Kotlin Lists

A list is a generic ordered collection of elements. A list can be mutable or immutable

An immutable list is created by

```
val classes = listOf("AnitaB", "Lisalab", "Lovelace")
```

A mutable list is created by

```
var colors = mutableListOf("red", "green", "blue", "yellow")
```

Inbuilt List Functions

```
var nums = listOf(31, 34, 4, 67, 213, 643, 28, 90, 55)

val len = nums.count()
val max = nums.max()
val min = nums.min()
val sum = nums.sum()
val avg = nums.average()
val firstItem = nums.first()
val lastItem = nums.last()
```

Kotlin List Indexing

Each element of a list has an index. The first list element's index is zero

```
var colors = mutableListOf("red", "green", "blue", "yellow")
println(colors[0])                  //red
println(colors.get(2))              //blue
println(colors.indexOf("green"))    //1
println(colors.lastIndex)           //3
```

List Iteration

```
var colors = mutableListOf("red", "green", "blue", "yellow")

for (color in colors){
    println(color)
}

colors.forEach { color-> println(color) }
```

List Sorting

```
val nums = listOf(11, 5, 3, 8, 1, 9, 6, 2)

val sortAsc = nums.sorted()
println(sortAsc)

val sortDesc = nums.sortedDescending()
println(sortDesc)

val revNums = nums.reversed()
println(revNums)


data class Car(var make: String, var model: String)

var cars = listOf(
    Car("Toyota", "Prado"),
    Car("Mazda", "Atenza"),
    Car("Subaru", "Legacy"),
    Car("Jeep", "Wrangler")
)

var sortedCars = cars.sortedBy { car -> car.make }
var descendingSortedCars = cars.sortedByDescending { car -> car.model }
```

List Contains

This function checks if a list contains a specified element.

```
val nums = listOf(11, 5, 3, 8, 1, 9, 6, 2)
println(nums.contains(5))
```

Mutable List

Mutable lists support adding and deleting elements as well as modifying existing elements.

```
val fruits = mutableListOf("Apple", "Banana")
println(fruits)             //[Apple, Banana]
fruits.add("Cherry")
fruits.add("Dragonfruit")
fruits.add("Mango")
fruits.add("Watermelon")
println(fruits)             //[Apple, Banana, Cherry, Dragonfruit,
Mango, Watermelon]

fruits.remove("Dragonfruit")
println(fruits)             //[Apple, Banana, Cherry, Mango, Watermelon]
```

List Filtering

We can filter lists to ensure that only items that meet certain criteria pass through

```
var names = listOf<String>("Toyota", "Pam", "Oncology", "Neurobiology",
"Mango")
var longNames = names.filter{name-> name.length>5}
println(longNames)          //[Toyota, Oncology, Neurobiology]
```

We can also filter lists of objects based on an object property

```
data class Person(var name: String, var age: Int)

var people =
    listOf(
        Person("Jane", 14),
        Person("Paul", 32),
        Person("Adrian", 16),
        Person("Muthoni", 25)
    )

var adults = people.filter { person -> person.age >= 18 }
println(adults)
//[Person(name=Paul, age=32), Person(name=Muthoni, age=25)]
```

References

https://zetcode.com/kotlin/lists/