Lesson 1 - Kotlin basics

Operators

Operators	Symbols
Mathematical operators	`+ - * / %`
Increment and decrement operators	`++ --`
Comparison operators	`< <= > >=`
Assignment operator	`=`
Equality operators	`== !=`

Numeric operator methods: Kotlin keeps numbers as primitives, but lets you call methods on numbers as if they were objects

```
2.times(3)
=> kotlin.Int = 6
```

Data types

- Integer types: `Long`, `Int`, `Short`, `Byte`
    - Underscores for long numbers

    ```
    val oneMillion = 1_000_000
    ```

- Floating-point and other numeric types: `Double`, `Float`, `Char`, `Boolean`
- Strings - any sequence of characters enclosed by double quotes or any arbitrary text delimited by a triple quote (`"""`)

    ```
    val s1 = "Hello World!"

    val text = """
        var bikes = 50
    """
    ```

Type casting

Convert `Int` to `Byte`

```
val i: Int = 6
println(t.toByte())
```

String concatenation and templates

```
val numberOfDogs = 3
val numberOfCats = 2

print("I have $numberOfDogs dogs" + " and $numberOfCats cats")

val s = "abc"
println("$s.length is ${s.length}")
```

Variables

Kotlin is a statically-typed language. The type is resolved at compile time and never changes

- The compiler infer the type or you can explicitly declare the type if needed

```
var width = 12
var width: Int = 12
```

- Mutable and immutable variables (recommended)

```
// mutable (changeable)
var score = 10
// immutable (unchangeable)
val name = "Jennifer"
```

> `val` cannot be reassigned

Conditionals

if/else statements

```
val guests = 30
if (guests == 0) {
    println("No guests")
} else if (guests < 20) {
    println("Small group of people)
} else {
```

```
    println("Large group of people)
}
```

Ranges

```
val numberOfStudents = 50
if (numberOfStudents in 1..100) {
    println(numberOfStudents)
}
```

when statement - like switch_case

```
when (results) {
    0 -> println("No results")
    in 1..39 -> println("Got results!")
    else -> println("That's lot of results!")
}
```

Loops

for loops

```
val pets = arrayOf("dog", "cat", "canary")
for (element in pets) {
    print(element + " ")
}

for ((index, element) in pets.withIndex()) {
    println("Item at $index is $element\n")
}

for (i in 1..5)         // 12345
for (i in 5 downTo 1)   // 54321
for (i in 3..6 step 2)  // 3 5
for (i in 'd'..'g')     // defg
```

while loops

```
var bicycles = 0
while (bicycles < 50) {
    bicycles++
}
```

```
do {
    bicycles--
} while (bicycles> 50)
```

repeat loops

```
repeat(2) {
    print("Hello")
}
```

Lists and Arrays

Lists

- Lists are ordered collection of elements, can be accessed programmatically through their indices
- Elements can occur **more than once** in a list

Immutable list using listOf() - can not add, remove, edit elements

```
val instruments = listOf("trumpet", "piano", "violin")
// [trumpet, piano, violin]
```

Mutable list using mutableListOf()

```
val myList = mutableListOf("trumpet", "piano", "violin")
myList.remove("violin") // return Boolean
```

Set data types

```
val list1 = mutableListOf<Int>()
val list2 = listOf<String>()
```

Arrays

- Arrays store multiple items
- Arrays elements can be accessed programmatically through their indices
- Array elements are **mutable**
- Array size is **fixed**

Array defining examples

```
val pets = arrayOf("dog", "cat", "canary")

// differents types
val mix = arrayOf("hats", 2)

// one type
val numbers = intArrayOf(1, 2, 3)
```

Combining arrays - Using `+` Operator

```
val num1 = intArrayOf(1, 2, 3)
val num2 = intArrayOf(4, 5, 6)
val combined = num1 + num2
```

Null safety

- In Kotlin, variables cannot be `null` by default

  ```
  var numberOfBooks: Int = null // error
  ```

- You can explicitly assign a variable to `null` using the safe call (`?`) operator

  ```
  var numberOfBooks: Int? = null // Int? as nullable
  ```

- Allow null-pointer exceptions using the `!!` operator. `!!` forces the variable into a non-null type

  ```
  var len = s!!.length
  // throws NullPointerException if s is null
  ```

- You can test for null using the elvis (`?:`) operator

  ```
  var numberOfBooks = 6
  if (numberOfBooks != null) {
      numberOfBooks = numberOfBooks.dec()
  } else { numberOfBooks = 0 }
  // or
  numberOfBooks = numberOfBooks?.dec() ?: 0
  ```