Strings and Characters

If you need a string that spans several lines, use a multiline string literal—a sequence of characters surrounded by three double quotation marks:

```
let singleLineString = "These are the same."

let multilineString = """

These are the same.

"""
```

Special Characters in String Literals

String literals can include the following special characters:

- The escaped special characters `\0` (null character), `\\` (backslash), `\t` (horizontal tab), `\n` (line feed), `\r` (carriage return), `\"` (double quotation mark) and `\'` (single quotation mark)
- An arbitrary Unicode scalar value, written as `\u{`*n*`}`, where *n* is a 1–8 digit hexadecimal number (Unicode is discussed in Unicode below)

```
let wiseWords = "\"Imagination is more important than knowledge\" - Einstein"

// "Imagination is more important than knowledge" - Einstein

let dollarSign = "\u{24}" // $, Unicode scalar U+0024

let blackHeart = "\u{2665}" // ♥, Unicode scalar U+2665

let sparklingHeart = "\u{1F496}" // 💖, Unicode scalar U+1F496
```

String Mutability

```
var variableString = "Horse"

variableString += " and carriage"

// variableString is now "Horse and carriage"


let constantString = "Highlander"

constantString += " and another Highlander"

// this reports a compile-time error - a constant string cannot be modified
```

Concatenating Strings and Characters

```
let badStart = """  one

two  """

let end = """  three

"""

print(badStart + end)

// Prints two lines:

// one

// two  three

let goodStart = """  one

two
```

```
"""

print(goodStart + end)

// Prints three lines:

// one

// two

// three
```

String interpolation is a way to construct a new `String` value from a mix of constants, variables, literals, and expressions by including their values inside a string literal. You can use string interpolation in both single-line and multiline string literals. Each item that you insert into the string literal is wrapped in a pair of parentheses, prefixed by a backslash (`\`):

```
let multiplier = 3

let message = "\(multiplier) times 2.5 is \(Double(multiplier) * 2.5)"

// message is "3 times 2.5 is 7.5"
```

To retrieve a count of the `Character` values in a string, use the `count` property of the string:

```
var word = "cafe"

print("the number of characters in \(word) is \(word.count)")

// Prints "the number of characters in cafe is 4"

word += "\u{301}" // COMBINING ACUTE ACCENT, U+0301
```

```
print("the number of characters in \(word) is \(word.count)")

// Prints "the number of characters in café is 4"
```

if you initialize a new string with the four-character word `cafe`, and then append a `COMBINING ACUTE ACCENT` (`U+0301`) to the end of the string, the resulting string will still have a character count of `4`, with a fourth character of `é`, not `e`: