip() Methods

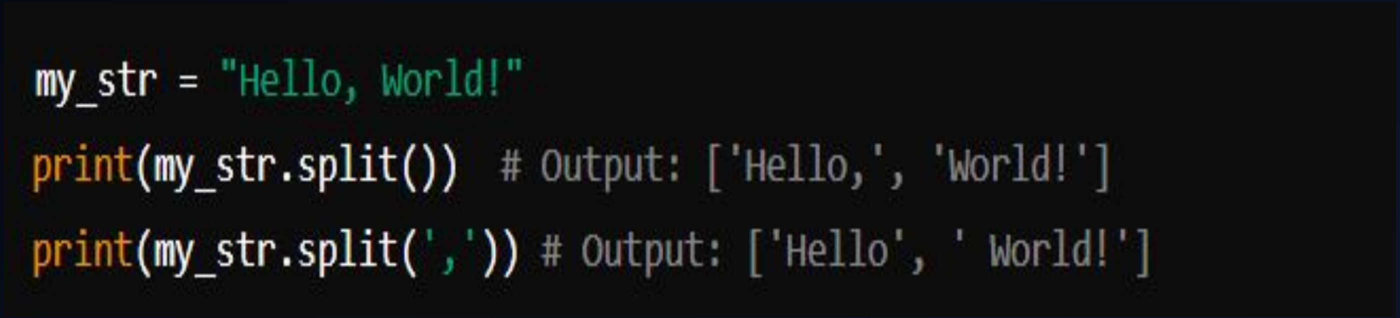
- strip(): Removes leading and trailing whitespace.
- lstrip(): Removes leading whitespace.
- rstrip(): Removes trailing whitespace.



```python
my_str = "   Hello, World!   "
print(my_str.strip())  # Output: 'Hello, World!'
print(my_str.lstrip()) # Output: 'Hello, World!   '
print(my_str.rstrip()) # Output: '   Hello, World!'
```

split() Method

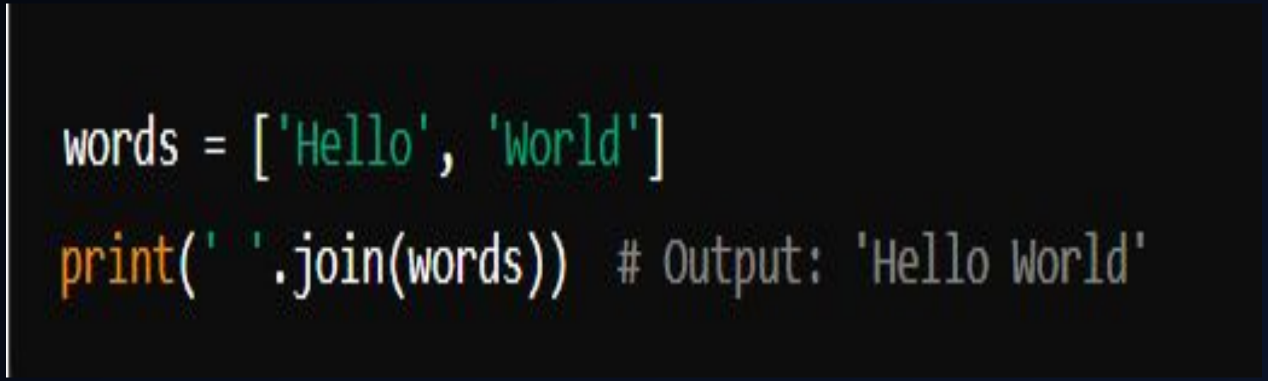
- Splits a string into a list of substrings based on a delimiter.
- Default delimiter is space.



```python
my_str = "Hello, World!"
print(my_str.split())  # Output: ['Hello,', 'World!']
print(my_str.split(',')) # Output: ['Hello', ' World!']
```

