**Taller Estructuras de Datos en Kotlin**

**¿Qué son las estructuras de datos y para qué se utilizan?**

La estructura de datos nos permite almacenar varios valores , generalmente del mismo tipo de datos, de manera ordenada que puede tener un agrupación de valores únicos o una asignación de valores de un tipo de datos.

**Ventajas de utilizar estructuras de datos en Kotlin**

Las ventajas de trabajar estructuras de datos con kotlin es que nos permite controlar esa incógnita de error más habitual (“null”) a la hora de desarrollar las aplicaciones dando la tranquiliada de que , si lo hace bien, la app no va a fallar en ningún momento.

**Diferencias entre las estructuras de datos en kotlin y java**

Kotlin se basa en la infraestructura de Java y, por lo tanto, comparte muchos de los mismos tipos de datos y estructuras, como ArrayList, LinkedList, HashMap y TreeMap. Sin embargo, existen algunas diferencias clave. Por ejemplo, Kotlin tiene una estructura de datos llamada MutableList que no está disponible en Java. Esta estructura de datos permite que los elementos de una lista sean modificados, lo que no es posible con la Lista de Java.

**Arreglos en kotlin**

Un arreglo en Kotlin es una estructura de datos que almacena un conjunto de elementos de un tipo específico. Estos elementos se almacenan en una ubicación contigua de memoria, lo que significa que en un arreglo los elementos se almacenan uno después del otro. Esto permite que se acceda a ellos rápidamente mediante un índice.

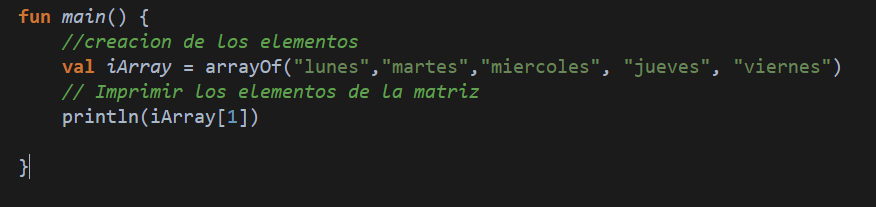
**Ejemplo**

Val x = arrayOf(“luis”, “juan”,”luisa”)

Println(x[1])

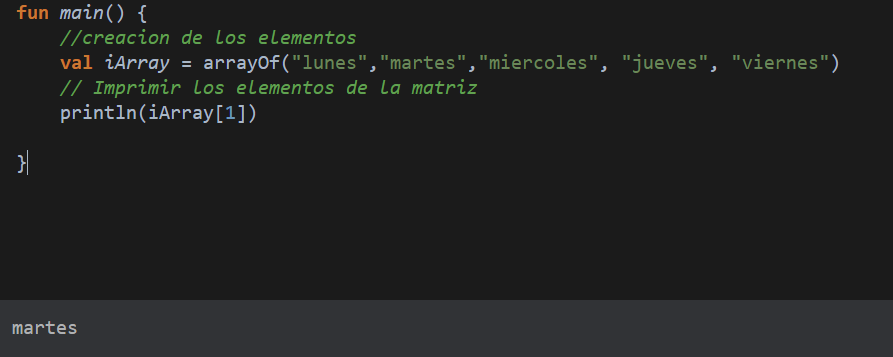
**Creacion de arreglos en kotlin**

Para crear un arreglo usamos la matriz “arrayOf” para agregar los elementos

****

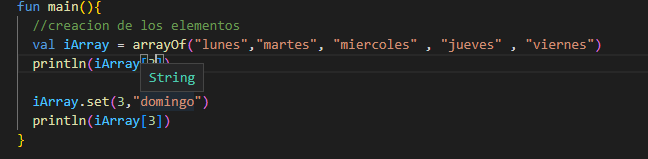
**Accediendo a los elementos de un arrgelo**

Para acceder a un elemento debemos saber que la posición inicial de un elemento empieza desde 0 para capturar un elemento se tiene que llamar la variable y entre corchetes ponemos la posición del elemento que queremos mostrar

