Fundamental notions of programming (NFP)

Class 1

Programming

What is programming

Programming consists into encoding an algorithm into a notation, series of command line arguments, using a programming language so a computer may perform different tasks, resolve certain problems, and fill certain needs.

Difference between programming and coding

Most of people confuses coding and programming and they tend to use both terms as synonyms. But coding and programming are two very different things.

Coding:
Coding means writing codes from one language to another, from the human language (concept/algorithm) to one a computer will able to understand. It is part of programming, but it is limited to a production level as coders are working based on instructions given to them about what the computer needs to accomplish. Although coding implies to master the language a coder uses, it is much easier to code than to program.

Programming:
Programming consists into analyzing a problem and developing a logical abstract system in order to enable a computer to resolve it. Programmers must take in account every possibilities an anticipate all problems in order for an application to run without any errors. Programming is the global aspect while coding is only a part of the programming process.

Software development

Software development is mostly associated with desktop applications and it encompasses the full cycle of developing a software product, from creating specification to testing and launch.

Web development

Most of people think web development consists into creating websites, which isn't entirely wrong. But web design is mostly graphic design and visual representation of information done by using markup languages like HTML and CSS.

Web development refers to web related programming. More and more websites act more like software, rather then a website in a traditional sense. They store data and interact with databases, they execute business logics, and process information in a more intricate way. Of course, those websites have a web interface which is the visible part of the iceberg, but it is not the most significant part in that sense that the hidden back-end part is the one really bringing a solution to a problematic, the one filling various needs.

Types of computer programming languages

Low level programming languages

Low level programming languages are machine dependent, such as *binary* or *assembly* languages. Since computers only understand binary language, those are the best, fastest and most direct way to give them instructions. But they are difficult to write, read, edit and understand.

Understanding the language used to program, computers then don't need any *interpreter* or *compiler* to convert the language. Although, *Assembly* language needs to be converted to binary code for the computers to understand. To do so, *Assembler* is used to convert the source code to its equivalent Binary code.

High level programming languages

High level programming languages are machine independent such as Java, .Net, Pascal, COBOL, C++, C, C#, etc. They are much easier to write, read, edit and understand.

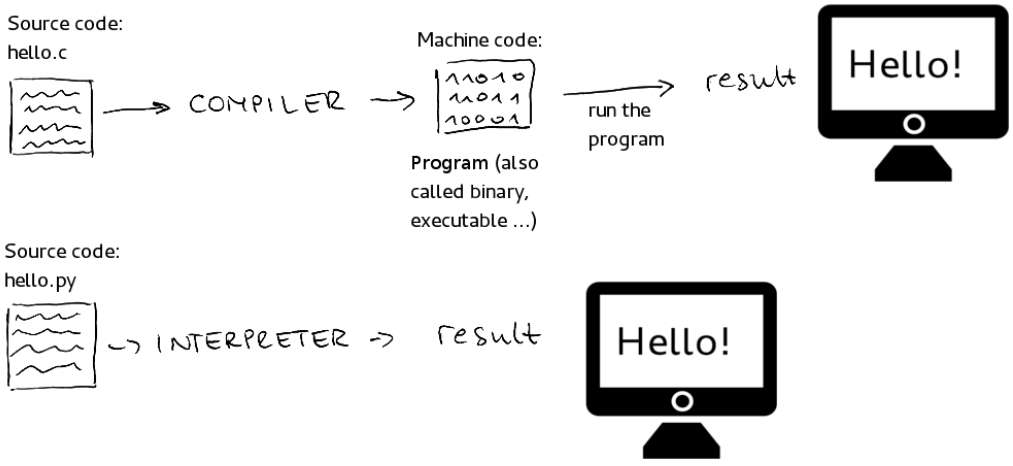
These languages are formed by special keywords, functions and class libraries so it can be easier to program computer applications. But since computers don't understand these languages made for humans, translator softwares must then be used to convert it to its equivalent Machine code. Those translators are named *compilers* and *Interpreters*.

Compiled and interpreted languages

Programming languages generally are considered *compiled* or *interpreted* languages.

With a ***compiled language***, the code is reduced to a group of computer instructions saved as an executable file and generally run faster than interpreted languages.

With ***interpreted languages***, the code is saved without being modified but generally run slower than compiled languages. Those are generally called script languages, such as JavaScript which is interpreted client-side by the web browser. Interpreted languages are easier to use as it does not require to compile the code every time it needs to be tested. Finally, it also allows to do things that can't be done with compiled languages. For instance, interpreted languages can modify themselves while being executed.



