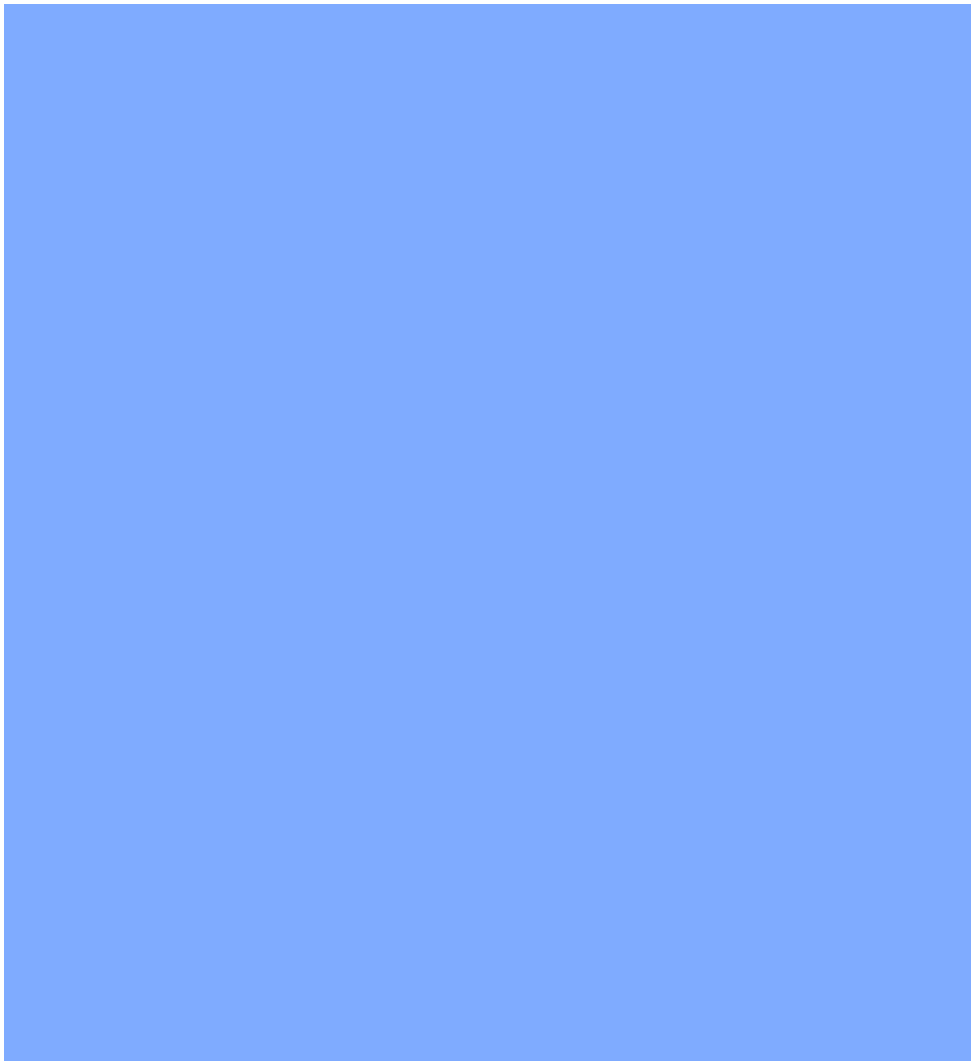
EDEM - Tech Knowledge



Introducción a la programación

Martín San José de Vicente

Director de Contenidos de Imagina Formación

OBJETIVO

Esta es la primera sesión y nuestro objetivo será el de realizar la primera toma de contacto con el mundo de la programación.

Esto quiere decir que empezaremos a entender en qué consiste la programación, la diferencia entre lenguajes de programación, los algoritmos y plantearemos una manera de resolver problemas previo a la programación que servirá para que te vayas haciendo a la idea de la forma de pensar que se tiene en el mundo de la programación.

