Python – String Operations

Shahriar's Intelligence Academy

String Formats

Definition:

A string is a **sequence of characters enclosed in quotes**, representing text data in Python. Strings are **immutable**, meaning they cannot be modified after creation.

Various Formats of String:

- **Word String**: A string containing a single word.

  Example: "Python”

- **String with Spaces**: A string containing multiple words separated by spaces.

  Example: "Learn Python Programming”

- **String with Numbers**: A string can also include numeric characters (treated as text).

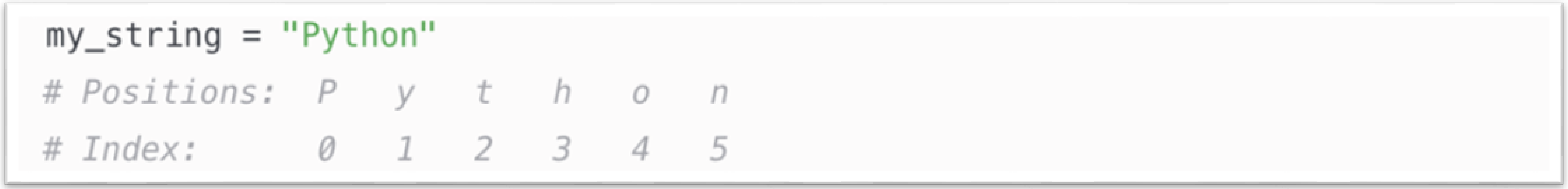
  Example: "12345" or "Python3"

Forward Indexing

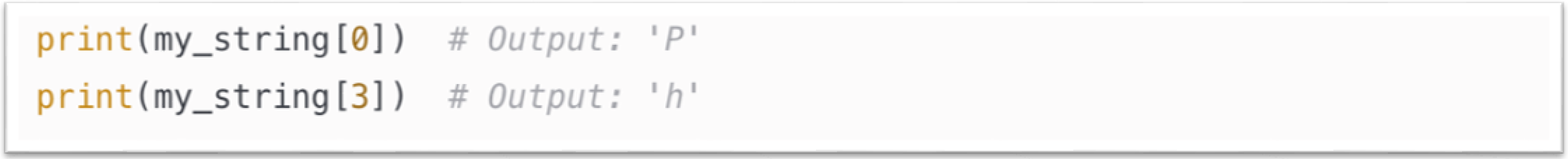
Key Features:

- Strings are **indexed** starting from **0**.
- Each character in a string can be accessed using its **position/index**.

Example:



```python
my_string = "Python"
# Positions:  P    y    t    h    o    n
# Index:      0    1    2    3    4    5
```



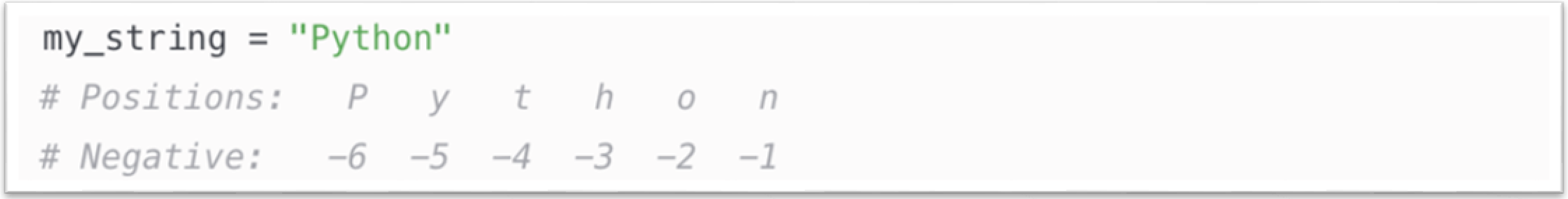
```python
print(my_string[0])  # Output: 'P'
print(my_string[3])  # Output: 'h'
```

Backward Indexing

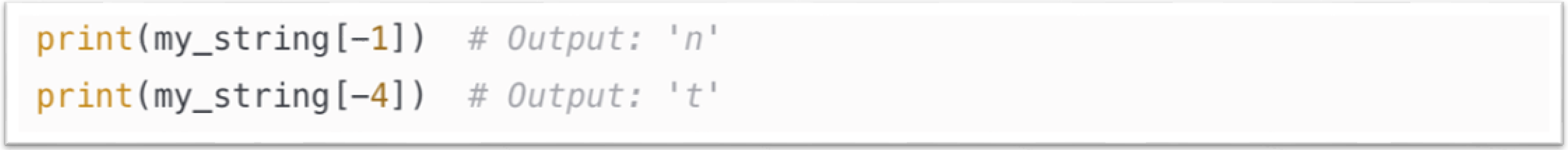
Key Features:

- Strings can also be indexed using **negative indexing**, where -1 represents the last character, -2 the second last, and so on.

