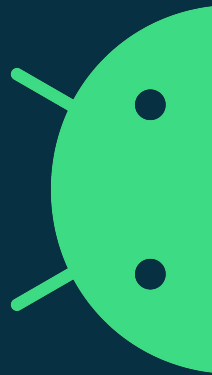


# Hour of Code Kotlin

Iniciación a kotlin

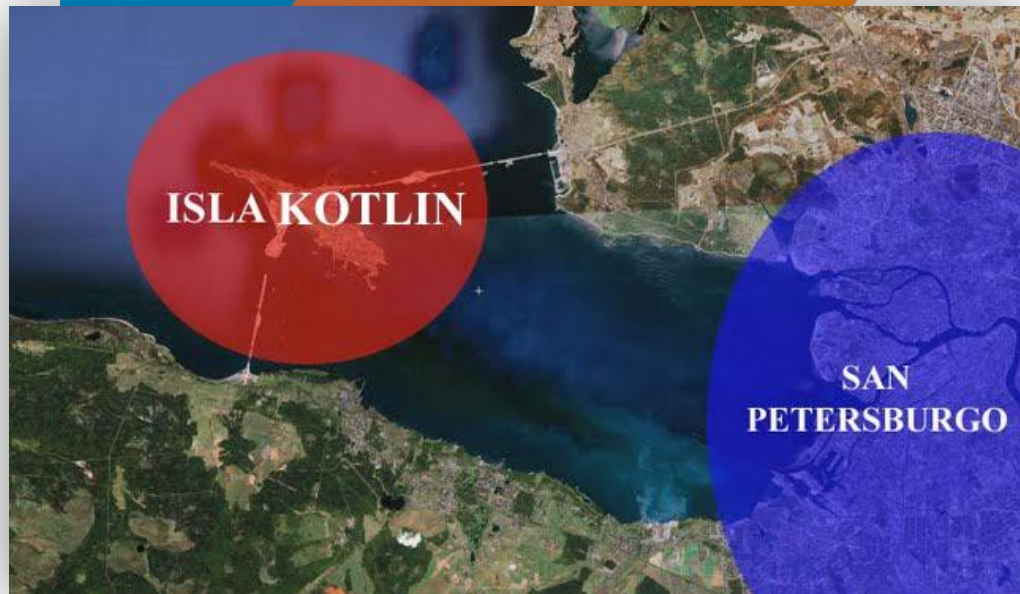


# Kotlin

¿Que es?

# ¿Que es kotlin?

- lenguaje tipado y estático
- corre sobre la máquina virtual de Java
- puede ser compilado a código fuente de JavaScript
- desarrollado principalmente por JetBrains



# Usos

- Desarrollo móvil multiplataforma.
- Programación del lado del servidor.
- Ciencia de datos.
- Android.



