### Ruby is

- Object oriented interpreted scripting language
  - Object Oriented?
    - · Objects & State
  - Interpreted

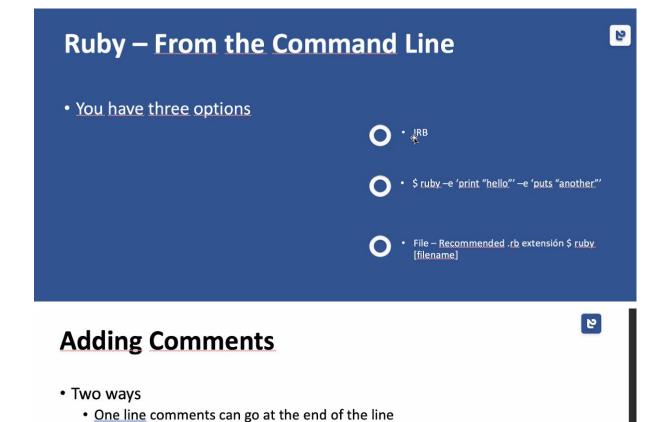
# This is a comment line

# a welcome message

displays

# it explains that the next line of code

- Compiled Read the whole file and then pass it to the processor
- Interpreted Executed line by line
- Scripting languages are a <u>high level</u> dynamic programming language



It stills not a good practice to use commentaries, your code should be self explanatory by names...

This is a comment line

a welcome message

it explains that the next line of code displays

#### **Constants**

Go on uppercase. They can change but it is not good practice.

- CONSTANTS = "go in uppercase"
  - Unlike other programming languages, they can change but it's not a good practice
- Dynamic Typing They can change types at run-time



# **VARIABLES** – Declaring them

Two ways

$$a, b, c, d = 10, 20, 30, 40$$

It is a matter of preference which one to use. But the first one is more readable.

# **VARIABLES – Knowing the current type**

Two ways

y.class
=> Fixnum

is\_a? is like .kind\_of, but is\_a also do another thing?

### 5

## **VARIABLES** – Casting

- To float .to\_f
- To string .to\_s
- To binary .to\_s(2) [please note this return what looks like a string but represents a binary]

```
irb(main):001:0> 1000.to_s(2)
=> "1111101000"
irb(main):002:0>
```

To\_s can convert it to binary?

Name Begins With	Variable Scope
\$	A global variable
@	An instance variable
[a-z] or _	A local variable
[A-Z]	A constant
@ @	A class variable

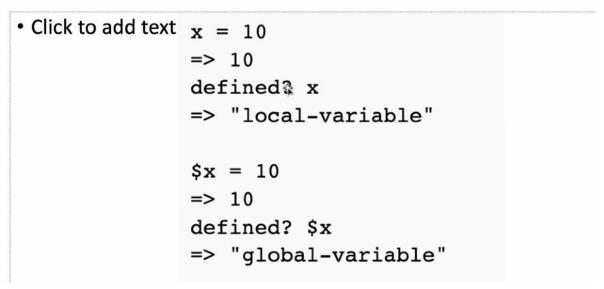
\$global variable -> most useful when we are using something stored in our session.

Communicate something from the controller to the module (not good practice but necessary).

local variable -> if put inside if, only available inside if.

constants -> declare normally at the beginning of the class.

@@ class variable -> static variable, also if inherited, the child will have the same value.



.defined? -> tell you what type of variable it is.

#### Why do some methods have "?"

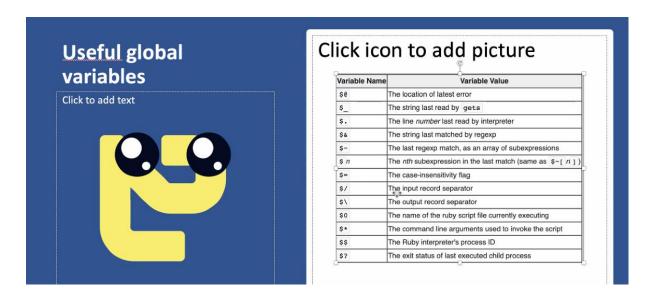
usually the ones that end with ?, returns a true or false, or the type, etc.

#### The "!"

some methods do not modify the variable itself, only return another value. the ones that end with "!" wil always change the variable

#### Clean irb console

control + L



# **Types of Objects - Numeric**

• Base class – From this class the rest are derived

Integer (0b01110101)
=> 117

All classes in ruby, specially the numeric ones, inherited from that one.

#### Fixnum is deprecated

We only use integers now.

It is computer architecture dependent.

Only maintainer because of code maintainament.

## **Types of Objects - Fixnum**

- 5
- Value of the <u>fixnum</u> depends on the architecture of the system where it is executed
- If it exceeds the range defined in the system, the value is interpreted into <u>Bignum</u>
  - Both classes were deprecated on Ruby 2.4.0
  - · Both classes are now covered by the Integer class

### Global variables

You can see how they work, defining one in a method, executing it, and calling it on another method.

Also try other ideas.

Also creating in the controller and then using it on the module.