

Universidad Nacional Politécnica
Departamento de Ciencia

Ingeniería en Computación
Programación móvil

Nombre: Victor Manuel Leiva

Docente: Ing. Luis

4- Codelabs: Functions vale 4 puntos

<https://developer.android.com/codelabs/kotlin-bootcamp-functions>

Answer these questions

Question 1

The `contains(element: String)` function returns true if the string element is contained in the string it's called on. What will be the output of the following code?

```
val decorations = listOf("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
```

```
println(decorations.filter {it.contains('p')})
```

[pagoda, plastic, plant]

[pagoda, plastic plant]

[pagoda, plastic plant, flowerpot]

[rock, alligator]

Question 2

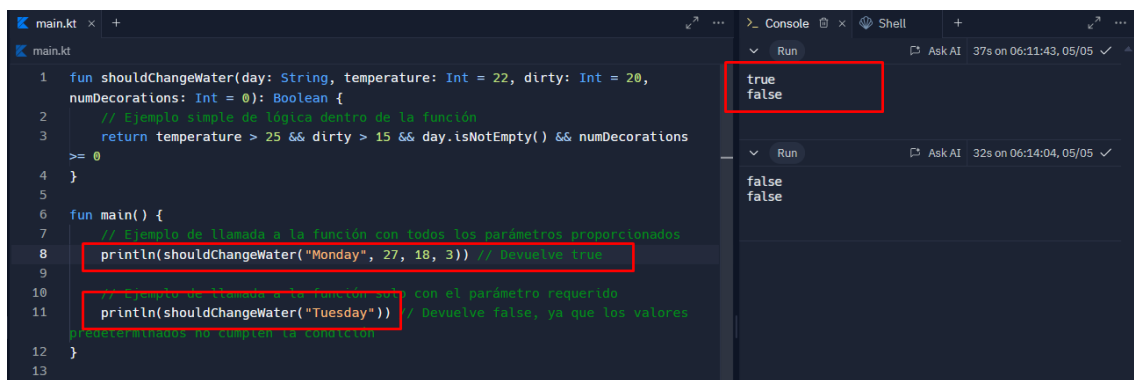
In the following function definition, which one of the parameters is required? `fun shouldChangeWater (day: String, temperature: Int = 22, dirty: Int = 20, numDecorations: Int = 0): Boolean {...}`

`numDecorations`

`dirty`

`day`

`temperature`

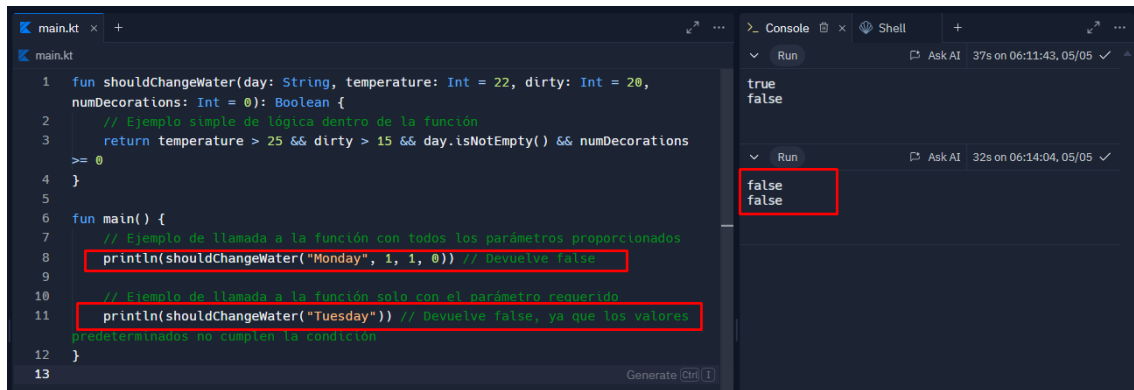


The screenshot shows an IDE with a Kotlin file named `main.kt`. The code defines a function `shouldChangeWater` with parameters `day: String`, `temperature: Int = 22`, `dirty: Int = 20`, and `numDecorations: Int = 0`. The function returns `true` if `temperature > 25` and `dirty > 15` and `day.isNotEmpty()` and `numDecorations >= 0`. In the `main` function, two calls to `shouldChangeWater` are shown. The first call is `println(shouldChangeWater("Monday", 27, 18, 3))` and the second is `println(shouldChangeWater("Tuesday"))`. The output console shows `true` for the first call and `false` for the second call.

```
1 fun shouldChangeWater(day: String, temperature: Int = 22, dirty: Int = 20,
2   numDecorations: Int = 0): Boolean {
3   // Ejemplo simple de lógica dentro de la función
4   return temperature > 25 && dirty > 15 && day.isNotEmpty() && numDecorations
5   >= 0
6 }
7
8 fun main() {
9   // Ejemplo de llamada a la función con todos los parámetros proporcionados
10  println(shouldChangeWater("Monday", 27, 18, 3)) // Devuelve true
11
12  // Ejemplo de llamada a la función con el parámetro requerido
13  println(shouldChangeWater("Tuesday")) // Devuelve false, ya que los valores
14  predeterminados no cumplen la condición
15 }
```

Output:

```
true
false
```



```

1 fun shouldChangeWater(day: String, temperature: Int = 22, dirty: Int = 20,
2   numDecorations: Int = 0): Boolean {
3   // Ejemplo simple de lógica dentro de la función
4   return temperature > 25 && dirty > 15 && day.isNotEmpty() && numDecorations
5   >= 0
6 }
7
8 fun main() {
9   // Ejemplo de llamada a la función con todos los parámetros proporcionados
10  println(shouldChangeWater("Monday", 1, 1, 0)) // Devuelve false
11
12  // Ejemplo de llamada a la función solo con el parámetro requerido
13  println(shouldChangeWater("Tuesday")) // Devuelve false, ya que los valores
14  predeterminados no cumplen la condición
15 }
  
```

Console output:

```

true
false
false
false
  
```

Question 3


You can pass a regular named function (not the result of calling it) to another function. How would you pass `increaseDirty(start: Int) = start + 1` to `updateDirty(dirty: Int, operation: (Int) -> Int)?`

`updateDirty(15, &increaseDirty())`

`updateDirty(15, increaseDirty())`

`updateDirty(15, ("increaseDirty()"))`

`updateDirty(15, ::increaseDirty)`



```

1 fun increaseDirty(start: Int): Int {
2   return start + 1
3 }
4
5 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
6   return operation(dirty)
7 }
8
9 fun main() {
10  val result = updateDirty(15, ::increaseDirty)
11  println(result) // Output: 16
12 }
13
  
```

Console output:

```

16
  
```