ÍNDICE

- Programar. ¿En qué consiste y para qué sirve?
- Lenguajes de programación ¿Qué son?
- Fundamentos Básicos de la Programación
- Lenguajes de alto y bajo nivel ¿Hay diferencia?
- Lenguajes interpretados vs lenguajes compilados. El eterno debate.
- ¿Qué es eso de un algoritmo?
- Pseudocódigo como vía de iniciación

Programar. ¿En qué consiste y para qué sirve?

Picar código, tirar líneas, desarrollar...

¿Realmente en qué consiste?

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es un programa?

Una **secuencia de instrucciones que dictan lo que debe realizar el ordenador al ser convertidas en impulsos eléctricos.**

Los impulsos eléctricos actúan sobre elementos hardware que realizan operaciones, modifican la memoria o interactúan con la pantalla, ratón, teclado u otros ordenadores que estén en una red local o en la internet.

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es un programa?

Los programas se crean con la intención de automatizar tareas rutinarias, tareas con coste muy elevado o crear valor por otros medios a través de máquinas más potentes que el propio cuerpo humano.

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es un programa?

Los programas permiten que una computadora se convierta en, por ejemplo:

- una calculadora científica
- un sistema de mensajería
- en un sistema financiero
- en un reproductor de películas en streaming.
- ¡CUALQUIER COSA QUE PODAMOS IMAGINAR!

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es un programa?

Gracias a los programas, se consigue que un usuario pueda realizar tareas muy complejas, sin la necesidad de conocer cómo funciona internamente.

Incluso, actualmente, ni siquiera los programadores necesitan conocer demasiado acerca de cómo está construida la máquina* sobre la que va a correr el programa que está creando.

*Aunque es muy conveniente si se busca el uso eficiente de los recursos...

> Programar. ¿En qué consiste y para qué sirve?

¿En qué consiste programar?

Programar consiste, simple y llanamente, en conseguir decirle a una máquina que realice una determinada tarea.

Los conceptos que se aplican son los de la lógica y la matemática.

> Programar. ¿En qué consiste y para qué sirve?

¿En qué consiste programar?

Los programadores son capaces de argumentar la viabilidad o la eficiencia de un programa cuando crean y examinan los distintos bloques de código que lo conforman.

> Programar. ¿En qué consiste y para qué sirve?

¿En qué consiste programar?

A pesar de no necesitar un ordenador para llegar a entender el código de un programa, sí que es necesario para poder ejecutarlo.

El Software está escrito para se consumido por máquinas.

> Programar. ¿En qué consiste y para qué sirve?

¿En qué consiste programar?

Durante la programación, también llamado desarrollo de software, se emplean herramientas concretas como podrían ser **entornos de desarrollo**, **compiladores**, **intérpretes** o **editores** para crear los programas.

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es la Ingeniería del Software?

La Ingeniería del Software es la **ciencia** que estudia el **diseño** y la **creación** de **productos software** para que sean **EEERM**:

- **Eficaces**
- **Eficientes**
- **Extensibles**
- **Rentables**
- **Mantenibles** en el tiempo

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es la Ingeniería del Software?

Esta ingeniería se encarga de estudiar y ofrecer un **enfoque metodológico disciplinado**, **sistemático** y **cuantificable** al desarrollo y mantenimiento del software.

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es la Ingeniería del Software?

A pesar de que se suele emplear el término **ingeniero/a** de software para definir a los **programadores/desarrolladores**, sólo una parte de éstos cuentan con un **título universitario que lo acredite**.

> Programar. ¿En qué consiste y para qué sirve?

¿Qué es la Ingeniería del Software?

Podríamos diferenciar entre:

Desarrollo de Software

> Creación de Productos Software.

> Objetivo: Crear un producto que funcione.

Ingeniería del Software

> Estudio metodológico y sistematizado para la creación de Productos Software confiables y de calidad.

> Objetivo: Establecer los criterios y herramientas necesarias y crear un producto óptimo en todos los sentidos.

