databrickspracticas

(https://databricks.com)

Ejercicios: Listas

2

```
// 1. Crea una lista inmutable de números enteros del 1 al 10. Filtra y genera una nueva lista que contenga solo l
son mayores que 5.
println("\n1. Filtrar números mayores que 5")
val numeros = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) // Crear la lista inmutable
val mayoresQueCinco = numeros.filter(_ > 5) // Filtrar números mayores que 5
println(s"Números mayores que 5: $mayoresQueCinco")

// 2. Crea una lista inmutable de números del 1 al 100. Calcula la suma de todos los números en la lista y muestra
println("\n2. Calcular la suma de los números del 1 al 100")
val numerosHastaCien = (1 to 100).toList // Crear la lista inmutable del 1 al 100
val sumaTotal = numerosHastaCien.sum // Calcular la suma de los números
println(s"Suma total de los números del 1 al 100: $sumaTotal")

// 3. Crea una lista inmutable de nombres de personas. Utiliza map para crear una nueva lista que contenga todos l
mayúsculas.
println("\n3. Convertir nombres a mayúsculas")
val nombres = List("Juan", "María", "Pedro", "Ana", "Luis") // Crear la lista inmutable de nombres
val nombresMayusculas = nombres.map(_.toUpperCase) // Convertir cada nombre a mayúsculas
println(s"Nombres en mayúsculas: $nombresMayusculas")
```

```
1. Filtrar números mayores que 5
Números mayores que 5: List(6, 7, 8, 9, 10)

2. Calcular la suma de los números del 1 al 100
Suma total de los números del 1 al 100: 5050

3. Convertir nombres a mayúsculas
Nombres en mayúsculas: List(JUAN, MARÍA, PEDRO, ANA, LUIS)
numeros: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
mayoresQueCinco: List[Int] = List(6, 7, 8, 9, 10)
numerosHastaCien: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 5
3, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 8
2, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100)
sumaTotal: Int = 5050
nombres: List[String] = List(Juan, María, Pedro, Ana, Luis)
nombresMayusculas: List[String] = List(JUAN, MARÍA, PEDRO, ANA, LUIS)
```

Ejercicios: Conjuntos

4

```
// 1. Crea un conjunto inmutable de números del 1 al 5. Añade los números 6 y 7 al conjunto. Imprime el conjunto o
actualizado.
println("\n1. Conjunto original y actualizado")
val conjuntoOriginal = Set(1, 2, 3, 4, 5) // Conjunto inmutable original
val conjuntoActualizado = conjuntoOriginal + 6 + 7 // Crear un nuevo conjunto añadiendo 6 y 7
println(s"Conjunto original: $conjuntoOriginal")
println(s"Conjunto actualizado: $conjuntoActualizado")

// 2. Crea un conjunto de números del 1 al 10. Filtra los números impares y genera un nuevo conjunto con esos núme
el conjunto resultante.
println("\n2. Conjunto de números impares")
val conjuntoNumeros = Set(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) // Conjunto inmutable del 1 al 10
val conjuntoImpares = conjuntoNumeros.filter(_ % 2 != 0) // Filtrar números impares
println(s"Conjunto de números impares: $conjuntoImpares")

// 3. Crea dos conjuntos: uno con los días de la semana y otro con días de fin de semana. Encuentra la intersecció
conjuntos y muestra los días que son fines de semana.
println("\n3. Intersección: Días de fin de semana")
val diasSemana = Set("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo") // Conjunto de días
val diasFinDeSemana = Set("Sábado", "Domingo") // Conjunto de días de fin de semana
val interseccion = diasSemana.intersect(diasFinDeSemana) // Encontrar la intersección
println(s"Días de fin de semana: $interseccion")
```

```
1. Conjunto original y actualizado
Conjunto original: Set(5, 1, 2, 3, 4)
Conjunto actualizado: Set(5, 1, 6, 2, 7, 3, 4)

2. Conjunto de números impares
Conjunto de números impares: Set(5, 1, 9, 7, 3)

3. Intersección: Días de fin de semana
Días de fin de semana: Set(Sábado, Domingo)
conjuntoOriginal: scala.collection.immutable.Set[Int] = Set(5, 1, 2, 3, 4)
conjuntoActualizado: scala.collection.immutable.Set[Int] = Set(5, 1, 6, 2, 7, 3, 4)
conjuntoNumeros: scala.collection.immutable.Set[Int] = Set(5, 10, 1, 6, 9, 2, 7, 3, 8, 4)
conjuntoImpares: scala.collection.immutable.Set[Int] = Set(5, 1, 9, 7, 3)
diasSemana: scala.collection.immutable.Set[String] = Set(Miércoles, Sábado, Jueves, Viernes, Domingo, Martes, Lunes)
diasFinDeSemana: scala.collection.immutable.Set[String] = Set(Sábado, Domingo)
interseccion: scala.collection.immutable.Set[String] = Set(Sábado, Domingo)
```

Ejercicios: Mapas

```
6
```

```
1. Mapa de productos y precios: original y actualizado
Mapa original: Map(Manzana -> 0.5, Plátano -> 0.3, Naranja -> 0.4)
Mapa actualizado: Map(Manzana -> 0.5, Plátano -> 0.3, Naranja -> 0.4, Pera -> 0.6)

2. Filtrar productos con precio mayor a 0.4
Productos con precio mayor a 0.4: Map(Manzana -> 0.5)
mapaOriginal: scala.collection.immutable.Map[String,Double] = Map(Manzana -> 0.5, Plátano -> 0.3, Naranja -> 0.4)
mapaActualizado: scala.collection.immutable.Map[String,Double] = Map(Manzana -> 0.5, Plátano -> 0.3, Naranja -> 0.4,
Pera -> 0.6)
productosFiltrados: scala.collection.immutable.Map[String,Double] = Map(Manzana -> 0.5)
```