# Prerequisites

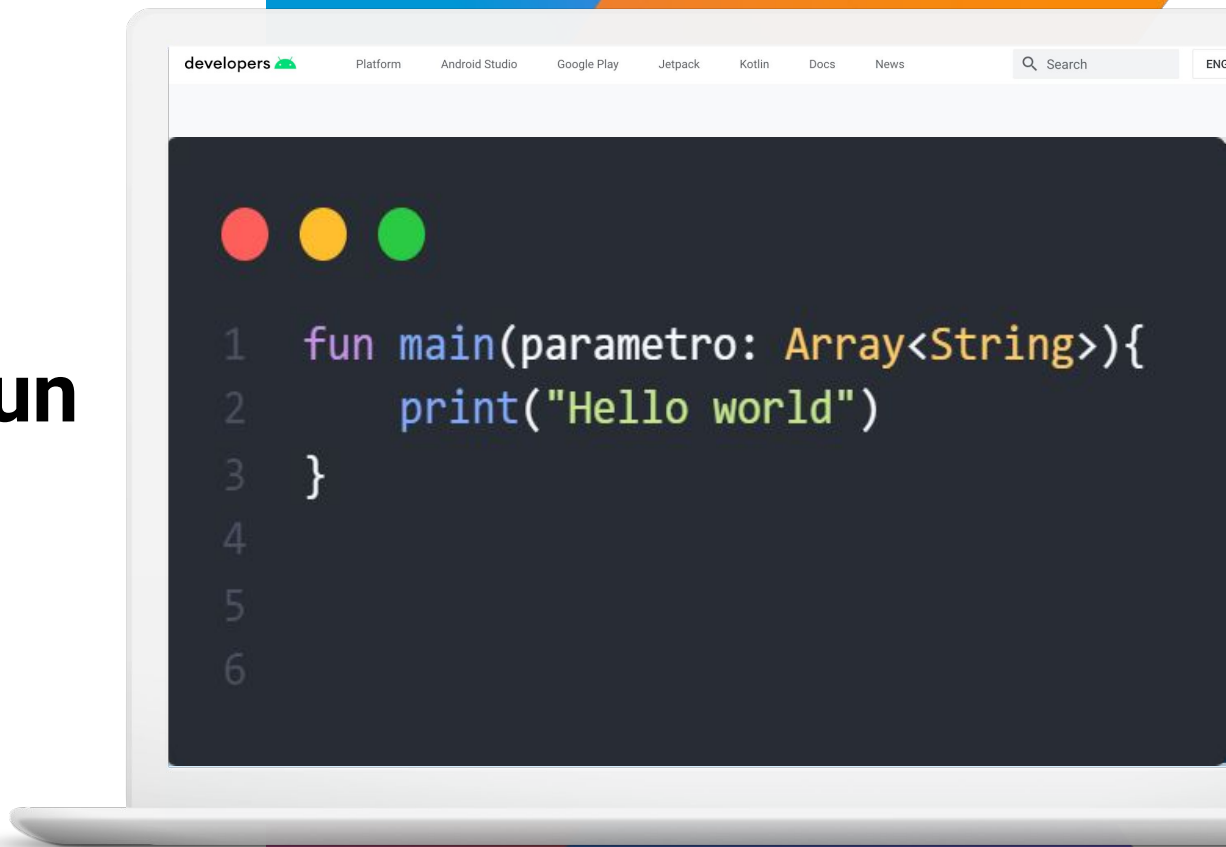
- Tener instalado IntelliJ IDEA o Kotlin.
- Saber algún lenguaje de programación.
- Orientación a objetos.
- Ganas de aprender.
- Ganas de programar.



