

## ENTORNOS DE DESARROLLO

### UNIDAD 1. DESARROLLO DE SOFTWARE

#### 1.1. CONCEPTOS BÁSICOS Y LENGUAJES DE PROGRAMACIÓN

Departamento de Informática  
Raúl Palao

**Cicles  
Formatius**



## ÍNDEX

<b>1 CONCEPTOS BÁSICOS .....</b>	<b>3</b>
1.1 Informática, binario y software .....	3
1.2 Programar vs. desarrollar software .....	4
1.3 Programa, librería, aplicación y suite .....	5
<b>2 LENGUAJES DE PROGRAMACIÓN .....</b>	<b>6</b>
2.1 Tipos de código (fuente, objeto, ejecutable y máquina) .....	6
2.2 Ejecutables portables y scripts .....	8
2.3 Máquinas virtuales .....	9
2.4 Clasificación de los lenguajes .....	9
2.5 ¿Cómo escoger un lenguaje de programación? .....	13
<b>3 BIBLIOGRAFÍA .....</b>	<b>14</b>

# 1 CONCEPTOS BÁSICOS

## 1.1 Informática, binario y software

La palabra **informática** viene de “información” y “automática”. La información es un conjunto de datos, y automático es que funciona por sí mismo. Así pues, la informática es el proceso de la información y su tratamiento, mediante un sistema automático, como puede ser un “ordenador”.

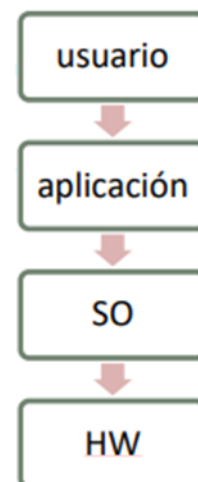


La informática permite automatizar el procesamiento de información. Toda la información que usa un “ordenador” debe estar representada en una secuencia de ceros y unos (dígitos) llamada código binario. Esto significa que cada unidad básica de información puede tener dos estados (0, 1). A dicha unidad se le conoce como bit o binary digit (b) y a la unión de ocho bits se le llama byte (B). Un byte puede representar 28 caracteres, es decir, 256 posibilidades distintas.

Cuando un dispositivo cualquiera maneja o almacena información en forma de bits (dígitos) decimos que ese dispositivo es digital. Cuando la información que maneja no es finita, como un reloj clásico o un termómetro de mercurio, decimos que es analógico.

El software es la parte intangible de un sistema informático, el equivalente al equipamiento lógico. Todo software está diseñado para realizar una tarea determinada en nuestro sistema y está presente en el sistema operativo (SO), en las aplicaciones que utilizamos y en, prácticamente, cualquier parte de un dispositivo electrónico moderno.

El software suele ser el traductor entre el hombre y la máquina. Es el encargado de comunicarse con el hardware, es decir, se encarga de traducir todas las órdenes que el usuario comunica a El software suele ser el traductor



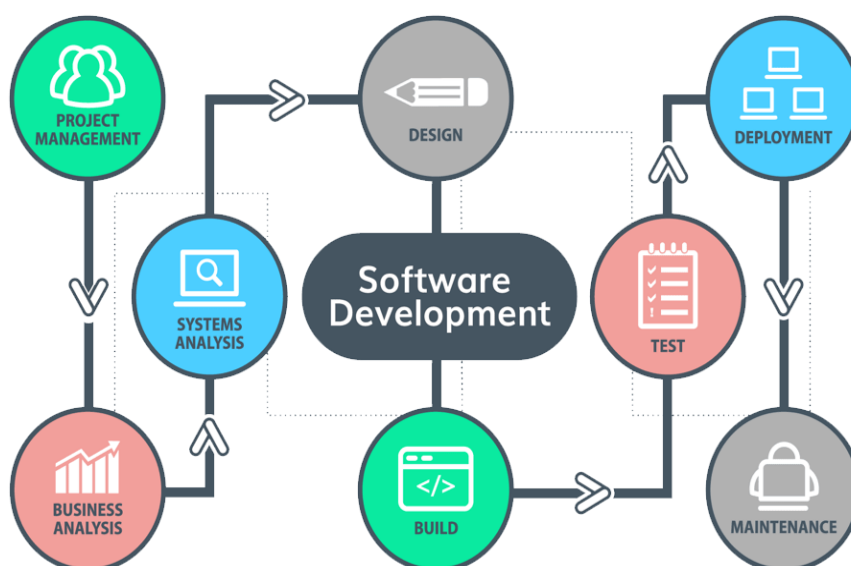
entre el hombre y la máquina. Es el encargado de comunicarse con el **hardware**, es decir, se encarga de traducir todas las órdenes que el usuario comunica a órdenes comprensibles por el hardware (p. ej. el controlador de una impresora).

