



Introducción a Python.



Curso teórico práctico de programación en Python.
PYTHON 2024.

Introducción.

A menudo cuando nos proponemos comenzar el aprendizaje de alguna nueva tecnología o lenguaje de programación que despierta nuestro interés, no nos detenemos a ver en detalle el ámbito de aplicación de lo que queremos aprender.

No es un tema menor ya que saber exactamente cuáles son las aplicaciones posibles de lo que estamos encarando, nos ayuda a elegir lo que realmente necesitamos y aprenderlo convencidos de su utilidad futura.

Es por eso por lo que comenzaremos explorando el maravilloso mundo de la programación con Python, conociendo un poco de las múltiples aplicaciones que este lenguaje tiene y sus usos presentes y futuros, ya que es un lenguaje moderno, versátil y con una inmensa comunidad de programadores detrás.

Python.

Python es un lenguaje de programación de alto nivel y propósito general, creado en 1991.

Actualmente es uno de los lenguajes más utilizados principalmente por su simpleza y su versatilidad, lo que hace de este, uno de los lenguajes más indicados para iniciarse en el mundo de la programación.



Los programas en este lenguaje suelen ser bastante compactos, por lo general suelen ser más cortos que el mismo programa escrito en otros lenguajes, como por ejemplo C.

Es un lenguaje interpretado, multiplataforma y orientado a objetos.

Python nos permite la codificación orientada a objetos, en forma imperativa y en menor medida, la programación funcional.

Otro detalle no menor es la gratuidad de su intérprete que tiene versiones para casi cualquier plataforma.

Características.

En lo que su versatilidad se refiere, Python nos permite crear, desde aplicaciones de uso local o web hasta entrenar algoritmos de Machine Learning y hoy en día es el lenguaje más utilizado en todo lo que se refiere a inteligencia artificial, redes neuronales y ciencia de datos.

Sus principales características son:

- Sintaxis legible.
- Versatilidad.
- Es interpretado.
- Posee múltiples implementaciones.
- Multiparadigma.
- Posee tipado dinámico.

Que sea de sintaxis legible, significa que podemos leer e interpretar con relativa facilidad el código escrito por otros programadores, captando rápidamente cual es la función de este y sus principales elementos.

Cuando hablamos de versatilidad, nos referimos a que podemos desarrollar aplicaciones, ya sea para equipos móviles, computadoras hogareñas, podemos hacer SCRIPTING, levantar servidores y páginas web y aplicar toda su potencia en el análisis de datos o sencillez para representar, mediante código, ideas complejas de manera muy sencilla.

Estas aplicaciones del lenguaje, a su vez, están potenciadas por una enorme comunidad de programadores e investigadores que hay por detrás y que posibilita que tengamos a nuestro alcance un sinfín de documentos, ejemplos e información de todo tipo.

Para quienes no conocen la diferencia entre lenguaje compilado e interpretado, al decir que Python es interpretado, decimos que necesita de un intérprete para traducirle al dispositivo el código que genera el programador.

Esto nos permite entre otras cosas, poder ejecutar el código en bloques sin necesidad de hacer una ejecución completa, permitiendo el avance en nuestro desarrollo de una manera mucho más ágil detectando y corrigiendo errores y verificando si el resultado de la ejecución de dicho bloque es la esperada o no.

Cabe aclarar que, la ejecución de un programa compilado es más rápida que la de un programa interpretado.

Otro de sus atributos es que posee múltiples implementaciones en otros lenguajes, como por ejemplo Java, permitiendo escribir código para ser utilizado en dicho lenguaje o por el contrario, hacer uso de las diferentes librerías de Java.

Un paradigma de programación es un estilo, una forma de escribir código de manera sistemática obedeciendo determinadas reglas o imposiciones de dicho paradigma, lo que nos aporta el beneficio de generar código de manera más ordenada, lo que redundará en las ventajas a la hora de mantener o actualizar nuestros desarrollos.

Python admite trabajar con varios paradigmas, como por ejemplo la programación orientada a objetos, la programación estructurada o la programación funcional.

En este curso en particular, nos vamos a enfocar en lo que es la programación orientada a objetos para poder sacar un mejor provecho de toda la potencia de este lenguaje.

