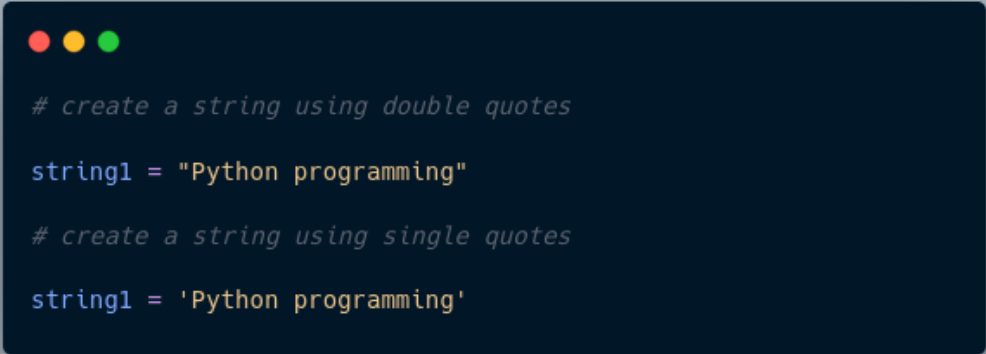


In computer programming, a string is a sequence of characters.

For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.

We use single quotes or double quotes to represent a string in Python.


For example,



```
# create a string using double quotes  
string1 = "Python programming"  
  
# create a string using single quotes  
string1 = 'Python programming'
```

Here, we have created a string variable named string1. The variable is initialized with the string Python Programming.

Example: Python String



```
# create string type variables  
  
name = "Python"  
  
print(name)  
  
message = "I love Python."  
  
print(message)  
  
# Output  
  
# Python  
  
# I love Python.
```

In the above example, we have created string-type variables: name and message with values "Python" and "I love Python" respectively.

Here, we have used double quotes to represent strings but we can use single quotes too.

Access String Characters in Python

We can access the characters in a string in three ways.

- **Indexing:** One way is to treat strings as a list and use index values.

For example,



```
greet = 'hello'

# access 1st index element

print(greet[1]) # "e"
```

- **Negative Indexing:** Similar to a list, Python allows negative indexing for its strings.

For example,




```
greet = 'hello'

# access 4th last element

print(greet[-4]) # "e"
```

- **Slicing:** Access a range of characters in a string by using the slicing operator colon :

For example,



```
greet = 'Hello'

# access character from 1st index to 3rd index


print(greet[1:4]) # "ell"
```

Note: If we try to access an index out of the range or use numbers other than an integer, we will get errors.

Python Strings are immutable

In Python, strings are immutable. That means the characters of a string cannot be changed.

For example,



```
message = 'Hola Amigos'


message[0] = 'H'

print(message)

# Output
# TypeError: 'str' object does not support item assignment
```

However, we can assign the variable name to a new string.

For example,




```
message = 'Hola Amigos'  
  
# assign new string to message variable  
message = 'Hello Friends'  
  
prints(message); # prints "Hello Friends"
```

Python Multiline String

We can also create a multiline string in Python. For this, we use triple double quotes `"""` or triple single quotes `'''`.

For example,



```
# multiline string

message = """

Never gonna give you up

Never gonna let you down

"""

print(message)

# Output

# Never gonna give you up

# Never gonna let you down
```

In the above example, anything inside the enclosing triple quotes is one multiline string.

