Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn't need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```
var myVariable = 42
myVariable = 50
let myConstant = 42
```

A constant or variable must have the same type as the value you want to assign to it. However, you don't always have to write the type explicitly. Providing a value when you create a constant or variable lets the compiler infer its type. In the example above, the compiler infers that `myVariable` is an integer because its initial value is an integer.

If the initial value doesn't provide enough information (or if isn't an initial value), specify the type by writing it after the variable, separated by a colon.

```
let implicitInteger = 70
let implicitDouble = 70.0
let explicitDouble: Double = 70
```

> EXPERIMENT
>
> Create a constant with an explicit type of `Float` and a value of `4`. → let a: Float = 4

Values are never implicitly converted to another type. If you need to convert a value to a different type, explicitly make an instance of the desired type.

```
let label = "The width is "
let width = 94
let widthLabel = label + String(width)
```

> EXPERIMENT
>
> Try removing the conversion to `String` from the last line. What error do you get?

→ Binary operator '+' cannot be applied to operands of type 'String' and 'Int'

There's an even simpler way to include values in strings: Write the value in parentheses, and write a backslash (\) before the parentheses. For example:

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces of fruit."
```

EXPERIMENT

Use \( ) to include a floating-point calculation in a string and to include someone's name in a greeting.

Use three double quotation marks (""") for strings that take up multiple lines. Indentation at the start of each quoted line is removed, as long as it matches the indentation of the closing quotation marks. For example:

```
let quotation = """
I said "I have \(apples) apples."
And then I said "I have \(apples + oranges) pieces of fruit."
"""
```

Create arrays and dictionaries using brackets (`[]`), and access their elements by writing the index or key in brackets. A comma is allowed after the last element.

```
var shoppingList = ["catfish", "water", "tulips"]
shoppingList[1] = "bottle of water"

var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
occupations["Jayne"] = "Public Relations"
```

Arrays automatically grow as you add elements.

```
shoppingList.append("blue paint")
print(shoppingList)
```

To create an empty array or dictionary, use the initializer syntax.

```
let emptyArray: [String] = []
let emptyDictionary: [String: Float] = [:]
```

If type information can be inferred, you can write an empty array as `[]` and an empty dictionary as `[:]`—for example, when you set a new value for a variable or pass an argument to a function.

```
shoppingList = []
occupations = [:]
```