Higher Order Functions

Lesson Objectives

- After completing this lesson, you should be able to:
  - Describe the application of functions to data
  - Outline basic usages of higher order functions in Scala

Higher Order Functions

- A function which takes another function
- Typically describes the “how” for work to be done in a container
- The function passed to it describes the “what” that should be done to elements in the container

map

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.map(number => number + 1)
res1: scala.collection.immutable.IndexedSeq[Int] =
  Vector(2, 3, 4, 5, 6)

scala> res0.map(_ + 1)
res2: scala.collection.immutable.IndexedSeq[Int] =
  Vector(2, 3, 4, 5, 6)
```

flatMap

```
scala> List("Scala", "Python", "R")
res0: List[String] = List(Scala, Python, R)

scala> res0.map(lang => lang + "#")
res1: List[String] = List(Scala#, Python#, R#)

scala> res0.flatMap(lang => lang + "#")
res2: List[Char] =
  List(S, c, a, l, a, #, P, y, t, h, o, n, #, R, #)
```

Typesafe

filter

```
scala> List("Scala", "Python", "R", "SQL")
res0: List[String] = List(Scala, Python, R, SQL)

scala> res0.filter(lang => lang.contains("S"))
res1: List[String] = List(Scala, SQL)
```

foreach

```
scala> List(1, 2)
res0: List[Int] = List(1, 2)

scala> res0.map(println)
1
2
res1: List[Unit] = List((), ())

scala> res0.foreach(println)
1
2
```

Typesafe

forall

```
scala> List("Scala", "Simple", "Stellar")
res0: List[String] = List(Scala, Simple, Stellar)

scala> res0.forall(lang => lang.contains("S"))
res1: Boolean = true

scala> res0.forall(lang => lang.contains("a"))
res2: Boolean = false
```

Typesafe

reduce

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.reduce((acc, cur) => acc + cur)
res1: Int = 15

scala> res0.reduce(_ + _)
res2: Int = 15

scala> List[Int]().reduce((acc, cur) => acc + cur)
java.lang.UnsupportedOperationException: empty.reduceLeft
  ... 37 elided
```

fold, foldLeft, foldRight

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.foldLeft(0){ case (acc, cur) => acc + cur}
res1: Int = 15

scala> List[Int]().foldLeft(0){ case (acc, cur) => acc + cur}
res2: Int = 0
```

Typesafe

product

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.product
res1: Int = 120
```

Typesafe

exists

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.exists(num => num == 3)
res1: Boolean = true

scala> res0.exists(num => num == 6)
res2: Boolean = false
```

Typesafe

find

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.find(num => num == 3)
res1: Option[Int] = Some(3)

scala> res0.find(num => num == 6)
res2: Option[Int] = None
```

groupBy

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.groupBy(num => num % 2)
res1: Map[Int,scala.collection.immutable.IndexedSeq[Int]] =
  Map(1 -> Vector(1, 3, 5), 0 -> Vector(2, 4))
```

`takeWhile` and `dropWhile`

```
scala> 1 to 5
res0: scala.collection.immutable.Range.Inclusive =
  Range(1, 2, 3, 4, 5)

scala> res0.takeWhile(num => num < 3)
res1: scala.collection.immutable.Range = Range(1, 2)

scala> res0.dropWhile(num => num < 3)
res2: scala.collection.immutable.Range = Range(3, 4, 5)
```

Lesson Summary

- Having completing this lesson, you should be able to:
  - Describe the application of functions to data
  - Outline basic usages of higher order functions in Scala