Lenguajes de Programación.

¿Qué son?

Aprendamos a comunicarnos con las máquinas.

> Lenguajes de programación. ¿Qué son?

¿Qué son los lenguajes de programación?

Son **lenguajes formales** que nos ofrecen la posibilidad de controlar una computadora.

A través de los lenguajes de programación, podemos crear secuencias de órdenes que nos permitan emplear la lógica y capacidad de cómputo de una máquina para crear soluciones a problemas humanos.

> Lenguajes de programación. ¿Qué son?

¿Qué son los lenguajes de programación?

Todos los lenguajes de programación disponen de sus propias reglas y estándares, pero en esencia todos ellos comparten elementos.

Disponen de **alfabeto**, **reglas sintácticas**, **reglas semánticas**, **convencionalismos, reglas léxicas**, **estándares**. A través de esto, se establecen las llamadas estructuras válidas de cada lenguaje.

> Lenguajes de programación. ¿Qué son?

¿Qué son los lenguajes de programación?

Estas reglas son empleadas por las distintas herramientas, que se usan para desarrollar, para validar las líneas de código.

> Lenguajes de programación. ¿Qué son?

¿Qué son los lenguajes de programación?

¡OJO!

Un lenguaje informático no tiene por qué se un lenguaje de programación.

Por ejemplo, **HTML** es un **Lenguaje de Marcado**, es decir, sirve para estructurar y codificar documentos.

Se trata de un **Lenguaje Informático**, pero **no un lenguaje de programación,** ya que no sirve para elaborar instrucciones para la máquina.

Tampoco serían Lenguajes de Programación, pero sí lenguajes informáticos: XML, **XAML**, **YAML**, **EBML**, etc

> Lenguajes de programación. ¿Qué son?

¿Qué son los lenguajes de programación?

Los lenguajes de programación buscan establecer un **lenguaje común** para que toda persona que desarrolla software pueda llegar a entenderlo.

Estos lenguajes, actualmente, se acercan al lenguaje humano. Es decir, se acercan a como hablamos "habitualmente",

pero no siempre ha sido así...

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes

Como cualquier habla humana, la **evolución** de los lenguajes de programación no se ha detenido desde que se creó el primero microprocesador capaz de facilitar la comunicación entre máquina y humano.

Históricamente, se habla de que han existido, hasta la fecha: **5 GENERACIONES**.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 1ª Gen

Fueron los **primeros lenguajes de la historia**. Sus instrucciones accedían directamente al control del **hardware** de la computadora.

Eran lenguajes muy **poco inteligibles**, ya que apenas se asemejaban a ningún lenguaje humanos.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 1ª Gen

Las primeras programaciones de la historia se realizaban en código máquina (**código binario**).

A esta generación de lenguajes se les conoce ahora como lenguajes de **Bajo Nivel**.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 2ª Gen

A estos lenguajes se les conoce como los lenguajes de **programación simbólica**.

Permitían que el planteamiento de las **instrucciones** fueran **más sencillas y comprensibles para el humano** (aunque no demasiado).

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 2ª Gen

Se trata de una serie de instrucciones muy potentes que equivalen a varias instrucciones de **Lenguaje Máquina** (del que hablaremos en el siguiente punto)**.**

Básicamente son **instrucciones legibles** que se traducen a **Lenguaje Máquina** y que **luego el CPU ejecuta**.

A estos lenguajes se les llama: **Lenguajes Ensamblador**.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 3ª Gen

A estos lenguajes se les conoce como los **lenguajes de Medio y Alto Nivel**.

Incorporan instrucciones más parecidas al lenguaje humano y **se alejan de las instrucciones máquina**.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 3ª Gen

Los primeros lenguajes de esta generación son actualmente considerados como lenguajes de **Medio Nivel**, ya que su abstracción no es tan grande como para considerarlos de alto nivel.

