Exercises - Session 19

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!

The Todolist App - Part 1

In this exercise, we'll start to implement a Todo list application. Each todo has these two properties:

- A title
- The fact that the todo is completed or not completed

1. Let's start by thinking about the data structures we are going to use for our program. What data types should we use for each of the following things? Think of "simple" data types like numbers and strings but also about the more complex ones like arrays, hashes and objects.

    - The title of the todo
    - The property of the todo to be completed or not
    - The whole todo
    - The collection of all of your todos

    After you have thought about this for a while, talk to a coach to verify your findings. You will need the solution of this exercise for the rest of them!

2. We are going to use object orientation for our todo application. Each of the todos is going to be a distinct object, but they will all have a common class called `Todo`. It's your task to implement that class now!
    Create a new Ruby file called `todo.rb`. For now, just define an empty class called `Todo` (without any methods, instance variables or initializers).
    We need a way to try out the class while writing it. As it turns out, `irb` is a great tool for that. In your command line, change to the directory containing your `todo.rb` file. Then type `irb`. Now your code is in that file, but not loaded into `irb` yet. To load the `todo.rb` file in `irb`, just type this: `load 'todo.rb'`. This is what it should look like in your command line:

```
irb(main):001:0> load 'todo.rb'
=> true
irb(main):002:0>
```

    The `true` means that your file was loaded correctly. Now you can play around with your Todo class. Try creating an instance of it and saving that to a variable called `todo`.

3. Next, let's make that class do something useful! We'll start with creating an initializer for the class. It should take one argument, the `title` for the todo. And it should initialize both the `title` and the

`completed` status in the new Todo object it creates. Initially, a Todo should of course not be completed. Below you see what that should look like when you try it in `irb`.
Note: Every time you change something in your `todo.rb` file, you have to re-load the file in irb by running `load 'todo.rb'` again. However, you can go back in the history of previously typed commands by using the up arrows on your keyboard!

```
irb(main):001:0> load 'todo.rb'
=> true
irb(main):002:0> Todo.new('Mow lawn')
=> #<Todo:0x007f96eb0474a8 @title="Mow lawn", @completed=false>
```

4. That `#<Todo:0x007f96eb0474a8 @title="Mow lawn", @completed=false>` isn't very legible, is it? We'd like to be able to show a todo to the user in a nice way. In order to do that, we can implement the `to_s` method (read "to string") in our `Todo` class. Here's what the output should look like:

```
irb(main):014:0> Todo.new('Mow lawn').to_s
=> "not completed: Mow lawn"
```

5. Of course, our todo doesn't make much sense without the ability to complete it. So we'll do that next. Here's what you should be able to do in `irb`:

```
irb(main):016:0> todo = Todo.new('Mow lawn')
=> #<Todo:0x007f96eb0ce9f8 @title="Mow lawn", @completed=false>
irb(main):017:0> todo.complete!
=> true
irb(main):018:0> todo.to_s
=> "completed: Mow lawn"
```

Optional part

1. Did you know you can use pretty much any character in your Ruby programs? Try to modify your `Todo` class so it does this (or something similar):

```
irb(main):026:0> todo = Todo.new('Mow lawn')
=> #<Todo:0x007f96ec0ef328 @title="Mow lawn", @completed=false>
irb(main):027:0> todo.to_s
=> "❌   Mow lawn"
irb(main):028:0> todo.complete!
=> true
irb(main):029:0> todo.to_s
=> "✅   Mow lawn"
```