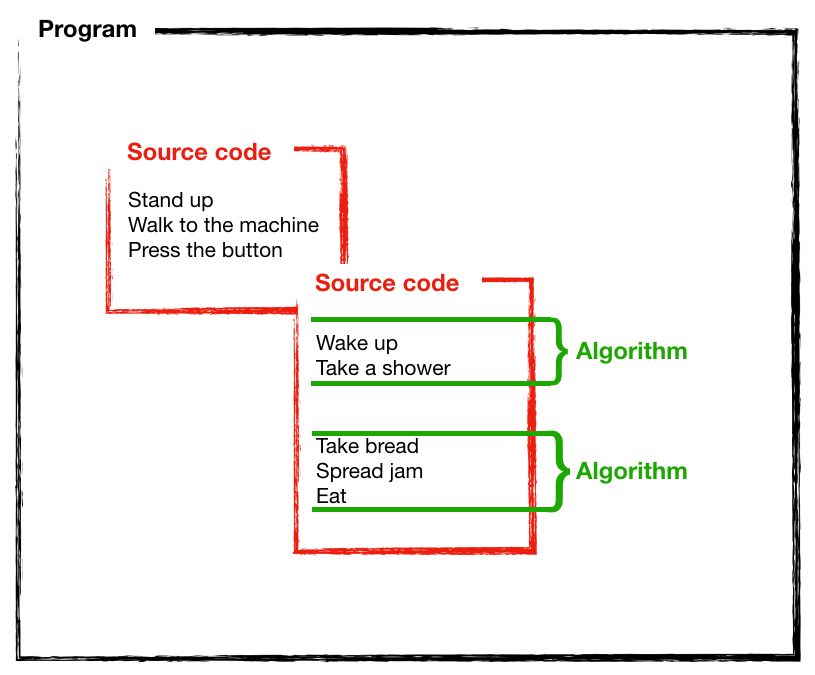
Programming Extension School

**02. What Is Programming**

Wait! That's a whole lot of new material. Let's recap, shall we?

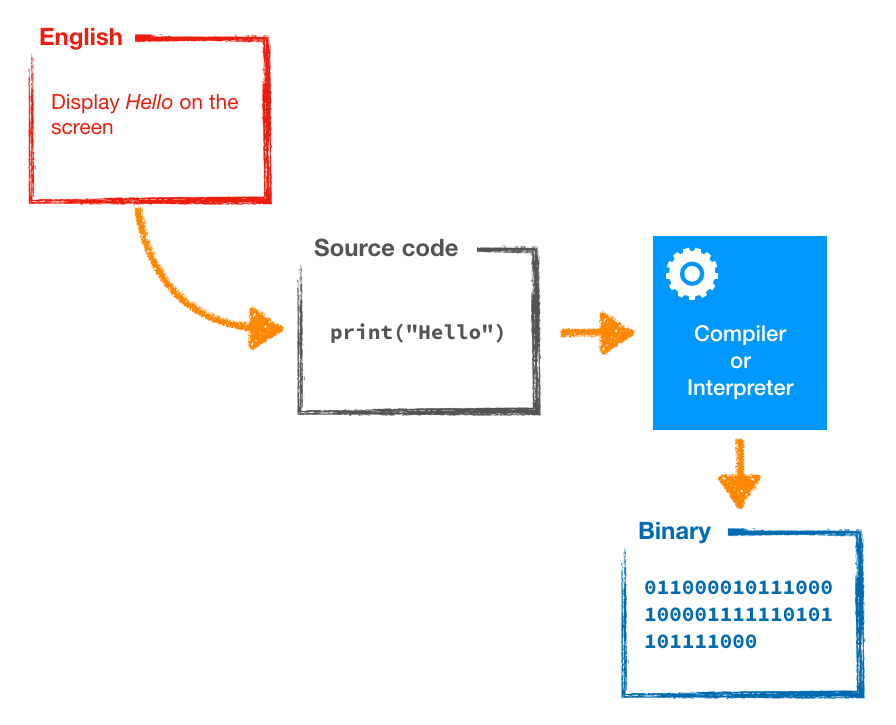


* **Source code:** The sheet of paper (or most likely the computer file) containing the written instructions.
* **Algorithm:** The grouping of the instructions that one must take to achieve something.
* **Program:** The result of one or more source code files containing one or more algorithms.

Even though we can use different programming languages, your computer actually understands only one: **binary code**.

## The need for speed (productivity!)

Writing programs using binary code as the programming language would be a tedious task (even though that's exactly what the early programmers did using punched cards: a hole for a 1 and no hole for a 0!). This is precisely why higher-level languages were created. BASIC, C, Java, Ruby, Python, PHP, JavaScript and many more. By using a programming language that resembles human languages (usually English), programmers can be way more productive. Whichever programming language we choose, though, its source code will ultimately be translated into binary code by a **compiler** or an **interpreter**.



# 03. Compiled Or Interpreted

## Compiled

Once a program is written using a compiled language, the source code needs to pass through a **compiler** that will translate it into binary code and output what is known as the executable

**Running a program**

Programs written with them are quicker to execute. The downside is that they're not easily portable. A program compiled for Microsoft Windows will run on Microsoft Windows only. A program compiled for Mac will only run on a Mac.

Examples of compiled languages include *C, C++, Objective-C, COBOL and Swift*.

## Interpreted

A program written using an interpreted language also gets translated to binary code, but it is done by an **interpreter** each time the program is run

One compelling reason to use an interpreted language is that in most cases, you'll find an interpreter for its source code targeting multiple platforms. For example, the same Python code can be run on Microsoft Windows, Mac, Linux

the speed of execution is slower (sometimes, much slower) than programs written using a compiled language because they need to be translated each time from source code to binary code by the interpreter before the computer can run them.

Examples of interpreted languages include *Python, Ruby, JavaScript*.

Now you know what **programming is: writing instructions in source code files as collections of algorithms, using a programming language.** Try saying that three times fast! **The source code will ultimately result in an interpreted or compiled computer program.**

What else do you now know? **That computers can only understand binary code** and that programming languages were created as an *intermediary between human languages and binary code*.

# 07. Walkthrough: Your first algorithm

Programming is write Instructions to breakdown Task ! One instruction is a Statement

# 08. The two languages of TCC: Python and JavaScript

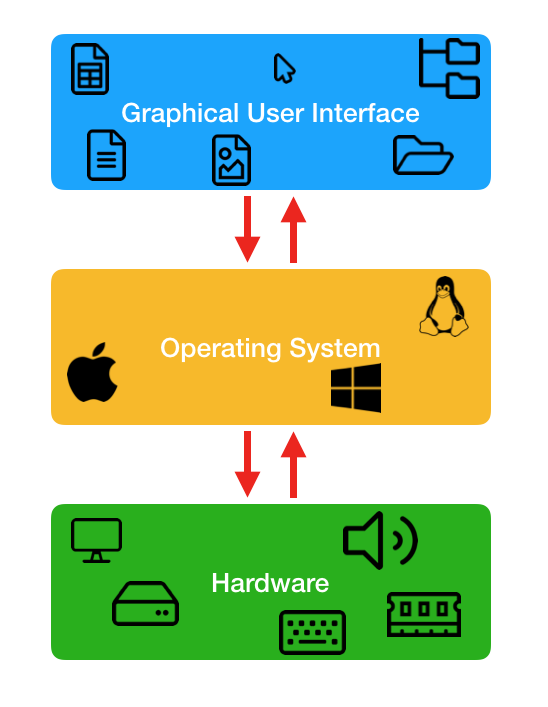
Python is a very popular general-purpose programming language that is used by companies big and small, from NASA to Instagram. JavaScript is **the** language of the web and any kind of web application uses it in one way or another

# 02. Common tools for programming

the command-line interface also sometimes called the terminal.