Estos lenguajes pueden emplearse para la creación de cualquier programa, dado que dejan de ser de un único propósito (manejar el ordenador y sus piezas) para ser de **propósito general o amplio**.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 4ª Gen

Los lenguajes de la cuarta generación **son lenguajes de alto nivel** que **se asemejan aún más al lenguaje humano** que los de 3ª generación.

Creados para **reducir el tiempo de desarrollo** y para **minimizar el coste y esfuerzo** del desarrollo.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: 4ª Gen

Estos lenguajes, en su gran mayoría, pertenecen a las familias de las: **Bases de Datos**, **Análisis y Manipulación de Datos**, **Generación de Informes**, **Desarrollo Web** y **Móvil** y otros **GUI***.

***(GUI)Graphical User Interface:** Software que crea Interfaces de Usuario para mostrar información, mediante el uso de recursos gráficos como imágenes, textos y formas.

> Lenguajes de programación. ¿Qué son?

Las generaciones históricas de los lenguajes: ¿5ª Gen?

No es una denominación oficial, pero durante mucho tiempo se habló de los lenguajes de 5ª Generación para denominar a todos aquellos lenguajes relacionados con la **IA*** y **ML****.

A día de hoy, este concepto a penas se escucha o lee, a pesar de que **la época dorada de la IA y el ML no ha hecho más que empezar**.

***(IA)Inteligencia Artificial:** Desarrollos que permiten que una máquina "imite" las funciones cognitivas de un humano.
****(ML)Machine Learning:** Es una rama de la inteligencia artificial que se centra en desarrollos que permiten a una máquina aprender y mejorarse a sí misma a través de la experiencia.

Fundamentos Básicos de la Programación

¿Existe alguna relación entre todos los lenguajes?

> Fundamentos Básicos de la Programación

Breve Introducción a los Elementos de los Lenguajes

Todo lenguaje de programación cuneta con una serie de elementos comunes:

- Sintaxis
- Tipos de datos
- Variables
- Identificadores
- Comentarios
- Funciones
- Directivas
- Condicionales y Bucles
- Palabras Reservadas
- Operadores
- Signos de Puntuación
- Separadores
- Expresiones
- Errores de diversos tipos
- Depuración

La Sintaxis en un Lenguaje de Programación

Se trata de las **reglas** que establecen los criterios para construir y secuenciar los elementos del lenguaje de programación.

Los errores sintácticos causarán errores en el código. A pesar de que cada lenguaje cuenta con una sintaxis propia, si conoces un lenguaje de programación con fluidez, pasar a desarrollar en otro resulta muy sencillo, pues todos comparten bases comunes que iremos viendo durante el curso.

La Semántica en un Lenguaje de Programación

Establece el **significado** a cada uno de los elementos del lenguaje.

Lenguajes Alto y Bajo Nivel. ¿Hay Diferencias?

Cada día es más fácil entendernos con las máquinas y que éstas entiendan lo que queremos que hagan.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel

Los lenguajes de bajo nivel son aquellos cuyo **nivel de abstracción es mínimo**.

Actualmente **se siguen usando**, en menor medida y a pesar de su complejidad a la hora de plantear la lógica en sus instrucciones, a la hora de programar determinadas tareas para **sistemas operativos***.

***Sistema Operativo:** Conjunto de programas y órdenes que controla los procesos básicos de una computadora. Además, permiten que otros programas funcionen en la misma. (Ej: Windows, Mac, Distribuciones de Linux como Ubuntu, etc.)

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Lenguaje Máquina

El lenguaje máquina es el Lenguaje de Programación de **más bajo nivel que existe**.

Está compuesto por **0s y 1s** que se ejecutan directamente por parte de la **CPU*.**

***CPU:** Central Processing Unit - Unidad Central de Procesamiento. Parte esencial de la computadora que se encarga de interpretar y ejecutar las instrucciones de un programa.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Lenguaje Máquina

Es el más alejado del lenguaje humano y **el más cercano al lenguaje de hablaría una máquina**.

