

# Programming basics

---

Desarrollo de Aplicaciones Web

1er Curso

Curso 2020-2021

Autor: Vicent Bosch

[vicent.bosch@ceedcv.es](mailto:vicent.bosch@ceedcv.es)



**Reconocimiento - NoComercial - Compartirlgual (by-nc-sa):** No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Esta obra esta sujeta a la Licencia Reconocimiento-NoComercial-Compartirlgual 4.0 Internacional de Creative Commons. Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by-nc-sa/4.0/> o envíe una carta Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

# Recordatorio

---



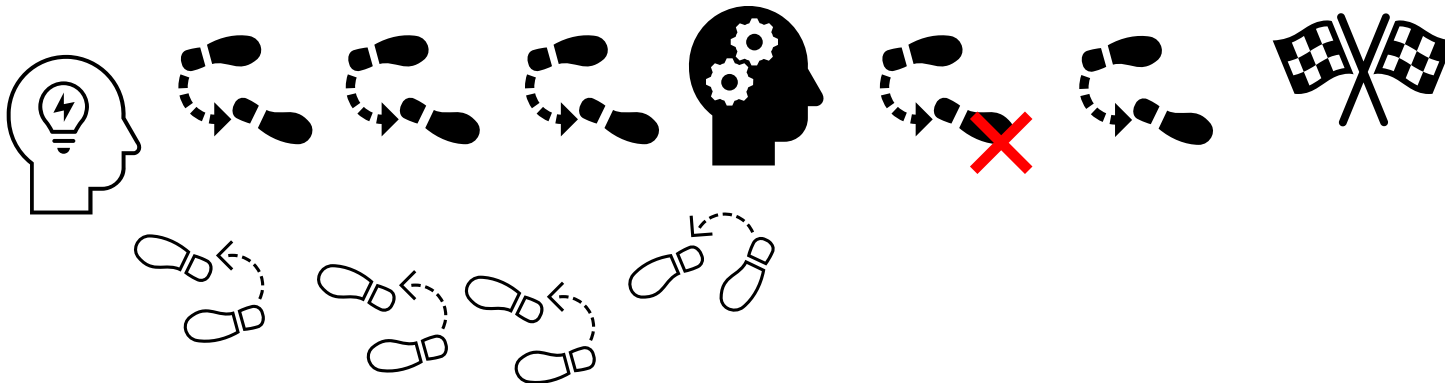
**Esta  
presentación no  
sustituye los  
apuntes  
disponibles en  
el aula virtual.**



**Las apuntes  
oficiales son los  
que tenéis en el  
aula virtual**

# Algorithm

- ✓ We use algorithms everyday
- ✓ We need algorithms to solve problems
- ✓ An algorithm is a sequence of **steps** we have to follow to solve a problem, get a result, etc.
- ✓ Sometimes we have to **decide** the correct path.
- ✓ Sometimes we have to go **backwards** and start again.
- ✓ But we must **test** very well each step to avoid errors.



# Algorithm: example.

## Preparing a cup of coffee

- ✓ First of all we must think about the *ingredients* we will need.

*Maybe not all of them are mandatory.*

*We call  
these:*

***variables***



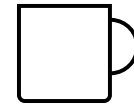
Water\*



Coffee\*



Sugar (not salt)



The cup\*

- ✓ Now we must write down the steps in the correct order.
- ✓ A proposed solution could be:

1. Put water and coffee in the coffee pot
2. Put the coffee pot on the stove
3. Turn on the stove
4. Wait until we get the coffee.
5. Decide whether you want sugar or not.



# Algorithm: think one ...

---



✓ First of all you must think about the *variables* you will need.

✓ Now write down the steps in the correct order.

**... and share in the forum.**

# Algorithm: example.

---

- ✓ Sometimes your algorithm has to check the value of a variable to decide what to do. (evaluates the variable).
- ✓ For example, imagine a traffic light.
  - If the color is green, you can move.
  - If the color is red, you must stand still.
- ✓ Therefore, your algorithm has to evaluate the content of a variable that we could call *color\_light* before continue.
  - If `color_light="green"` then move 
  - If `color_light="red"` then stand still 

# Algorithm: variables (1)

- ✓ Variables are used to store values.
- ✓ One variable stores a value, but it can change while executing the algorithm.
- ✓ You can perform different operations on the variables (i.e. their values).
- ✓ For example, if you need an algorithm to calculate the average of three numbers. How many variables will you need?
  - 1 to store the three numbers and the average? Not recommended for beginners.
  - 3 variables (three numbers), and 1 variable (the average). Recommended solution for beginners.

# Algorithm: variables (2)

## Calculate the average of three numbers:

- ✓ Store each value in its associated variable.

(the values can be introduced through the keyword)

```
number1=value1
```

```
number2=value2
```

```
number3=value3
```

- ✓ Sum the value of the three variables and divide the result into three and Store the result in the *average* variable.

```
average=(number1+number2+number3)/3
```

- ✓ Show the result.

```
print average (print is a function)
```



# Algorithm: conditionals(1)

- ✓ When your algorithm has to evaluate the content of a variable and depending on this evaluation the algorithm has different *paths*, your algorithm needs a *conditional*.
- ✓ For example, an algorithm has to answer the question “Did I pass the exam?”
- ✓ The algorithm needs a value to respond to that question, in this case the final note, and in an advanced version, the algorithm would also need the minimum note to pass the exam.

Print “enter your note”

Store the note

Print “enter the minimum note to pass”

Store the minimum note

Compare both values using conditionals:

if (note is less than minimum note) then “you did not pass”

if (note is greater or equal than minimum note) then “you passed”

# Algorithm: conditionals(2)

- ✓ Another versions of the conditional could be:

if (note is less than minimum note) then “you did not pass”  
else “you passed”

if (note is greater or equal than minimum note) then “you passed”  
else “you did not pass”

- ✓ You can use *else* when there are no more possible *paths*.