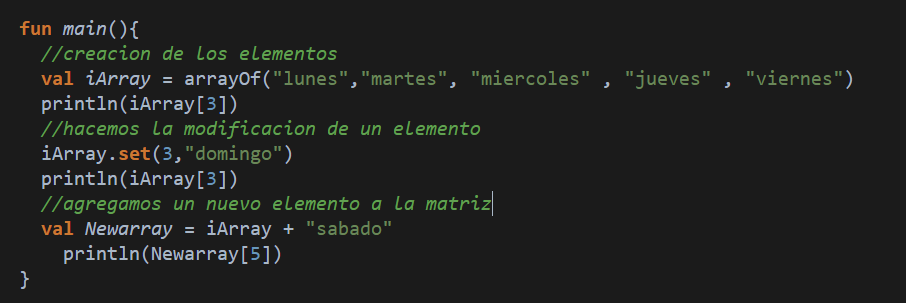


**Modificando los eventos de una lista**

Para modificar los elementos de una lista en Kotlin, puedes usar la función set() para cambiar el valor de un elemento en particular. Por ejemplo, para cambiar el valor del elemento



También puedes crear una nueva variable en la cual vamos agregar un nuevo elemento

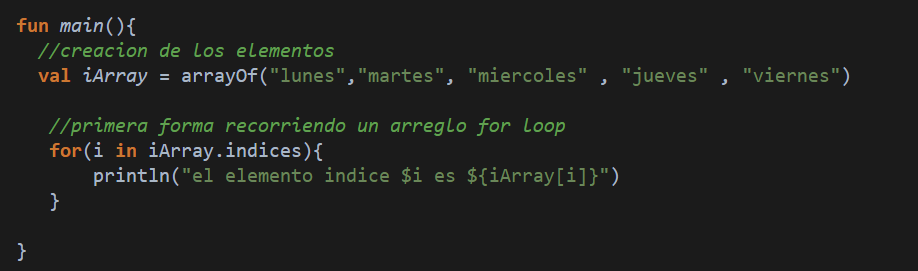


**Recorriendo un arreglo**

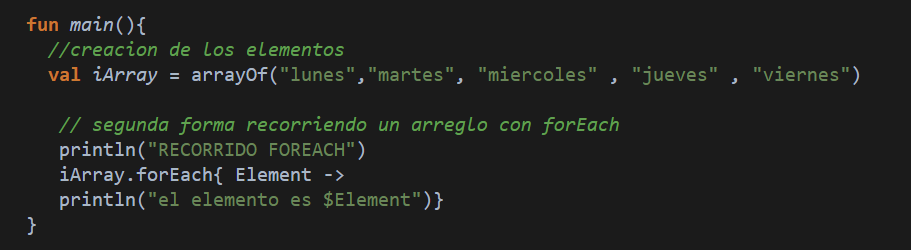
hay varias formas de recorrer un arreglo. Una forma es utilizar un "for loop" o “forEach” para recorrer cada elemento en el arreglo. Esto se vería algo así:

for (item in array){

For loop



forEach

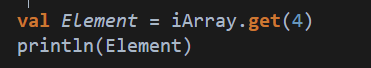


**Funciones útiles para trabajar con arreglos en kotlin**

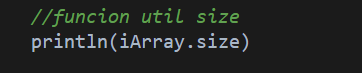
**Set()**

****

**Get()**

****

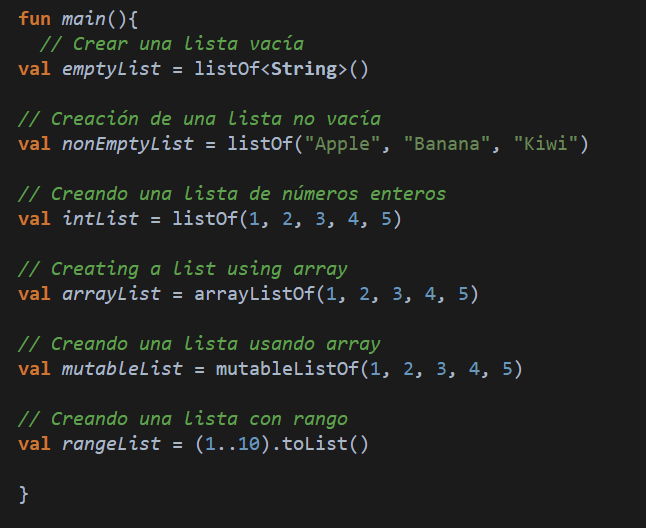
**Size**

****

**Listas en kotlin**

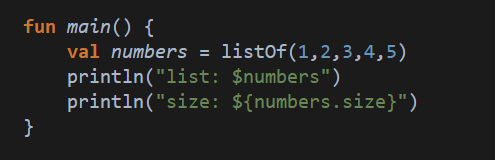
**¿Qué es una lista?**

Una lista en Kotlin es una secuencia ordenada de elementos. Está compuesto por un conjunto de objetos que se pueden referenciar usando índices. La clase List de Kotlin proporciona operaciones genéricas para almacenar, recuperar, eliminar y reemplazar elementos de una lista.