Example:



```
my_string = "Python"
# Positions:   P    y    t    h    o    n
# Negative:   -6   -5   -4   -3   -2   -1
```



```
print(my_string[-1])  # Output: 'n'
print(my_string[-4])  # Output: 't'
```

Slicing Method for Strings

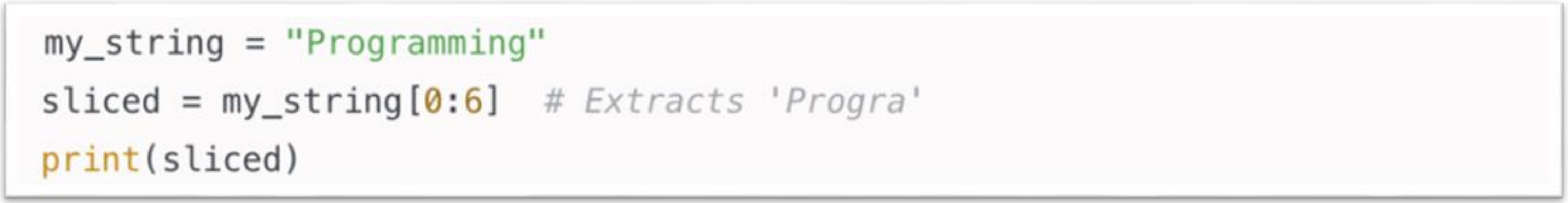
Definition:

Slicing allows extracting a **portion** of a string.

Syntax:

string[start : end] (end is exclusive).

Example:



```
my_string = "Programming"
sliced = my_string[0:6]  # Extracts 'Progra'
print(sliced)
```

Striding Method for Strings

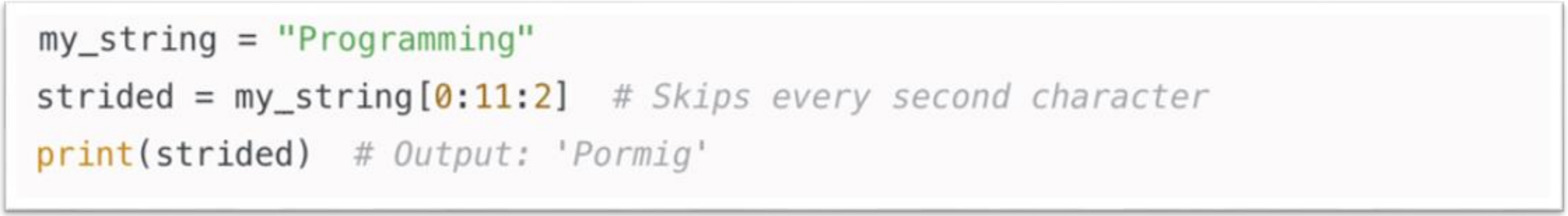
Definition:

Striding is used to **skip characters** while slicing.

Syntax:

- **string[start : end : step]**
- step defines how many characters to skip.

Example:



```
my_string = "Programming"
strided = my_string[0:11:2]  # Skips every second character
print(strided)  # Output: 'Pormig'
```

Len() Function for Strings

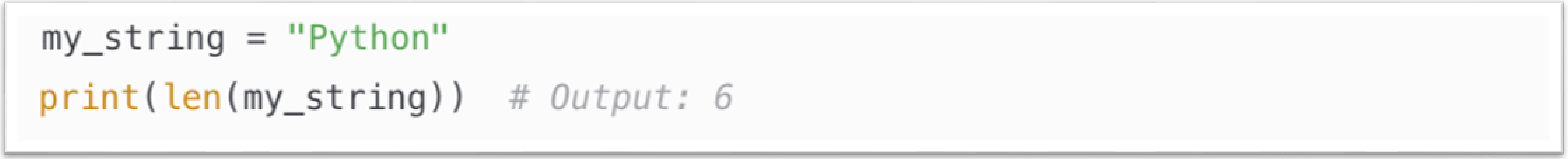
Definition:

The len() function returns the **length** of a string.

Syntax:

len(the_value**)**

Example:



```python
my_string = "Python"
print(len(my_string))  # Output: 6
```

Concatenation of Strings

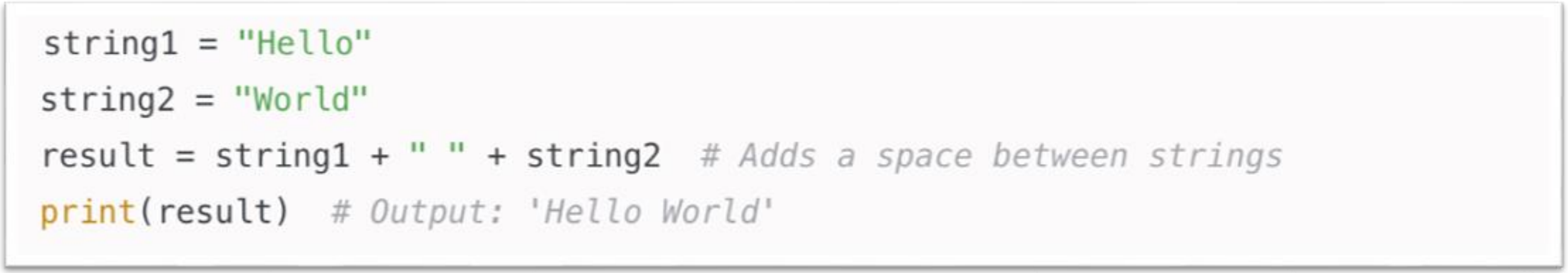
Definition:

