**Basic Datatypes in Kotlin**

**Numbers**

Similar to Java.

Kotlin does not allow internal conversion of different data types.

|  |  |
| --- | --- |
| **Type** | **Size** |
| Double | 64 |
| Float | 32 |
| Long | 64 |
| Int | 32 |
| Short | 16 |
| Byte | 8 |

Example:

fun main(args: Array<String>) {

val a: Int = 10000

val d: Double = 100.00

val f: Float = 100.00f

val l: Long = 1000000004

val s: Short = 10

val b: Byte = 1

println("Your Int Value is "+a);

println("Your Double Value is "+d);

println("Your Float Value is "+f);

println("Your Long Value is "+l);

println("Your Short Value is "+s);

println("Your Byte Value is "+b);

}

**Literal Constants**

There are the following kinds of literal constants for integral values:

Decimals: 123

Longs are tagged by a capital L: 123L

Hexadecimals: 0x0F

Binaries: 0b00001011

NOTE: Octal literals are not supported.

Kotlin also supports a conventional notation for floating-point numbers:

Doubles by default: 123.5, 123.5e10

Floats are tagged by f or F: 123.5f

**Underscores in numeric literals (since 1.1)**

You can use underscores to make number constants more readable:

val oneMillion = 1\_000\_000

val creditCardNumber = 1234\_5678\_9012\_3456L

val socialSecurityNumber = 999\_99\_9999L

val hexBytes = 0xFF\_EC\_DE\_5E

val bytes = 0b11010010\_01101001\_10010100\_10010010

### **Explicit Conversions**

we cannot assign a value of type Byte to an Int variable without an explicit conversion

b: Byte = 1 // OK, literals are checked statically

val i: Int = b // ERROR

We can use explicit conversions to widen numbers

val i: Int = b.toInt() // OK: explicitly widened

print(i)

Every number type supports the following conversions:

toByte(): Byte

toShort(): Short

toInt(): Int

toLong(): Long

toFloat(): Float

toDouble(): Double

toChar(): Char

val l = 1L + 3 // Long + Int => Long

**bitwise operations (**available for Int and Long only**)**

* shl(bits) – signed shift left (Java's <<)
* shr(bits) – signed shift right (Java's >>)
* ushr(bits) – unsigned shift right (Java's >>>)
* and(bits) – bitwise and
* or(bits) – bitwise or
* xor(bits) – bitwise xor
* inv() – bitwise inversion

**Floating Point Numbers Comparison**

The operations on floating point numbers discussed in this section are:

* Equality checks: a == b and a != b
* Comparison operators: a < b, a > b, a <= b, a >= b
* Range instantiation and range checks:

a..b, x in a..b, x !in a..b

**Characters**

Kotlin represents character using **char**.

Character should be declared ***in a single quote*** like ‘c’.

Character variable cannot be declared like number variables.

Kotlin variable can be declared in two ways - one using “**var**” and another using “**val**”.

fun main(args: Array<String>) {

val letter: Char // defining a variable

letter = 'A' // Assigning a value to it

println("$letter")

}

**Boolean**

Like other language, boolean is very simple.

There are two values for Boolean – ***true*** or false.

fun main(args: Array<String>) {

val letter: Boolean // defining a variable

letter = true // Assinging a value to it

println("Your character value is "+"$letter")

}

**Strings**

Strings are character arrays.

Like Java, they are **immutable** in nature.

There are two kinds of string available in Kotlin

- one is called raw String and

- another is called escaped String.

fun main(args: Array<String>) {

var rawString :String = "I am Raw String!"

val escapedString : String = "I am escaped String!\n"

println("Hello!"+escapedString)

println("Hey!!"+rawString)

}

# **Kotlin Variable**

# Variable refers to a memory location. It is used to store data. The data of variable can be changed and reused depending on condition or on information passed to the program.

# **Variable Declaration**

# Kotlin variable is declared using keyword var and val.

var language ="Java"

val salary = 30000

## **Arrays**

Arrays are a collection of homogeneous data. Like Java, Kotlin supports arrays of different data types.

fun main(args: Array<String>) {

val numbers: IntArray = intArrayOf(1, 2, 3, 4, 5)

println("Hey!! I am array Example"+numbers[2])

}

**Kotlin Comment**

* Comments are the statements that are used for documentation purpose.
* Comments are ignored by compiler so that don't execute.
* We can also used it for providing information about the line of code.

There are **two types** of comments in Kotlin.

1. Single line comment.
2. Multi line comment.

**Single line comment**

Single line comment is used for commenting single line of statement. It is done by using '//' (double slash).

For example:

fun main(args: Array<String>) {

// this statement used for print

println("Hello World!")

}

**Multi line comment**

Multi line comment is used for commenting multiple line of statement. It is done by using /\* \*/ (start with slash strict and end with star slash).

For example:

fun main(args: Array<String>) {

/\* this statement

is used

for print \*/

println("Hello World!")

}

<https://kotlinlang.org/docs/reference/basic-syntax.html>

<https://www.javatpoint.com/kotlin-comment>

[https://codelabs.developers.google.com/codelabs/build-your-first-android-app-kotlin/index.html#0](https://codelabs.developers.google.com/codelabs/build-your-first-android-app-kotlin/index.html" \l "0)