**Creación de listas en kotlin**

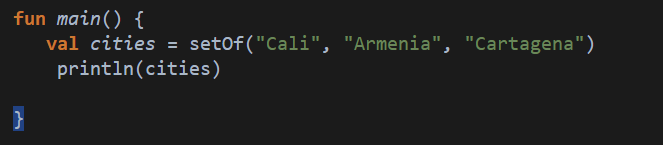
**Accediendo a los elementos de una lista**

Para acceder a una lista usamos el mismo método que un array nos suelta la lista de elementos y su tamaño



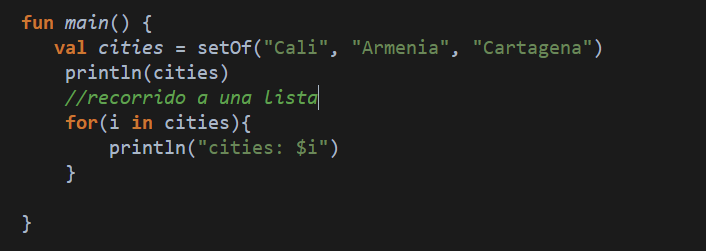
**Modificando los elementos de una lista**

Modificar los elementos de una lista en Kotlin es sencillo, ya que se puede utilizar la función set(), pasándole como parámetros el índice del elemento a modificar y el nuevo valor.

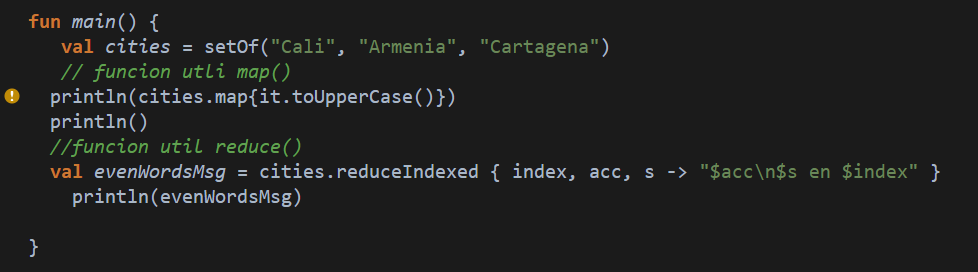
****

**Recorriendo una lista**

Recorriendo una lista en Kotlin es muy sencillo, gracias al uso de los ciclos for. Estos permiten recorrer todos los elementos de la lista, accediendo a su contenido y realizando alguna operación con ellos.



**Funciones útiles para trabajar con listas en kotlin**

****

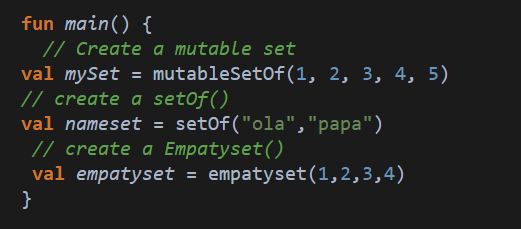
**Conjuntos en kotlin**

**¿Que es un conjunto?**

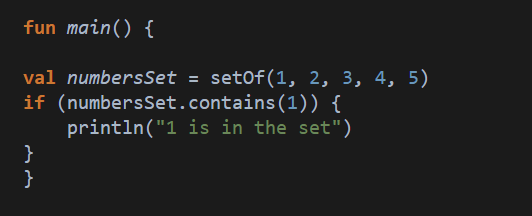
Un conjunto es una colección de objetos diferentes que están relacionados entre sí. Los conjuntos no tienen ningún orden específico y no contienen elementos duplicados.