join() Method

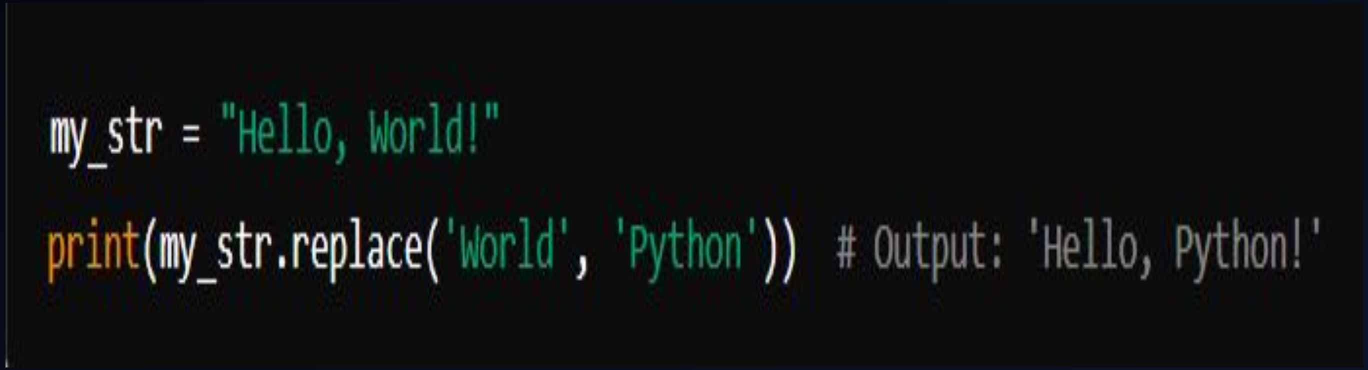
- Joins elements of an iterable (e.g., list) into a single string with a specified separator.



```python
words = ['Hello', 'World']
print(' '.join(words))  # Output: 'Hello World'
```

replace() Method

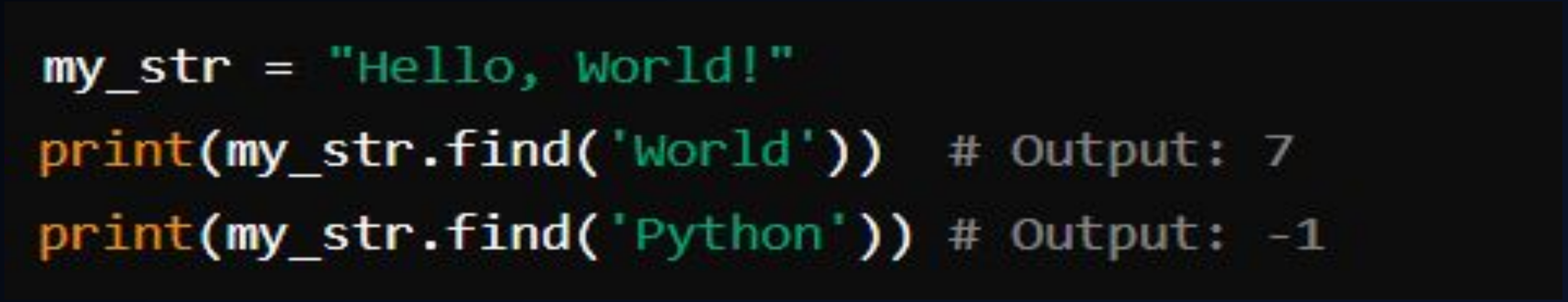
 Replaces occurrences of a substring with another substring.



```
my_str = "Hello, World!"
print(my_str.replace('World', 'Python'))  # Output: 'Hello, Python!'
```

find() Method

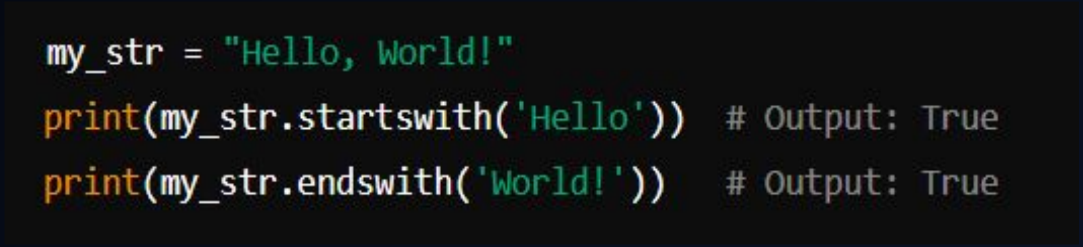
- Returns the index of the first occurrence of a substring. Returns -1 if not found.



```python
my_str = "Hello, World!"
print(my_str.find('World'))  # Output: 7
print(my_str.find('Python')) # Output: -1
```

