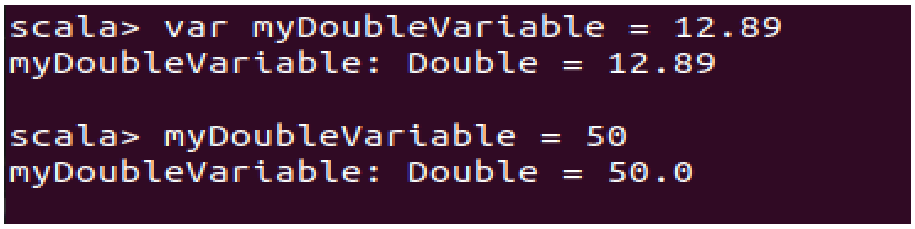
Chapter 4: Variables

Exercises :

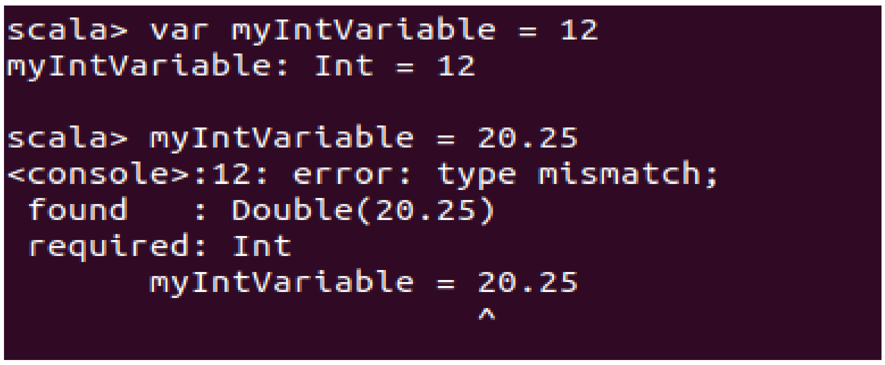
1. Try creating a variable of type Double and assigning an integer value to it. Does it work? If yes, research this. Do it the other way (storing a Double value in an Integer variable).
2. Try creating a variable (e.g., x) and assigning a value to it (e.g., 10). Then create another variable (e.g., y) and assign it to another variable (i.e., x=y). Now change the value of x. Check whether it changed the value of y or not. If not, research this concept (specifically what is meant by pass by value and pass by reference).
3. Try creating multiple variables on one line.

Answers :

1.

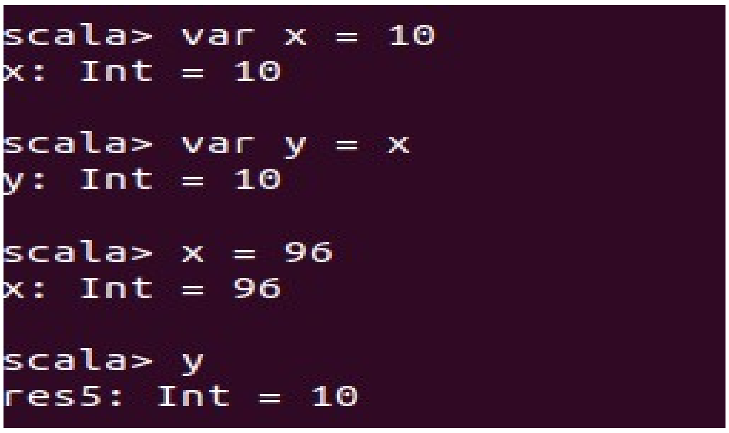


```
scala> var myDoubleVariable = 12.89
myDoubleVariable: Double = 12.89

scala> myDoubleVariable = 50
myDoubleVariable: Double = 50.0
```



```
scala> var myIntVariable = 12
myIntVariable: Int = 12

scala> myIntVariable = 20.25
<console>:12: error: type mismatch;
 found   : Double(20.25)
 required: Int
       myIntVariable = 20.25
                       ^
```

2.



```
scala> var x = 10
x: Int = 10

scala> var y = x
y: Int = 10

scala> x = 96
x: Int = 96

scala> y
res5: Int = 10
```

In value passing, a copy of the actual arguments is passed to the respective formal arguments. While in reference passing, the location (address) of the actual arguments is passed to formal arguments, any changes made to the formal arguments will also be reflected in the actual arguments.