Popular languages in use

Python

Advanced interpreted and object-oriented programming language using flexible and solid semantics released in 1991. Easy to read and code, it is a free and open source language offering a large standard library, but its speed is limited, and it is weak when it comes to mobile computing and browsers, and design restrictions.

Java

Released in 1995, Java (not JavaScript) is a high-level object-oriented general-purpose programming language offering several features ideal for web-based development. It was developed according to the WORA principle (Write Once Run Anywhere).

Java is mainly used to develop enterprise-level applications for video games and mobile apps, and to create web-based applications with Java Server Pages (JSP). When used online, it allows applets to be downloaded and used through a browser so it can execute normally unavailable functions.

Ruby / Ruby on Rails

Released in 1995, Ruby is open-sourced, object-oriented scripting language that can be used independently or as part of the Ruby on Rails web framework which is excellent for quick application development an insure cost-effective web development and maintenance.

C

Released in 1972, C is a highly portable structure-oriented, middle-level programming language mostly used to develop low-level applications as those integrated into systems such as Windows (graphics packages, word processors, spreadsheets, operating system development, database systems, compilers and assemblers, network drivers, interpreters, etc).

Factors to be considered when choosing a programming language

With several types of programming languages, it can be difficult for a web developer to decide which one to choose. However, certain factor should be considered before making such an important decision :

Targeted platform

First ask yourself where will the program be run. All languages aren't capable of running on all platforms. For instance, a program written in C language requires compilers to run on Windows and Linux based systems.

Elasticity and Performance

The chosen programming language must be flexible enough to let you add more programs or features in it.

Availability of libraries

A library that is capable of solving all of your problems with the chosen language should be available.

Project size

There are large and small types of programming. You must select a language that can support the application you're building as well as the project size well.

Expressiveness and production time

Make sure to choose a language that is sufficiently expressive for your project's goals and avoid selecting a language that is uselessly complicated to use.

What is a software program?

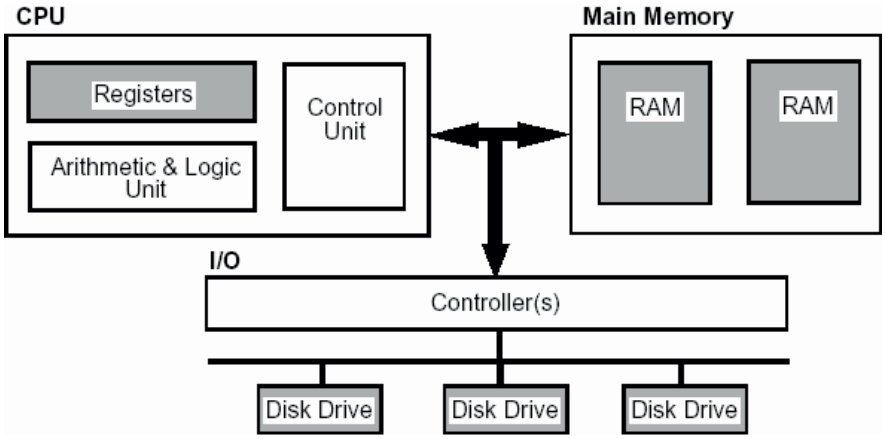
As it was priorly briefly explained, a computer is an assemblage of different components (hardware) that can process data.

To benefit from the computer's potential, to have it perform different tasks, the machine needs to be told what to do in the form of a set of instructions. These instructions are called software programs which are stored as files most commonly on the computer's the hard disk. When they need to run, the programs are loaded into RAM memory

So, for a program to be run, two major core components of a computer are needed : the central processing unit (CPU) and the random access memory (RAM).

CPU (Central Processing Unit)
Can be compared to the brain and it runs a set of instructions.

RAM (Random Access Memory)
Consists in a short term memory and it is used as a temporary storage to help the CPU perform its tasks.



Programming constructs

Sequence

A program consist into a set of instructions performed in a certain order. The programming construct is called a sequence when instructions are performed one after another. Let's take the following simple equation: A + B = C. If A = 100 and B = 200 and the computer is asked to addition A and B, the result (300) will be affected to C.

In a computer program, our values would be assigned to variables we would declare:

```
var A = 100
var B = 200
var C                    // No value assigned
```

Explanations

As we write *var A* or *var B*, spaces are allocated in RAM because the variables of a program reside in an actual physical location. A and B can be imagined as being rooms in a house (RAM) that can hold values that are assigned to them.