## 1.2 Programar vs. desarrollar software

Verás muchas definiciones de lo que es programar, pero en esencia, consiste en darle órdenes y datos a un ordenador para recibir una serie de resultados o provocar cierto comportamiento. Para ser técnicamente correctos, usaremos dispositivo en lugar de ordenador e instrucción en lugar de orden.

***Es importante usar la palabra “dispositivo”, ya que hoy en día podemos programar desde un ordenador hasta una nevera, pasando por unas zapatillas deportivas o, por qué no, hasta una alfombra si estos manejan información digital. Cada vez más elementos de nuestra vida cotidiana aceptan órdenes en forma de comandos, líneas de código o programas, dentro de lo que se conoce como Internet de las Cosas, Internet of Things o con sus siglas en inglés IoT, un concepto que evoluciona del concepto clásico de domótica.***

Desarrollar software es o, al menos, debería ser mucho más que darle órdenes a un dispositivo, mucho más que programar. De hecho, el tiempo que dedicamos, o que deberíamos dedicar, a analizar, diseñar, probar, documentar y mantener el software es mucho mayor que el tiempo que dedicamos exclusivamente a escribir líneas de código. Preferimos decir desarrollador de software (o simplemente desarrollador) en lugar de programador, si consideramos que nuestra labor se basa en “algo más” que codificar.



### 1.3 Programa, librería, aplicación y suite

Del mismo modo que verás muchas definiciones y usos del término programar también verás muchas interpretaciones de los términos programa y aplicación.

Podríamos decir que un programa es una serie de órdenes instrucciones secuenciadas u ordenadas con una finalidad concreta y que devuelven un valor o realizan una función determinada. Por ejemplo, una función que dados números te devuelva la suma o un procedimiento que minimiza las ventanas de tu escritorio.

Por otro lado, una librería es un archivo, o contenedor lógico, que contiene una serie de programas. Por ejemplo, cualquiera de los archivos DLL que encuentras en la carpeta System32 de Windows.

***Nuestro código fuente puede usar funciones de cientos de líneas de código que otras personas ya han escrito y que están almacenadas en archivos externos llamados librerías. Imagina que estás programando una calculadora científica que necesita realizar cálculos complejos. Tú escribirás el código fuente de la calculadora y le dirás (ya verás cómo hacerlo) que utilice determinadas funciones almacenadas en determinados archivos.***

A su vez, una aplicación está formada por varios programas con sus librerías correspondientes, aunque podrían constar solamente de un programa.

