#Ejercicios listas:

*1.*

val numbers = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

val greaterThanFive = numbers.filter(\_ > 5)

println(greaterThanFive) // Output: List(6, 7, 8, 9, 10)

*2.*

val numbers = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) // Puedes extender hasta 100 si lo deseas

val sum = numbers.sum

println(sum) // Output: 55 (si solo se toman del 1 al 10)

*3.*

val names = List("Alice", "Bob", "Charlie", "David")

val uppercaseNames = names.map(\_.toUpperCase)

println(uppercaseNames) // Output: List(ALICE, BOB, CHARLIE, DAVID)

#Ejercicios conjuntos:

*1.*

val numbers = Set(1, 2, 3, 4, 5)

// Añadir los números 6 y 7

val updatedNumbers = numbers + 6 + 7

println(s"Original: $numbers") // Output: Original: Set(1, 2, 3, 4, 5)

println(s"Updated: $updatedNumbers") // Output: Updated: Set(1, 2, 3, 4, 5, 6, 7)

*2.*

val numbers = Set(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

// Filtrar números impares

val oddNumbers = numbers.filter(\_ % 2 != 0)

println(oddNumbers) // Output: Set(1, 3, 5, 7, 9)

*3.*

val allDays = Set("lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo")

val weekendDays = Set("sábado", "domingo")

// Encontrar la intersección

val weekendIntersection = allDays.intersect(weekendDays)

println(weekendIntersection) // Output: Set(sábado, domingo)

#Ejercicios mapas:

*1.*

val products = Map("manzana" -> 0.5, "plátano" -> 0.3, "naranja" -> 0.4)

// Añadir un nuevo producto

val updatedProducts = products + ("kiwi" -> 0.6)

println(s"Original: $products") // Output: Original: Map(manzana -> 0.5, plátano -> 0.3, naranja -> 0.4)

println(s"Updated: $updatedProducts") // Output: Updated: Map(manzana -> 0.5, plátano -> 0.3, naranja -> 0.4, kiwi -> 0.6)

*2.*

val products = Map("manzana" -> 0.5, "plátano" -> 0.3, "naranja" -> 0.4, "kiwi" -> 0.6)

// Filtrar productos que cuestan más de 0.4

val expensiveProducts = products.filter { case (name, price) => price > 0.4 }

println(expensiveProducts) // Output: Map(manzana -> 0.5, kiwi -> 0.6)