**Start-Over Recap**

**\*QTJ: Question to Juan**

jueves, 7 de julio de 2022

4:08 p. m.

**Llamadas con Juan / Notas propias**

**Fundamentals - 2 Apr '20**

Procesador: "El que opera".

RAM: Donde se guarda la info para operar.

Un ejecutable (.exe) es un "Binario".

Runtime: El intérprete para comunicarse con el sistema operativo.

Sistema Operativo: Intérprete entre el "Ensamblador" y los controladores (Software).

e.g.: Un Web Browser es un "Runtime" de HTML, CSS, JS

".NET": La plataforma de C#.

**Video Call - 7 Apr '20**

For x in range (0,20): { Así se escribe un ciclo o bucle.

En Python, siempre que se trabaje con funciones, tienen que existir dos statements: "Def" & "Return", y se escribiría así:

**Def** "Alguna cosa"

& **Return** "Otra cosa"

**QTJ:** Si la función definida no tiene un "Return" se asume cómo un valor Nulo, es decir retorna "NULL"

R/ Functions always return something (at least None, when no return-statement was reached during execution and the end of the function is reached). From StackOverflow.

Python es un lenguaje indentado, lo que significa que los niveles los determino yo, pero también que tienen un significado.

**PEP:** Son como las normas "APA" pero en Python.

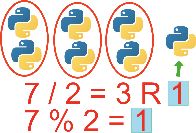
*PEP stands for Python Enhancement Proposal, and there are several of them. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.*

Why?!

*“Readability counts.”*

— *The Zen of Python*

El símbolo '%' significa 'Modulus', es un operador y entrega el residuo de una división



Una "Librería Estándar" de un lenguaje son las funciones básicas de cada lenguaje.

**Video Call - 14 Apr '20**

Lint: Buscar errores de sintaxis antes de ejecutar.

Shebang (#): The shebang line in any script determines the script's ability to be executed like a standalone executable without typing python beforehand in the terminal or when double clicking it in a file manager (when configured properly). It isn't necessary but generally put there so when someone sees the file opened in an editor, they immediately know what they're looking at. However, which shebang line you use *is* important.

**Correct** usage for (defaults to version 3.latest) **Python 3** scripts is:

#!/usr/bin/env python3

NT: Windows / Posix: Linux.

Variable de Entorno / Environment Variables: An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer. They are part of the environment in which a process runs. For example, a running process can query the value of the TEMP environment variable to discover a suitable location to store temporary files, or the HOME or USERPROFILE variable to find the directory structure owned by the user running the process. THEY SHOULD BE CALLE ALWAYS UPPERCASED

[What are Environment Variables, and how do I use them? (get,set)](https://www.youtube.com/watch?v=ADh_OFBfdEE)

Environment 
Variables 

Simple Array Sum Problem - HackerRank

**Import OS :** Esto me da herramientas para interactuar con el sistema operativo.

La sección " if \_\_name\_\_ == '\_\_main\_\_': " sacada de HackerRank se le llama "Name Guard" funciona principalmente para separar bloques de código que van a ser ejecutados si el módulo que contiene el Name Guard es la entrada principal del programa que estamos escribiendo. La Variable " \_\_name\_\_ " es una variable especial que utiliza Python para nombrar los módulos cuando se ejecutan, bien sea desde el mismo módulo o si se importa. Esto importa porque puede servir para ejecutar o no código que sirva en testeos o solo si son los módulos principales de un programa.

Aquí hay un articulo de StackOverflow que lo explica: <https://stackoverflow.com/questions/419163/what-does-if-name-main-do#:~:text=__name__%20is%20a,(as%20a%20str%20type).&text=the%20otherwise%20anonymous%20global%20namespace,to%20its%20__name__%20>.

Corey Schafer lo explica también: [Python Tutorial: if \_\_name\_\_ == '\_\_main\_\_'](https://www.youtube.com/watch?v=sugvnHA7ElY)

لاPإلاJ 
¯9LLlell¯ !J¯ 
ى1إا0للا 

**Cursos**

**Fundamentals of Programming - By A. Davis (2019) - 1 Apr '20**

Chapter #1

What's coding? Basically Instructions, which should be specific & sequential.

**Bug:** When something unexpected happen.

**Crash:** When the machine early stop of freezes.

* On programming languages Syntax (Rules) & Semantics (Meaning) are unique & specific.

