Kotlin Programming

Kotlin Variables and Basic Types

In this Lesson, you will learn about variables, how to create them, and basic data types that Kotlin supports for creating variables.

As you know, a variable is a location in memory (storage area) to hold data.

To indicate the storage area, each variable should be given a unique name (identifier). Learn more about How to name a variable in Kotlin?

How to declare a variable in Kotlin?

To declare a variable in Kotlin, either `var` or `val` keyword is used. Here is an example:

```
var language = "French"
val score = 95
```

The difference in using *var* and *val* is discussed later in the article. For now, let's focus on variable declaration.

Here, *language* is a variable of type `String`, and `score` is a variable of type `Int`. You don't have to specify the type of variables; Kotlin implicitly does that for you. The compiler knows this by initializer expression (*"French"* is a `String`, and *95* is an integer value in the above program). This is called type inference in programming.

However, you can explicitly specify the type if you want to:

```
var language: String = "French"
val score: Int = 95
```

We have initialized variable during declaration in above examples. However, it's not necessary. You can declare variable and specify its type in one statement, and initialize the variable in another statement later in the program.

```
var language: String      // variable declaration of type String
... .. ...
language = "French"       // variable initialization

val score: Int          // variable declaration of type Int
... .. ...
```

```
score = 95              // variable initialization
```

Here are few examples that results into error.

```
var language            // Error
language = "French"
```

Here, the type of *language* variable is not explicitly specified, nor the variable is initialized during declaration.

```
var language: String
language = 14          // Error
```

Here, we are trying to assign 14 (integer value) to variable of different type (`String`).

Difference Between var and val

- **val** (Immutable reference) - The variable declared using `val` keyword cannot be changed once the value is assigned. It is similar to *final variable in Java*.
- **var** (Mutable reference) - The variable declared using `var` keyword can be changed later in the program. It corresponds to regular Java variable.

Here are few examples:

```
var language = "French"
language = "German"
```

Here, `language` variable is reassigned to *German*. Since, the variable is declared using `var`, this code work perfectly.

```
val language = "French"
language = "German"      // Error
```

You cannot reassign *language* variable to `German` in the above example because the variable is declared using `val`.

Now, you know what Kotlin variables are, it's time to learn different values a Kotlin variable can take.

Kotlin Basic Types

Kotlin is a statically typed language like Java. That is, the type of a variable is known during the compile time.

----------Number Types----------

NB: to run the codes in this document you can go to this link and practice https://play.kotlinlang.org

Numbers in Kotlin are similar to Java. There are 6 built-in types representing numbers.

- Byte
- Short
- Int
- Long
- Float
- Double

1. Byte

- The `Byte` data type can have values from -128 to 127 (8-bit signed two's complement integer).
- It is used instead of `Int` or other integer data types to save memory if it's certain that the value of a variable will be within [-128, 127]
- Example:

```
fun main(args : Array<String>) {
    val range: Byte = 112
    println("$range")
}
```

When you run the program, the output will be:

```
112
```

2. Short

- The `Short` data type can have values from -32768 to 32767 (16-bit signed two's complement integer).
- It is used instead of other integer data types to save memory if it's certain that the value of the variable will be within [-32768, 32767].
- Example:

```
fun main(args : Array<String>) {
```

```
    val temperature: Short = -11245
    println("$temperature")
}
```

When you run the program, the output will be:

```
-11245
```

3. Int

- The `Int` data type can have values from -2^{31} to $2^{31}-1$ (32-bit signed two's complement integer).
- Example:
-

```
fun main(args : Array<String>) {
    val score: Int =  100000
    println("$score")
}
```

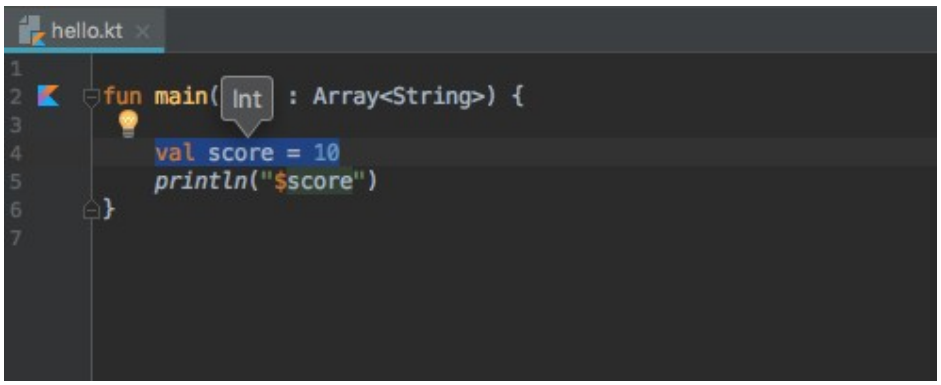
When you run the program, the output will be:

```
100000
```

If you assign an integer between -2^{31} to $2^{31}-1$ to a variable without explicitly specifying its type, the variable will be of `Int` type. For example,

```
fun main(args : Array<String>) {

   // score is of type Int
    val score = 10
    println("$score")
}
```



4. Long

- The `Long` data type can have values from -2^{63} to $2^{63}-1$ (64-bit signed two's complement integer).
- Example:

```
fun main(args : Array<String>) {
    val highestScore: Long = 9999
    println("$highestScore")
}
```

When you run the program, the output will be:

```
9999
```

If you assign an integer value greater than $2^{31}-1$ or less than -2^{31} to a variable (without explicitly specifying its type), the variable will be of `Long` type. For example,

```
val distance = 10000000000  // distance variable of type Long
```

Similarly, you can use capital letter *L* to specify that the variable is of type `Long`. For example,

```
val distance = 100L    // distance value of type Long
```

5. Double

- The `Double` type is a double-precision 64-bit floating point.
- Example:

```
fun main(args : Array<String>) {
    // distance is of type Double
    val distance = 999.5
```

```
    println("$distance")
}
```

When you run the program, the output will be:

```
999.5
```

Float

- The `Float` data type is a single-precision 32-bit floating point. Learn more about single precision and double precision floating point if you are interested.
- Example:

```
fun main(args : Array<String>) {
    // distance is of type Float
    val distance = 19.5F
    println("$distance")
}
```

When you run the program, the output will be:

```
19.5
```

Char

To represent a character in Kotlin, *Char* types are used.

Unlike Java, *Char* types cannot be treated as numbers.

```
fun main(args : Array<String>) {
    val letter: Char
    letter = 'k'
    println("$letter")
}
```

When you run the program, the output will be:

```
k
```

Boolean

- The `Boolean` data type has two possible values, either `true` or `false`.

- Example:

```
fun main(args : Array<String>) {
    val flag = true
    println("$flag")
}
```

Booleans are used in decision making statements (will be discussed in later chapter).