**Conceptos básicos de computación y programación.**

\*\*Algoritmo -> **descripción** de la solución de un problema, no ambigua, en una secuencia ordenada y finita de pasos. Algo que queremos resolver, (ej. Calcular bien las nóminas) el algoritmo me describe como lo puedo solucionar, una serie de pasos para resolver ese problema. En la solución de un problema, incluimos la información asociada, la información que lleva a ese problema, ya que el problema a veces es algo abstracto, es importante saber la información, y que esta información no tenga otras posibles interpretaciones, siendo **muy claro y conciso** (p.e. si hago referencia a la nómina del mes pasado, o de otra persona). Esto se debe traducir en una secuencia **ordenada.** Es una idea una explicación.

Codificación -> adaptar el algoritmo a un lenguaje, dando como resultado un programa. P.e., la codificación en Python, es lo mismo que programar, se trata de transformar en otro contexto u otro tipo de información. Se transforma esa idea o explicación del algoritmo al lenguaje de ordenador, para que todos interpreten lo mismo.

Programa -> conjunto de instrucciones comprensibles por un computador que permite ordenar la secuencia de pasos que resuelve un problema dado (usando información). Se le suele llamar **código fuente** (source). Lenguaje maquinal, es un concepto, es un algoritmo ya adaptado, transformado a un lenguaje de programación, es por ello que tiene propiedades muy parecidas. El código fuente son los algoritmos con los que se trabaja internamente y da lugar a las aplicaciones. Tras compilarlo, lo recibe la gente.

Computador -> máquina programable capaz de almacenar y ejecutar programas. Memoria + CPU + periféricos (pantalla, entrada/salida, comunicación del ordenador con el exterior) (I/O). CPU (unidad central de procesos, chip principal del ordenador, el que ejecuta los programas, es el procesador), hará lo que digamos. Sirve como almacenamiento.

S.O. -> conjunto de programas para facilitar el uso (ficheros, seguridad, herramientas) y controlar el funcionamiento de la máquina (HW: memoria, perif., asignación de recursos, scheduling …). Es un programa más, una aplicación ejecutable, o un conjunto de programas, especiales, es el que manda, y controla el ordenador, se encarga de la seguridad, es el que está más cerca del hadware.

IDE -> Entorno de desarrollo integrado: programa ejecutable que facilita la programación con uno o varios lenguajes (edición, compilación pasar de mi programa al programa ejecutable, depuración, ejecución, control de versiones …), es un programa más especializado.

Dato -> abstracción de la realidad, con un valor asociado, y que permite operar; con una representación interna determinada. Lo importante es la información sobre el objeto que queremos actuar, que representa algo más concreto.

Tipo de dato -> dominio de posibles valores + representación + operadores asociados. Suele haber unos predefinidos siempre, las operaciones más básicas. Ej. Números naturales vs enteros (o vs reales o complejos). En un principio siempre serán finitos, pero se hace una abstracción, para que en el momento de convertirlo en un dato se hacen unas simplificaciones, para poder operar con ellos. Trabajamos con operaciones ya definidas, o que definiremos nosotros.

*\*programa ejecutable = aplicación.*

Un módulo es un trozo de programa que se ha escrito antes.

**Intérpretes y compiladores**

\*\*Ejecución de un programa:

* Compilada: traducción previa a lenguaje máquina, generando un **programa ejecutable** o aplicación. La traducción del algoritmo al programa ejecutable (el de verdad). Lo normal es que los programas se ejecuten compilados, esto tiene un coste, es complejo, más óptimo, eficiente, etc. Es muy útil cuando el programa está rematado, es decir, ya está listo para usar. Traduce y lo hace todo a la vez.
* Interpretada: paso a paso (cada instrucción), usando un intérprete necesariamente (entorno de ejecución, el complemento que necesita el programa para funcionar). Se suele hablar de **scripts** (es un guion lo que yo sigo, lo que se va a ejecutar paso a paso). Es la política contraria a la compilación. Compila sobre la marcha, se ejecuta más despacio, primero traduce y después lo hace. No generamos un ejecutable. Ej. Pyhton es un intérprete, el entorno ejecución lo tendremos cuando lo instalemos.

Suele haber opciones “híbridas” (…). Por ejemplo, Java es un lenguaje que se ejecuta de las dos maneras, de esta manera es más rápida, ese interprete traduce sobre la marcha (compilando) y se lo guarda, generan un lenguaje semi media, y la máquina virtual. Esto lo hacen así, para poder ejecutarlo en “cualquier” entorno de ejecución. Estos híbridos, a veces se pueden compilar, como es el caso de Python.

Nosotros hacemos un lenguaje específico para el sistema operativo de nuestro ordenador.

Para que un programa sea eficiente, hay que optimizar los recursos, los recursos en este caso son tiempo de procesador y memoria.

MIT Course 6.0001 -> entender la parte formal, teórica, matemática, es público. Curso para empezar.

**¿Por qué Python?**

* Versatilidad (muchos ámbitos, en especial Data Science)
* Multi-plataforma
* Gran popularidad: documentación, ejemplos, ayuda …
* Multitud de bibliotecas disponibles
* Facilidad de aprendizaje, escritura y lectura
* Código conciso y compacto
* Lenguaje interpretado (a diferencia de los demás)
* Lenguaje de alto nivel (abstrae de muchos detalles)
* Rápido para muchas cosas

¿Qué es apache? Un servidor web, algo que me da un servicio en la web. Http es un protocolo, que es como habla el servidor y el cliente, el navegador es un cliente.

Sudo -> es como el super usuario, pero se pone delante para realizar sólo esa acción, es más recompensable que el su -, ya que evitas más riesgos, permite más cosas.

Debox, es un conjunto de prácticas.

Entorno de desarrollo integrado, Visual Studio Code,

**Ejercicios extra**

* Ver tutoriales como repaso de lo que hemos hecho hasta ahora.
* Jugar con los entornos.
* Estudiar diferencias entre Python 2 y Python 3-