IT2010 – Mobile Application Development
BSc (Hons) in Information Technology
2nd Year
Faculty of Computing
SLIIT
2025 – Tutorial 01

Kotlin Basics

What is Kotlin?

Kotlin is a modern, trending programming language that was released in 2016 by JetBrains. It has grown in popularity because it is compatible with JAVA (one of the most popular programming languages).

Kotlin is used for:

- Mobile applications (especially Android apps)
- Web development
- Server-side applications
- Data science

Why Kotlin for Android?

- Less code combined with greater readability.
- Mature language and environment.
- Kotlin support in Android Jetpack and other libraries.
- Interoperability with Java.
- Support for multiplatform development.
- Code safety.
- Easy learning.
- Big community.

Hello World

```
fun main (){
    println("Hello World")
}
```

```
Hello World
```

Functions

```
fun printAge(a: Int, b: String) {
    println("The age of $b is $a")
}

fun main (){
    printAge( a: 12,  b: "Saman")
}
```

```
The age of Saman is 12
```

Comments

```
// This is a single line comment
println("Hello World")
```

```
/* This is a multi-line comment
    (a comment block) */
println("Hello World")
```

Variables

Unlike many other programming languages, variables in Kotlin do not need to be declared with a specified type (like "String" for text or "Int" for numbers, if you are familiar with those).

Only 2 keywords are needed to declare a variable in Kotlin

- Var keyword - can be changed/modified
- Val keyword – the value assigned with the val keyword cannot be changed.

```
val name = "Saman"
var grade = 6
println("$name is in grade $grade")
```

```
Saman is in grade 6
```

```
val name = "Saman"
var grade = 6
grade = 7
println("$name is in grade $grade")
```

```
Saman is in grade 7
```

Basic Data Types

- Numbers and their unsigned counterparts
- Booleans
- Characters
- Strings
- Arrays

```
//Data Types in Kotlin

    //Numbers
    val intvalue = 1 //Integer
    val longValue1 = 100000 //Long
    val longValue2 = 1L //Long
    val byteValue : Byte = 1 //Byte
    val doubleVal = 1.0 //Double
    val floatValue = 1.2384789f //Float
```

```
   /*Boolean - Either TRUE or FALSE can be stored
   in a boolean data type variable
    */
   val trueValue = true //True
   val falseValue = false //False
```

```
    //Characters
    val characterValue = "A"

    //Strings
    val strValue = "kotlin"
```

Arrays

As opposed to generating distinct variables for each value, arrays are used to store numerous values in a single variable.

```
var clothingbrands = arrayOf("GUCCI","Chanel", "Adidas", "ZARA", "Puma")

    //Size of the array
    println("Size of the array is " + clothingbrands.size)

    println("*************")

    //Access an element in an array
    println(clothingbrands[4])

    println("*************")

    //Change an array element
    clothingbrands[0] = "Levi's"

    //Looping through the array
    for (i in clothingbrands){
        println(i)
    }
```

```
Size of the array is 5
*************
Puma
*************
Levi's
Chanel
Adidas
ZARA
Puma
```

1. Create an array that contains the module names, that you'll learn in this semester.

Operators

```
//Operators

val a = 5
val b = 3

//Arithmetic Operators
val sum = a+b //summation
val sub = a-b //subtraction
val mul = a*b //multiplication
val div = a/b.toFloat() //division

//Comparison Operators
val val1 =a>b
val val2 = a==b
val val3 = a!=b

//Logical Operators
val t = true
val f = false

val val4 = t&&f //AND operator
val val5 = t||f //OR operator
val val6 = !t //NOT operator
```

```
sum = 8
sub = 2
mul = 15
div = 1.6666666
val1 = true
val2 = false
val3 = true
val4 = false
val5 = true
val6 = false
```

2. Write a Kotlin function to calculate the circumference of a circle for a given radius value.

Conditions and Loops

If Expression

```
var age = 5
if (age > 18){
    println("The person is an adult.")
}
else {
    println("The person is a child.")
}
```

```
The person is a child.
```