Concatenation means **joining two or more strings** using the + operator.

Syntax:

“String 1” **+** “String 2” **+ …. +** “String n”

Example:



```
string1 = "Hello"
string2 = "World"
result = string1 + " " + string2  # Adds a space between strings
print(result)  # Output: 'Hello World'
```

Escape Sequence in Strings

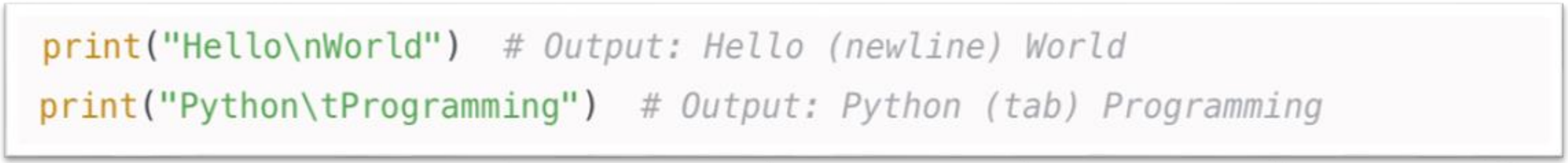
Definition:

Escape sequences are used to represent **special characters** in strings.

Syntax:

- **\n**: Newline
- **\t**: Tab

Example:



```python
print("Hello\nWorld")  # Output: Hello (newline) World
print("Python\tProgramming")  # Output: Python (tab) Programming
```

Upper() & Lower() for Strings

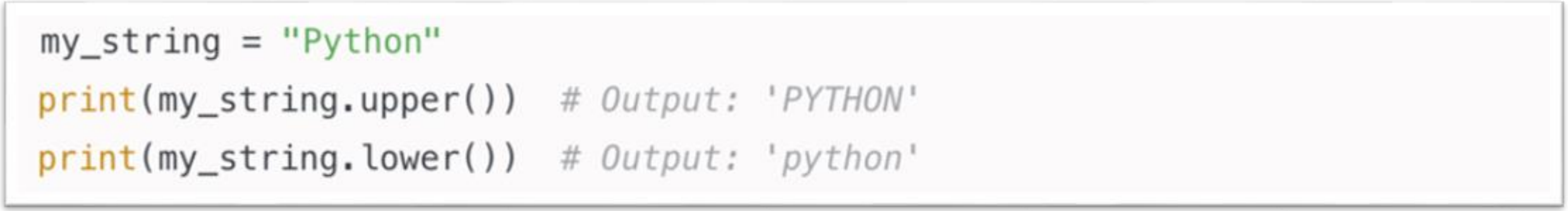
Definition:

Convert a string to **uppercase** or **lowercase**.

Syntax:

- String**.upper()**
- String**.lower()**

Example:



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

Replace() Function for Strings

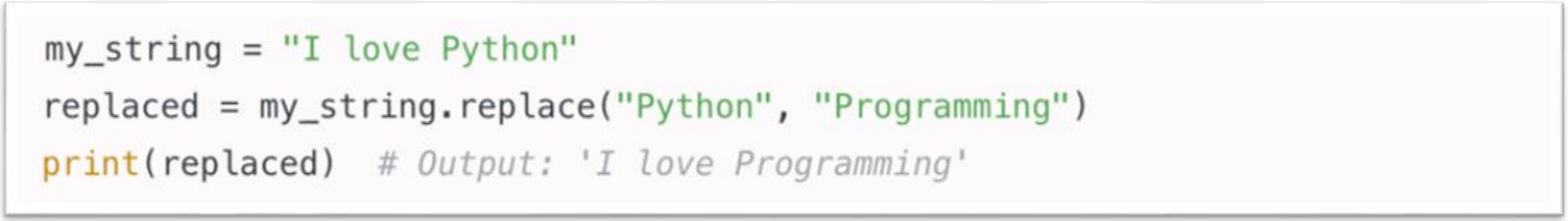
Definition:

Replace occurrences of a substring with another string.

Syntax:

string.replace(old, new)

Example:



```
my_string = "I love Python"
replaced = my_string.replace("Python", "Programming")
print(replaced)  # Output: 'I love Programming'
```

Thank You!