Finalmente, cuando decimos que Python tiene un tipado dinámico, significa que, cuando definimos una variable u otro almacén temporal de datos, no debemos especificar el tipo de datos que va a contener, pudiendo en una misma variable cargar un dato numérico, una cadena de texto o cualquiera de los tipos que maneja el lenguaje.

Resumiendo lo visto hasta, podemos armar un cuadro resumen de las ventajas y desventajas de Python, atributos a tener en cuenta al momento

de decidir que herramientas vamos a utilizar para llevar a buen término nuestro desarrollo.

Ventajas	Desventajas
Facilidad de uso	Ejecución lenta
Popularidad	No se tiene control de la gestión de memoria
Open Source	Requiere testing en tiempo de ejecución
Fácil integración con sistemas empresariales	
Desarrollo asincrónico	

Versiones de Python.

Python posee dos versiones diferentes del intérprete actualmente en uso

- Python 2.
- Python 3.

Si bien la versión 2 fue deprecada, es decir, ya no tiene más actualizaciones y su uso está totalmente desalentado, se continúa usando ya que hay muchos sistemas armados en base a esta versión aún funcionando y su desarrollo sobre la nueva implica en muchos casos, un desarrollo desde cero.

Big Data.

Sin duda una de las primeras y más populares aplicaciones de Python, fue todo el ecosistema referido al mundo del Big Data.

Los informáticos estábamos acostumbrados a trabajar con grandes volúmenes de datos, verificar su veracidad y hacer desarrollos basados en dichos datos, de manera eficiente.

Con el paso de los años las empresas, en particular las de telecomunicaciones y grandes redes sociales, comenzaron a ver la gran cantidad de datos que los usuarios generamos día a día y decidieron invertir

en analizar si era posible sacar algún rédito económico de toda esa enorme cantidad de datos, lo que comúnmente llamamos monetizar la información.



Fue así como se fue desarrollando una nueva tecnología, un nuevo paradigma de ver el mundo de los datos que hoy conocemos como Big Data.

Dicha definición empezó a circular por el año 1.999, en una publicación académica acerca de la exploración de datos en tiempo real.

Luego se definieron las bases para trabajar el concepto de Big Data, tomando como pilares fundamentales las tres V, volumen, velocidad y variedad.

Como casi todas las nuevas ideas, esta también fue impulsada por el avance constante de internet y el crecimiento exponencial de usuarios en todo el mundo.

En 2.007 surge la definición actual de Big Data, en 2.010 el entonces presidente ejecutivo de Google, Eric Schmidt, dice en una conferencia que la cantidad de datos que en ese momento se estaban creando cada dos días, era mayor que la creada desde el comienzo de la civilización humana hasta el 2.003.

Si bien es un dato difícil de comprobar, no estaba tan alejado de la realidad y la cantidad de datos que generamos, muchos de ellos de manera inconsciente, es de un volumen muy, pero muy grande a lo que estábamos acostumbrados a manejar.

Finalmente, la ONU formaliza una definición formal en el año 2.012, definiendo al Big Data como:

- *Volumen masivo de datos, tanto estructurados como no estructurados, los cuales son demasiado grandes y difíciles de procesar con las bases de datos y el software tradicional.*

Resumiendo, definimos a Big Data como un gran volumen de datos, estructurados y no estructurados y la tecnología necesaria para procesarlos.

Cuando hablamos de tecnología, nos referimos tanto a la de extracción y almacenamiento como a la de análisis y proceso, que es justamente dónde Python se hace fuerte entre los profesionales del área.

Dentro de las empresas, Big Data es el sector de IT que hace referencia a grandes conjuntos de datos que por la velocidad a la que se generan, la capacidad para tratarlos y los múltiples formatos y fuentes de origen, es necesario procesarlos con mecanismos distintos a los tradicionales.

A las clásicas tres V fundamentales en las primeras ideas del Big Data, sumamos una cuarta referida a la veracidad de la información, completando así el concepto formal de todo el trabajo necesario para desempeñarse en este ámbito.



