Basic_Collections

January 2, 2025

[1]:
```
// Creating an immutable list of animals
val animals = List("Dog", "Cat", "Rabbit")

// Accessing elements
println(animals.head)  // First element
println(animals.last)  // Last element

// Iterating over the list
animals.foreach(animal => println(s"Animal: $animal"))
```

```
Dog
Rabbit
Animal: Dog
Animal: Cat
Animal: Rabbit
```

```
animals = List(Dog, Cat, Rabbit)
```

[1]:
```
List(Dog, Cat, Rabbit)
```

[2]:
```
// Adding elements to an immutable list
val moreAnimals = "Bird" :: animals

// Printing the new list
println(s"Original List: ${animals.mkString(", ")}")
println(s"New List: ${moreAnimals.mkString(", ")}")
```

```
Original List: Dog, Cat, Rabbit
New List: Bird, Dog, Cat, Rabbit
```

```
moreAnimals = List(Bird, Dog, Cat, Rabbit)
```

[2]:
```
List(Bird, Dog, Cat, Rabbit)
```

[3]:
```
// Combining two immutable lists
val wildAnimals = List("Lion", "Tiger")
val combinedAnimals = animals ++ wildAnimals
```

```
// Printing the combined list
println(s"Combined List: ${combinedAnimals.mkString(", ")}")
```

```
Combined List: Dog, Cat, Rabbit, Lion, Tiger
```

```
wildAnimals = List(Lion, Tiger)
combinedAnimals = List(Dog, Cat, Rabbit, Lion, Tiger)
```

[3]:
```
List(Dog, Cat, Rabbit, Lion, Tiger)
```

[4]:
```
import scala.collection.mutable.ListBuffer

// Creating a mutable list
val colors = ListBuffer("Red", "Green", "Blue")

// Adding elements
colors += "Yellow"
colors += ("Purple", "Orange")

// Removing elements
colors -= "Green"

// Printing the modified list
println(s"Mutable List: ${colors.mkString(", ")}")
```

```
Mutable List: Red, Blue, Yellow, Purple, Orange
```

```
colors = ListBuffer(Red, Blue, Yellow, Purple, Orange)
```

[4]:
```
ListBuffer(Red, Blue, Yellow, Purple, Orange)
```

[5]:
```
// Immutable to mutable
val immutableList = List(10, 20, 30)
val mutableList = scala.collection.mutable.ListBuffer[Int]() ++= immutableList
mutableList += 40

// Mutable to immutable
val convertedImmutable = mutableList.toList

println(s"Mutable List: ${mutableList.mkString(", ")}")
println(s"Converted Immutable List: ${convertedImmutable.mkString(", ")}")
```

```
Mutable List: 10, 20, 30, 40
Converted Immutable List: 10, 20, 30, 40
```

```
immutableList = List(10, 20, 30)
mutableList = ListBuffer(10, 20, 30, 40)
convertedImmutable = List(10, 20, 30, 40)
```

[5]: List(10, 20, 30, 40)

[6]:
```scala
// Operations on lists
val numbers = List(5, 10, 15, 20)

// Filtering even numbers
val evens = numbers.filter(_ % 2 == 0)

// Doubling each number
val doubled = numbers.map(_ * 2)

// Finding the sum
val sum = numbers.sum

println(s"Original List: ${numbers.mkString(", ")}")
println(s"Even Numbers: ${evens.mkString(", ")}")
println(s"Doubled Numbers: ${doubled.mkString(", ")}")
println(s"Sum: $sum")
```

```
Original List: 5, 10, 15, 20
Even Numbers: 10, 20
Doubled Numbers: 10, 20, 30, 40
Sum: 50
```

```
numbers = List(5, 10, 15, 20)
evens = List(10, 20)
doubled = List(10, 20, 30, 40)
sum = 50
```

[6]: 50

[7]:
```scala
// Sorting and reversing
val fruits = List("Banana", "Apple", "Cherry")

val sortedFruits = fruits.sorted
val reversedFruits = fruits.reverse

println(s"Original List: ${fruits.mkString(", ")}")
println(s"Sorted List: ${sortedFruits.mkString(", ")}")
println(s"Reversed List: ${reversedFruits.mkString(", ")}")
```

```
Original List: Banana, Apple, Cherry
Sorted List: Apple, Banana, Cherry
Reversed List: Cherry, Apple, Banana
```

```
fruits = List(Banana, Apple, Cherry)
sortedFruits = List(Apple, Banana, Cherry)
```

```
reversedFruits = List(Cherry, Apple, Banana)
```

```
[7]: List(Cherry, Apple, Banana)
```

```
[ ]:
```