# 03. Command-line 101

The graphical user interface is a layer built on top of the operating system that lets you work quickly and intuitively. A basic representation of these layers would look like this:



**The top block is the graphical user interface**. There are your icons, your mouse **pointer**, your apps, etc. The operating system will translate the actions you do in the graphical user interface into something that it can understand.

**The middle block is the operating system core**. That's actually a program (the most essential one on a computer!) and its purpose is to translate the inputs received from the graphical user interface into something that the hardware can understand. It works the other way around too: it translates back what the hardware does into something that can be displayed by the graphical user interface. We will see an example in a minute.

**The bottom block is the hardware**. Your keyboard, your mouse or touchpad, your screen, the hard disk inside the computer, the processor and all the other components that make the machine works. The hardware receives its commands from the operating system core, processes them and respond with an output back to the operating system core.

## An alternative way to communicate with your computer

t's called the *command-line interface*. It is know by other names too, maybe you have heard one of them:

* Terminal
* CLI
* Command prompt
* Prompt
* Console
* CMD

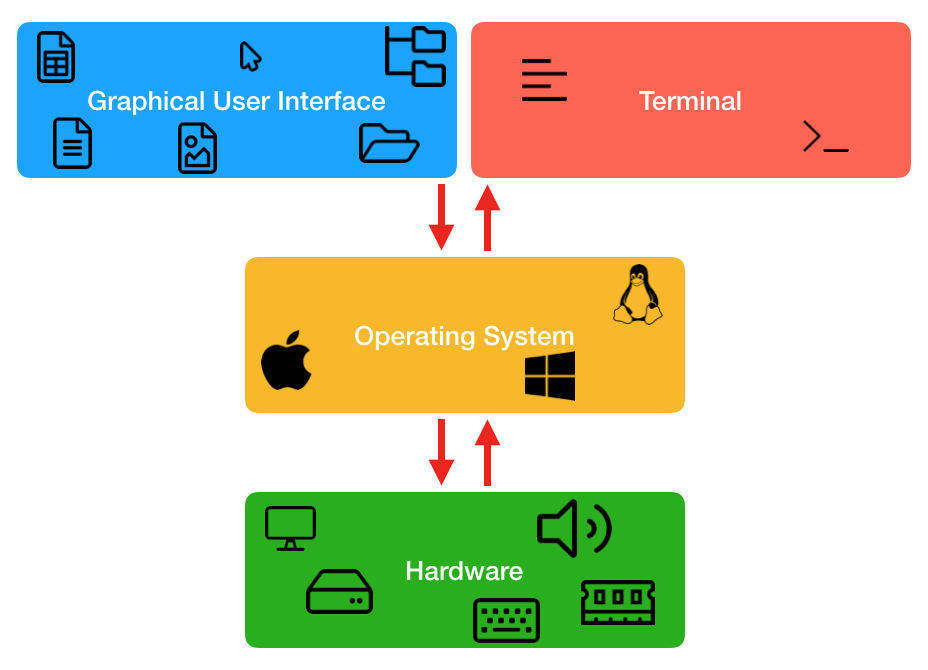
ll these terms refer to the command-line interface and are synonymous. For simplicity, from this point forward, **we will use the term terminal**.

Everything that you do using the graphical user interface can be done from the terminal by issuing the relevant commands. **On the contrary, everything you do from the terminal doesn't always have a counterpart in the graphical user interface!**

Another reason to get familiar with the terminal is that most server operating systems simply don't have a graphical user interface.

The reason for this is that a terminal doesn't consume much resources in terms of processing power and memory, and this is exactly what we want from a server: to have as much resources to do its job, not to waste them on fancy eye-candy icons and semi-transparent interfaces!

We can also update the previous schema with this one:



### Windows

The Windows terminal is also known as the Command Prompt (abbreviated cmd).

# 04. Useful terminal commands

Windows uses different commands that were used by its own parent, [MS-DOS](https://en.wikipedia.org/wiki/MS-DOS).

Mac and Linux use essentially the same commands. This is because both operating systems share the same ancestry: [UNIX](https://en.wikipedia.org/wiki/Unix).

### Windows

Listing files on Windows is done with the dir

Here are some options that you can use with dir: dir /slash

* /p Displays one screen of the listing at a time. To see the next screen, press any key on the keyboard.
* /q Displays file ownership information. Affiche la propriété des fichiers
* /w Displays the listing in wide format, with as many as five file names or directory names on each line.
* /d Displays the listing in the same format as /w, but the files are sorted by column.

To know more about the dir options, have a look here: <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/dir>.

## Changing folder

**macOS/Linux**

cd Photos/Holidays

**Windows**

cd Photos\Holidays **back slash**

**Note:** On macOS and Linux, file and folder names are case sensitive. Music and music are not the same folder. On Windows, there is no such difference.

To go back to the parent level

cd ..

## Creating folders

mkdir Courses

## Help

Help xxx

Exemple : Help robocopy

# 05. Setting up a code editor

A text editor designed to write code is also called a code editor

you **can't** use a word processor such as Microsoft Word

These extra informations would mess with the programs you are writing and the interpreter wouldn't know what to do with them. In short, use a code editor, not a word processor!

**Atom:** Available at [atom.io](https://atom.io/). Free and open-source

**Sublime Text 3:** Available at [www.sublimetext.com](https://www.sublimetext.com/). For many years, it has been the leading code editor for the Python language and Sublime Text itself was made with Python. It is extremely fast and stable, and offers support for all the main programming languages, including JavaScript and Python.

**Visual Studio Code:** Available at [code.visualstudio.com](https://code.visualstudio.com/). Free and open-source, Visual Studio Code shares the same base source code as Atom. It has been developed by Microsoft, using some of the open-source code in Atom

# 06. Setting up your GitHub account

User Name : Roulierj

Email Adresse : learner-974da0@extensionschool.ch

User Name : jdroul45,01

# 07. Your JavaScript interpreter: the browser

## Your JS interpreter: the browser

web browsers have a JavaScript interpreter built right into them.

That is how they can process and execute JavaScript code on web pages. This is what we are going to use too. Each time you will need to execute JavaScript code, we will use your browser.

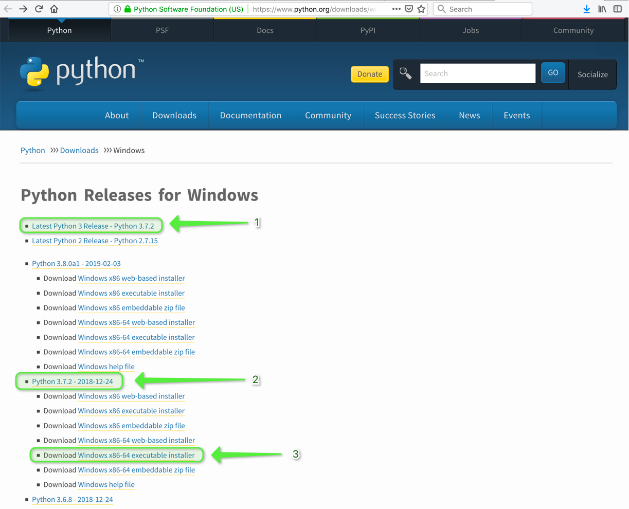
Firefox JavaScript interpreter, also known as Firefox JavaScript Engine is named [SpiderMonkey](https://en.wikipedia.org/wiki/SpiderMonkey).

# 08. Getting Python on your computer

## Windows

Go to the official Python website. They have a download page here: [www.python.org/downloads/windows/](https://www.python.org/downloads/windows/).

Lastly, find the installer that is labeled Windows x86-64 executable installer (*3*).



!!! path length limit !!