# Empezemos

**Who would like to share?**

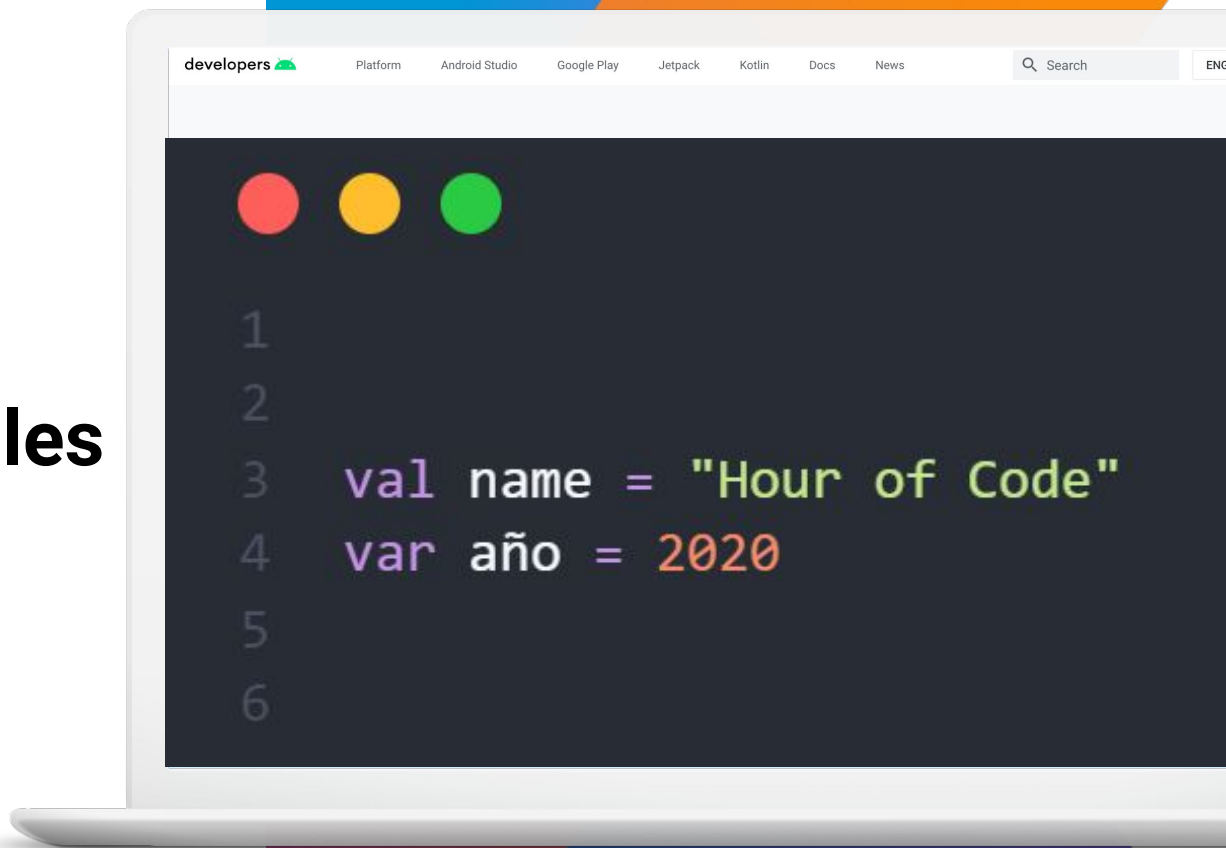
# Calentemos con un “Hola Mundo”





# Tipos de variables

# Dos tipos de variables



# Tipos básicos

- Numbers
- String
- Characters
- Boolean
- Arrays



# Numbers

- Long: 64 Bits
- Int: 32 Bits
- Short: 16 Bits
- Byte: 8 Bits
- Double: 64 Bits
- Float: 32 Bits





```
1
2  val myNumber: Long = 40
3  val MyNumber: Int = 30
4  val sueldo = 1200.55f
5  val titulo = "Sistema de Ventas"
6  var valor1 = 10
7  var valor2 = 100
8  println("La suma de $valor1 + $valor2 es $resultado")
9
10
11
12
13
```

# Conversiones de datos

- `toByte(): Byte`
- `toInt(): Int`
- `toLong(): Long`
- `toFloat(): Float`
- `toChar(): Char`
- `toDouble(): Double`



```
1  val b: Byte = 1 // OK, literals are checked statically
2  val i: Int = b // ERROR
3  val i: Int = b.toInt() // OK: explicitly widened
4  print(i)
```

# Bitwise operations

- shl(bits) – signed shift left
- shr(bits) – signed shift right
- ushr(bits) – unsigned shift right
- and(bits) – bitwise and
- or(bits) – bitwise or
- xor(bits) – bitwise xor
- inv() – bitwise inversion



```
1  val b: Byte = 1 // OK, literals are checked statically
2  val i: Int = b // ERROR
3  val i: Int = b.toInt() // OK: explicitly widened
4  print(i)
```

# Arrays

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1  class Array<T> private constructor() {  
2      val size: Int  
3      operator fun get(index: Int): T  
4      operator fun set(index: Int, value: T): Unit  
5  
6      operator fun iterator(): Iterator<T>  
7      // ...  
8  }  
9  
10
```

Kotlin





