Exercises - Session 6

In case you get stuck anywhere, don't be afraid to ask the coaches! They are here to help and will gladly explain everything to you!
Take notes during the exercises. Even if you never look at them again, they will help you memorise things!

Recap previous sessions

1. Write a method called "get_user_number_input". When you call the method, it should ask the user for a number with a nice text. The method should then return this number as a number, not a string. Use your method in the example code:

   ```
   def get_user_number_input
     # your code goes here…
   end

   number = get_user_number_input()
   puts "The user entered the number #{number} and it's a #{number.class}"
   ```

   When you execute it, you should get:

   ```
   Hi, please enter a number: 42
   The user entered the number 42 and it's a Fixnum
   ```

Boolean Operators

1. The following exercise might be a bit dry, but it's quite important in the daily life of a programmer.

   Image the "condition" is part of your code, like this:

   ```
   if condition
     puts "A"
   else
     puts "B"
   end
   ```

   Also, we have the following variables:

   ```
   list  = [2, 3, 4]
   title = "Ruby Monstas"
   ```

Analyze the following conditions and note their return type like in the first example. Also write what the code would execute in the if statement above.
Fill this out without trying it on your computer. After that, check your answers with irb.

condition	**result (return value)**	**puts ...**
`1 < 2`	true	A
`title.include?(“Ruby”)`		
`list.length == 3`		
`"test" == 1`		
`true || false`		
`true && false`		
`1 < 2 || 1 > 2`		
`list.length > 3 && title.length == 12`		
`!(list.length == 3)`		
`!(list[1] == 3 || 10 != 12)`		
`1 == 1 && (!("testing" == 1 || 1 == 0))`		
`3 != 4 && !("A" != "a" || "Ruby" == "Ruby")`		

2. Let’s turn it around. Let’s say we have two variables called “one” and “two”. This is our desired truth table:

one	**two**	**condition result**
false	false	false
true	false	true
false	true	true
true	true	false

a) How would you describe this condition? When is it true and when is it false?
b) What does your condition look like that the result is correct?

3. Consider this boolean condition:

```
false && x
```

Can you tell what the result is without knowing the value of x? Why?

<u>Optional Part</u>

1. Write nice methods for the boolean operators ||, && and ! so you can use them like this:

   ```
   not( and( or( true, false ), or( false, true ) ) )
   ```

   Here's a template for these methods:

   ```
   def not(argument)
     # your code goes here…
   end

   def and(a, b)
     # your code goes here…
   end

   def or(a, b)
     # your code goes here…
   end
   ```