Estas combinaciones de **0s y 1s** están basados en el sistema binario, en el que **1 = pasa la electricidad** y **0 = no pasa la electricidad**.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Se trata de un sistema de codificación que se utiliza para representar textos (cadenas de caracteres), números, etc.

Cada número, letra u otros símbolos son representados por un número binario (**0s y 1s**) de una **longitud fija**.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Por ejemplo:

M	A	R	T	I	N
01001101	01100001	01110010	01110100	01101001	01101110

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Entender cómo funciona el código binario para los valores numéricos es especialmente fácil. Tienes que conocer las potencias de 2:

- 2^0 = **1**
- 2^1 = **2**
- 2^2 = **4**
- 2^3 = **8**
- 2^4 = **16**
- 2^5 = **32**
- 2^6 = **64**
- 2^7 = **128**
- 2^8 = **256**
- 2^9 = **512**
- 2^10 = **1024**
- etc.

En el mundo de la informática, el hecho de hacer uso de un sistema binario, hace que, en este mundo, contemos siempre con unidades de potencia de 2.

De ahí que haya , por ejemplo, memorias externas (USBs) de 1GB, 2GB, 4GB 8GB o 16GB, pero nunca de 3 GB o de 127GB.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Para construir los números binarios, simplemente debes poner un **1 en la posición que quieres que sume** y un **0 donde no quieres que sume** para llegar al total esperado.

Las posiciones se cuentan **desde la derecha a la izquierda** y comienzan en **0**.

Es decir, poner un **1** en la **posición 0** sería realizar **2^0 = 1**

Mientras que poner un **1** en la **posición 3** sería realizar **2^3 = 8**

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Posición N	Posición 2	Posición 1	Posición 0
2^N	2^2	2^1	2^0
	4	2	1

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Pongamos un ejemplo: Vamos a sacar el número **67**

Posición 7	Posición 6	Posición 5	Posición 4	Posición 3	Posición 2	Posición 1	Posición 0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	**1**	**0**	**0**	**0**	**0**	**1**	**1**

$$64 + 2 + 1 = 67 = 0100011$$

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Bajo Nivel: El Código Binario

Te planteo un ejercicio...

¿Sabrías calcular tu edad en binario?

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Medio Nivel:

Son los lenguajes cuya abstracción sobre el lenguaje máquina es **suficiente como para que un humano pueda comprenderlo "sin demasiadas" complicaciones**.

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Lenguajes de Alto Nivel:

Son los lenguajes cuya abstracción sobre el lenguaje máquina bastante grande y **cualquier humano podría llegar comprenderlo "sin demasiadas" complicaciones.**

> Lenguajes de Alto y Bajo Nivel ¿Hay Diferencias?

Ejemplos de Lenguajes según su nivel

Bajo Nivel	Medio/Bajo Nivel	Medio/Alto Nivel	Alto Nivel
Ensamblador	C	C++	Java
	Basic	Fortran	Python
		Cobol	C#
		Lisp	JavaScript
		Pascal	TypeScript
			Dart
			Go
			Kotlin
			Swift
			Rust
			SQL
			PHP
			Visual Basic
			etc.

Lenguajes Interpretados vs Lenguajes Compilados.