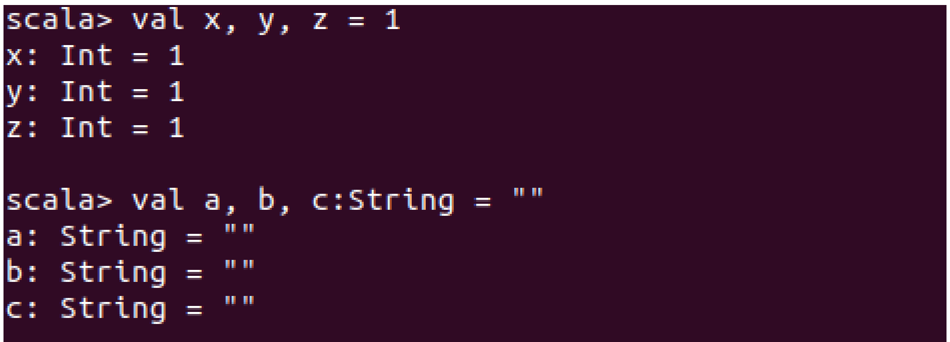
Passing by value
Passing a parameter by value copies the value of the parameter to a variable local to the function. Thus, if the argument is modified, the parameter passed to the function is not.

Passing by reference
Passing a parameter by reference copies the reference of the parameter into an argument that is a variable local to the scope of the function. Thus, if the argument is modified, the parameter passed to the function is also modified.

	Pass by value	Pass by reference
Variable	A copy of the variable is passed.	The variable itself is passed.
The Effect	The modification of the variable copy does not change the original value of the variable outside the function.	The modification of the variable also affects the value of the variable outside the function.

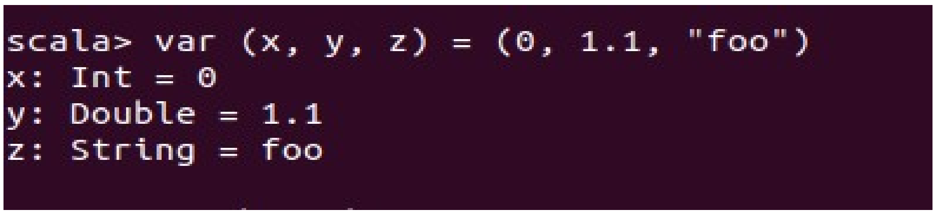
I thought I'd have a little fun with Scala today, and show different ways to declare multiple variables on one line. These aren't the most useful or common things to do, but they're interesting, and I know I learned at least one thing while looking into this.

You can define multiple fields on one line, separated by commas, as long as they are all the same type and have the same value:



```
scala> val x, y, z = 1
x: Int = 1
y: Int = 1
z: Int = 1

scala> val a, b, c:String = ""
a: String = ""
b: String = ""
c: String = ""
```

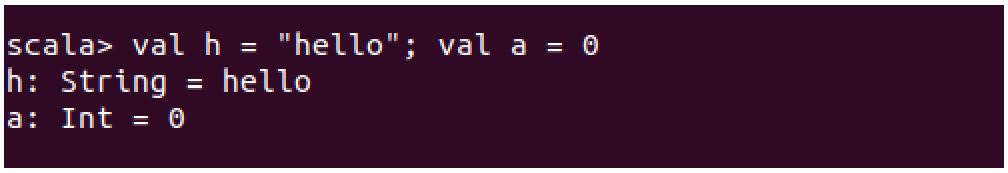
If you really want to have some fun, you can take advantage of Scala’s extractor functionality to declare fields with different types like this:



```
scala> var (x, y, z) = (0, 1.1, "foo")
x: Int = 0
y: Double = 1.1
z: String = foo
```

As you can see from the output, that example creates three var fields of different types in one line of code.

Depending on your definition of “one line,” you can also define multiple fields on one line if they are separated by a semicolon:



```
scala> val h = "hello"; val a = 0
h: String = hello
a: Int = 0
```