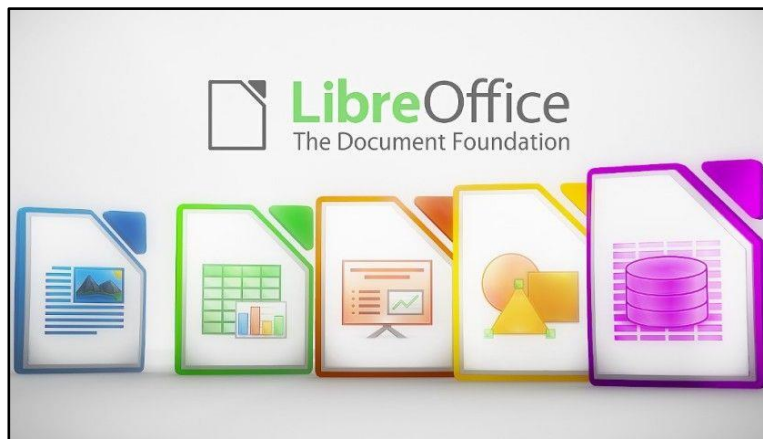
Por último, y con esto ya acabamos, cuando lo que tenemos son varias aplicaciones que pueden ejecutarse independientemente, una de otra, suele denominarse suite o “paquete integrado”, como por ejemplo nos sucede con MS Office o con Libre y Open Office.

Del mismo modo que verás muchas definiciones y usos del término programar, también verás muchas interpretaciones de los términos programa y aplicación.

Podríamos decir que un programa es una serie de instrucciones secuenciadas u ordenadas con una finalidad concreta y que devuelven un valor o realizan una función determinada. Por ejemplo, una función que dados números te devuelva la suma o un procedimiento que minimiza las ventanas de tu escritorio.

Por otro lado, una librería es un archivo, o contenedor lógico, que contiene una serie de programas. Por ejemplo, cualquiera de los archivos DLL que encuentras en la carpeta System32 de Windows.





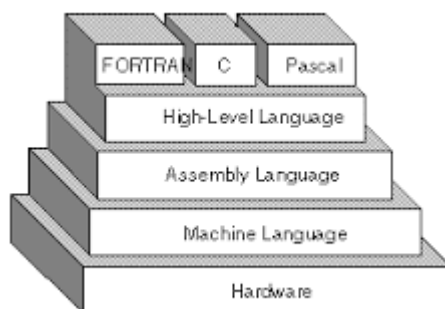
## 2 LENGUAJES DE PROGRAMACIÓN

## 2.1 Tipos de código (fuente, objeto, ejecutable y máquina)

Los dispositivos solo entienden un lenguaje muy complejo llamado lenguaje máquina y nadie, en su sano juicio, sabe/quiere escribir código en ese lenguaje. Para facilitarnos la labor a los desarrolladores existen unos lenguajes intermedios que, mediante unos traductores llamados compiladores, nos ayudan a comunicarnos con esos dispositivos.



Por tanto, todos los programas se desarrollan en algún lenguaje de programación. Los lenguajes de programación son, por lo tanto, lenguajes artificiales creados para que, al traducirse a código máquina, cada una de las instrucciones de dicho lenguaje dé lugar a una o a varias instrucciones máquina.



### Código fuente

- Conjunto de instrucciones escritas en un lenguaje de programación

Al conjunto de TODAS las instrucciones que forman un programa le llamaremos código fuente. En el caso de java, el código fuente se guarda en un fichero con la extensión “.java”.

### Código objeto

- El resultado de compilar el código fuente (bytecode/c.máquina)

Como hemos dicho, tenemos que transformar el código fuente de nuestro programa al código que entienda nuestro hardware. Esto se realiza con los compiladores, que generan una especie de código intermedio al que llamamos código objeto, bytecode o código intermedio. En el caso de Java, el compilador genera un archivo de extensión “.class”.

### Código ejecutable

- El resultado final de entrelazar los c.objetos + librerías
- Programa ejecutable

Si a estos objetos les añadimos las funciones que hemos usado de librerías y las peculiaridades del sistema operativo en el que se va a ejecutar obtenemos el código