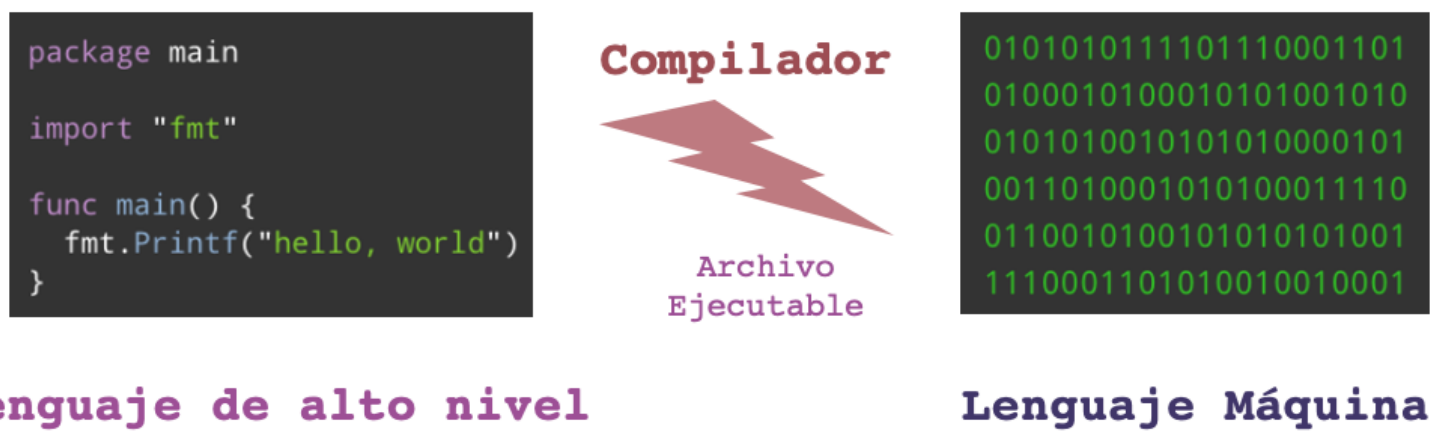
El eterno debate

Hay que escoger un lenguaje, pero ¿de qué tipo?

> Lenguajes Interpretados vs Compilados

Lenguajes Compilados

Son **lenguajes de alto nivel** que necesitan de otro programa (llamado **compilador**) que se encarga de **traducir el código de alto nivel a lenguaje máquina** y así poder ejecutarlo.



> Lenguajes Interpretados vs Compilados

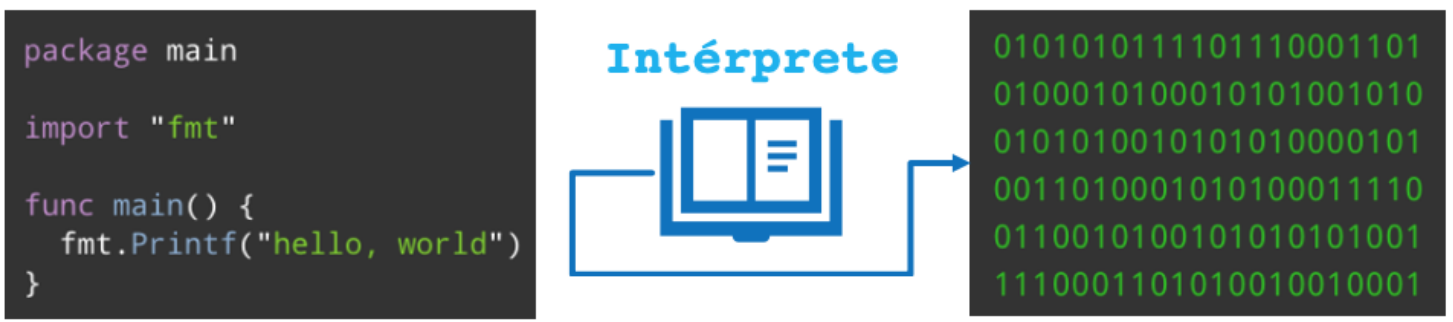
Lenguajes Compilados

Algunos ejemplos de lenguajes compilados. ¿Reconoces alguno?

> Lenguajes Interpretados vs Compilados

Lenguajes Interpretados

Un lenguaje interpretado es aquel lenguaje de alto nivel cuyo código es traducido directamente a lenguaje máquina.



> Lenguajes Interpretados vs Compilados

Lenguajes Interpretados

Algunos ejemplos de lenguajes interpretados. Uno de ellos es el que vamos a usar durante los siguientes módulos del curso:

> Lenguajes Interpretados vs Compilados

Principal Diferencia entre Compilados e Interpretados

-El lenguaje **COMPILADO** requiere del proceso de compilación para convertirlo a máquina y después poderse ejecutar.