```
1 // Crea un Array<String> con valores ["0", "1", "4", "9", "16"]
2 val asc = Array(5) { i -> (i * i).toString() }
3 asc.forEach { println(it) }
4
5 val x: IntArray = intArrayOf(1, 2, 3)
6 x[0] = x[1] + x[2]
7 // Array de enteros con tamaño 5 con valores [0, 0, 0, 0, 0]
8
8 val arr = IntArray(5)
9
10 // e.g. Inicializar los valores de un array con una constant
   e
11 // Array de tamaño 5, con valores [42, 42, 42, 42, 42]
12 val arr = IntArray(5) { 42 }
13
14 // e.g. inicializar los valores con una función Lambda
15 // Array de enteros con tamaño 5 y valores
16 // [0, 1, 2, 3, 4]
17 var arr = IntArray(5) { it * 1 }
```

# String

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1  val s = "Hello, world!\n"
2
3  val text = """
4      |Tell me and I forget.
5      |Teach me and I remember.
6      |Involve me and I learn.
7      |(Benjamin Franklin)
8      """.trimMargin()
9
10 val i = 10
11 println("i = $i") // prints "i = 10"
12 val s = "abc"
13 println("$s.length is ${s.length}")
    // prints "abc.length is 3"
```

Kotlin

# Clases y objetos

# Constructor

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1  class Constructors {  
2      init {  
3          println("Init block")  
4      }  
5  
6      constructor(i: Int) {  
7          println("Constructor")  
8      }  
9  }
```

Kotlin

# Clases abstractas

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

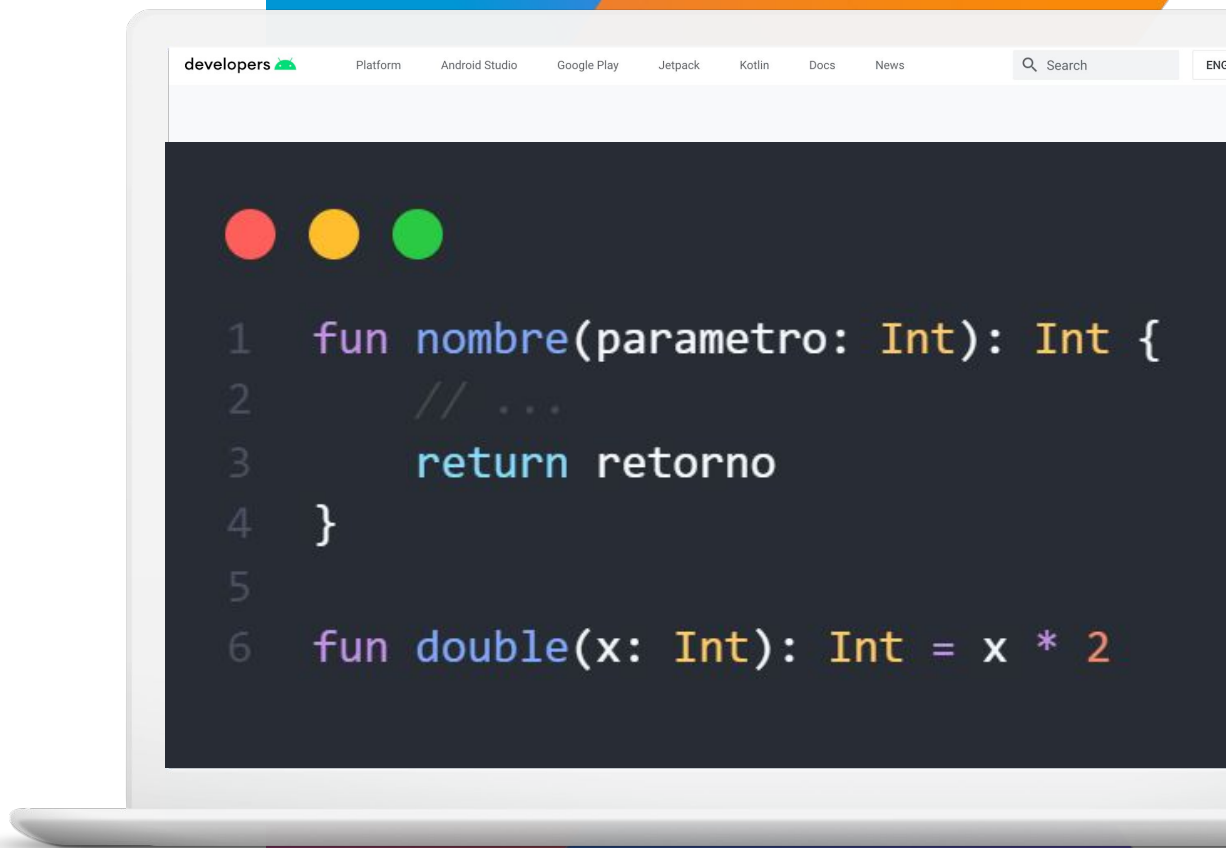
ENG