startswith() and endswith() Methods

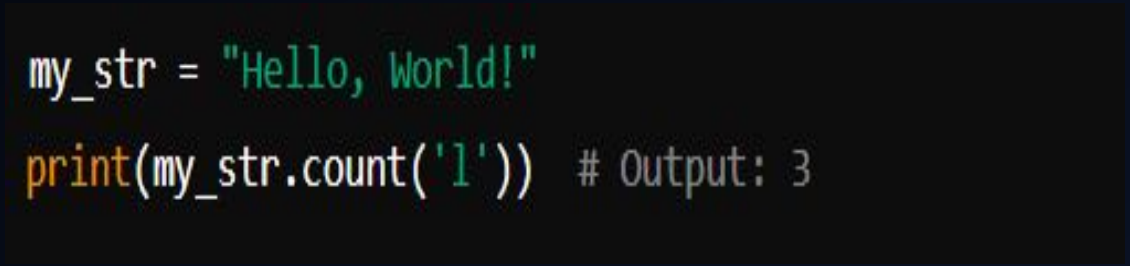
- startswith(): Checks if a string starts with a specified prefix.
- endswith(): Checks if a string ends with a specified suffix.



```
my_str = "Hello, World!"
print(my_str.startswith('Hello'))  # Output: True
print(my_str.endswith('World!'))   # Output: True
```

count() Method

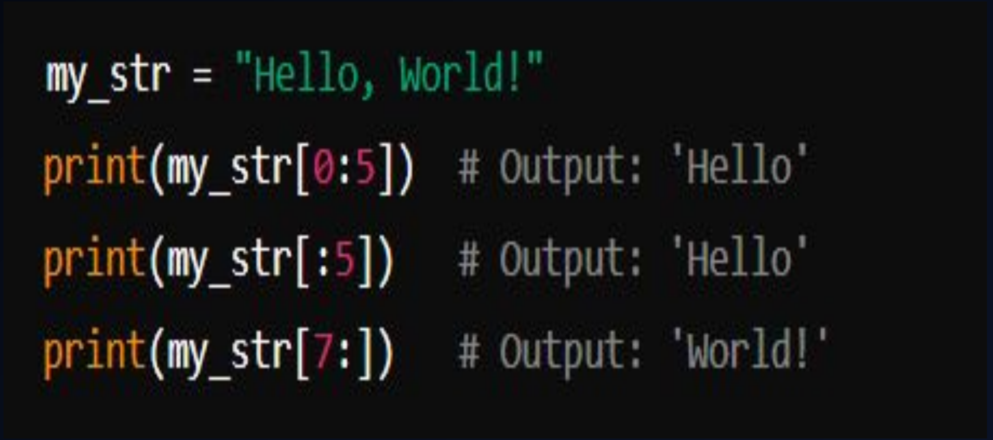
- Counts the occurrences of a substring in a string.



```python
my_str = "Hello, World!"
print(my_str.count('l'))  # Output: 3
```

String Slicing

- Extracts a substring using a range of indices.

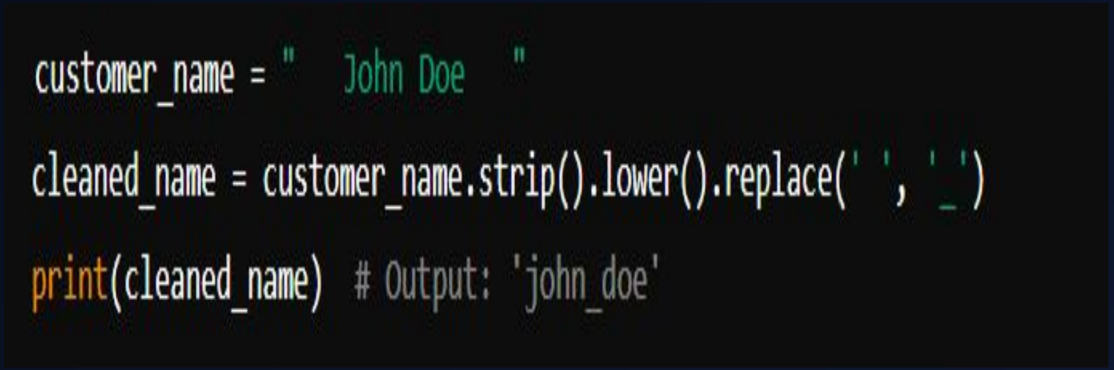


```python
my_str = "Hello, World!"
print(my_str[0:5])  # Output: 'Hello'
print(my_str[:5])   # Output: 'Hello'
print(my_str[7:])   # Output: 'World!'
```

Business Use cases

- Use Case1: Data Cleaning and Preparation
- Scenario: Cleaning up customer data imported from different sources.
- Methods Used: strip(), lower(), replace()

- **Problem:** Customer names have extra spaces and inconsistent capitalization.



```python
customer_name = "   John Doe   "
cleaned_name = customer_name.strip().lower().replace(' ', '_')
print(cleaned_name)  # Output: 'john_doe'
```

THANK YOU

HAPPY LEARNING!