String Operations

This lesson showcases some of the most commonly used string operations.

We'll cover the following

The string data type has numerous utilities that make string computations much easier. We'll explore all of them in the future, but first, let's get down to the basics.

Comparison Operators

Strings are compatible with the comparison operators we studied earlier. Each character has a Unicode value.

This allows strings to be compared on the basis of their Unicode values.

When two strings have different lengths, the string which comes first in the dictionary is said to have the smaller value.

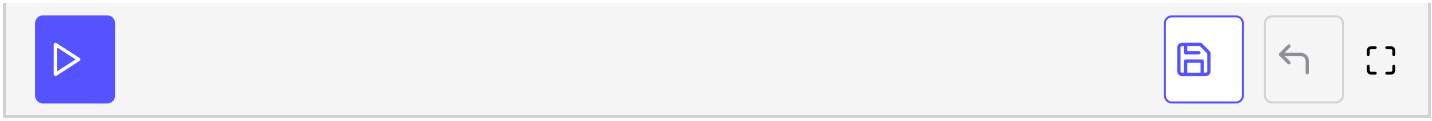
Let's look at a few examples:

```
print('a' < 'b')  # 'a' has a smaller Unicode value

house = "Gryffindor"
house_copy = "Gryffindor"

print(house == house_copy)

new_house = "Slytherin"

print(house == new_house)

print(new_house <= house)

print(new_house >= house)
```

Concatenation

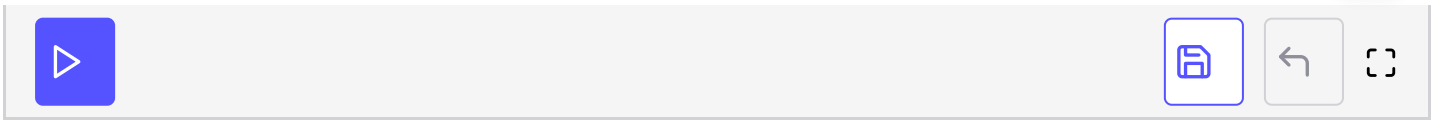
The `+` operator can be used to merge two strings together:

```
first_half = "Bat"
second_half = "man"

full_name = first_half + second_half
print(full_name)
```



The `*` operator allows us to multiply a string, resulting in a repeating pattern:

```
print("ha" * 3)
```

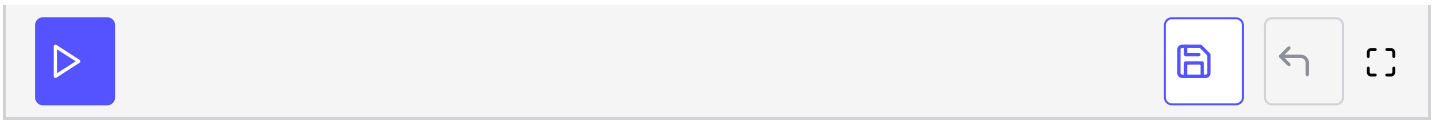


Search

The `in` keyword can be used to check if a particular substring exists in another string. If the substring is found, the operation returns `true`.

Here's how it works:

```
random_string = "This is a random string"

print('of' in random_string)  # Check whether 'of' exists in randomString
print('random' in random_string)  # 'random' exists!
```



This ends our discussion on strings...for now.

Next, we'll learn how to store and organize multiple data items in one variable.