-EL lenguaje **COMPILADO** está optimizado para el momento de ejecución.

-El lenguaje **INTERPRETADO** es convertido a lenguaje máquina mientras se ejecuta.

-El lenguaje **INTERPRETADO** está optimizado para el desarrollo.

> Lenguajes Interpretados vs Compilados

Ventajas y Desventajas entre ambos

Ventajas Lenguajes Compilados

> Son lenguajes **mucho más rápidos**, ya que **el ejecutable ya está en lenguaje máquina y no necesita ser traducido** a medida que se va ejecutando.

> Lenguajes Interpretados vs Compilados

Ventajas y Desventajas entre ambos

Desventajas Lenguajes Compilados:

> Cada vez que se cambia el código, **se debe realizar el proceso de compilación**, que a pesar de poderse automatizar, es más lento para poder probarse.

> Los lenguajes compilados **exigen la creación de un archivo ejecutable que va íntimamente ligado con el Sistema Operativo** para el que se ha creado.

> Lenguajes Interpretados vs Compilados

Ventajas y Desventajas entre ambos

Ventajas Lenguajes interpretados:

> El **tiempo** entre la **escritura de código** y **poder probarlo** es **mucho menor**.

> Lenguajes Interpretados vs Compilados

Ventajas y Desventajas entre ambos

Desventajas Lenguajes interpretados:

> La **velocidad de ejecución es menor**, ya que **necesita ser traducido** a Lenguaje Máquina a medida que se va ejecutando.

> Se **necesita tener un intérprete instalado** para poder ejecutar el programa

> Lenguajes Interpretados vs Compilados

La Particularidad de JAVA

Mientras que casi todos los lenguajes de programación actuales son compilados o interpretados, Java supone un curioso caso.

Java es un lenguaje compilado, pero es compilado a un lenguaje intermedio llamado **ByteCode**.

> Lenguajes Interpretados vs Compilados

La Particularidad de JAVA

Éste después es **interpretado cuando se ejecuta y traducido a Lenguaje Máquina.**

> La razón detrás de esta extraña decisión:

Los creadores de Java querían que Java fuera un lenguaje compilado, pero **que se pudiera ejecutar en cualquier Sistema Operativo** o Procesador sin tener que crear ejecutables distintos para cada uno.

> Lenguajes Interpretados vs Compilados

La Particularidad de JAVA

Por esta misma razón, cuando se quiere desarrollar en Java, se tienen que realizar dos instalaciones en el equipo indispensables:

> JDK de Java (Java Development Kit)

Incluye, entre muchas otras cosas, el **compilador** de Java a **ByteCode**.

> JRE (Java Runtime Environment)

Es el software encargado de **interpretar** el código Bytecode creado.

¿Qué es eso de un algoritmo?

El proceso reiterativo para obtener la solución de un problema.

> ¿Qué es eso de un algoritmo?

¿Qué es un algoritmo?

Un Algoritmo es la **secuencia de pasos que hay que realizar para poder resolver un problema**.

En el mundo de la programación, **todo programa está lleno de algoritmos**.

> ¿Qué es eso de un algoritmo?

¿Qué es un algoritmo?

Los algoritmos pueden realizarse fuera de los lenguajes de programación. Es más, lo ideal es tratar de pensar:

1. ¿Cuál es el problema?
2. ¿Cuál es la solución que busco?
3. ¿Qué pasos debería realizar para llegar a esa solución?
4. ¿Hay más de una alternativa para llegar a la solución?
5. ¿Qué alternativa es la más eficiente?
6. ¿En qué lenguaje de programación puedo crearlo y probarlo?
7. ¿Cómo depuro los problemas que vayan surgiendo al probar el algoritmo?

> ¿Qué es eso de un algoritmo?