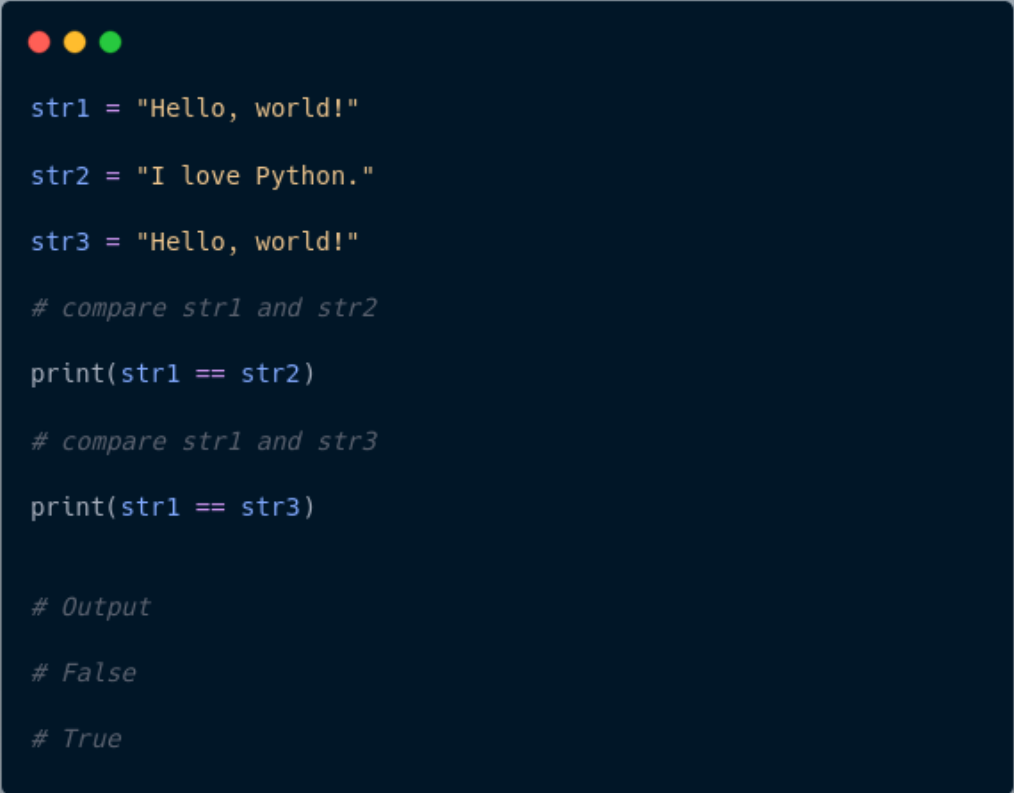
Python String Operations

There are many operations that can be performed with strings which makes it one of the most used data types in Python.

1. Compare Two Strings

We use the `==` operator to compare two strings. If two strings are equal, the operator returns `True`. Otherwise, it returns `False`.

For example,




```
str1 = "Hello, world!"  
  
str2 = "I love Python."  
  
str3 = "Hello, world!"  
  
# compare str1 and str2  
print(str1 == str2)  
  
# compare str1 and str3  
print(str1 == str3)  
  
# Output  
# False  
  
# True
```

In the above example,

- str1 and str2 are not equal. Hence, the result is False.
- str1 and str3 are equal. Hence, the result is True.

2. Join Two or More Strings

In Python, we can join (concatenate) two or more strings using the + operator.



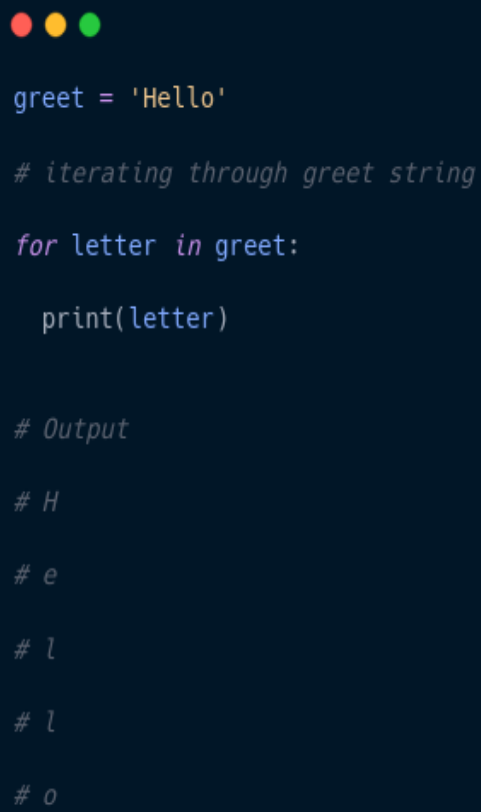
```
greet = "Hello, "  
name = "Jack"  
  
# using + operator  
  
result = greet + name  
print(result)  
  
# Output: Hello, Jack
```

In the above example, we have used the + operator to join two strings: greet and name.