**Creación de conjuntos en Kotlin**

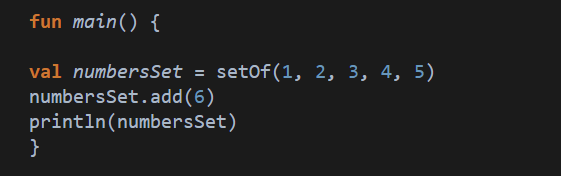
En Kotlin, los conjuntos se pueden crear usando la palabra clave "setOf ()" para crear un conjunto inmutable, o "mutableSetOf ()" para crear un conjunto modificable. Estas funciones devuelven un conjunto inmutable o modificable, respectivamente, que contiene los elementos especificados como argumentos. También se pueden crear conjuntos vacíos usando la función "emptySet ()".

****

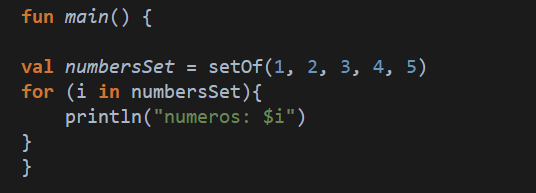
**Accediendo a los elementos de un conjunto**



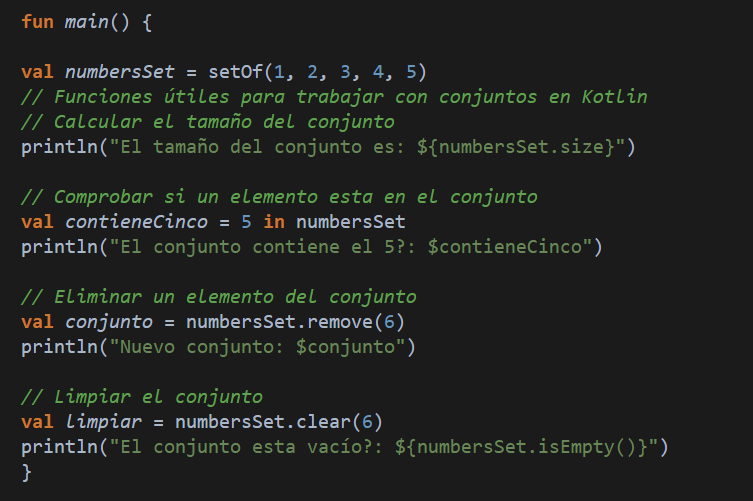
**Modificando los elementos de un conjunto**

****

**Recorriendo un conjunto**

****

**Funciones útiles para trabajar con conjuntos en Kotlin**

****

Mapas en Kotlin

**¿Qué es un mapa?**

Un mapa es una estructura de datos que almacena pares de clave-valor. Se trata de una colección no ordenada de elementos que vincula un valor a una clave única, permitiendo así a los usuarios buscar elementos rápidamente basándose en su clave.

**Creación de mapas en Kotlin**

**Texto

Descripción generada automáticamente**

**Accediendo a los elementos de un mapa**

**Texto

Descripción generada automáticamente**

**Modificando los elementos de un mapa**

**Texto

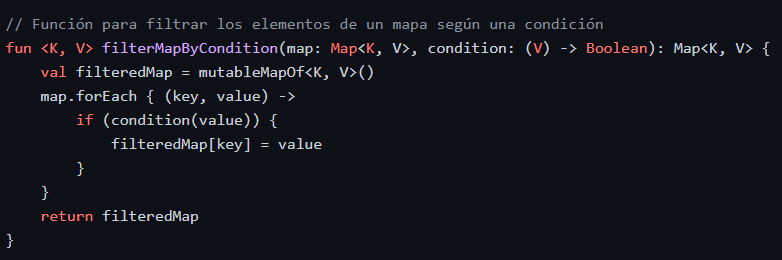
Descripción generada automáticamente**

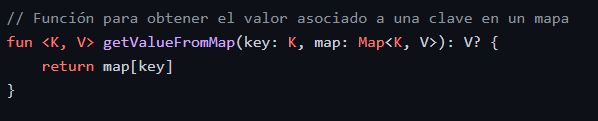
**Recorriendo un mapa**

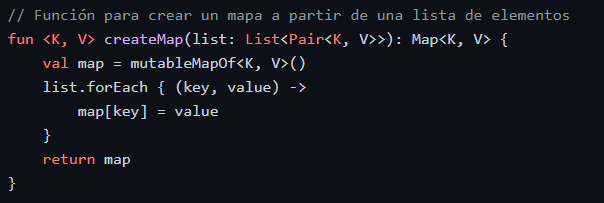
**Texto

Descripción generada automáticamente**

**Funciones útiles para trabajar con mapas en Kotlin**

****

****

****

**Pares en Kotlin**

**¿Qué es un par?**

Un par en programación es una estructura de datos que contiene dos valores relacionados entre sí. Estos dos valores se conocen como los elementos del par y pueden ser del mismo tipo o de diferentes tipos. Un par es útil cuando desea almacenar dos valores juntos para su posterior procesamiento o para enviarlos a una función.