Un detalle no menor y que contribuyó al desarrollo de esta nueva tecnología, es el hecho de que el precio del almacenamiento de datos fue bajando año a año, llegando a ser muchísimo más barato almacenar información que procesarla, esto implicaba directamente que la captura de información en tiempo real era un hecho posible y relativamente fácil de integrar con las tecnologías disponibles.

Aplicaciones del Big Data en los negocios.

Ya establecida esta tecnología, fue madurando y tanto desarrolladores como usuarios, rápidamente fueron encontrando aplicaciones comerciales derivadas de las características de la nueva información disponible, porque justamente, lo que antes eran solo datos almacenados, pasaron a ser información útil y la demanda de nuevos usos creció casi al ritmo que el Big Data evolucionaba.

Básicamente, en donde tenemos mucha gente generando muchos datos, es ahí donde necesitamos aplicar esta tecnología, para poder relacionarnos en tiempo real con nuestros usuarios y clientes actuales o potenciales, gracias al análisis de los datos con Python, entre los principales lenguajes.



Sistemas de gestión de relacionamiento con clientes (CRM), redes sociales, compras, ventas, oferta de servicios y marketing en tiempo real, son algunos de los principales usos del Big Data.

Esto posibilita un conocimiento más detallado de los clientes, pasamos de modelo estático a uno dinámico, se crearon las campañas publicitarias personalizadas, la retención de clientes insatisfechos, mejora de experiencia de usuarios, anticipar el comportamiento de las personas en cuanto a sus perfiles de comportamiento y muchas aplicaciones más se hicieron posibles gracias a lenguajes como Python, que es, dentro del mundo de la ciencia de datos, el más utilizado y los profesionales especializados, son muy requeridos.

Casos de uso.

Dentro de los casos de uso más conocidos, nos encontramos con el gigante Amazon, que hace una evaluación continua en tiempo real del comportamiento de sus potenciales clientes, trabajando con modelos analíticos (son los que trabajan sobre datos ya generados) y con modelos predictivos (aquellos que anticipan los gustos y necesidades del cliente).

Otro caso muy conocido es el de la serie de Netflix, HOUSE OF CARDS, una producción que se creó en base a los gustos de los usuarios, estableciendo un patrón de consumo en base a más de 40 millones de usuarios, detectando que era lo que más les atraía y en base a ello, crearon la serie.

Otro detalle es que se crearon diez avances diferentes de la serie y en base al perfil de quien quisiera verla, le mostraban el que más se ajustaba a sus preferencias.

Las empresas de venta en línea y los bancos trabajan mucho con el tema del comportamiento del usuario para detectar posibles fraudes, algo que también se aplica al uso de las redes sociales.

Bases de datos.

Todos hemos trabajado con bases de datos alguna vez, desde una simple agenda telefónica hasta un complicado sistema de facturación.

Cuando hablamos de Big Data, se mencionaron los tipos de datos, un tema no menor cuando hablamos de bases.

Todos conocemos las bases de datos tradicionales, en las que puedo guardar los tipos de datos más usados, ya sean numéricos, de texto, fechas y todo lo que se derive de cada uno.

Para cada dato, conocemos exactamente su tipo y por ello es relativamente sencillo modelar las bases de datos donde serán guardados.

A estos datos los llamamos estructurados.

Pero en los volúmenes de datos que comenzamos a trabajar en Big Data, nos encontramos con la necesidad de almacenar datos como, por ejemplo, fotos, audios, videos, geolocalizaciones y muchos otros. A este tipo de datos los llamamos no estructurados y fue necesario avanzar en el concepto tradicional de base de datos para aprender a trabajar con esta nueva información.

Para quienes no trabajaron nunca con bases de datos tradicionales, los datos se consultaban escribiendo las consultas en un lenguaje particular, el principal para la explotación de este tipo de datos, el SQL.

Es un lenguaje estructurado para la ejecución de consultas a bases de datos.

Pero para los nuevos tipos de datos, necesitábamos algo diferente, fue así como comenzamos a hablar de bases NoSQL (NOT ONLY SQL).

Son sistemas de gestión de bases de datos que difieren del modelo clásico de las bases relacionales, no usan SQL como lenguaje de consulta y los datos almacenados no requieren tener una estructura fija como una tabla.

Surgieron para complementar las bases de datos tradicionales y no para reemplazarlas.

