Practical class # 11 – Collections

1) Open the Kotlin playground.

2) Add support for images in the Affirmation class

Create a list and print it as is and sorted.

```
fun main() {
        val numbers = listOf(0, 3, 8, 4, 0, 5, 5, 8, 9, 2)
        println("list:   ${numbers}")
        println("list:   ${numbers.sorted()}")
}
```

3) Convert the list to a set

```
val setOfNumbers = numbers.toSet()
println("set:    ${setOfNumbers}")
```

4) Define mutable and immutable sets

```
val set1 = setOf(1,2,3)
val set2 = mutableSetOf(3,2,1)
```

check that the two sets are identical

```
println("$set1 == $set2: ${set1 == set2}")
```

As for lists, you can use the contains function.

```
println("contains 7: ${setOfNumbers.contains(7)}")
```

5) Define a map

A map is a set of key-value pairs, designed to make it easy to look up a value given a particular key. Keys are unique, and each key maps to exactly one value, but the values can have duplicates. Values in a map can be strings, numbers, or objects—even another collection like a list or a set.

```
fun main() {
        val peopleAges = mutableMapOf<String, Int>(
        "Fred" to 30,
        "Ann" to 23
        )
        println(peopleAges)
}
```

Use the put function to add elements or use indexing with the key.

```
peopleAges.put("Barbara", 42)
peopleAges["Joe"] = 51
```

Try to add the following line and print the result.

```
peopleAges["Fred"] = 31
```

6) Try useful functions for collections

```
peopleAges.forEach { print("${it.key} is ${it.value}, ") }
```

The extra comma at the end can be deleted with

```
println(peopleAges.map { "${it.key} is ${it.value}" }.joinToString(", ") )
```

Filter a set

```
val filteredNames = peopleAges.filter { it.key.length < 4 } //generates a LinkedHashMap
println(filteredNames)
```

7) Lambdas functions

```
fun main() {
        val triple: (Int) -> Int = { a: Int -> a * 3 }
        println(triple(5))
}
```

Within the curly braces, you can omit explicitly declaring the parameter (a: Int), omit the function arrow (->), and just have the function body.

```
val triple: (Int) -> Int = { it * 3 }
```

8) Higher-order functions

This just means passing a function (in this case a lambda) to another function, or returning a function from another function.

```
fun main() {
        val peopleNames = listOf("Fred", "Ann", "Barbara", "Joe")
        println(peopleNames.sorted())
        println(peopleNames.sortedWith { str1: String, str2: String -> str1.length -
str2.length})

}
```

9) Create a word list

```
fun main() {
        val words = listOf("about", "acute", "awesome", "balloon", "best", "brief", "class",
"coffee", "creative")
}
```

filter the list to extract the words starting with b

```
val filteredWords = words.filter{ it.startsWith("b", ignoreCase = true) }
```

Randomize the order with the shuffle function.

```
val filteredWords = words.filter{ it.startsWith("b", ignoreCase = true) }.shuffled()
```

Take only 2 words.

```
val filteredWords = words.filter{ it.startsWith("b", ignoreCase = true) }.shuffled().take(2)
```