¿Cómo idear un Algoritmo?

Existen diversas herramientas que nos pueden ayudar a plantear respuestas a todas esas preguntas anteriores:

> Herramientas Escritas:

- Pseudocódigo

> Herramientas Gráficas:

- Diagramas de Flujo
- UML*

***(UML)Unified Modified Language**: Lenguaje Unificado Modulado - Lenguaje Gráfico para visualizar, documentar, construir y especificar un sistema o algoritmo.

> ¿Qué es eso de un algoritmo?

¿Cómo idear un Algoritmo?

También hay **técnicas estandarizadas** para crear algoritmos de calidad. Destacaríamos las siguientes:

> Divide y vencerás (DYV)

Basado en la resolución recursiva de un problema a través de dividirlo en problemas cada vez más pequeños y fáciles de resolver.

> ¿Qué es eso de un algoritmo?

¿Cómo idear un Algoritmo?

También hay **técnicas estandarizadas** para crear algoritmos de calidad. Destacaríamos las siguientes:

> Algoritmos voraces o golosos

Estrategia de búsqueda que sigue una heurística para encontrar la opción óptima en cada paso para llegar a una solución general óptima.

> ¿Qué es eso de un algoritmo?

¿Cómo idear un Algoritmo?

También hay **técnicas estandarizadas** para crear algoritmos de calidad. Destacaríamos las siguientes:

> Búsqueda Exhaustiva

Método de fuerza bruta que evalúa todas las posibles opciones para ver cuál es la más óptima.

> Programación Dinámica

Método para reducir el tiempo de ejecución del algoritmo.

> ¿Qué es eso de un algoritmo?

¿Cómo evaluar un Algoritmo?

Para evaluar un Algoritmo podemos basarnos en:

- Si obtiene una **solución**
- Si la solución es la **óptima**
- Para esto, es necesario calcular su **complejidad computacional**

Pseudocódigo como vía de iniciación

¿Por dónde empiezo a programar?

> Pseudocódigo como vía de iniciación

¿Qué es el pseudocódigo?

El Pseudocódigo es un **Lenguaje de Descripción Algorítmico** que se emplea para:

> Plantear y describir algoritmos

Se tratan de descripciones compactas a alto nivel, es decir: con lenguaje muy cercano al humano.

Aún así, emplea convenciones de los lenguajes de programación, ya que a continuación, se buscará la implementación en alguno de estos lenguajes.

> Pseudocódigo como vía de iniciación

¿Qué es el pseudocódigo?

Se trata de un **proceso bastante informal que busca estructurar el problema existente** y el **encontrar los pasos necesarios para resolverlo** de la mejor manera posible.

> Pseudocódigo como vía de iniciación

Primer ejemplo de Pseudocódigo

Plantear en Pseudocódigo un algoritmo que calcule el **área de un triángulo** al recibir el tamaño de la **base** y la **altura** del mismo como entrada del usuario:

```
Inicio
Variables: area_triangulo, base_triangulo, altura_triangulo

Imprimir "Hola Usuario, para calcular el área debes introducir…"
Imprimir "La base del triángulo"
Leer base_triangulo
Imprimir "La altura del triángulo"
Leer altura_triangulo
area_triangulo = (base_triangulo * altura_triangulo) / 2
Imprimir "El Área de tu triángulo es: " + area_triangulo

FIN
```

> Pseudocódigo como vía de iniciación

Primer ejemplo de Pseudocódigo

Plantear en Pseudocódigo la **lectura de 3 números e indicar cuál de ellos es el mayor.**

```
Inicio
Variables a, b, c = enteros.
Imprimir ”Hola Usuario. Introduce 3 números para saber cuál es el mayor: "
Leer a, b, c
SI a > b y a > c
   entonces
   Imprime "El mayor es: ", a
SINO
   SI b > a y b > c
     ENTONCES
     Imprime "El mayor es: ” + b
   SINO
     Imprime "El mayor es: ” + c
Fin
```