Dentro de estos nuevos ecosistemas de datos, el más conocido y utilizado es HADOOP, un sistema de código abierto que se utiliza para almacenar, procesar en paralelo y analizar grandes volúmenes de datos.

DataWareHouse.

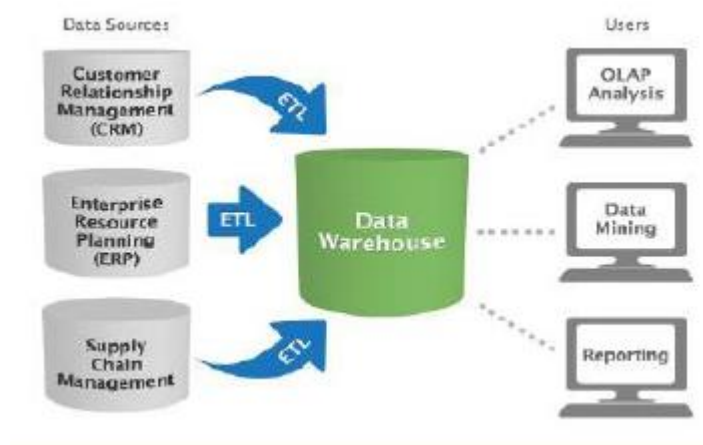
Un DataWareHouse es un repositorio de datos integrado, no volátil, variable en el tiempo, con clara orientación al negocio, organizado de forma tal que facilita el análisis de grandes volúmenes de datos para la toma de decisiones.

En las empresas habitualmente se trabaja con más de un sistema de información, por ejemplo, sistemas de RR. HH, sistemas de cobros y facturación, sistemas de venta, etc.

Si tuviéramos que tomar decisiones en base a los datos de todos ellos, sería un poco complicado, es por ello que existe el DataWareHouse, que se encarga de realizar la extracción de datos de todos estos sistemas, transformarlos a un modelo sencillo orientado al modelo de negocio de la empresa y cargarlos en el para que estén disponibles.

Es una gran base de datos que contiene la información más relevante de todos los sistemas de la compañía, en un formato que sea de fácil lectura y uso.

En base a estos datos se arman reportes, tableros de datos y diferentes estudios o análisis de los principales indicadores de la compañía.



El DataWareHouse es una herramienta fundamental para la toma de decisiones, en cualquier área funcional, basándose en información integrada y global de la empresa.

Facilita mucho el uso y aplicación de técnicas estadísticas de análisis y modelización, para encontrar las relaciones ocultas entre los datos almacenados, y obtener de esa forma, un valor extra para el negocio.

Además, proporciona la capacidad de aprender de los datos del pasado y predecir situaciones futuras, en diversos escenarios.

Cuando decimos que un DataWareHouse (DW) debía ser integrado, nos referíamos a que debe proporcionar información proveniente de sistemas heterogéneos (diferentes tipos de origen de datos) además de contar con procesos de integración de dichos datos y limpieza de la información.

Por no volátil, nos referimos a que los datos deben persistir en el tiempo.

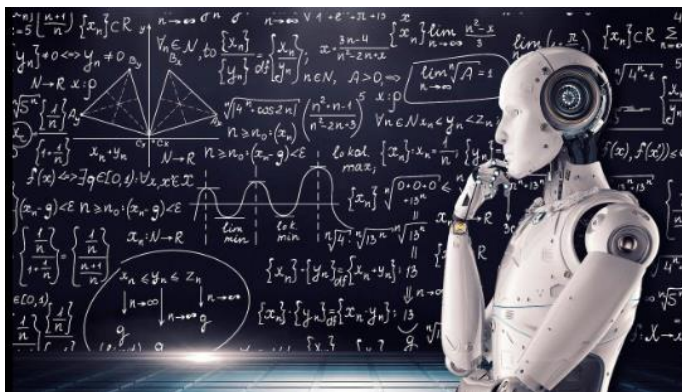
Es, como ya mencionamos, un repositorio de datos históricos, con un tiempo de conservación de los mismos mucho mayor al de los sistemas tradicionales, en general, la fecha es un dato fundamental para poder analizar en el tiempo, trabajar en el estudio de las evoluciones.

