Variables in Scala

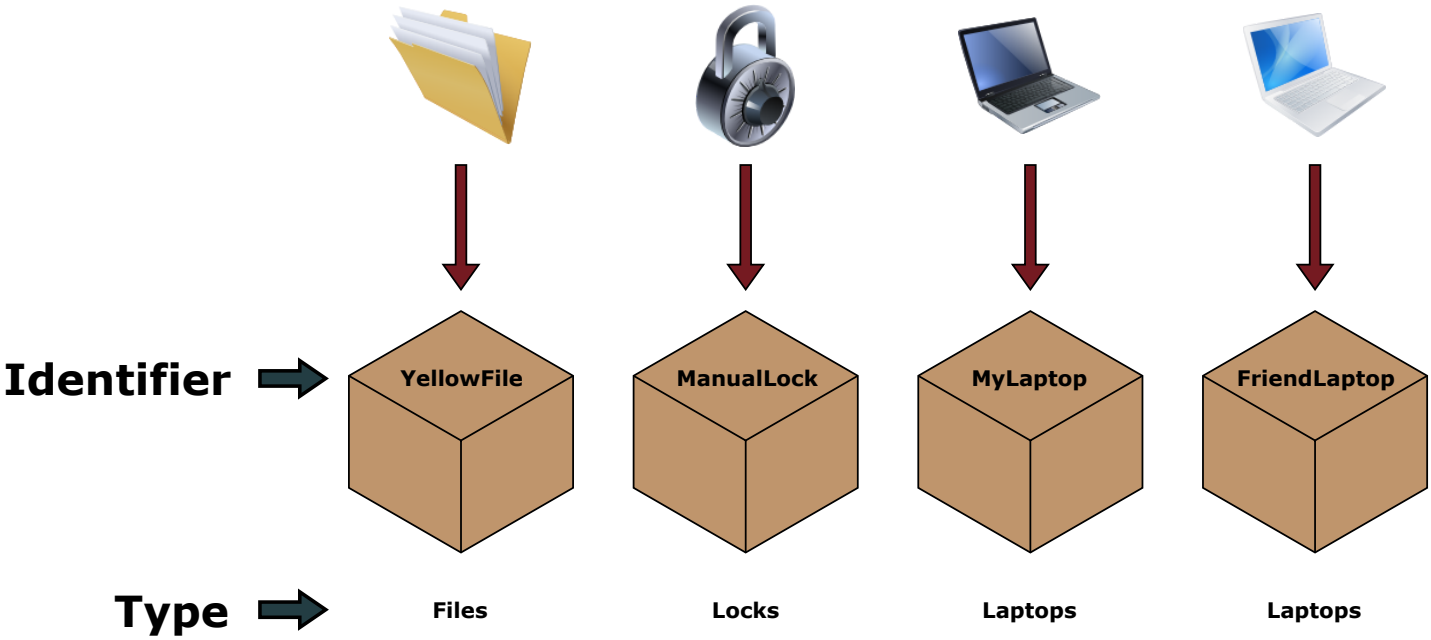
In the following lesson, you will be introduced to variables and how they are declared in Scala.

We'll cover the following

- Introduction
- Declaring a Variable
- Choosing the Right Keyword
- Multiple Variables

Introduction

Imagine you have multiple boxes and in each box, you can store one item. Before you can store anything, you need to decide what type of item can be stored in each box. But now you have so many boxes, you don't remember what is in each one. You decide to give each box a unique label. This will help you keep track of what each one contains. The labels act as identifiers, giving each box a unique identity.



In the same way, a variable is a small box used to store data. When we *assign* a value to a variable, we are basically putting something in a box. When you declare a variable, you give it a unique name or identifier, define the type of data it can store, and set its initial value. While most programming languages have a similar

way of declaring variables, Scala is a little different.

Declaring a Variable

Let's look at the syntax of how to declare a variable in Scala.

```
keyword variableName: DataType = Initial Value
```

Now, let's map the syntax to actual code in Scala.

```
val myFirstScalaVariable: Int = 5
```

In the code above, we are declaring a variable with the name `myFirstScalaVariable`. `myFirstScalaVariable` can store data of type `Int` and is assigned an initial value of 5. It is an immutable variable because we chose the keyword `val`.

As stated above, when declaring a variable, we need to define its type of data and set an initial value. However, in Scala, we also need to define the type of variable by mentioning a *keyword*.

But what are keywords and how do we choose the right one for our variable?

Choosing the Right Keyword

For this chapter, we are only going to concern ourselves with two keywords: `val` and `var`.

- Variables of type `val` are immutable variables; once they are initialized, they can never be reassigned.
- Variables of type `var` are mutable variables; they can be reassigned throughout their lifetime as long as they are valid.

We will discuss immutable and mutable variables in greater detail in the coming lessons.

Multiple Variables

In large programs, we can have a great number of variables. It becomes difficult to remember what each variable stores, hence, when choosing a name for your variable, make sure it makes sense so that later on when you read the name again,

you will know what the variable stores and why you declared it in the first place.

Let's look at an example where we need to store data about a book:

```
val bookTitle: String = "Lord of the Rings: The Fellowship of the Ring"
val bookAuthor: String = "J. R. R. Tolkien"
val bookNoOfPages: Int = 423
```

Declaring a variable, 'myFirstVariable'

In the code above, we are declaring three variables:

- The first variable is named `bookTitle` and is of type `String`. It is assigned an initial value of "Lord of the Rings: The Fellowship of the Ring" and stores the title of the book.
- The second variable is named `bookAuthor` and is of type `String`. It is assigned an initial value of "J. R. R. Tolkien" and stores the name of the author of the book.
- The third variable is named `bookNoOfPages` and is of type `Int`. It is assigned an initial value of **423** and stores the number of pages in a book.

All the variables are immutable variables because we mention the keyword `val` before each identifier. We can easily deduce what data each variable stores because of how we chose to name them.

Before we move on to immutable and mutable variables, let's learn how to use Scala's print statement in the next lesson.