**Ordinateur\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem**

**LongPathsEnable = 1**

# 02. Running your own code

**Python code**

D:\ES\_Data\Code\Python>code ; Start Simply Visual Studio Code

D:\ES\_Data\Code\Python>**code .** ; Start Visual Studio Code from the Folder = **Code dot !!**

Create new File from Visual Studio , To safe is ctrl+s

To Start a program Python

Option 1 : From Terminal write : python example.py

Option 2 : From Visual Studio Code open Terminal and write python example.py

**Java Script code**

D:\ES\_Data\Code\JS\_Java>code ; Start Simply Visual Studio Code

D:\ES\_Data\Code\JS\_Java>code . ; Start Visual Studio Code from the Folder

Create new JS from Visual Studio , To safe is ctrl+s

Create new HTML file index.html

Solution 1

Copy, change the name corresponding to your program and Save Ctrl+S

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>JavaScript</title>

<script src="**script.js**"></script>

</head>

<body>

</body>

</html>

Solution 2

In Html file type Html , in the list select html:5

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

After title>Document</title Insert : script, and select script:src

 <script src=""></script>

Write the file name between “” and save it !

To Start a program JavaScript

1-Open la console web de FireFox :

Option 1 ; Outils/Développement web/console web

Option 2 ; Ctrl+Maj+K

2-In Visual Studio Code

Option 1 ; Clic right on Index.html , reveal in File Folder

Option 2 ; Shift+Alt+R

3-Drag and drop the index file in the browser / up windows and the program start automatically , you can restart from the browser with Ctrl+A

# 03. Storing values with variables

## Creating your own variables

**Python**

In programming, = doesn't mean equals! It means **assign**.

* You can use any letter or the underscore character (\_).
* You can use numbers in your variable names but *you can't start the name with a number*.
* You can't use whitespaces or special characters that have a meaning in Python such as + or -.
* Variable names are case sensitive. That means that fruits and Fruits are two different variables.
* Start your variable names with a lower case letter. fruits and ***not* Fruits**. It's not a rule but a convention.
* Separate multiple words with an underscore like number\_of\_rooms, car\_color.
* Python reserved words cannot be used as variable names. We will see a list of these later.

Color /ok

Temperature /ok

player3 /ok

first\_name /ok

phone\_number /ok

**~~3~~**~~rd\_player~~ /Not OK

~~last~~**~~+~~**~~name~~/Not OK

~~total amount~~/Not OK

## Reassigning variables

apples = 5

print(apples)

apples = 7

print(apples)

apples = 11

apples = 3

apples = 9

print(apples)

Here is a Python example: Comment start with #

**#** This line won't be seen by the Python interpreter

apples = 5

oranges = 3

## Numbers

temperature = 21**.**6

## Text / text is named a **string** in programming.

message = "Hello everyone!" Normalement toujours entre guillemet double !

Exception! ‘ texte ‘’ texte ‘’ ‘

print('the difference between "strings" and "numbers" is really important' )

## Numbers

* To add, you use the plus sign: +
* To subtract, you use the minus sign: -
* To multiply, you use the asterisk sign: \*
* To divide, you use the forward slash: /

Text!

name = "Rick"

greetings = "Hello"

message = greetings + name

print(message)

HelloRick

To add space :

space = " " / new variable

message = greetings + space + name / written space

message = greetings + " " + name / “” = space

greetings = "Hello " / “Hello + space”

**Java**

In order to tell the JavaScript interpreter that you are going to define and **assign** a value to a variable, you need to start your line with the keyword var

You also have to tell JavaScript when you line of code ends. Unlike Python, JavaScript doesn't see the new line character. To the eyes of the interpreter, JavaScript code is just one long line of code so you'll need to specify when your statement ends. You use the semi-colon character for this**: ;.**

Here is an example in JavaScript:

var fruits = 7;

* Variable names can contain letters, digits, underscores (\_) and dollar signs ($).
* Variable names are case sensitive.
* Variable names ***cannot* start with a number**.
* Start your variable names with a lower case letter.
* JavaScript reserved words cannot be used as variable names. We will see a list of these later.
* As a convention, when a variable name contains multiple words, we separate them with upper case letters. This is called *camel case*. For example numberOfRooms, carColor.

color

temperature

player3

firstName

phoneNumber

~~3rdPlayer~~

~~last name~~

~~total/amount~~

## Reassigning variables / var = only one time !!

var apples = 5;

console.log(apples);

apples = 7;

apples = 11;

console.log(apples);

JavaScript example: Comment start with //

**//** This is a comment line!

var a = 3;

var b = 10;

**//** This is another comment line

## Numbers

var fruits = 11;

console.log(fruits);

console.log(58**.**45);

## Text / text is named a **string** in programming.

console.log("c'est bon")

Exception ‘ texte ‘’ texte ‘’ ‘

console.log('tout est bon dans le "cochon"')

## Numbers

* To add, you use the plus sign: +
* To subtract, you use the minus sign: -
* To multiply, you use the asterisk sign: \*
* To divide, you use the forward slash: /

# 10. Implementing reusability with functions

We will do this using **functions**.

## in Python

def say\_hello(): This first line is also called the **function signature**

print("Hello!") called the **function body**.

Attention to the **4** whitespaces in front of print("Hello!"):

say\_hello() Executing a function is named **calling a function**.

say\_hello()

say\_hello**()**

Lorsque vous définissez une variable entre parenthèses d'une signature de fonction, nous appelons cela un paramètre

When you define a variable within the parentheses of a function signature, we call this a **parameter**. It is simply a variable.

**Note:** Now that you know about function parameters, look closer at the print statement that we use everywhere. Do you see it? Yes! print() is a **function** itself! **printing on terminal !!**

## Return values

def add(number\_one, number\_two):

result = number\_one + number\_two

**return result !!!** return is actually a keyword, not a function, then without ()

total = add(3,7)

print(total)

What it will do is that the content of the result (in this case, 10) will be returned outside of the function.

Simplify …

def add(number\_one, number\_two):

return number\_one + number\_two

print(add(3,7))

## in Java

// A function that says hello to someone

**function** sayHello(name) **{**

console.log("Hello " + name);

**}**

// Calling the function

sayHello("Rick");

sayHello("Lory");

sayHello("Daryl");

# 18. Mixing data types

## Converting types in Python

**Converter functions**

There are three that are specially designed to solve this common issue:

* str() will convert any number to a string.
* int() will convert a string or a decimal number to an integer (eg. 3.7 will be converted to 3).
* float() will convert a string or an integer to a decimal number (eg. 3 will be converted to 3.0).

string = châine de caractères

integer = nombre entier qui n’est pas une fraction / La gamme de valeurs décimales qu'il est possible d'encoder sur 4 octets seulement s'étend de -2147483648 à + 2147483647

float = [Nombre](https://fr.wiktionary.org/wiki/nombre) à [virgule](https://fr.wiktionary.org/wiki/virgule) [flottante](https://fr.wiktionary.org/wiki/flottante) Le type ***float*** utilisé dans notre exemple permet de manipuler des nombres (positifs ou négatifs) compris entre 10-308 et 10308 avec une précision de 12 chiffres significatifs

**Important:** When converting a string to a number, if the string value is not a number, both int() and float() will generate an error. For example, int("what?") would fail. **Make sure that when using these, the string can actually be converted to a number!**

def add\_function(number\_one,number\_two,number\_tree):

    result = number\_one + number\_two + number\_tree

    return result

total = add\_function(10,12,13)

first\_message = "Le résultat = "

second\_message = " tout est ok"

print (first\_message + str(total) + second\_message)

[**https://docs.python.org/3/library/index.html#library-index**](https://docs.python.org/3/library/index.html#library-index)

# Concerter celcius to Farenheit

def cnv\_farenheit(celsius):

    return  str(celsius) + " Degré C is " + str(((celsius \*9/5)+32)) + " Fahrenheit"

print(cnv\_farenheit(21.5))

print(cnv\_farenheit(-7))

print(cnv\_farenheit(11))

print(cnv\_farenheit(0))

## Converting types in JavaScript

JavaScript also gives us some functions to convert strings and numbers: **toString() and Number()**

The function toString() is a little bit different in its usage as it used as a *postfix*. Look at the following example:

var aNumber = 5;

var aString = "Hello";

var result = aNumber.**toString(**) + aString;

console.log(result);

var aNumber = 5;

var aString = "3";

var result = aNumber + **Number**(aString); **Number avec N majuscule !!**

console.log(result);

**Important:** If you try to use Number() on a variable that can't be converted to an actual number, JavaScript will output the special value **NaN which stands for Not a Number**. It is a special value that we use sometimes and that we will discuss later when we will talk about equality.