Space allocated to variable A **A = 100**
Space allocated to variable B **B = 200**

To calculate the value of A + B, we could write: A + B, and the operation takes place in the CPU. If the result is not stored anywhere, it is unaccessible and lost, that is why variable C has been declared, to hold the result. Our program would then take a form similar to the following:

```
var A = 100
var B = 200
var C                    // To hold the result
C = A + B                // Result of the operation stored in C
```

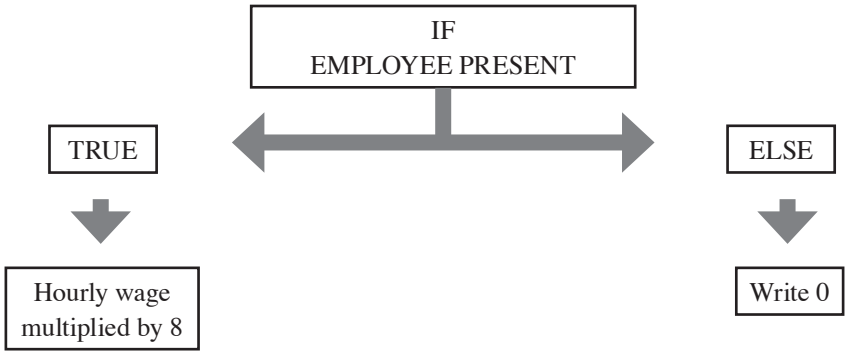
Space allocated to variable A **A = 100**
Space allocated to variable B **B = 200**
Space allocated to variable C **C = 300**

Programming constructs

Control structure (selection)

A control structure allows to create options to select from based on conditions. Depending if a condition is met (verified), the program will execute different instructions. This is commonly known as the IF/ELSE structure.

Example
Let's take the example of a business daily payroll system. If the employee is present, the program calculates the employee's hourly wage multiplied by 8 hours and the result will be the employee's daily gain. If the employee is absent, no calculation is needed and the daily gain is then zero.



In the example, the condition consists into verifying if the employee is present or not. If the condition is verified (true), the mathematical operation is performed, if the condition is not verified (false), zero is supplied and printed as a result.

The program could have the following aspect:

```
var WAGE = 20                    // 20$ per hour
var HOURS = 8                    // 8 hours a day
var GAIN                         // Result of operation

IF (employee = present)          // If condition true
        GAIN = WAGE * HOURS
        PRINT GAIN
ELSE                             // If condition false
        PRINT 0
```

Programming constructs

Repetition (loops)

A repetition (or loop) allows for a set of instructions to be repeated as long as a condition is verified. The most common structures are *while* and *for*.

While

The while structure executes a set of instruction as long as a condition (between parenthesis) is verified (true).

```
while (age < 18) {

        // you are minor
        // No wine for you!
}

        // Let's have a drink!
```

Explanation :
As long as the variable is ***true*** (age is inferior to 18), the refusal message is output. If or when the condition (between parenthesis) is or becomes ***false***, the loop stops and the program continues, in this case outputting the message « Let's have a drink!».

For

The for structure allows to count how many times the instructions of a loop are executed. This is useful to get the total count or to set a limit.

Example :
A program asks the user for their 5 favourite meals. The user enter the answers, and once 5 answers has been submitted, the loop stops and the five answers are output.

```
for (i=0; meals < 5; i++) {

        // i is the count value at the beginning
        // meals is a variable containing the answers
        // i++ increases i every time the loop is executed
}

        // Your 5 favourite meals are : ...
```

What is an algorithm?

We already know that programs are made of sets of instructions a computer needs to execute in order to perform a given task, to resolve a problem. But, in order to know what instructions a computer should be given, analyses must be conducted and programming must be planned. The result is the algorithm : a step by step list of things to be done in order to solve a problem.

An algorithm is basically made of 3 elements : an input, a process and a goal.

For instance, if my goal is to know the sum of two numbers, the two numbers need to be input so the computer can process them and output the result of the operation.

But Algorithms are much more complicated than this as they need to anticipate each steps and possibilities. They are like the recipes of a cookbook explaining precisely what to do.

Example : taking a bath

Turn on water tap

If water is too cold : add hot water

If water is too hot : add cold water

Let the water run until the bath tub is full

As the water level : get undressed

When totally naked : get in the bath tub

etc.

Assignment 1 : Create a basic algorithm

Doing just like it was done in the last example, create the algorithm necessary to cook spaghettis.

The be reviewed in group at the beginning of class 2.