**Creación de pares en Kotlin**

En Kotlin, un par se crea usando el operador “to”. El operador “to” toma dos argumentos, los cuales se convertirán en los elementos del par. Por ejemplo, para crear un par con dos números enteros sería:

****

También se puede crear un par usando la función Pair():

****

**Accediendo a los elementos de un par**

Para acceder a los elementos de un par, puede usar el operador “.componentN” donde N es el número del elemento. Por ejemplo, para acceder al primer elemento del par anterior:

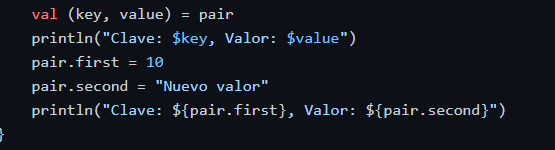
****

También puede usar el operador de desempaquetado para acceder a los elementos de un par. Por ejemplo:

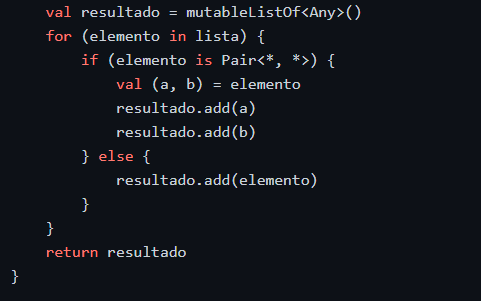
****

**Modificando los elementos de un par**

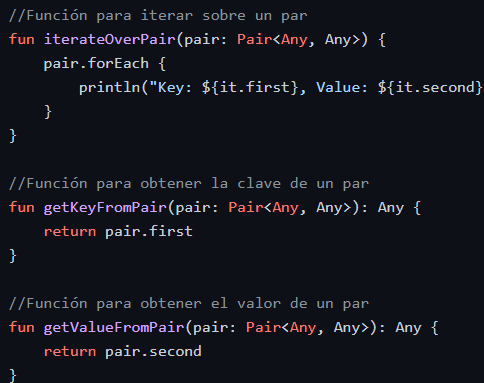
Los elementos de un par no se pueden modificar directamente, ya que son inmutables. Sin embargo, puede reasignar los elementos a un par nuevo. Por ejemplo, para cambiar el primer elemento del par anterior:



**Recorriendo un par**

****

**Funciones útiles para trabajar con pares en Kotlin**

****

**Prácticas de estructuras de datos en Kotlin**

**Ejercicios prácticos para aplicar los conceptos aprendidos**

1. Crea un par que asocie un nombre de una fruta con su precio y muestra sus elementos por pantalla.
2. Crea un mapa que asocie nombres de países con sus capitales y muestra sus elementos por pantalla.
3. Crea un conjunto de números y muestra sus elementos por pantalla.
4. Crea una lista de strings y muestra sus elementos por pantalla.

**Solución a los ejercicios prácticos**

**1.**

fun main() {

    val frutaPrecio = Pair("Manzana", 2.5)

    println("La ${frutaPrecio.first} cuesta ${frutaPrecio.second} euros.")

}

**2.**

fun main() {

    val capitales = mapOf(

        "España" to "Madrid",

        "Francia" to "París",

        "Italia" to "Roma",

        "Alemania" to "Berlín"

    )

    for ((pais, capital) in capitales) {

        println("La capital de $pais es $capital")

    }

}

**3.**

fun main() {

    val numeros = setOf(5, 8, 3, 1, 5, 8, 2)

    for (numero in numeros) {

        println(numero)

    }

}

**4.**

fun main() {

    val nombres = listOf("Juan", "María", "Pedro", "Ana")

    for (nombre in nombres) {

        println(nombre)

    }

}