```
1  open class Polygon {  
2      open fun draw() {}  
3  }  
4  
5  abstract class Rectangle : Polygon() {  
6      abstract override fun draw()  
7  }
```

# Función

# Estructura



# Funciones locales

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

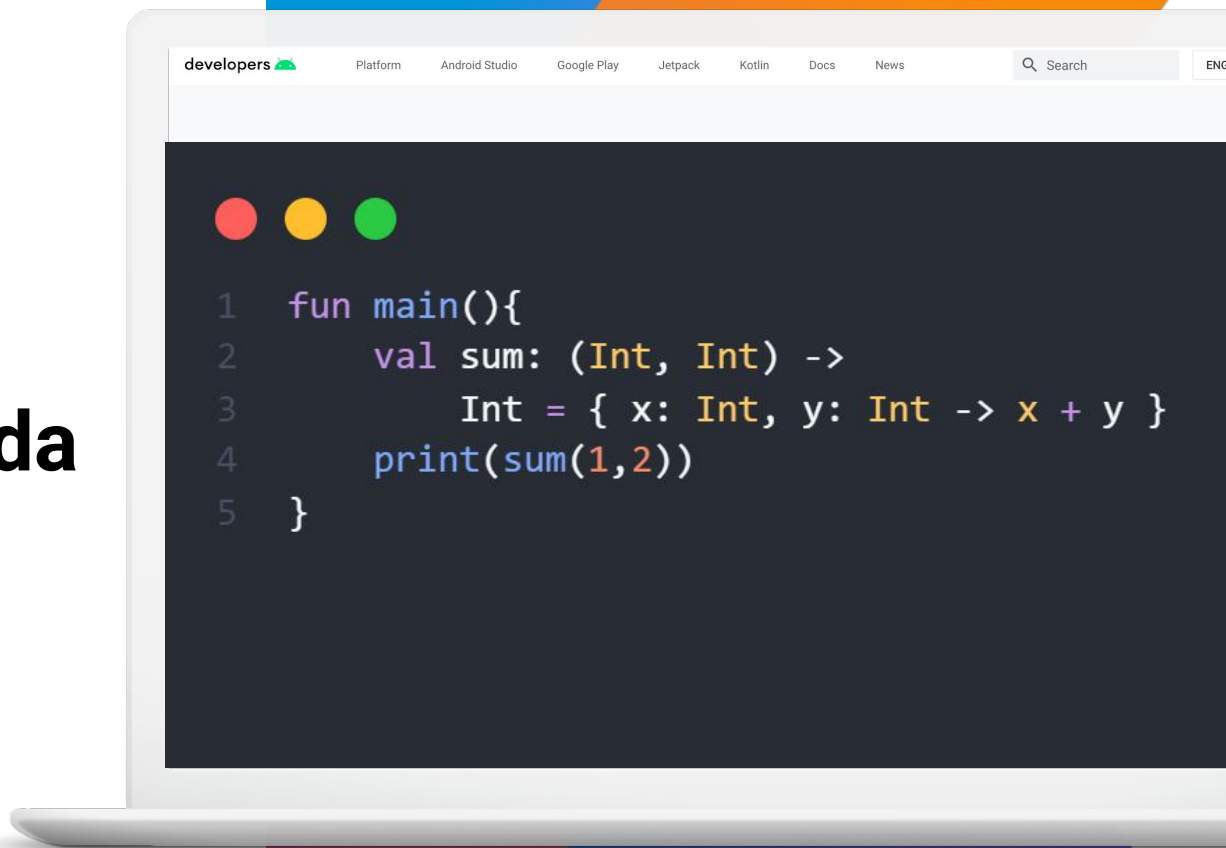
```
1 fun nombre(parametro: Int): Int {  
2     fun funLocal(){  
3         //..  
4     }  
5     return retorno  
6 }
```

Kotlin



# Funciones Lambda

# Ejemplo Lambda



# Funciones inLine