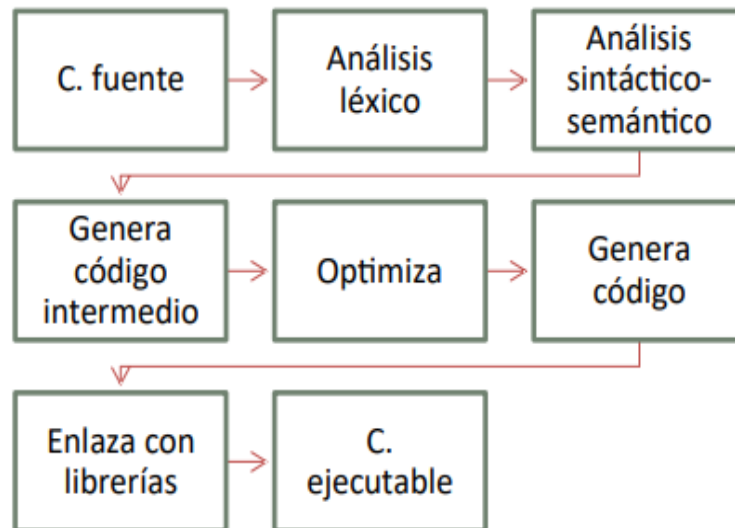
ejecutable, que es el conjunto de órdenes instrucciones directamente interpretable por el dispositivo que estamos usando. Suele usarse un enlazador.

Piensa que, un mismo código fuente, para poder ser ejecutado en diferentes sistemas operativos necesita ser compilado de varias maneras. Por ejemplo, la calculadora de la que hablamos puede que tenga un único código fuente, pero seguro que tendrá diferentes ejecutables, uno por cada sistema operativo en el que quieras que funcione.

***Nuestro código fuente puede usar funciones de cientos de líneas de código que otras personas ya han escrito y que están almacenadas en archivos externos llamados librerías. Imagina que estás programando una calculadora científica que necesita realizar cálculos complejos. Tú escribirás el código fuente de la calculadora y le dirás (ya verás cómo hacerlo) que utilice determinadas funciones almacenadas en determinados archivos.***



El esquema de traducción que sigue el compilador es el que se muestra en la siguiente figura:



*Verás que se usa de manera indistinta los términos plataforma, arquitectura, máquina, procesador y sistema operativo cuando hablamos de que una aplicación determinada está compilada para poder “correr” (ser ejecutada) en diferentes entornos. Ten en cuenta que, en la mayoría de los casos, estos términos son sinónimos.*

## 2.2 Ejecutables portables y scripts

Sobre lo que te acabamos de contar, existen varias peculiaridades.

Hay distintos tipos de archivos ejecutables, dependiendo cómo se traten las órdenes instrucciones:

- **Ejecutables portables.** Se pueden ejecutar en varias plataformas. Por ejemplo, un ejecutable Java es portable ya que utiliza un bytecode no asociado a un procesador en concreto.
- **Ejecutables no portables.** Destinado a una plataforma concreta. Por ejemplo, un ejecutable en C.

Sin embargo, en un sentido más general, un programa ejecutable no tiene por qué necesariamente contener código máquina, sino que puede tener instrucciones a interpretar por otro programa. Este tipo de ejecutables son conocidos con el nombre de scripts. Ejemplo: Script en bash de Linux o un script en PHP.

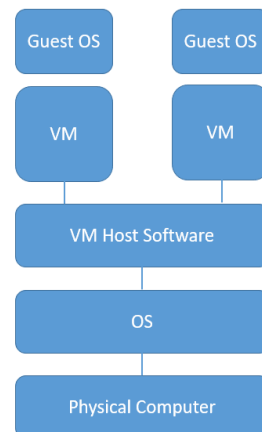
Determinar si un archivo es ejecutable es sobre todo una cuestión de convención. Unos sistemas operativos se basan en la extensión de archivo (como la terminación .exe) y otros lo hacen leyendo los metadatos (como los bits de permiso de ejecución en Unix).