La orientación al negocio del DW implica que los datos que allí se almacenan, están organizados y presentados de la manera en que se manejan en el negocio para facilitar su explotación.

Los datos deben tener el nivel de detalle y estructura necesarios para los que toman decisiones.

Inteligencia artificial.

Cuando hablamos de inteligencia artificial, nos basamos en dos ramas de trabajo principales, la interpretación del lenguaje natural y la imitación del razonamiento humano para la resolución de problemas, añadiendo la capacidad de memoria y procesamiento que nos ofrecen los equipos de hardware hoy en día.



Python basa su versatilidad en dos premisas muy útiles, la filosofía DRY (Don't Repeat Yourself) y RAD (Rapid Application Development).

Ambas características han hecho de Python el lenguaje por excelencia para el desarrollo de aplicaciones de inteligencia artificial (AI).

Es un lenguaje que permite desarrollar ideas muy complejas de una manera muy sencilla por la sintaxis y semántica que tiene, muy fácil de entender y con una facilidad para el aprendizaje que ha posibilitado que mucha gente lo utilice.

Al lenguaje en si se le pueden agregar de manera muy simple muchas funcionalidades, lo que conocemos habitualmente como módulos.

MACHINE LEARNING (ML).

El MACHINE LEARNING o aprendizaje automático, es un derivado de la inteligencia artificial, cuya principal finalidad es desarrollar técnicas para que las máquinas aprendan una determinada tarea o proceso.

Dentro de lo que hace un desarrollador en el proceso de creación de dichas técnicas, está el explorar los datos para identificar el problema a resolver, y para ello Python nos proporciona herramientas tales como Pandas y Numpy.

El estudio de dichas librerías es parte del curso avanzado de ciencia de datos.

Otro trabajo típico de esta rama es el de graficar los datos explorados, con librerías como por ejemplo MATPLOTLIB.

Una vez llegado a este punto, llega el momento de empezar a aplicar técnicas de ML para lo cual contamos con excelentes librerías como SCIPY y SKLEARN.



Redes neuronales.

A menudo se habla de redes neuronales, como una forma de emular algunas características propias del ser humano, asociándolo solo a la capacidad de memorizar y de asociar ciertos hechos.

Cuando nos encontramos con uno de esos problemas que no pueden ser resueltos con un algoritmo, notamos que todos tienen algo en común, la experiencia o el entrenamiento requeridos para su resolución.

Los seres humanos somos capaces de resolver estos problemas, apelando a la experiencia y a la asociación de la situación a resolver con otras vividas previamente.

Es así como surge esta idea de acudir a la construcción de sistemas que sean capaces de reproducir esta conducta humana para la resolución de problemas.

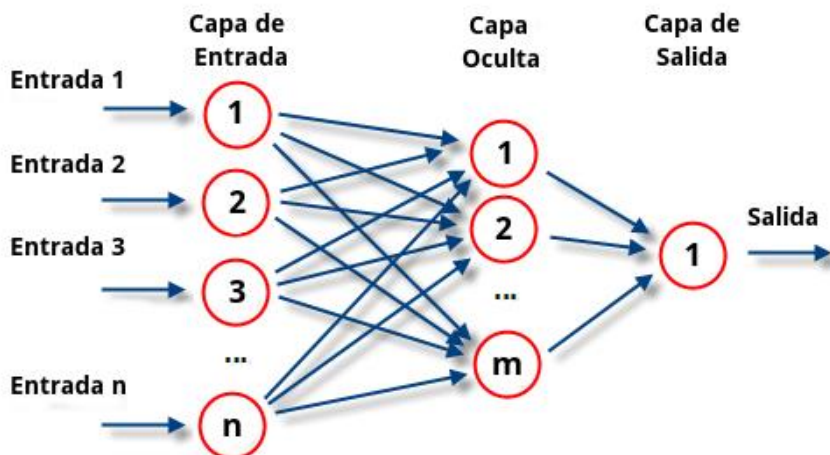
Resumiendo, las redes neuronales no son más que un modelo artificial y simplificado del cerebro humano, que es el modelo más acertado del que disponemos para emular y desarrollar un sistema que sea capaz de adquirir conocimiento a través de la experiencia.