// Test add + mixing data types

function add(number\_1,number\_2,text\_1,text\_2){

    var total = number\_1 + number\_2;

    return (text\_1+total.toString()+text\_2);

   }

console.log(add(99,99,"hello "," Test"))

[**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference)

/ Converter Celcius to Farenheit

function cnvFarenheit(celcius){

    var cnv=((celcius \* 9/5) + 32)

    return celcius.toString() + " Degré C is " + ((celcius \* 9/5) + 32).toString() + " Fahrenheit";

}

console.log(cnvFarenheit(21.5));

console.log(cnvFarenheit(-7));

console.log(cnvFarenheit(11));

console.log(cnvFarenheit(0));

# 21. Exercise: Your own converter

// Converter Pa / mbar / Atm / Psi

// mbar Formula diviser la valeur "pression" par 100

// atm Formula diviser la valeur "pression" par 101325

// psi Formula pour obtenir un résultat approximatif, diviser la valeur "pression" par 6895

function converterPa(valPa){

    var valMbar = valPa / 100

    var valAtm = valPa / 101325

    var ValPsi = valPa / 6895

    var textStart = "The conversion of "+ valPa.toString() + " Pa in MBar/Atm/Psi" + " ="

    var textMiddle = " , "

    var cnvMbar = textStart + valMbar.toString() + " mbar";

    var cnvAtm = textMiddle + valAtm.toString() + " atm";

    var cnvPsi = textMiddle + ValPsi.toString() + " psi";

    //console.log (cnvMbar);

    //console.log (cnvAtm);

    //console.log (cnvPsi);

    return cnvMbar + cnvAtm + cnvPsi;

}

console.log(converterPa(100))

console.log(converterPa(1000))

console.log(converterPa(650))

# 25. Getting user input

## The input() function

name = input("Please enter your name: ")

print("Your name is " + name)

Your program will now display the following:

Please enter your name: Nico

Your name is Nico

# 02. The most basic of all types: Booleans

## True and false

Python

day = **T**rue

night = **F**alse

print(day)

print(night)

The output will be:

True

False

Javascript

var day = true;

var night = false;

console.log(day);

console.log(night);

And it outputs the following in the console:

true

false

## Equality

print(5 == 5)

Python

# Test fonction Boolean

secret\_number = 5

guess = input("Rentré un numéro de 1 à 10 ")

print(secret\_number == int(guess))

Javascript

Comme il n y a pas d’input , guess est déjà un chiffre !

// Test fonction Boolean

var secretNumber = 5;

var guess = 6;

console.log(secretNumber == guess);

## Not equal

**print(5 != 5)**

**console.log(5 != 5)**

# 06. If this, then that!

This is the Python syntax for the *if…then* logic:

if test:

what to do if the test evaluates to true

# test fonction if

password = "secret"

user\_input =input("Entrez le mot de passe ici = ")

if password == user\_input:

    print("You are now logged")

weather = "sunny"

if weather == "sunny"**:**

print("It is sunny outside.")

print("It is a beautiful day!")

**!!!!! Fin des 4 espaces = fin de la condition IF**

print("This line of code will execute no matter what.")

## **Beyond equal: greater and smaller**

To express greater than or lesser than we use the mathematical symbols > and <. Here is an example:

apples = 7

oranges = 3

if apples > oranges**:**

print("There are more apples than oranges")

-------

You can also say greater than or equal with the combination of the two signs >= or lesser than or equal with <=. Here is another example:

apples = 3

oranges = 3

if apples <= oranges**:**

print("There are less apples than oranges...")

print("...or the same amount!")

# Find the result of an operation

guess = input(" What is the result of 5+2? = ")

if float(guess) == 7 :

    print("Congartulations ! You fund the correct result ")

if float(guess) < 7:

    print("Wrong answer too small, try again")

if float(guess) >7:

    print("Wrong answer too big, try again")

print("The programm terminated sucessfully")

## If…then: JavaScript edition

Here is the whole syntax of a JavaScript if…then block:

**if (test) {**

code to be executed if test evaluates to true

}

code that is executed regardless of the test result

// Test fonction Boolean

var apples = 7;

var oranges = 3;

if (apples <= oranges){

    console.log("The are less apples than oranges...");

    console.log("... or the same amount!");

}

console.log(" The program terminated sucessfully")

# conversion pa / mbar / atm / psi

def multi\_cnv\_pression(val,unit):

    if(unit == "pa"):

        start\_txt = "The convertion of " + str(val) + " pa in mbar/atm/psi = "

        pa\_txt = ""

        mbar\_txt = ",   " + str(val/100) + " mbar"

        atm\_txt = ",   " + str(val/101325) + " atm"

        psi\_txt = ",   " + str(val/6895) + " psi"

    if(unit == "mbar"):

        start\_txt = "The convertion of " + str(val) + " pa in pa/atm/psi = "

        pa\_txt = ",   " + str(val\*100) + " pa"

        mbar\_txt =""

        atm\_txt = ",   " + str(val/1013.25) + " atm"

        psi\_txt = ",   " + str(val/68.948) + " psi"

    if(unit == "atm"):

        start\_txt = "The convertion of " + str(val) + " pa in pa/mbar/psi = "

        pa\_txt = ",   " + str(val\*101325) + " pa"

        mbar\_txt = ",   " + str(val\*1013.25) + " mbar"

        atm\_txt =""

        psi\_txt = ",   " + str(val\*14.6959) + " psi"

    if(unit == "psi"):

        start\_txt = "The convertion of " + str(val) + " pa in pa/mbar/atm = "

        pa\_txt = ",   " + str(val\*6894.76) + " pa"

        mbar\_txt = ",   " + str(val\*68.9476) + " mbar"

        atm\_txt = ",   " + str(val/14.696) + " atm"

        psi\_txt = ""

    return start\_txt + pa\_txt + mbar\_txt + atm\_txt + psi\_txt

print("Small routine to convert pression pa/mbar/atm/psi")

unit = input("Give the unit of pression to convert: ")

value = input("Give the value of pression: ")

print(multi\_cnv\_pression(float(value),unit))

again = input("Would you convert a new value? Yes or No : " )

if(again == "Yes" ):

    print("Ok lets try again")

if(again== "No"):

    print("Program terminated , see you soon")

# 16. Say Else in Python

if apples <= oranges:

print("There are less apples than oranges...")

print("...or the same amount!")

else:

print("There are more apples than oranges.")

# Test If then else

print(" Hello I just want to play with you")

def test\_number(number):

    if(int(number) < 0):

        message = ("Number to small")

    else:

        if(int(number) > 10):

            message = ("Number to big")

        else:

            message = ("Its a good number")

    return message

def test\_calcul(number\_1,number\_2,select):

    if(select == "add"):

        result = int(number\_1)+int(number\_2)

    if(select == "mult"):

        result = int(number\_1)\*int(number\_2)

    if(select == "div"):

        result = int(number\_1)/int(number\_2)

    print("First " + test\_number(number\_1))

    print("Second "+ test\_number(number\_2))

    return result

# Input

number\_1 = input("Give a first number from 1 to 10 : ")

number\_2 = input("Give a second number from 1 to 10 : ")

select = input("Select the calcul : add , mult , div : ")

test = input("Give your result : ")

# Control

result=int(test\_calcul(number\_1,number\_2,select))

if(result == int(test)):

    print("Excellent")

    print("Your answer " + str(test) + "= control " + str(result))

else:

    print("Wrong ! Try again")

    print(str(result))

# 16. Say Else in Javascript

Without further ado, here is the example we will analyze. Try to understand the syntax:

var apples = 3;

var oranges = 11;

if (apples >= oranges) {

console.log("There are more apples than oranges...");

console.log("...or the same amount!");

} else {

console.log("There are more oranges!");

}

console.log("The program executed properly.")