Iterate Through a Python String

We can iterate through a string using a for loop.

For example,

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The code inside is as follows:

```
greet = 'Hello'  
  
# iterating through greet string  
  
for letter in greet:  
    print(letter)  
  
# Output  
  
# H  
  
# e  
  
# l  
  
# l  
  
# o
```

Python String Length

In Python, we use the `len()` method to find the length of a string.


For example,



```
greet = 'Hello'  
  
# count length of greet string  
print(len(greet))  
  
# Output: 5
```

String Membership Test

We can test if a substring exists within a string or not, using the keyword `in`.



```
print('a' in 'program') # True  
print('at' not in 'battle') False
```

Methods of Python String

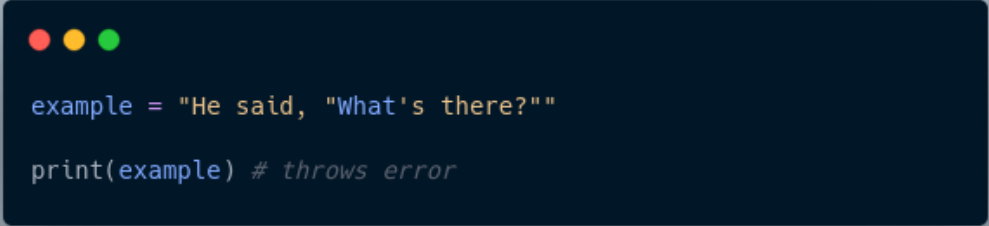
Besides those mentioned above, there are various string methods present in Python. Here are some of those methods:

| Methods | Description |
|---------------------------|--|
| <code>upper()</code> | converts the string to uppercase |
| <code>lower()</code> | converts the string to lowercase |
| <code>partition()</code> | returns a tuple |
| <code>replace()</code> | replaces substring inside |
| <code>find()</code> | returns the index of first occurrence of substring |
| <code>rstrip()</code> | removes trailing characters |
| <code>split()</code> | splits string from left |
| <code>startswith()</code> | checks if string starts with the specified string |
| <code>isnumeric()</code> | checks numeric characters |
| <code>index()</code> | returns index of substring |

Escape Sequences in Python

The escape sequence is used to escape some of the characters present inside a string.