## 2.3 Máquinas virtuales

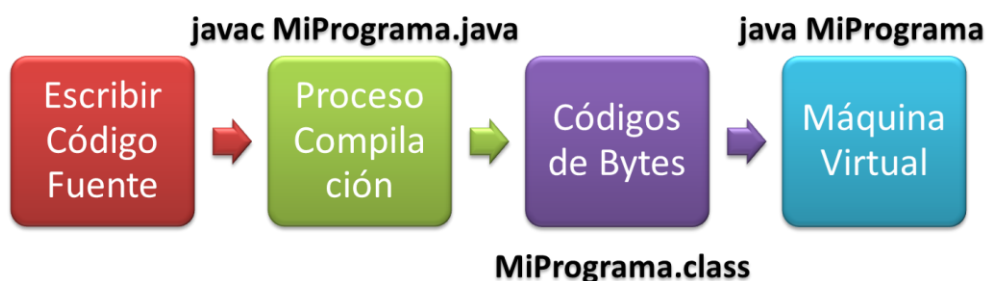
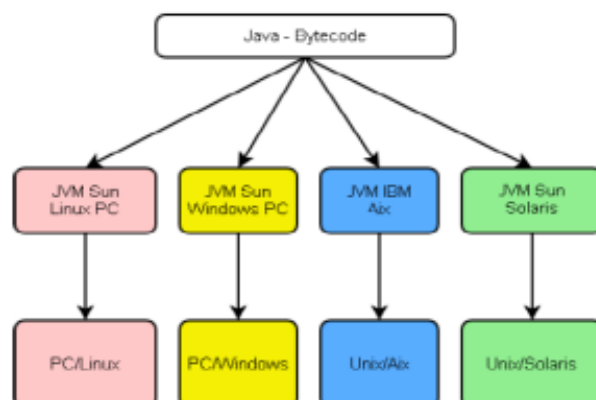
Algunos lenguajes como Java usan una aplicación que emula un SO dentro de otro SO y que se conoce como máquina virtual.

Existen dos tipos de máquinas virtuales:

- **Las máquinas virtuales de sistema:** Se simula un ordenador completo dentro del nuestro a través de un programa de virtualización. Estos programas simulan una arquitectura hardware lo que permite instalar sistemas operativos diferentes a los creados para nuestra arquitectura. Otras aplicaciones son realizar pruebas o simular una red de ordenadores. Ejemplos: VMWare, VirtualBox...



- **Las máquinas virtuales de proceso:** se utilizan para independizar la ejecución de un proceso del hardware subyacente. El bytecode (código objeto) es interpretado por la máquina virtual que se encarga de traducir las operaciones para que funcionen con el hardware específico de la máquina que estemos usando. Sirve de entorno de ejecución para estos programas. La más conocida es la máquina virtual de java o JVM (Java Virtual Machine). La JVM es un ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.



## 2.4 Clasificación de los lenguajes

Existen casi tantas clasificaciones de lenguajes como lenguajes en sí, dependiendo de los autores. Aquí te presentamos una de las muchas posibles:



## a) Tipo de ejecución

### • Lenguajes interpretados

El uso de los lenguajes interpretados ha venido en crecimiento y cuyos máximos representantes son los lenguajes usados para el desarrollo web entre estos Ruby, Python, PHP (se interpreta del lado del servidor), JavaScript y otros como Perl, Smalltalk, MATLAB, Mathematica.

Básicamente un lenguaje interpretado es aquel en el cual sus instrucciones o más bien el código fuente, escrito por el programador en un lenguaje de alto nivel, es traducido por el intérprete a un lenguaje entendible para la máquina paso a paso, instrucción por instrucción.

Los lenguajes interpretados permiten el tipado dinámico de datos, es decir, no es necesario inicializar una variable con determinado tipo de dato, sino que esta puede cambiar su tipo en condición al dato que almacene entre otras características más.

A los lenguajes interpretados los vemos más en software de entornos web o terminales de comandos, ya que requieren de menores recursos y de acceso a archivos determinados.