```
1 fun retornarSuperficie(lado: Int) = lado * lado
2
3 fun main() {
4     print("Ingrese el valor del lado del cuafrado:")
5     val la = readLine()!!.toInt()
6     println("La superficie del cuadrado es ${
7     retornarSuperficie(la)}")
8 }
```

# Iterators

# while

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1  val numbers = listOf("one", "two", "three", "four")
2  val numbersIterator = numbers.iterator()
3  while (numbersIterator.hasNext()) {
4      println(numbersIterator.next())
5  }
6
7
8
```

Kotlin

# for

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

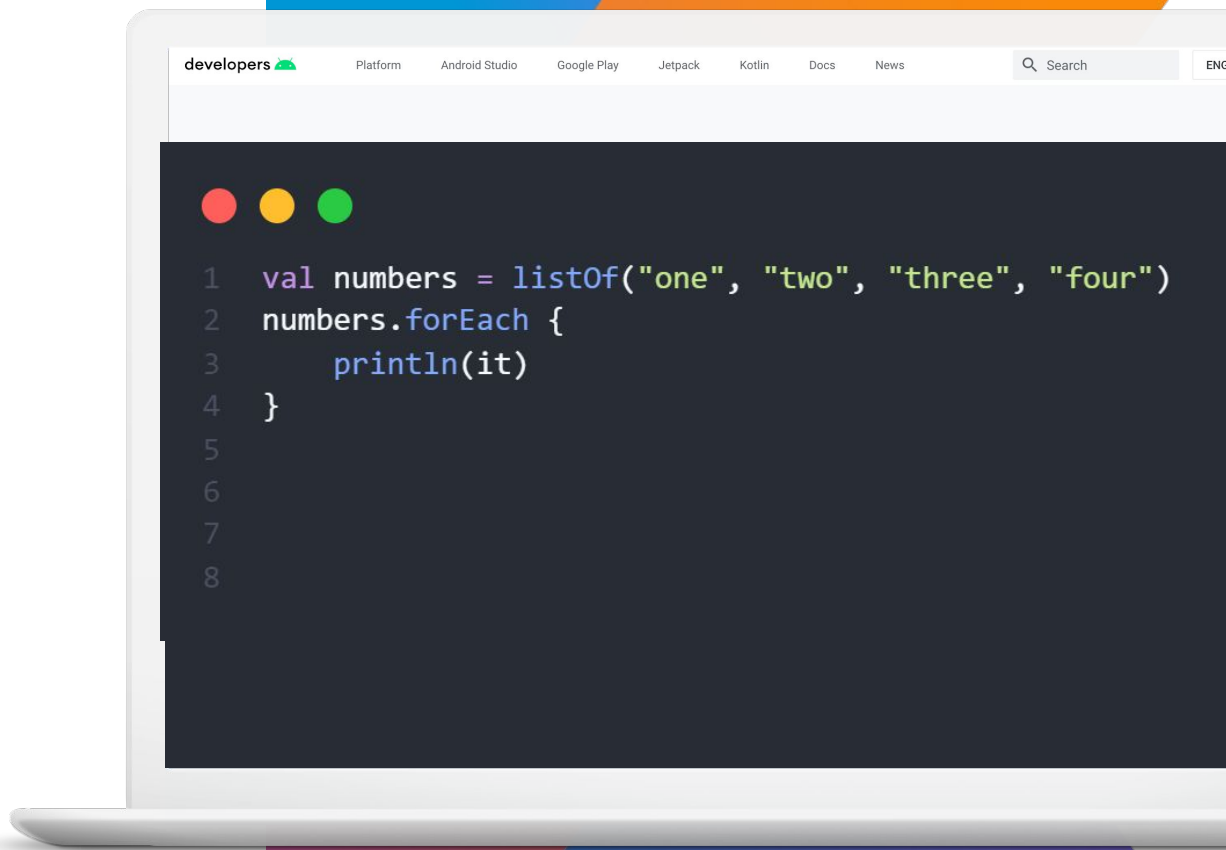
Search

ENG

```
1  val numbers = listOf("one", "two", "three", "four")
2  for (item in numbers) {
3      println(item)
4  }
```

Kotlin

# foreach





# Range

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1 for (i in 1..4) print(i)
2 for (i in 4 downTo 1) print(i)
3 for (i in 1..8 step 2) print(i)
4 println((1..10).filter { it % 2 == 0 })
5
6
7
8
```

Kotlin

# When

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1 print("Ingrese coordenada x del punto:")
2 val x = readLine()!!.toInt()
3 print("Ingrese coordenada y del punto:")
4 val y = readLine()!!.toInt()
5 when {
6     x > 0 && y > 0 -> println("Primer cuadrante")
7     x < 0 && y > 0 -> println("Segundo cuadrante")
8     x < 0 && y < 0 -> println("Tercer cuadrante")
9     x > 0 && y < 0 -> println("Cuarto cuadrante")
10    else -> println("El punto se encuentra en un eje")
11 }
```

