Programming for Everybody

9.Object-Oriented Programming, Part 1

Primitive Objects

Until now we've been assigning and manipulating different types of **primitive objects**

String	`"Hello, welcome to Le Wagon!"`
Integer	`4`
Float	`3.14`
TrueClass	`true`
FalseClass	`false`
NilClass	`nil`
Array	`["Hello!", 4, 3.14, true, false, nil, [], {}, :symbol]`
Hash	`{ first_key: "a string value", number: 4, array: ["👾"] }`
Symbol	`:gabriele`
Proc	`(param)-> { puts "The proc param is: #{}" }`

Attributes and Behaviours

Ruby is *object-oriented*, and uses objects to store **attributes** and execute **methods**

```ruby
"Hello!".length  # => the method :length (symbols for method names!) returns the attribute 6 of type Integer
4.even?  # => the method :even? returns the attribute 6 of type TrueClass
teachers = [  # => the assignment returns an attribute teachers of type Array, containing two Hashes
  {
    name: "Mariana",
    age: 37,
    country: “🇵🇹”
  },
  {
    name: "Gabriele",
    age: 27,
    country: "🇮🇹"
  },
]
get_flag = ->(teacher) { teacher[:country] } # => fastest and cleanest way to assign a Proc (omit the return!)
flags = teachers.map(&get_flag)  # pass it to the method, and store an Array called flags, which will contain 2 Strings
p flags  # returns an instance of Array (👉 see why) and sends the object to the console
```

Let’s see some more live examples!

In the next section we will create our own `Class`es

See you soon! 🙂 👋