Exercises - Session 3

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 enables the user to calculate his/her BMI. The user will be asked for the body mass in kg and the body height in m and the program calculates the BMI from that.

2. Write a program that lets the user input 3 strings. Save those strings in an array. Then print the first letter of each of the strings.

3. Find out how to do each of these things with an array (by googling or looking it up in the Ruby documentation):
    a. sort the array
    b. randomise the array
    c. delete duplicate entries from the array

    Try each of these in IRB.

String interpolation

1. Improve your temperature conversion programs from last session by using string interpolation.

2. The escape sequence `\r` is a carriage return. Run `puts "A\rB"` In IRB and find out what it does!

    Can you imagine where it could be useful? Maybe your coach knows?

Hashes

First, describe the characteristics for a Hash. If needed, research definitions and examples in the Ruby documentation:
http://ruby-doc.org/core-2.2.0/Hash.html

Create two hashes, using the two different ways to do so.

We are going to make a dictionary for whatever two languages you know. The key is always a word or expression in the source language, whereas the value is a word or expression in the target language.

1. Create a new file in your directory
2. Create an empty hash

3. Add at least seven key and value pairs
   -> e.g. The German translation for “Ruby” is “Rubin”
4. Access the value of a certain key
5. Check if a certain value is in your hash. Try with one value that is in the hash and with another that is not
6. Delete one key-value pair from your hash
7. Research more hash methods and try them on your hash

Optional Part

1. Improve your BMI program by informing the user about what BMI range means what. Store the table in a Ruby hash (one column being the keys and one being the values) and display each entry to the user:

Category	BMI range – kg/m^2
Very severely underweight	less than 15
Severely underweight	from 15.0 to 16.0
Underweight	from 16.0 to 18.5
Normal (healthy weight)	from 18.5 to 25
Overweight	from 25 to 30
Obese Class I (Moderately obese)	from 30 to 35
Obese Class II (Severely obese)	from 35 to 40
Obese Class III (Very severely obese)	over 40

   Source: https://en.wikipedia.org/wiki/Body_mass_index

2. The `sleep` command makes your program wait for a certain amount of time before carrying on.

   Lay down some sick beats by using `puts "\a"` and the `sleep` command to get the timing right.

3. Find out how to do each of these things with an array (by googling or looking it up in the Ruby documentation):
   d. delete all elements in the array
   e. turn around the order of the elements in the array

If you still have time, create another hash for locations with their respective weather conditions i.e. their temperature.

As you know, weather conditions are not permanent, so it can well be that you have to change the temperature of a certain location. Try to find out how to change the values of a hash.