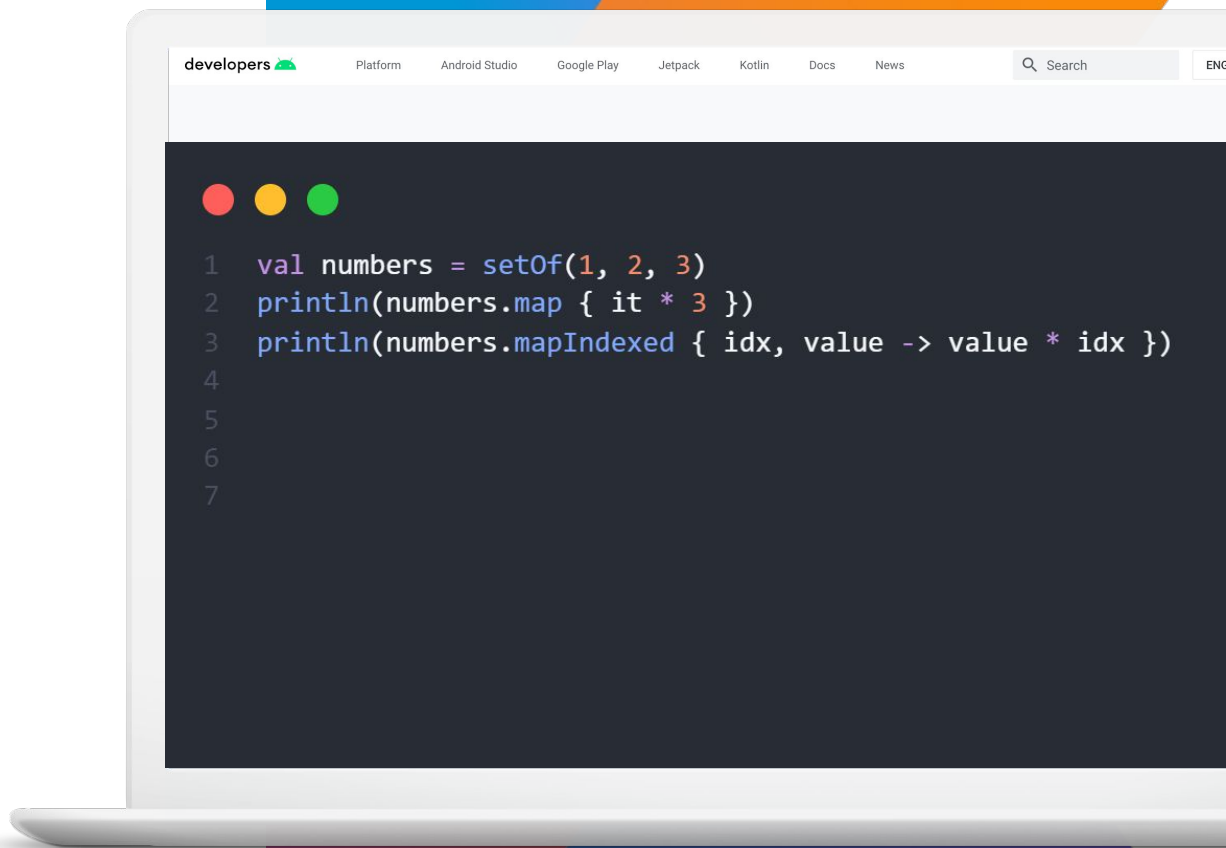
Kotlin



```
1  print("Ingrese un valor entero entre 1 y 5:")
2      val valor = readLine()!!.toInt()
3      when (valor) {
4          1 -> print("uno")
5          2 -> print("dos")
6          3 -> print("tres")
7          4 -> print("cuatro")
8          5 -> print("cinco")
9          else -> print("valor fuera de rango")
10     }
```

# Collection Transformation

# Mapping



# Zippping

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1  val colors = listOf("red", "brown", "grey")
2  val animals = listOf("fox", "bear", "wolf")
3  println(colors zip animals)
4
5  val twoAnimals = listOf("fox", "bear")
6  println(colors.zip(twoAnimals))
7
```

Kotlin

# Association

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1 val numbers = listOf("one", "two", "three", "four")
2 println(numbers.associateWith { it.length })
3
4 val names = listOf("Alice Adams", "Brian Brown",
5 "Clara Campbell")
6 println(names.associate { name -> parseFullName(name).let
7     { it.lastName to it.firstName } })
```

Kotlin

# Flattening

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1 val numberSets = listOf(setOf(1, 2, 3), setOf(4, 5, 6), setOf(
2     1, 2))
3     println(numberSets.flatten())
4
5 val containers = listOf(
6     StringContainer(listOf("one", "two", "three")),
7     StringContainer(listOf("four", "five", "six")),
8     StringContainer(listOf("seven", "eight"))
9 )
10 println(containers.flatMap { it.values })
```

Kotlin



# Filtering

developers

Platform

Android Studio

Google Play

Jetpack

Kotlin

Docs

News

Search

ENG

```
1 val numbers = listOf("one", "two", "three", "four")
2 val longerThan3 = numbers.filter { it.length > 3 }
3 println(longerThan3)
4
5 val filteredNot = numbers.filterNot { it.length <= 3 }
6 println(filteredNot)
7
```

Kotlin

# Muchas gracias por asistir

Si queréis contactar con nosotros:



Angel Moreno  
[github.com/angelmorenocalvo](https://github.com/angelmorenocalvo)  
[@\\_\\_angelmoreno](#)



Gonzalo Calvo  
[github.com/GonzaCS](https://github.com/GonzaCS)  
[@Gonza\\_cs175](#)