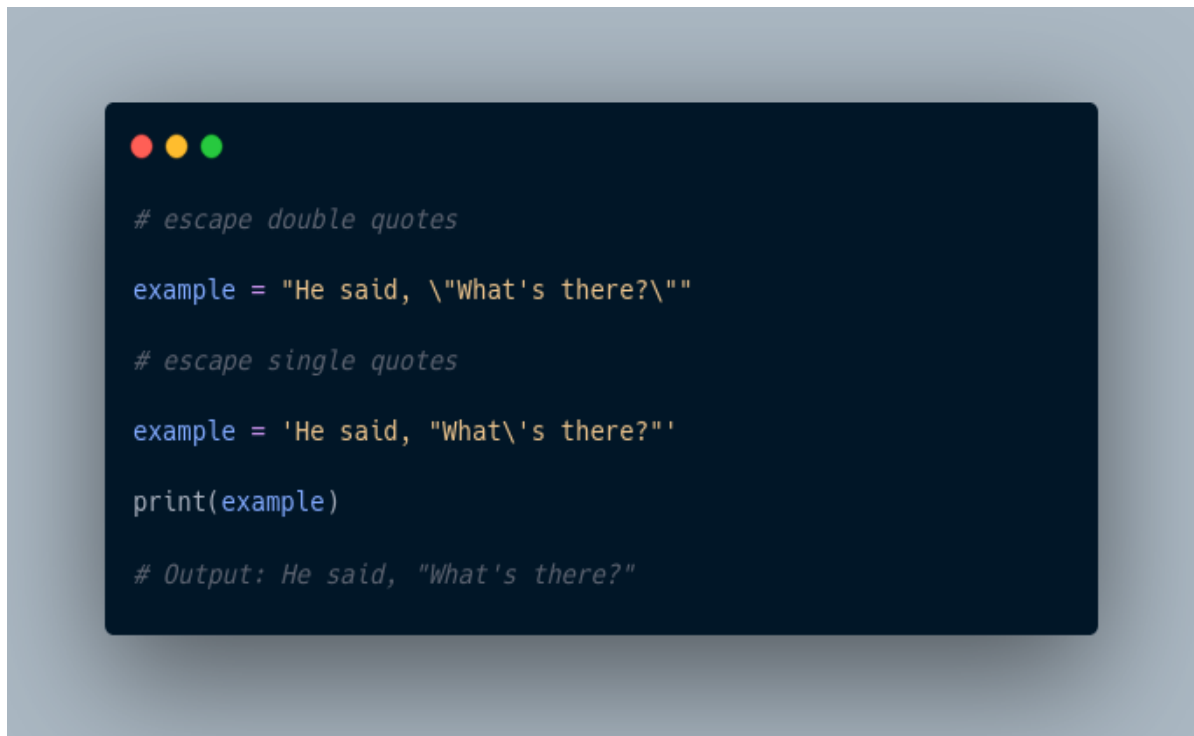
Suppose we need to include both double quote and single quote inside a string,



```
example = "He said, "what's there?""  
print(example) # throws error
```

Since strings are represented by single or double quotes, the compiler will treat "He said, " as the string. Hence, the above code will cause an error.

To solve this issue, we use the escape character \ in Python.

A terminal window with a dark blue background and light blue text. It shows Python code for escaping double and single quotes. The code includes comments, variable assignments, and a print statement. The output of the print statement is shown at the bottom.

```
# escape double quotes
example = "He said, \"What's there?\""

# escape single quotes
example = 'He said, "What\'s there?"'

print(example)

# Output: He said, "What's there?"
```


Here is a list of all the escape sequences supported by Python.

| Escape Sequence | Description |
|-------------------|-------------------------------------|
| <code>\\</code> | Backslash |
| <code>\'</code> | Single quote |
| <code>\"</code> | Double quote |
| <code>\a</code> | ASCII Bell |
| <code>\b</code> | ASCII Backspace |
| <code>\f</code> | ASCII Formfeed |
| <code>\n</code> | ASCII Linefeed |
| <code>\r</code> | ASCII Carriage Return |
| <code>\t</code> | ASCII Horizontal Tab |
| <code>\v</code> | ASCII Vertical Tab |
| <code>\ooo</code> | Character with octal value ooo |
| <code>\xHH</code> | Character with hexadecimal value HH |

Python String Formatting (f-Strings)

Python **f-Strings** make it really easy to print values and variables.

For example,

A code editor window with a dark blue background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in a light blue font. It defines two variables, 'name' and 'country', and uses an f-string to print their values. Below the code, the output is shown in a lighter blue font.

```
name = 'Cathy'
country = 'UK'
print(f'{name} is from {country}')

# Output
# Cathy is from UK
```

Here, `f'{name} is from {country}'` is an **f-string**.

1. This new formatting syntax is powerful and easy to use. From now on, we will use f-Strings to print strings and variables.

More Resources:

1. <https://www.simplilearn.com/tutorials/python-tutorial/python-strings>
2. <https://www.geeksforgeeks.org/python-string/>
3. https://www.tutorialspoint.com/python/python_strings.htm