### • Lenguajes compilados

Un lenguaje compilado es aquel cuyo código fuente, escrito en un lenguaje de alto nivel, es traducido por un compilador a un archivo ejecutable entendible para la máquina en determinada plataforma. Con ese archivo se puede ejecutar el programa cuantas veces sea necesario sin tener que

repetir el proceso por lo que el tiempo de espera entre ejecución y ejecución es ínfimo.

Dentro de los lenguajes de programación que son compilados tenemos la familia C que incluye a C++, Objective C, C# y también otros como Fortran, Pascal, Haskell y Visual Basic.

A los lenguajes compilados los vemos más en software de escritorio ya que requieren de mayores recursos y de acceso a archivos determinados. También por el peso mayor que estos suelen tener en sus archivos ejecutables.

- **Lenguajes virtuales**

Algunos autores clasifican a lenguajes como Java lenguajes virtuales o híbridos. Es un caso particular ya que hace uso de una máquina virtual que se encarga de la traducción del código fuente por lo que hay veces es denominado compilado e interpretado o virtual. Otra ventaja de la máquina virtual que usar Java es que le permite ejecutar código Java en cualquier máquina que tenga instalada la JVM.

## **b) Nivel de abstracción**

Respecto del nivel de abstracción, podría definirse como el nivel de cercanía al lenguaje natural, siendo un lenguaje de alto nivel el que se parece más al lenguaje natural y un lenguaje de bajo nivel el que es más similar al lenguaje máquina.

- **Bajo nivel**

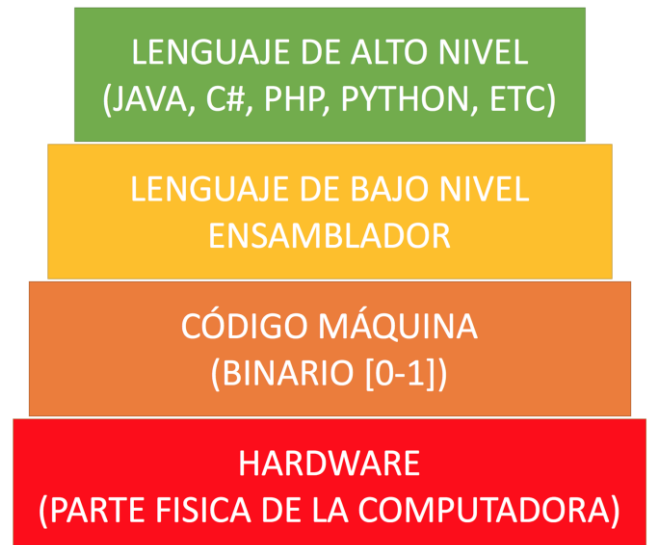
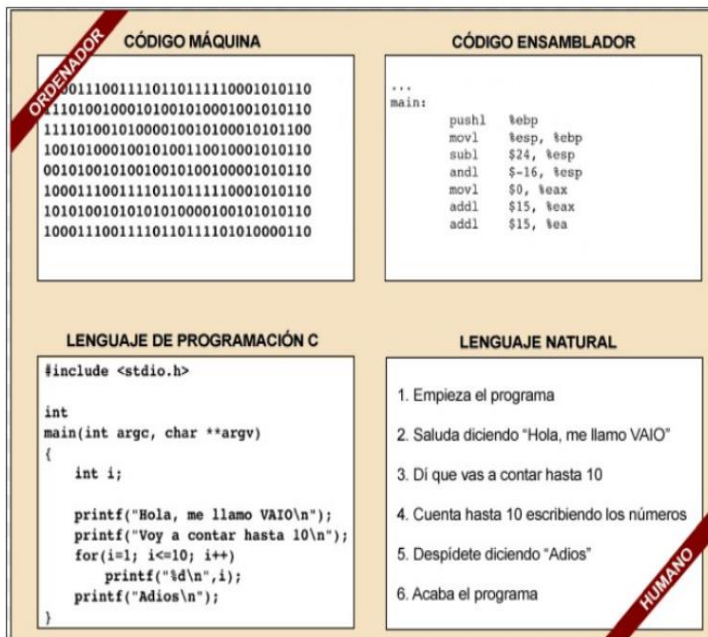
Los lenguajes de programación de bajo nivel son los que dan un conjunto de instrucciones aritméticas que no permiten el uso de funciones que no estén ya contempladas en la arquitectura del hardware. Estos suelen ser una mera traducción del código máquina para facilitar su legibilidad y escritura.

