

Programming for Everybody

5. Methods, Blocks, & Sorting

Methods

Methods (a.k.a functions) are reusable blocks of code written to perform a (repeatable) specific task.

As a mathematical function, methods can take one or multiple *arguments* (the **input**).

The role of a method is then to compute some calculations (using the eventual inputs) and give back 1 result (the **output**)

Why methods?

1

They are reusable and dynamic (the output depends on the input!)

2

They help keep your code organised by separating the different parts of your app:
a specific method executes a specific task

3

For “top-down” programming: as your code becomes more complex, methods can help you solve a big problem (the “top”) by breaking it down into little problems

Syntax

Method name

Argument(s)

```
def print_reversed_name(name)
  puts name.reverse
end
```

Method body

When I write a method (with the def-end keywords), there is no code getting executed, because I'm just declaring (assigning) it.

To execute it, I have to call it (and specify any eventual dynamic argument, otherwise I will get an error!)

```
print_reversed_name("gabriele")
```

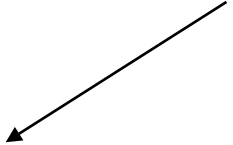
Spat

What if I don't know the exact numbers of arguments to expect?

Splat arguments!

With a star before!

```
def print_reversed_names(*names)
  names.each do |name|
    name.reverse
  end
end
```



I can now pass one or more arguments, and they will all go inside an array called names

The **return** keyword

Returning a value is different from printing it

When I return, the value I get is available in my code, and can be assigned to a variable or reused

When I print, I just show the value in the console (terminal), but I don't have it available in my code

```
def reverse_name(name)
    return name.reverse
end
```



Is not printing, but just giving me back the value in my code!

Blocks

Blocks (a.k.a anonymous/nameless functions) are block of codes executed inside a pre-defined method (like each, sort, etc.)

They are like methods, but they don't need a name or to be assigned, because they are already inside another (pre-defined) method

```
names = ["gabriele", "mariana", "shannon"]
```

```
names.each do |name|  
  reversed_name = name.reverse  
  puts reversed_name  
end
```



Nameless block go code

Thank you! :)