Variables

Let's start by learning about variables. If you are new to programming, then you should think of a variable as a labeled box that contains an item or a set of items.

Type out the following example:

```
var greeting = "Hello Swift"
```

Here, you are creating a variable called **greeting**, which contains the text "Hello Swift". So, **greeting** would be the box, and "Hello Swift" would be the data you are storing inside the box.

If you type out the following print command and run it, then you will notice it will output your variable to the right-hand pane and to the Debug Area.

```
print(greeting)
```

Only code that has "print" before it will be shown inside the Debug Area. The results of everything else will be shown in the right-hand pane when you click the **Play** button.

You could change the value of greeting by typing:

```
greeting = "Hello John"
```

The value of **greeting** would now be "Hello John".

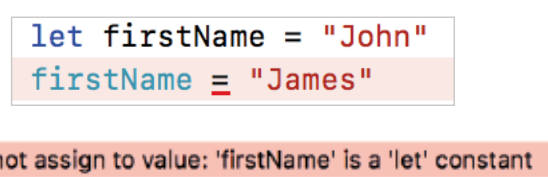
Constants

Type the following inside Playgrounds:

```
let firstName = "John"
```

In this example, you would then be assigning "John" to the variable called **firstName.** Usually, you can change a variable at any time; however, when you put **let** at the start it means the variable cannot be changed because the variable has been defined as a constant. So, if you tried to change the constant by typing the following line of code, then you would get an error:

```
firstName = "James"
```



If you changed **firstName** to a **var** (`var firstName = "John"`) instead of a **let**, it would work fine. Constants would work well for things like date of birth, or even URLs that will not change across your app.

Data types

There are various data types you can use with Swift. If you are coming from another programming language, then you will probably already be familiar with these. Make sure you type out the examples inside Playgrounds, as it will help you get used to each type.

- **Strings** are a sequence of characters like "Good Morning". The examples we covered on the last page were all strings. However, when you wrote your example, Swift presumed that your variables were strings because of the context of the code. If you wanted to make sure that Swift knew for sure that your variable was a string, you would write something like this:

```
var message: String = "Hello"
```

- **Integers** are used for whole numbers like 29, 101; etc. Below is an example:

```
var myNo: Int = 100
```

- **Float** is used to represent numbers with a fractional component. It can be used to hold numbers with larger or smaller decimal points, like 3.12 and -435.3244.

```
var floatval: Float = 3.12
```

- **Double** can be used when floating-point values are very large – for example, 415.324544 and -4254.3255543.

```
var doubleval: Double = 415.324544
```

- **Booleans** would be used for values that are either true or false.

```
var isSuccess: Bool = true
```

Tuples

These can be used to store multiple values in a single value.

```
let playerInfo = ("Mark", 20)
```

Even though the syntax is different for each language, learning about data types lays down a foundation for learning new languages.

Variables must be initialized when declared unless using type annotation.

Tuples don't have to hold the same type of values. For example, they could hold a string and an integer.

Combining variables

As well as creating variables, you also need to be able to combine them. You might want to combine string values or use two or more values for arithmetic operations.

If you were looking to combine two string variables, it might look something like this:

```
var message1: String = "Hello there"
var message2: String = "Swift fans"
message1 +  " " + message2
```

This example creates two different string variables, which both contain different messages. It then combines the two strings to create a message. A space has been added (" ") so that the sentence is spaced out. It should output "Hello there Swift fans".

If you wanted to do mathematical equations, it would look something like this:

```
var num1: Int = 12
var num2: Int = 30
num1 + num2
```

This example creates two integers, and then adds them up: **12 + 30 = 42**.

If you are trying to do a mathematical calculation, make sure you are using the right data type. For example, if you tried adding two strings, it would just join one string to the other.

Swift supports all four standard arithmetic operators:

- **Addition** (+)
- **Subtraction** (-)
- **Multiplication** (*)
- **Division** (/)

Swift has a Remainder operator (**%**), which will return the value that is left over when dividing two numbers.

```
var first: Int = 13
var second: Int = 6
first % second
```

Giving the result: **1**

Value comparisons

When you build your apps, there are going to be times when you will need to compare values. For example, in a game you might want to check whether or not a user has beaten their own personal best score.

Here are the operators you will need for comparisons:

- **Equal to**: (==)
- **Not equal to**: (!=)
- **Greater than**: (>)
- **Less than**: (<)
- **Greater than or equal to**: (>=)
- **Less than or equal to**: (<=)

Type out these examples inside Playgrounds. Each one will return a Boolean value (true or false) depending on whether or not the statement is true.

```
4 == 4          //True: 4 is equal to 4
3 != 2          //True: 3 is not equal to 2
4 > 5           //False: 4 is not greater than 5
6 < 8           //True: 6 is less than 8
2 >= 2          //True: 2 is equal to 2
5 <= 4          //False: 5 is not less than or equal to 4
```

You can also do comparisons with variables of the same type.

```
var number1 = 8
var number2 = 10
//This will return false
number1 == number2
```

Lines of code that have "//" at the start are code comments. These are a good way to document your code.

Converting data types

The example below takes a string and converts it to an integer. This would come in handy if you were taking a value from a text box and using it for a mathematical calculation.

```
let strNo = "215"
let myInt1 = Int(strNo)
print(myInt1!)
```

If you want to compare string values, you will need to use (==).