Es un sistema para el tratamiento de la información, cuya entidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano, la neurona.

Casi todos los procesos del cuerpo humano se relacionan en menor o mayor medida con la actividad de dichas neuronas.

Como sabemos, el pensamiento tiene lugar en el cerebro humano, que consta de billones de neuronas que están interconectadas, por lo tanto, las redes neuronales presentan las siguientes características:

- Son unidades de procesamiento que intercambian datos e información.
- Sirven para reconocer patrones y secuencias de tiempo.
- Son capaces de aprender y mejorar su funcionamiento.



Introducción al proceso de desarrollo.

Para meternos en el mundo de la programación, debemos tener presentes los elementos iniciales que forman parte de los procesos ligados al desarrollo de aplicaciones.

Como futuros programadores, nos encontraremos con problemas a resolver, y para encontrar la solución adecuada iremos confeccionando aplicaciones informáticas.

En el mundo del desarrollo, veremos que cada problema es único, pero son únicas las soluciones posibles y es parte de nuestro trabajo diseñar la respuesta más eficiente para cada problema al que nos enfrentamos.

En el contexto actual nos encontramos a diario con diferentes tipos de sistemas y aplicaciones, como, por ejemplo, sistemas operativos para controlar dispositivos, juegos, gestores de información, plataformas de compras, páginas web, etc.

Detrás de todo este mundo de soluciones se encuentran los programadores, analistas, diseñadores y muchos profesionales más, cuyo trabajo conjunto hace posible que existan estos elementos que ya forman parte de nuestras tareas diarias, sin importar a que nos dediquemos.

Una definición bastante aceptada de lo que significa en el mundo IT el desarrollo, es confeccionar, probar y gestionar errores en un programa destinado a dar solución a un determinado problema.

Cuando nos proponemos aprender a programar, lo hacemos con el fin de cubrir determinadas necesidades, ya sean personales o de terceros (clientes) a cambio de un rédito económico.



El primer paso que tendremos que dar como futuros programadores, es aprender lo que llamamos, programación lógica, muy importante porque si bien cada lenguaje tiene sus particularidades, los problemas son analizados desde un punto de vista lógico único y de un mismo modo sin importar las herramientas que utilicemos para diseñar y desarrollar nuestras soluciones. Esto es sumamente conveniente ya que nos permite migrar nuestra solución a cualquier lenguaje de programación que necesitemos implementar.

Resolución de problemas.

Cuando desarrollamos una solución, ya sea por trabajo o con fines académicos, partimos siempre de un problema a resolver, al cual debemos encontrarle la mejor solución.

El problema a resolver o la necesidad para la cual programamos, puede ser de índole personal, para realizar pequeñas tareas con un fin determinado que nos beneficie, como puede ser una aplicación para nuestro móvil que nos informe nuestras tareas pendientes, una web o un CV en línea o bien puede ser del ámbito empresarial, donde podríamos necesitar realizar sistemas, partes de sistemas o nuevos módulos, incluso arreglar un código escrito por alguien más, o automatizar procesos que se estén llevando a cabo de manera manual.

Sea cual fuere el caso, debemos tener siempre presente nuestras metas, determinar fehacientemente el objetivo al cual apuntamos, analizar las

posibles etapas e incumbencias del proyecto y evaluar correctamente la viabilidad del mismo.

Cuando programamos podemos hacerlo para diferentes entornos o ecosistemas, trabajar con diferentes arquitecturas (formas de estructurar nuestros desarrollos) y de esa manera hacer aplicaciones móviles, web, de escritorio, sistemas operativos, SCRIPTS (porciones de código con un fin específico) o procesos.



Aplicaciones web.

Todos conocemos y utilizamos, cada día más, las aplicaciones desarrolladas para el mundo web. En general las agrupamos en dos grandes categorías, las aplicaciones estáticas, más orientadas a mostrar imágenes y datos estáticos o animaciones simples y las dinámicas, que son aquellas más complejas que, además de toda parte que se ve, cuenta además con toda una estructura por detrás, lo que llamamos BACKEND que, entre otras cosas, incluyen un servidor, un intérprete del lenguaje de programación y un sistema de gestión de bases de datos.