// secret word test in java

var secretWord = "kiwi";

var guess =prompt("give me the secret world :");

if(secretWord == guess){

    console.log("Yes ou get it");

}else {

    console.log("Try again");

}

console.log("Program is executed");

# 20. Chaining if statements in Python

weather = "sunny"

if weather == "raining":

print("Take an umbrella.")

elif weather == "sunny":

print("Take sunglasses.")

elif weather == "snowing":

print("Take gloves.")

else:

print("Stay home.")

## Test fonction elif

number\_1 = input("Give me a first number : ")

number\_2 = input("Give me a second number : ")

result = float(number\_1)-float(number\_2)

if result > 10:

    print("The result is greater than 10")

elif result > 0:

    print("The result is greater than 0 but not than 10")

elif result == 0:

    print("The result is 0")

else:

    print("The result is a negative number")

print("You can try again")

# 21. Chaining conditions with JavaScript

var weather = "sunny";

if (weather == "raining") {

console.log("Take an umbrella.");

} else if (weather == "sunny") {

console.log("Take sunglasses.");

} else if (weather == "snowing") {

console.log("Take gloves.");

} else {

console.log("Stay home.");

}

// test if else in Java script

var number\_1 = prompt("Give me number 1: ");

var number\_2 = prompt("Give me number 2: ");

var result = Number(number\_1)-Number(number\_2);

if ( result >10 ){

    console.log("The result is greater than 10");

} else if (result >0 ){

    console.log("The result is greater than 0 but not bigger than 10");

} else if (result == 0){

    console.log("The resukt is zero");

} else {

    console.log("The result is a negative value")

}

console.log("try again")

// Test gender

var gender = prompt("Give you gender: ");

function greet(gender){

    if (gender == "male"){

    return ("Hello sir ! Welcome back ");

    } else if (gender =="female"){

    return ("Hello Madam ! Welcome back ");

    } else {

    return ("Hello ! Welcome back ");

    }

}

console.log(greet(gender));

# 25. The syntax of AND

**Python**

a = input("Enter a number: ")

b = input("Enter another number: ")

if int(a) > 10 and int(b) > 10:

print("Both numbers are greater than 10")

else:

print("At least one of the numbers you entered is not greater than 10")

# Test and function

a = input("Give me a: ")

b = input("Give me b: ")

def check(number\_1,number\_2):

    if float(number\_1) > 10 and float(number\_2) > 1210 :

        return "Both number are greater than 10"

    else :

        return "At leas one number is not greater than 10"

print (check(a,b))

# Test consitions meteo

temp = input("Give me temp ; warm/cold : ")

clim = input("Give me climat ; sunny/rainny ")

def action\_to\_do(temp,clim):

    if clim == "rainny" and temp == "cold":

        return "take an umbrella an a warm jacket"

    elif clim == "rainny" and temp == "warm":

            return "take an umbrella and a t-shirt"

    elif clim == "sunny" and temp == "cold":

            return "take sunglasses and a warm jacket"

    elif clim == "sunny" and temp == "warm":

            return "take sunglasses and a t-shirt0"

    else :

        return " stay home"

print(action\_to\_do(temp,clim))

**Java**

var weather = "sunny";

var temperature = "warm";

if (weather == "sunny" && temperature == "warm") {

console.log("Take sunglasses and a t-shirt");

}

/ Test And fonction

var a = prompt("Give me first number: ");

var b = prompt("Give me second number: ");

function check(number\_1,number\_2){

    if (Number(number\_1) > 10 && Number(number\_2) > 10) {

        return ("Both number are greater than 10 ");

    } else {

        return ("At leat one number is smaller than 10 ");

    }

}console.log(check(a,b));

// Test fonction And + If else

var clim = prompt("Give ne the climat ; sunny/rainny: ");

var temp = prompt("Give me the temperature ; warm/cold: ");

function action(clim,temp){

if ((clim == "rainny") && (temp == "cold")){

    return ("take an mbrella and a warm jacket");

} else if ((clim == "rainny") && (temp == "warm")){

    return ("take an umbrella and a t-shirt");

} else if ((clim == "sunny") && (temp == "cold")){

    return ("take sunglasses and a wwarm jacket");

} else if ((clim == "sunny") && (temp == "warm")){

    return ("take sunglasses and a t-shirt");

}else {

    return ("stay home");

}

}

console.log(action(clim,temp));

# 29. Saying OR

**Python**

a = 8

b = 12

if a > 10 **or** b > 10:

print("At least one of the two numbers is greater than 10!")

else:

print("Both numbers are not greater than 10.")

**JavaScript**

var a = 8;

var b = 12;

if (a > 10 **||** b > 10) {

console.log("At least one of the numbers is greater than 10");

} else {

console.log("Both numbers are not greater than 10");

}

The important part is at line 4. Check out how we say or in JavaScript. **With two pipe characters: ||. (AltGr + 7 )**

# 33. Walkthrough: Guess the number and the color

**My solution**

# Test my color and number

my\_number = 20

my\_color = "blue"

number = input("Please give me a number betwee 0 and 20: ")

color = input("Please give me a color: ")

def checksecret(my\_number,my\_color,number,color):

    if my\_number == float(number) and my\_color == color:

        return("You've found both the secret number and the color number")

    elif my\_number == float(number) or my\_color == color:

        return("You found a least one of the secrets")

    else:

        return("You didn't fine any of the secrets !\nBetter luck next time")

print(checksecret(my\_number,my\_color,number,color))

print("Try again")

**Solution of the course :**

**!!! Just write the program like here and Test it !! ( Normal program and not directly a function )!!**

# Test my color and number

my\_number = 20

my\_color = "blue"

number = input("Please give me a number betwee 0 and 20: ")

color = input("Please give me a color: ")

if my\_number == float(number) and my\_color == color:

    print("You've found both the secret number and the color number")

elif my\_number == float(number) or my\_color == color:

    print("You found a least one of the secrets")

else:

    print("You didn't fine any of the secrets !")

    print("Better luck next time")

print("Try again")

**And to transform in Function … define a function at the top like play()**

**and ad 4 space at each line**

def play():

    # Test my color and number

    my\_number = 20

    my\_color = "blue"

    number = input("Please give me a number betwee 0 and 20: ")

    color = input("Please give me a color: ")

    if my\_number == float(number) and my\_color == color:

        print("You've found both the secret number and the color number")

    elif my\_number == float(number) or my\_color == color:

        print("You found a least one of the secrets")

    else:

        print("You didn't fine any of the secrets !")

        print("Better luck next time")

    print("Try again")

play()

**In JAVA :**

// Test my secret

var myNumber = 20

var myColor = "blue"

var number = prompt("Give me a number : ")

var color = prompt("Give me a color: ")

if(myNumber == Number(number) && myColor == color){

    console.log("You've found both of my secret !")

}else if(myNumber == Number(number) || myColor == color){

    console.log("You found at least on of my secret !")

}else{

    console.log("You didn't find any of the secret \nBetterluck next time")

}

console.log("Try agaîn")

**And to transform in Function … define a function at the top like play()**

**and ad 4 space at each line**

function play(){

    // Test my secret

    var myNumber = 20

    var myColor = "blue"

    var number = prompt("Give me a number : ")

    var color = prompt("Give me a color: ")

    if(myNumber == Number(number) && myColor == color){

        console.log("You've found both of my secret !")