- **Alto nivel**

Los lenguajes de programación de alto nivel son los más abstractos. Están más cerca de la forma natural de habla humana que de las instrucciones de código máquina. Son los lenguajes más usados en la actualidad e incluyen ejemplos como C++, Java, PHP, Python... (Cuesta, 2014).

- **Medio nivel**

Algunos autores clasifican a lenguajes como C un lenguaje de programación de medio nivel puede acceder a registros del sistema y direcciones de memoria, todas propias de lenguajes de bajo nivel.



### c) Paradigma de programación

- **Imperativos**

Los lenguajes imperativos se basan en conocer el estado de la máquina y modificarlo mediante instrucciones. Se llaman imperativos porque usamos órdenes para decir al ordenador dispositivo qué debe hacer y cómo hacerlo. La mayoría de las arquitecturas de ordenadores siguen una filosofía imperativa por lo que la traducción de un lenguaje imperativo a código máquina es más sencilla que la de los lenguajes declarativos.

Son lenguajes imperativos prácticamente todos los que conocemos: Perl, C, Java, PHP, Python, ...

- **Declarativos**

Los lenguajes declarativos fijan un objetivo, pero no el camino para llegar a él. Se le indica al ordenador qué queremos obtener y se le detalla una descripción del problema. La solución se encuentra con estos elementos



siguiendo una lógica interna. Son lenguajes más cercanos a las matemáticas.

Son lenguajes declarativos Prolog, LISP y uno que seguro que te resulta familiar: SQL.

## 2.5 ¿Cómo escoger un lenguaje de programación?

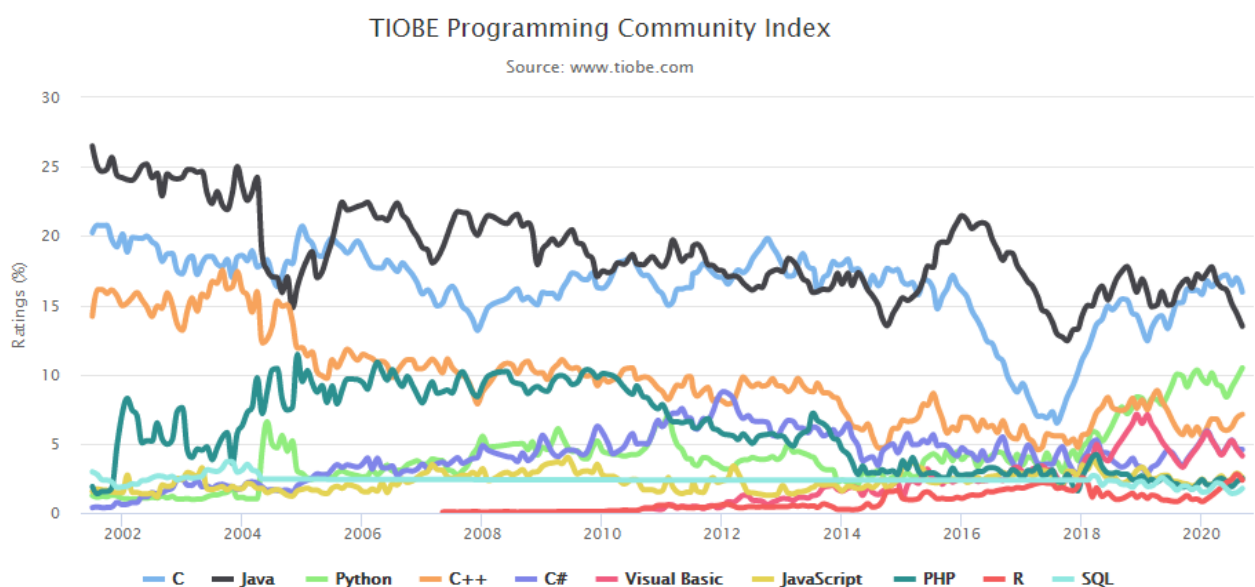
Resolver esta pregunta podría llevarnos días enteros divagando entre todos los posibles paradigmas y tipos de lenguajes y llegaríamos a la conclusión de: DEPENDE.