Muy populares en el mundo empresarial, les permiten a las compañías trascender las limitaciones de la actividad presencial y llegar literalmente a todo un mundo de potenciales clientes.

Trabajan sobre lo que llamamos arquitectura cliente – servidor, siendo quienes solicitan ver o acceder a determinada aplicación web, los clientes, y

dentro del servidor está implementada toda la infraestructura de proceso y bases de datos.

Este tipo de tecnología no solo es aplicable en los negocios, también son muy útiles desde hace ya bastante tiempo, en la educación, medicina y todo tipo de actividades que requieran interacción entre personas o de personas con sistemas complejos.



Existe una muy amplia gama de sitios web dedicadas a casi todas las actividades que se nos puedan ocurrir, es una rama de la programación muy requerida en el mundo laboral y que continuamente está incorporando nuevas tecnologías y métodos de trabajo, pasando a ser desde hace ya un tiempo, una forma de relacionamiento cada vez más presente y aceptada como alternativa de trabajo.

Aplicaciones de escritorio.

Las aplicaciones de escritorio son aquellas que funcionan sobre un sistema operativo, de dispositivos tales como un servidor, pc de escritorio o notebook.

Este tipo de desarrollos suelen ser bastante más grandes que sus pares de otra arquitectura, como la web, y a menudo más costosos, ya que están orientados a un público determinado, generalmente a demanda y con requerimientos específicos.

Muchos de estos programas cuentan con versiones de prueba (TRIAL) destinadas a hacer conocido el proyecto o al testeo de errores con usuarios reales, lo que llamamos versión BETA, y desde luego, las versiones finales, no gratuitas con diferentes configuraciones dependiendo del grupo de usuarios y objetivos al que esté destinado.



Aplicaciones móviles.

Son los programas que se desarrollan para dispositivos móviles como teléfonos celulares o tabletas.

A simple vista puede que nos resulten similares a las aplicaciones de escritorio, pero desde su construcción y diseño, se basan en paradigmas de programación muy distintos a los del software de escritorio.

Existen muchos sistemas operativos para estos dispositivos móviles, siendo tal vez el más conocido, Android, cuya arquitectura tiene entre sus características, la posibilidad de desarrollar aplicaciones seguras, portables y con una simpleza de acceso al hardware del dispositivo, que nos permite un nuevo mundo de posibilidades a la hora de programar.



En particular, el mundo de las aplicaciones móviles y nuestra orientación dentro del mismo estará determinado por el público objetivo al cual deseamos llegar, siendo el sistema operativo Android uno de los más populares y que no requiere demasiada inversión a la hora de iniciar un proyecto.

Se trabaja con herramientas de uso gratuito, en cuanto a lenguajes de programación, los dos lenguajes oficiales aceptados por Google son Java y Kotlin y como herramienta para el desarrollo integral de aplicaciones, Android Studio, que, si bien es gratuito y cuenta con todo lo necesario para desarrollar cualquier tipo de actividad, tiene algunos requerimientos de hardware bastante más exigentes que la mayoría de los entornos.

Sistema.

En programación entendemos como sistema a todo conjunto de instrucciones, trabajando en conjunto con un fin común.

Puede tratarse de un complejo sistema de información, hasta un simple tratamiento de datos, que, ya que lo mencionamos, siempre contará con los mismos elementos básicos, una entrada de datos, un proceso de los mismos, y una salida.



Una entrada es un ingreso de datos o comandos, ejecutados sobre un dispositivo, como, por ejemplo, un teclado, una pantalla táctil, un comando de inserción de datos desde un origen, como puede ser un archivo que se cargue en una memoria, una cámara web o un micrófono.

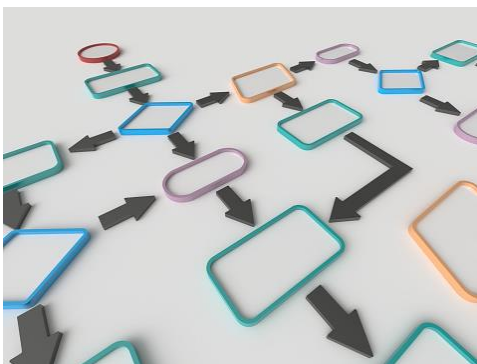
El trabajo que hacemos con los datos que fueron ingresados, se llama proceso y va desde un simple ordenamiento hasta un complejo sistema de cálculos o transformaciones, diferentes pasos todos programados en algún lenguaje informático, donde entran en acción elementos tales como el procesador y la memoria del dispositivo donde se esté ejecutando este procesamiento.