    }else if(myNumber == Number(number) || myColor == color){

        console.log("You found at least on of my secret !")

    }else{

        console.log("You didn't find any of the secret \nBetterluck next time")

    }console.log("Try agaîn")

}

play()

# 03. The common types of errors

## Syntax errors

The most useful one would be to check your code editor: does it highlight a particular line? Does it alert you that a line of code is potentially wrong? Pay attention to these clues.

## Implementation errors

Example: forgot that multiplications or divisions are processed before additions.

## Logic errors

**They are those that you make in the algorithm itself**. If the algorithm and formulas you translate from English to code are already wrong when said in English, you are going to have a tough time tracking them down!

# 04. Debugging JavaScript strategies

## JavaScript implementation errors

var a = 0;

console.log(a); // log

a = a + 10;

**// at this point, a == 10**

console.log(a); // log

a = a \* 2 + 1;

**// at this point, a == 21**

console.log(a); // log

a = a + 3 / 2;

**// at this point, a == 12**

console.log(a); // log

a = a - 11;

**// at this point, a == 1**

console.log(a); // log

console.log(a);

**Boolean parameter means , only if true or false !**

**To test this function example : login(false) or login(true) !!!!**

**The error here was the condition else !**

/ This function takes a boolean parameter

// that says if password was entered correctly

// (true = yes, false = no)

function logIn(passwordIsCorrect) {

    if (passwordIsCorrect != true) {

      console.log("Your password is not correct, you can't log in");

    }

    console.log("Welcome back! You are now signed in.");

  }

  function logIn(passwordIsCorrect) {

    if (passwordIsCorrect != true) {

        console.log("Your password is not correct, you can't log in");

    } else {

        console.log("Welcome back! You are now signed in.");

}

# 02. What is Source Control (and why use it)

First, the most important choice you should make is what source control management (or SCM) app you are going to use. In this course, I will show you the one called Git. There are many others such as Subversion, Team Foundation Server, CVS, Mercurial, and more.

Although there are third party apps that let you work with Git by using a graphical user interface, we will learn how to use Git from the command line interface. I'll assume that you know the basics of working with your terminal.

we will concentrate on the following workflow:

1. Installing and configuring Git
2. Creating a repository
3. Adding and committing your files
4. Reverting to an earlier version of your project
5. Uploading your work online for collaboration and safekeeping using GitHub
6. Creating branches and merging them in your project

## Configuring Git

D:\>git config --global user.name "Roulier"

D:\>git config --global user.email jean-daniel.roulier@epfl.ch

mkdir mybook

cd mybook

You are now within the mybook folder and we will tell Git "*Hey Git! Please track everything that happens in this folder so later, I can review the changes I made to my book and maybe revert back to an earlier version*". **This long sentence is translated to Git by the command git init**.:

git init

git status

git add file.txt Add file four git tracking

git commit -m "Added the first line of the first chapter"

After each change we have to save again + mentioned the change with commit -m

git add chapter1.txt

git commit -m "Changed the text in chapter 1"

## Reverting to a previous version

git log

A log is read from bottom to top. The last line of the output gives you your first commit message

To create an empty txt file

type nul > file\_name.txt

visit the [Git Handbook](https://guides.github.com/introduction/git-handbook/) at GitHub.

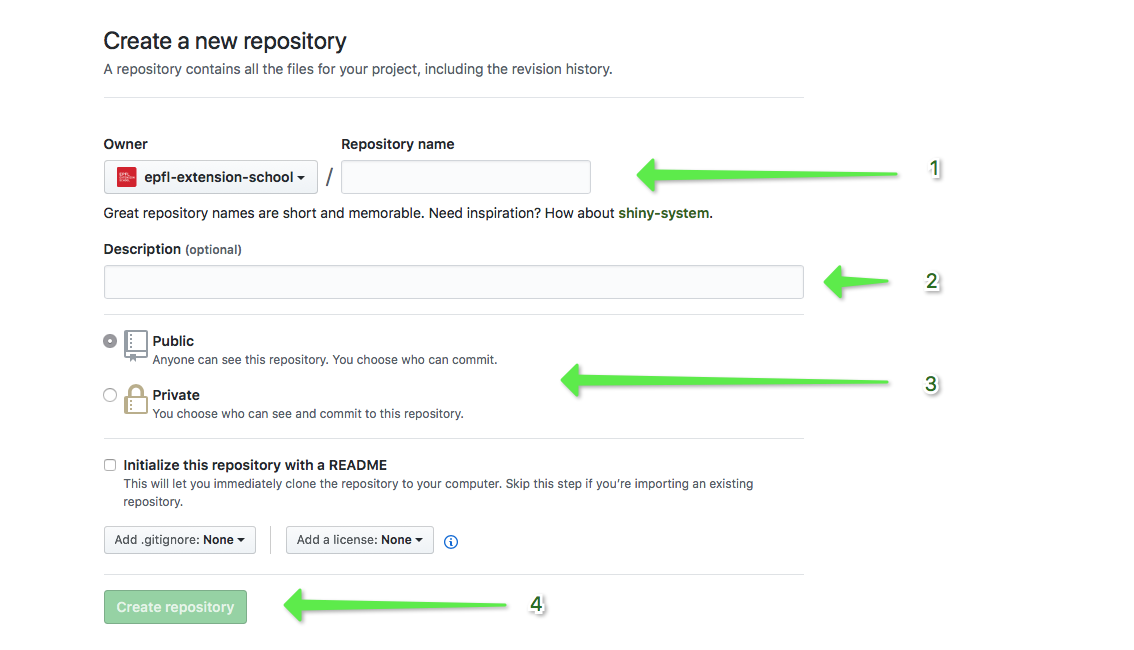
# 05. Your online Git repository: GitHub

User Name : Roulierj

Email Adresse : learner-974da0@extensionschool.ch

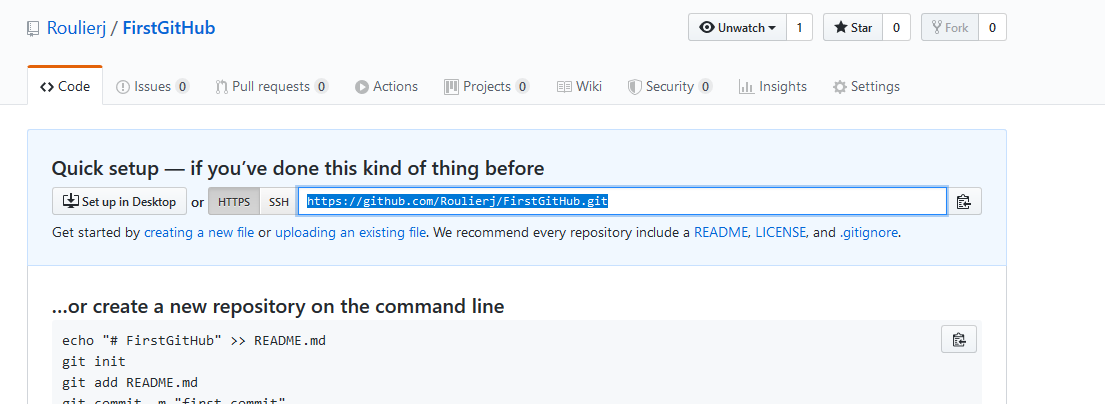
User Name : jdroul45,01

You will be presented with the following form:



1. Type the name of the repository you want to use. You can't use spaces or special characters.
2. Optionally, you can add a description such as *the main repository for my fantastic book*.
3. GitHub lets you choose if you want your repository to be public (open for anyone to see but not change, ideal for publishing your portfolio of code or creating a new open-source project) or private (only you can see its content). Note that by being a learner at the EPFL Extension School, you will get private repositories created automatically for you in order to store your exercises and final course project (more on that later!).
4. When you've filled everything in, press *Create repository*.