* There are three ways to bring plain text or "Source Code" into machine language: to Compile it; to Interpret it; a combinations of both

Compiled languages are converted directly into machine code that the processor can execute. As a result, they tend to be faster and more efficient to execute than interpreted languages. They also give the developer more control over hardware aspects, like memory management and CPU usage.

Compiled languages need a “build” step – they need to be manually compiled first. You need to “rebuild” the program every time you need to make a change. In our hummus example, the entire translation is written before it gets to you. If the original author decides that he wants to use a different kind of olive oil, the entire recipe would need to be translated again and resent to you.

Examples of pure compiled languages are C, C++, Erlang, Haskell, Rust, and Go.

Interpreters run through a program line by line and execute each command. Here, if the author decides he wants to use a different kind of olive oil, he could scratch the old one out and add the new one. Your translator friend can then convey that change to you as it happens.

Interpreted languages were once significantly slower than compiled languages. But, with the development of just-in-time compilation, that gap is shrinking.

Examples of common interpreted languages are PHP, Ruby, Python, and JavaScript.

**\*QTJ:** Why both types of languages exists? Is it like one of those was thought first and later came he second?

Chapter #2

Nowadays the top 5 programming languages are: Java ; Python ; JavaScript ; C++ & C#

*"There are two ways to write error-free programs; only the third one works"*

* *Alan Pertis.*

Operators: in the expression "3 \* 2" ; 3 & 2 are Operators input, "\*" would be the operator.

Error Categories

Syntax: Languages rules broken.

Runtime: Unable to execute.

Semantic: Unexpected Output.

**First Dive into Software Dev, Agile & Scrum**

**Software Development (Dive) - 9 May '19**

Basic steps: Conceiving > Specifying > Designing > Programming > Documenting > Testing > Bug Fixing.

Main programming languages:

* Assembly
* COBOL
* C
* C++
* C#
* Java
* LISP
* Python

**Python Course - Coursera - 21 May '19**

\* IDE: Integrated Development Environment.

Program:

* Chapter #1 - Why we program?
* Chapter #2 - Using Python
* Chapter #3 - Writing Code's Paragraph
* Chapter #4 - Variables & Expressions
* Chapter #5 - Conditional Coding
* Chapter #6 - Functions
* Chapter #7 - Loops & Iterations

**Agile Principles & Methodologies - Skillsoft - 23 Mar '20**

Understanding Agile

There's two approaches: Linear & Iterative.

There are specific stages by iterations:

1. Requirements
2. Design
3. Develop
4. Testing
5. Delivery

When developing a project the 1st iteration should address the highest priorities valued by the stakeholders.

*"Money talks"*

Agile Values & Principles

Agile Manifesto => "Thinking Compass"

4 Values

1. Individuals & Interactions over Processes & Tools.
2. Working software over Comprehensive Documentation.
3. Customer Collaboration over Contract Negotiation.
4. Responding to change over Following a Plan.

12 Principles

1. Satisfy the customer
2. Welcome change
3. Deliver software frequently
4. Work together
5. Motivate individuals
6. Use face-to-face communication
7. Working software = Progress
8. Constant pace
9. Tech. Experience
10. Simplicity
11. Self-organizing
12. Reflection

Agile PM

1. Envisioning > 2. Speculating > 3. Exploring > 4. Adapting > 5. Closing

V

1. Code Dev. > 2. Software Dev. > 3. Test > 4. Feedback

SCRUM

Is a "Framework" more than a methodology.

Iterations = Sprints

Each sprint has a clear goal: The agreed set of items to be implemented on that iteration.

This is why the team should gather up prior to begin a sprint, to set goals and expectations.

Usually occur each 1 - 4 weeks, but is mostly common 2 weeks length sprints.

There are 4 opportunities to inspect and adapt:

1. Sprint Planning

This is the first meeting to coming with deliverables for the sprints.

1. Daily Sprint

Daily stand-up, typically 15m-long. Every member should answer 3 things: What have I done since last Scrum; What I am planning to finish until the next Scrum; Which are the roadblocks where I needed support.

1. Sprint Review

By the end of the sprint, the team show &demo to the product owner &/or the Stakeholders to make sure the alignment and to adapt.

1. Sprint Retrospective

Like a "Lesson Learned" session to understand "What went well" and what does not and what can be done to improve as a unit.