

# Basic Data Types

Learn how to represent and store numbers and text in Kotlin.

## WE'LL COVER THE FOLLOWING



- Integers
- Floating Point Numbers
- Text
- Booleans
  - Note: Type Mapping to Java
- Quiz
- Exercise
- Summary

Kotlin is a statically typed language, meaning that the data type of every expression is known at compile time.

## Integers #

There are four basic data types to store integer numbers of different sizes in Kotlin:

```
val byte: Byte = 127
val short: Short = 32767
val int: Int = 2147483647
val long: Long = 9223372036854775807
```



**Note:** The values assigned to each variable above are the largest allowed for the corresponding data type. This gives you an indication of their

magnitude.

If you increase the assigned values even by one, the Kotlin compiler will complain because an overflow would occur at runtime (*try it!*).

## Floating Point Numbers #

Additionally, Kotlin has `Float` and `Double` to store floating point numbers up to different precision and sizes:

```
val float: Float = 3.4028235e38f
val double: Double = 1.7976931348623157e308
```



Here again, the assigned values are the maximum allowed for the corresponding type.

Two things to note:

- The `e` in both values denotes exponentiation, for instance `10e3 == 1000`.
- In order to denote a `Float` value, you have to add the `f` suffix. Otherwise, Kotlin infers `Double` as the type of the number.

## Text #

Kotlin uses the `Char` type for single characters and `String` for arbitrary sequences of characters:

```
val character: Char = '#'
val text: String = "Learning about Kotlin's data types"
```



Single characters are denoted using single quotes `' '`, whereas basic strings use double quotes `" "`.

However, you can also use multiline strings by wrapping your string into three double quotes: `"""<multiline string here>"""`.

# Booleans #

Finally, Kotlin uses `Boolean` to store either `true` or `false`:

```
val yes: Boolean = true
val no: Boolean = false
```



These two are the only valid values for the `Boolean` type.

## Note: Type Mapping to Java

Since Kotlin transpiles to Java bytecode (if you're using it on the JVM), it's important to explore how language concepts translate to Java.

In contrast to Java, Kotlin has no primitive types. All types discussed in this lesson are objects at runtime. However, they do transpile to Java's primitive types in Java bytecode:

Kotlin Type	Type in Java Bytecode
Byte	byte
Short	short
Int	int
Long	long
Float	float
Double	double
Char	char
Boolean	boolean

The mapping is quite trivial, but note that this mapping only works because Kotlin's types cannot be null by default, just like Java's primitive types cannot hold `null`. Kotlin's approach to null safety is explained in detail later in the course.

## Quiz #

### Data Types in Kotlin

1

Which of these are valid data types to hold *integers* in Kotlin?

COMPLETED 0%

1 of 3



## Exercise #

Create a variable `favoriteMovie` that stores the title of your favorite movie and a variable `rating` that contains your rating for it. Ratings range from 0.5 to 5.0 in increments of 0.5.

 Problem

 Solution

```
// Add your variables below
```



# Summary #

Here's what you should take away from this lesson:

- Kotlin has `Byte`, `Short`, `Int`, and `Long` as basic types for integer numbers.
  - Kotlin uses `Float` and `Double` for floating point numbers. A `Float` is denoted with a trailing `f`, as in `17f`.
  - Kotlin has `Char` to store single characters and `String` to store strings of text.
  - Kotlin's basic types map to Java's primitive types when targeting the JVM (and `String` maps to `String`).
- 

In the following lesson, you'll understand how and when the Kotlin compiler can infer the types used in your code.