In blue is the link used to push the files into GitHub



Lien : <https://github.com/Roulierj/FirstGitHub.git>

## Uploading your repository to GitHub

The procedure for pushing your repository to GitHub (yes, we use push and not upload when we talk about Git repositories) is straightforward. Make sure that you are located inside a folder containing a Git repository on your computer and type the following, replacing the address at the end with the one you just copied:

git remote add origin <https://github.com/Roulierj/FirstGitHub.git>

1. git remote: instructs Git that we want to do something with a remote repository.
2. add: well, we want to *add* a new remote repository (the one we created on GitHub).
3. **origin**: **this is the short name for your remote repository** so you don't have to type the url each time you want to push something. You can choose any name you want but the common and best practice is to use origin for naming your remote repository.
4. What follows is the address of the repository on GitHub.

Once you've added the remote repository, Git is aware that it can push your commits to it so let's do just that with:

git push origin master

Let's take a closer look at the command you just typed:

1. git push: instructs Git to push your local repository to a remote one.
2. origin: the name of the remote repository.
3. master: the name of the local repository (or the local *branch*) to push. By default, we use master.

## Retrieving your repository

Let's now say that you deleted your local project. No more folder, no more Git repository on your computer. You can very easily download it from GitHub. This operation is called cloning. You simply type the following, replacing the url by the address of your repository:

git clone https://github.com/nicoschuele/myproject.git

**Resume:**

1. git init / initialize the Git structure
2. git add filename.\* or add . / I have to for each change that I want to save ( add. = means add all what it’s a folder )
3. git commit -m “blablablaba” / I have to give a comment for each change
4. git status / To get the status
5. git log .. to see the different version with numbering
6. git checkout **20d89035** / To reverse an old version Id of version fund in log file
7. Create the repository in GitHub and copy the link
8. git remote add origin [https://github.com/Roulierj/myproject.git /](https://github.com/Roulierj/myproject.git%20%20/) defines the origin !! at this time nothing is saved !!!!
9. git push origin master / Means push the master to link origin !!
10. git clone [https://github.com/Roulierj/myproject.git /](https://github.com/Roulierj/myproject.git%20%20%20%20%20%20%20/) Copy from GitHub

# 02. Collections

**Note:** Collections are part of a wider subject known as data structures. The study of data structures is outside the scope of this course but remember that term. If you ever have an issue with collections, this is a keyword you can use in search engines.

# 03. Arrays Python

In the above example, I've just created what is known as an **array**. Arrays are variables that can hold multiple values. The Python syntax is fairly easy:

variable\_name = [value1, value2, value3, value4]

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot"]

print(employees)

The output would be the following:

['Richard Swift', 'Marie-Anne Petrie', 'Cody Lightfoot']

What if we only want *Richard Swift*? We access it by using its **position** in the array also known as an **index**. Each value in an array has an index. **It starts at 0**. Richard would be 0, Marie-Anne is 1 and Cody is 2.

1. You can store a set of related data into an array variable.
2. You can access each individual value by using its position (index) in the array.
3. Python arrays are *zero-based* because the first value in the array is always at index 0.

Its possible to break the variable array in several line : Example

# List of my favorite songs .. the difficult choice ;-)

favorite\_song = [ "Y No Hago Mas Na",

                "Heart of Courage",

                "Shiva Moon",

                "Sun in Your Eyes"]

print(favorite\_song[0])

print(favorite\_song[1])

print(favorite\_song[2])

print(favorite\_song[3])

To change the value of one Array

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot"]

employees[1] = "Marie-Anne Zimmer"

print(employees)

## Adding values to an array

We can add her using the array function **.append()**

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot"]

employees.append("Laure Simmons")

print(employees)

## Removing values from an array

They have the **pop()** function to remove a value at a particular position

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot", "Laure Simmons"]

employees.pop(2) = **index 2 = third element**

print(employees)

['Richard Swift', 'Marie-Anne Petrie', 'Laure Simmons']

**Note:** Before we removed "Cody Lightfoot", "Laure Simmons" was at index position 3. After removing Cody, Laure is now at index position 2. Pay attention to this when you are adding/removing values as it can become tricky quickly.

## How many values do I have?

You can also automatically get the count of values in an array with the len() function.

number\_of\_employees = **len(**employees**)**

## Sorting values in an array

As a bonus, here are two more array functions that you can use: sort() and reverse(). The first one will sort the values in your array from the smallest to the largest. In the case of string values, it will sort them in alphabetical order. Like this:

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot", "Laure Simmons"]

employees.sort()

print(employees)

Here is the result:

['Cody Lightfoot', 'Laure Simmons', 'Marie-Anne Petrie', 'Richard Swift']

The reverse() function is used exactly in the same way as sort().

.sort() **/// Dans l’ordre alphabétique !!!**

.reverse() **/// Inverse l’ordre dans la chaine .. pas de lien avec l’ordre alphabétique**

**Important:** Of course, when you use the sort() or reverse() functions, the indexes are modified as well. For example, what was at position 0 before sorting may be a different value after sorting!

To have both on variable with the function sorted() we can have names sorted in a variable !

names = ["Andrea", "Claudio", "Bruno"]

sorted\_names = sorted(names)

print(sorted\_names)

print(names)

To know the index of :

fruits = ["apple", "banana", "kiwi"]

print(fruits.index("banana"))

**Récap Array Python :**

variable\_name = [value1, value2, value3, value4]

employees = ["Richard Swift", "Marie-Anne Petrie", "Cody Lightfoot"]

employees.append("Laure Simmons")

employees.pop(2) = **index 2 = third element**

number\_of\_employees = **len(**employees**)**

employees.sort()

employees.reverse()

.sort() **/// Dans l’ordre alphabétique !!!**

.reverse() **/// Inverse l’ordre dans la chaine .. pas de lien avec l’ordre alphabétique**

print(employees.index("Richard Swift"))/// Pour connaître l’index

---------

song\_index = my\_favorites\_song.index(song)

# This will print the index of the song you passed as parameter

print(song\_index)

--------

sorted\_names = sorted(employees)

print(sorted\_names)/// Imprimer dans l’ordre alphabétique sans changer la variable d’origine

# 03. Arrays JavaScript

var fruits = ["oranges", "apples", "bananas"];

console.log(fruits);

Array(3) [ "oranges", "apples", "bananas" ]

## Accessing values

**JavaScript arrays are zero-based too!**

var fruits = ["oranges", "apples", "bananas"];

var message = "The second value contains apples: " + fruits[1];

console.log(message);

## Updating a value

var fruits = ["oranges", "apples", "bananas"];

fruits[1] = "kiwis";

var message = "The second value now contains kiwis: " + fruits[1];

console.log(message);

## Adding and removing values to a JavaScript array

JavaScript uses .**push()** / To add

var fruits = ["oranges", "apples", "bananas"];

fruits.push("tomatoes");

console.log(fruits);

You get the following (and yes, tomatoes are a fruit!):

Array(4) [ "oranges", "apples", "bananas", "tomatoes" ]

JavaScript use .**splice()** / To remove

var fruits = ["oranges", "apples", "bananas", "kiwis"];

fruits.splice(2, 1);

console.log(fruits);

Here I removed "bananas". I've used **splice(2, 1)** to do that. The first parameter, 2, indicates that I want to remove the 3rd element from the array. And the second parameter, 1, indicates that I want to remove one element starting at the said position.

