Exercises - Session 4/5

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 program that asks for the user's name and prints it out like this: Hello Michelle!

2. We're going to collect mushrooms! Enter the following into IRB:
   ```
   mushroom_count = { "Fabian" => 5, "Thomas" => 3, "Marion" => 7 }
   ```
   Now print out all the participants who collected mushrooms. Find that method in the Ruby Documentation of Hash: http://ruby-doc.org/core-2.2.0/Hash.html

Control structures

We're building a Todo script in the command line! Before we can start, we should prepare ourselves. First, create a folder for your new script. It could be called "todoscript". In that folder, create a Ruby file called "todo.rb". We will program in that file for the rest of the exercise. We'll do it step by step, so that we have a working program after each step.

1. Say hi to the user and ask for a command to enter. If the user enters "quit", you should print "Goodbye!". Otherwise, the command is ignored and your program ends as usual.

2. Now, allow the user to stay in the script. After the user entered her command, she should be able to type others commands. If she writes "quit", the program quits, otherwise she can enter commands indefinitely.

3. Now we want to be able to list todos. First, we prepare some todo items for the user. She should be able to list all todos with the command "list". Our script now supports two commands: list and quit. It will still ask for new commands if the command was not "quit".

4. The command we implement next is "add". After the user typed add, we will ask for a short text that is the new todo item. After the text was entered, we add this to the list of todos. After that, we can enter a new command.

5. The last command we implement is "done". If the user is done with an item, she can type "done" and then enter one of the todo texts. This todo item should then be removed from the todo list. After that, a new command can be entered as usual.

Optional Part

Let's build a number guessing game. Here's how it works:
When our program starts, we think of a random number like this:

secret_number = rand(100)

With "rand(100)" Ruby picks a random number between 0 and 100 for us.
The user can then enter a number and we will tell her whether it's smaller or greater than the secret number we stored in the beginning.
When the user is wrong, she can guess again as long as she wants. If she guesses the number, the game is over and we congratulate her.

Tip 1: While you are developing the program, print out the secret number. This way it's much easier to see what's going on. If you're done, you can just remove that line and your game is ready.

Tip 2: You're probably more successful if you do this exercise step by step. Figure out the smallest steps you can take to advance your program and run it regularly to see if your code works.

Practise: While

- Define a counter and set it to 10
- As long as the number is smaller than 15, multiply the number by 2
- Then increase the counter

-> Read about while loops in the Ruby documentation:
http://ruby-doc.org/docs/Tutorial/part_02/while.html
http://www.tutorialspoint.com/ruby/ruby_loops.htm

Practise: Until

- Research until-loops in the Ruby documentation:
  http://www.tutorialspoint.com/ruby/ruby_loops.htm
  http://ruby-doc.org/core-2.2.0/doc/syntax/control_expressions_rdoc.html
- Define your own while loop
- The until-loop has many similarities with another control structure you know. Which one is it?
- Research the similarities and the differences