Cada lenguaje de programación tiene unas características determinadas que lo hacen más o menos apropiado para el contexto donde lo vayas a usar, pudiendo ser determinante estos factores:

1. El sector productivo al que va dirigida tu aplicación (medicina, académico, videojuegos...)
2. En qué dispositivo va a ejecutarse (móviles, escritorio, IoT...)
3. La plataforma sobre la que va a ejecutarse (Windows, Linux, Mac, ...)










Saber qué idioma es más hablado del mundo es tan “sencillo” como contar el número de personas que lo hablan, pero saber cuál es el lenguaje más usado es algo más complicado. Para que te hagas una idea de “cómo está el patio” aquí tienes uno de los índices más reputados que miden eso tan difícil de medir como es “cuanto se usa un lenguaje de programación”. Se llama índice TIOBE y lo puedes consultar aquí:

<https://www.tiobe.com/tiobe-index/>



TIOBE quizás sea el más conocido. Lo elabora la empresa Tiobe, especializada en evaluación de calidad de programas software. Según indican en su propia página

establece la popularidad de los lenguajes en función del número de resultados que se producen en los 25 buscadores más utilizados.

Sep 2023	Sep 2022	Change	Programming Language	Ratings	Change
1	1		 Python	14.16%	-1.58%
2	2		 C	11.27%	-2.70%
3	4	▲	 C++	10.65%	+0.90%
4	3	▼	 Java	9.49%	-2.23%
5	5		 C#	7.31%	+2.42%
6	7	▲	 JavaScript	3.30%	+0.48%
7	6	▼	 Visual Basic	2.22%	-2.18%
8	10	▲	 PHP	1.55%	-0.13%
9	8	▼	 Assembly language	1.53%	-0.96%

### 3 BIBLIOGRAFÍA

- i. Cuesta Vicente, Sergio, S. C. V. (2011, 17 octubre). Tema 1 - Desarrollo de Software. Apuntes de Sergio Cuesta Vicente. [docs.google.com/viewer?a=v&pid=sites&srcid=ZG9tZW5pY29zY2FybGF0dGkuZXN8c2N1ZXN0YXxneDoyZGQ2MjBjNDcwODlyYmFj](https://docs.google.com/viewer?a=v&pid=sites&srcid=ZG9tZW5pY29zY2FybGF0dGkuZXN8c2N1ZXN0YXxneDoyZGQ2MjBjNDcwODlyYmFj)
- ii. The Open University, (2019, 6 noviembre). An introduction to software development. The Open University - England. [open.edu/openlearn/science-maths-technology/introduction-software-development/content-section-0?active-tab=description-tab](https://open.edu/openlearn/science-maths-technology/introduction-software-development/content-section-0?active-tab=description-tab)
- iii. Javier Garzás, J. G. (consultado online 2020, 2 octubre). Javier Garzás - Mentor Ágil, Ágil Coach. Gestión de proyectos y equipos. Blog oficial de Javier Garzás. [javiargarzas.com](https://javiargarzas.com)
- iv. Iglesias, C. C. (2020). Entornos de Desarrollo (GRADO SUPERIOR). RA-MA S.A. Editorial y Publicaciones.
- v. Aldarias, F. (2012): Apuntes de Entornos de Desarrollo, CEEDCV