Array(3) [ "oranges", "apples", "kiwis" ]

## Sorting

.sort() **/// Dans l’ordre alphabétique !!!**

.reverse() **/// Inverse l’ordre dans la chaine .. pas de lien avec l’ordre alphabétique**

## JavaScript array length

Like this: myArray**.length**

No! **This is called a property** and we will see really soon what it is but for now, you'll just have to remember that for getting the size of an array, you use myArray.length!

var fruits = ["oranges", "apples", "bananas", "kiwis"];

var fruitsCount = fruits.length;

console.log(fruitsCount);

**Récap Array Java :**

var fruits = ["oranges", "apples", "bananas"];

fruits.push("tomatoes");

fruits.splice(2, 1);

fruits.sort()

fruits.reverse()

.sort() **/// Dans l’ordre alphabétique !!!**

.reverse() **/// Inverse l’ordre dans la chaine .. pas de lien avec l’ordre alphabétique**

var fruitsCount = fruits.length;// to write the length in one variable

# 13. Write loops in Python

## For loops

for x in range(10)**:**

print("Hello!")

The "following task" that we want to be executed 10 times is written on the lines following the loop statement using a 4 spaces indent

Why from 0 to 9 and not from 1 to 10? This is an excellent question! It's because Python is zero-based

for x in range(5):

    value = str(x + 1) # Start at number 1

    if value == 1:

        message = " step"

    else:

        message = " steps"

    print(value + message)

## For loops in array !!

Fruits = [“Banana”,”Orange”,”Pommes”]

for i in Fruits**:**

print(i)

Fruits = [“Banana”,”Orange”,”Pommes”]

for fruit in Fruits**:**

print(fruit)

# 16. Loop while...Python

*while something is true, continue to execute a task.*

Let's see how we can do that with Python. I want to translate *while my number is smaller than 10, I want you to continue counting*. Here is the solution:

my\_number = 0

while my\_number < 10:

print(my\_number)

my\_number = my\_number + 1

!!! If the condition is always = True

while my\_number < 10:

print(my\_number)

# line in remark my\_number = my\_number + 1

Your program would enter a state known as an infinite loop. That is, the condition my\_number < 10 would always be True and the program would never quit the loop.

You can try it in your terminal but know this before: you can use the keyboard shortcut CTRL + C on Mac or **CTRL + C** on Windows to terminate a program that entered an infinite loop.

# List of my favorite songs .. the difficult choice ;-)

favorite\_song = [ "Y No Hago Mas Na",

                "Heart of Courage",

                "Shiva Moon",

                "Sun in Your Eyes","Del Monton"]

print("These are my favorite songs : ")

for song in favorite\_song:

    print(song)

print("What are yours?")

# 17. For Loop Java

The major difference between Python for loops and JavaScript for loops is that **we also need to increment our iterator ourselves when using JavaScript**.

for (var i = 0; i < 5; **i = i + 1**) {

console.log("Step: " + i.toString());

}

we set a semi-colon (;) after setting our iterator to let JavaScript know that we are moving to the next part of our loop definition.

For loop in Array

var fruits = ["apples", "oranges", "bananas"];

for(i = 0; i < fruits.length; i = i + 1) {

console.log(fruits[i]);

}

we have to use the size of the array in our loop condition:

i < fruits.length;

# 17. While Loop Java

var i = 0;

while (i < 5) {

console.log("Hello " + i.toString());

i = i + 1;

}

# 28. Named collections Pythons

In Python, they **are called dictionaries** but some refer to named collections as *associative arrays* or *hashes*. Python programmers use the word *dictionary*. Consider this:

Instead of 0 base , it s possible dictionary based,

person = {"firstname": "Aleqa",

"lastname": "Lyberth",

"city": "Nuuk",

"country": "Greenland"}

print(person**["city"]**)

In Python, **arrays are wrapped between square brackets ([])**. **Dictionaries, on the other hand are wrapped between curly braces ({}).**

Each value in an array has an index position. **Dictionary have a *key***. We can chose the name of the keys like in the above example. Keys are regular strings enclosed in double quotes and followed by a **colon (:).** Directly after the colon, we set the value we want for a particular key. This is known as a *key-value pair*.

In a dictionary, much like in an array, key-value pairs are separated by a **comma (,).**

# Test named collection

bike = {"Type":"Scott hardtile","size":"XL","vitesses":11}

for key in bike:

    print(key,":",bike[key])

town = {

    "name": "Courroux",

    "elevation": 418,

    "population": 3200

}

for key in town:

    print(key, ":", town[key])

# 31. Named Objects in JavaScript

In Java Script Dictionaries are called a JavaScript object

var city = {name: "Nyon", country: "Switzerland", population: 11000}

console.log(city.country);

console.log(city["name"]);

 The name of the key is followed by a colon (:) and then by the value for the key.

 Key-value pairs are separated by a comma (,).

 Objects are enclosed within curly braces ({}).

To access the value of a particular key, we can use two different syntaxes.

**city.country**

**city["country"]**

**!!!!! var card = {...} !!!semi-colon not necessary when {} is used !!** **the curly brace means "a block of code"**

In JavaScript you can also **use quotes for the key names**, but it is usually for cases where the name would be invalid without the quotes. **That is the case of names with spacing or reserved JS words (like if, for, etc.):**

var card = {

"**first name**": "Laura",

age: 15,

email: "laura@extensionschool.ch"

}

console.log(card["first name"]);

# 02. Increment a variable

i = i + 1

We assign a new value to i. This new value is the old one plus 1: i + 1. Because this task is so common in programming, most languages provide a shortcut for this. Here it is:

**i += 1** **the same as i = i + 1**

for (i = 0; i < 5; i += 1)

See? Instead of using i = i + 1, we've used i += 1.

# 03. Getting a random number

In Python

import random // **Written at the top of our program !!!**

random\_number = random.randint(1, 100)

**Note:**  Meaning random.randint(1, 51) generates a number between 1 and 51 (including 1 and 51).

# Test random

import random

number = random.randint(1,51)

print(number)

## Random numbers in JavaScript

JavaScript has a random() function, it can only generate numbers between 0 and 1. 0.823745 for example.

If we want to generate a random integer number, we will first need to multiply this result by the maximum number that we want to generate and then we need to floor that result.

For example, getting a random number between 0 and 100 would be written like this:

var a = Math.random() \* **101**;

a = Math.floor(a);

// Random test in JAVA

var a = Math.random() \* 101 // Number between 0-100;

console.log(a);

a = Math.floor(a);

console.log(a);

55.74751800382377 [randomex.js:4:9](file:///D:\ES_Data\Code\Java\random\randomex.js)

55 [randomex.js:6:9](file:///D:\ES_Data\Code\Java\random\randomex.js)  !!! Math.floor() !!!

We could simplify it on one line like this:

var a = Math.floor(Math.random() \* 101);

// Random test in JAVA

var a = Math.floor(Math.random()\*101);

console.log(a)

Option to get a random integer between **two integers min and max** is the following:

var a = Math.floor(Math.random() \* (max - min)) + min;

Note that the maximum is **exclusive** and the minimum is **inclusive**.

// Random test in JAVA

var a = Math.floor(Math.random()\*(101-50))+50;

console.log(a)

// Random test in JAVA

var a = Math.floor(Math.random()\*(101-50))+50;

console.log(a);

for (i=0;i<5;i+=1){

    var x =(Math.floor(Math.random()\*(20-10))+10);

    console.log(x)

}

for (i=0;i<5;i+=1){

    console.log(Math.floor(Math.random()\*(20-10))+10);

}

# 04. True is true!

answer\_is\_correct = True

if answer\_is\_correct == True:

print("The answer is correct!")

*If true is true then…*

But in proper English, you would say:

*If it is true, then…*

**We could simply rewrite this example like this:**

answer\_is\_correct = True

if answer\_is\_correct:

print("The answer is correct!")

# 05. Catching errors

## The Python Try block

number = input("Type a number: ")

try:

result = int(number) + 1

except:

print("Couldn't convert your input to a valid number")

quit()

print("The result is " + str(result))

# 06. Getting user input with JavaScript

var number = window.prompt("Give me a number : ")

var color = window.prompt("Give me a color: ")