When Expression

```
var x = 5
    when (x) {
        0, 1 -> println("x == 0 or x == 1")
        else -> println("otherwise")
    }
```

```
otherwise
```

For Loop

```
for (i in 1 ≤ .. ≤ 10){
    print("$i ")    //1 2 3 4 5 6 7 8 9 10
}
```

```
for (i in 0 ≤ .. ≤ 20 step 5){
    print("$i ")    //0 5 10 15 20
}
```

```
for (i in 1 ≤ until < 10){
    print("$i ")    //1 2 3 4 5 6 7 8 9
}
```

```
for (i in 1 ≤ .. ≤ 10){
    print("$i ")    //1 2 3 4 5

    if (i==5){
        break
    }
}
```

```
for (i in 1 ≤ .. ≤ 10){
    if (i==5){
        continue
    }
    print("$i ")    //1 2 3 4 6 7 8 9 10
}
```

While Loop

```
var i = 0
while (i < 5) {
    println(i)
    i++
}
```

```
0
1
2
3
4
```

3. Write a Kotlin function to calculate the grades of a student for a given mark according to the following criteria.

100-75	A
74-65	B
64-50	C
49-35	D
34-00	F

4. Write a Kotlin function to determine whether a given integer number is prime.

Kotlin Generics

Generics in Kotlin is a type-safe way of writing code that can work with different types. Instead of writing separate versions of the code for each type, generics allow you to write a single function or class that can work with multiple different types. To use generics in Kotlin, the type parameter in angle brackets is specified when defining a function or class.

```
fun <T> sort(items: List<T>) {
    /* Use the suitable method
    for the type T to run the function.*/
}
```

Kotlin Collections

A collection usually contains several objects of the same type and these objects in the collection are called elements or items.

Collection types in Kotlin

- List - an ordered collection with access to elements by indices.
- Set - a collection of unique elements. The order of set elements has no significance.
- Map - a set of key-value pairs. Keys are unique, and each of them maps to exactly one value. The values can be duplicates.

In Kotlin, collections are categorized into two forms.

- Immutable Collection – This category supports only read-only functionalities
    - **List** –listOf<T>()
    - **Set** – setOf<T>()
    - **Map** – mapOf<K,V>()

- Mutable Collection – This supports both read and write functionalities (adding, removing, updating)
    - **List** – mutablelistOf<T>()
    - **Set** – mutablesetOf<T>()
    - **Map** – mutablemapOf<K,V>()

List

```
//immutable list
val family = listOf("father","mother","sister")

println(family[0])  //father

for (i in family){
    print("$i ")    //father mother sister
}
```

```
println(family.size)    //3
```

```
//mutable list
val family = mutableListOf("father","mother")
family.add("brother")

family.remove( element: "father")

println(family[0])  //mother

for (i in family){
    print("$i ")    // mother brother
}
```

Set

```
//immutable set
val family = setOf("father","mother","mother","brother","brother","sister")

for(i in family){
    print("$i ") //father mother brother sister
}
```

```
//mutable set
val family = mutableSetOf("father","mother","mother","brother","brother","sister")

family.add("grandmother")
family.remove( element: "sister")

for(i in family){
    print("$i ") //father mother brother grandmother
}
```

Map

```
//immutable map
val grade = mapOf("stu1" to "A","stu2" to "B","stu3" to "A")

println(grade["stu1"])  //A
println("All students : ${grade.keys}")
        //All students : [stu1, stu2, stu3]
println("All grades : ${grade.values}")
        //All grades : [A, B, A]
```

```
//mutable map
val grade = mutableMapOf("stu1" to "A","stu2" to "B","stu3" to "A")

grade.remove( key: "stu1")
grade["stu4"] = "C"

println("All students : ${grade.keys}")
        //All students : [stu2, stu3, stu4]
println("All grades : ${grade.values}")
        //All grades : [B, A, C]
```

5. a. Create a mutable list that contains multiplications of 3 from 0 to 100.
   b. Replace all the even numbers of above list from 999.