3. Compare conditions and booleans

In this task, you learn about booleans and checking conditions in the Kotlin programming language. Like other languages, Kotlin has booleans and boolean operators such as less than, equal to, greater than, and so on (`<`, `==`, `>`, `!=`, `<=`, `>=`).

1. Write an `if`/`else` statement.

```
val numberOfFish = 50
val numberOfPlants = 23
if (numberOfFish > numberOfPlants) {
   println("Good ratio!")
} else {
   println("Unhealthy ratio")
}
⇒ Good ratio!
```

2. Kotlin offers the ability to easily define a succession of values with starting and terminating endpoints. This is called a range. The easiest way to create a range in Kotlin is with the ".." operator. Try using a range in an `if` statement. In Kotlin, the condition you test can use ranges, too.

```
val fish = 50
if (fish in 1..100) {
   println(fish)
}
⇒ 50
```

3. Write an `if` with multiple cases. For more complicated conditions, use logical and `&&` and logical or `||`. As in other languages, you can have multiple cases by using `else if`.

```
if (numberOfFish == 0) {
   println("Empty tank")
} else if (numberOfFish < 40) {
   println("Got fish!")
} else {
   println("That's a lot of fish!")
}
⇒ That's a lot of fish!
```

4. Try out a `when` statement. A `when` statement can be a convenient way to write that series of `if`/`else` statements in Kotlin. The `when` statement is like the `switch` statement in other languages. Conditions in a `when` statement can use ranges, too.

```
when (numberOfFish) {
   0  -> println("Empty tank")
   in 1..39 -> println("Got fish!")
   else -> println("That's a lot of fish!")
}
⇒ That's a lot of fish!
```