Exercises - Session 27

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!

1. Code Puzzle (Repetition Ruby Bascis)

In this puzzle, you get a bit of code as a precondition. So with this code executed, you have to program a single line (think irb) that yields the return value. Sometimes the code is already given and you have to provide the expected return value.

Hint: Don't just type code into your irb, think about what you expect or what needs to be done to yield the return value. You can obviously read documentation, but try to find the solution first yourself.

Precondition	**Code**	**Return value**
`a = [1, 2, 3]`	`a.class`	
`a = [“one”, “two”, “three”]`		`“one, two, three”`
`a = [“one”, “two”, “three”]`		`[“ONE”, “TWO”, “THREE”]`
`a = [1, 2, 3, 5, 8, 13, 20]`		`[2, 8, 20]`
`a = 45`	`a.class`	
`a = -23`	`a.abs`	
`a = 3`	`a.succ`	
	`a.zero?`	`true`
`a = “Hello world!”`	`a.class`	
`a = “Hello world!”`	`a.slice(6, 5)`	
`a = “Hello world!”`		
`a = “Hello world!”`		`“hello world!”`
`a = “star”`		`“rats”`
`a = “Every single word”`		`[“Every”, “single”, “word”]`
`a = “\tweird text \n\t”`		`“weird text”`
`“42”`		`42.0`
`a = [1, 2, 3, 5, 8, 13, 20]`		`7`

a = [“woman”, “router”, “text”]		[5, 6, 4]
a = [“woman”, “router”, “text”]		[“woman”, “text”]

a = [“cool”, “nice”]
b = [“party”, “weather”]
		[[“cool”, “party”], [“nice”, “weather”]]

a = [“woman”, “router”, “text”]	a.all? do |element|
	  element.length > 3
	end	

a = [[“cool”, “party”], [“nice”, “weather”]]		[“cool”, “party”, “nice”, “weather”]
a = [“cool”, “party”, “nice”, “weather”]		“weather”
a = [“cool”, “party”, “cool”, “weather”]		a = [“cool”, “party”, “weather”]
a = {name: “Helga”, age: 42}		42
a = {name: “Helga”, age: 42}		[:name, :age]
a = {name: “Helga”, age: 42}		[“Helga”, 42]

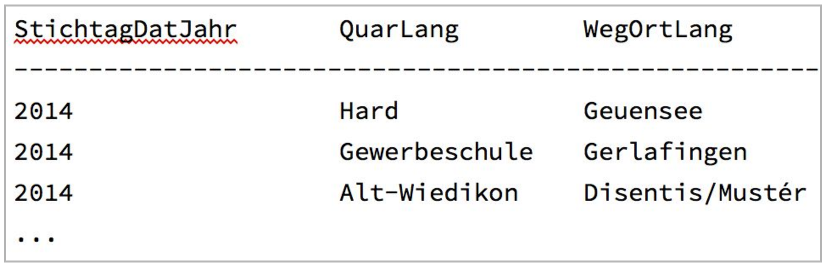
2. CSV. Again.

By this point you can probably put “CSV” as a skill in your CV. We don’t want to bore you, but CSVs and similar exchange formats are very popular and a lot of programs in use today can export to CSV. Moreover, it’s nice to work with them in a Ruby program and we’re sure you’ll use them somewhere in the future when you’ll have to solve a problem.
In addition, a lot of open data data-sets are published as CSV. Take for example the City of Zurich: https://data.stadt-zuerich.ch/
They publish quite a few data sets on their open data platform. One interesting example is the “Wegzüge nach Jahr und Quartier” data set. It records people leaving Zurich per city quarter. It also states whether they moved to another Canton, another country or even another continent: https://data.stadt-zuerich.ch/dataset/bev-wegz-jahr-quartier-v2/resource/a03afa87-ddf9-4c3a-8e9c-515d7bcb2918. This is actually the full data set since 1983. We prepared a sub-set from 2014 on the curriculum page for you to download.

a. Try to make sense of the data. What columns do they offer? What does each column mean? Is the data in any way redundant? Do you like the naming of the columns? If not, how would you rename them?

Hint: Use the CSV standard library (http://ruby-doc.org/stdlib-2.2.0/libdoc/csv/rdoc/CSV.html) to solve the following tasks.
Friendly reminder: always work in steps, don’t try to solve everything at once, try your ideas in irb.
Especially make use of `CSV.table()`. Read up what it offers you and try it in the irb, maybe with a smaller CSV file you used for an earlier exercise.

b. Now let’s go for our first task. Select a few columns you find useful and print out the data only with these columns.
What function on `Array` can you use to only print the first 10 elements? It could look something like this:



```
StichtagDatJahr      QuarLang        WegOrtLang
---------------------------------------------------------
2014                 Hard            Geuensee
2014                 Gewerbeschule   Gerlafingen
2014                 Alt-Wiedikon    Disentis/Mustér
...
```

c. Now pick out a column you want to filter for. Let’s say you chose “StichtagDatJahr”. Now when you start your script, ask the user to enter a specific value for this column. Then print all the values as above, but this time only if the value matches the value that the user has given you. If the value is for example “2015”, then only print rows that have the value “2015” for “StichtagDatJahr”.

d. Now pick 2 more columns and also ask for them in the beginning. It gets interesting, now you can filter for 3 different columns. But what if the user does not want to filter for this column? If they leave the value empty, we should not filter for it!

And that’s it. You’ve built a open data browser! :-)

Optional

1. Let the user analyze a certain column. If the user chooses for example “`QuarLang`”, print out a list with the top 10 values and how often this value was found.
Think about how you want to analyze this and how you want to store the information first.

2. Offer to store the filtered rows to a separate CSV file. What features from the CSV standard library can you use to generate CSV files?