Finalmente, la salida es el resultado de estas acciones, efectuadas durante el proceso, sobre los datos ingresados.

Puede ser una salida por pantalla, una impresión o directamente guardar los datos resultantes en una base de datos.

Algoritmo.

Si bien existen múltiples definiciones de lo que es y lo que no es un algoritmo, vamos a enfocarnos y trabajar sobre la definición formal que nos da la RAE respecto a los mismos, a los que define como un conjunto ordenado y finito de operaciones que nos permite hallar la solución a un problema.

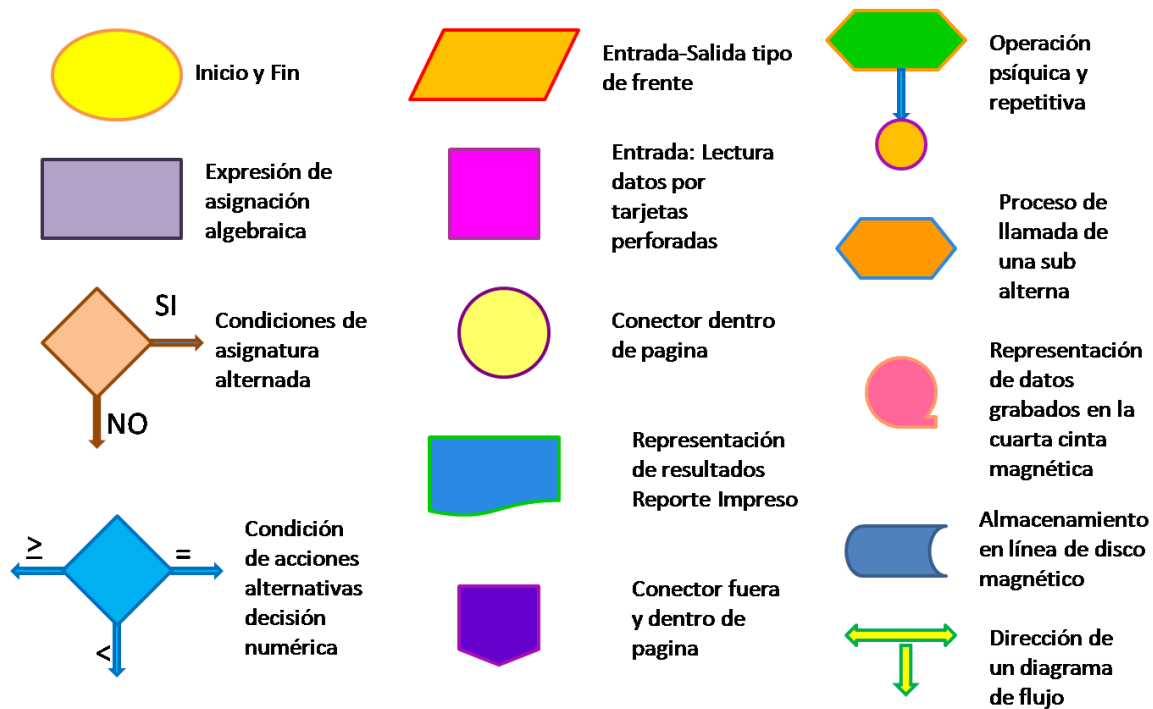


Las personas tenemos incorporados nuestros propios algoritmos de decisión y ejecución de procesos cotidianos.

Desde decidir con que ropa vestarnos hasta armar una lista de compras del supermercado. Y, como bien dijimos, todo esto lo hacemos en base a un número finito de operaciones, que, conociendo la simbología adecuada, podríamos expresarlos como un algoritmo.

En general, conociendo solo unos pocos símbolos de algoritmos o diagramas de flujo de datos como también se los conoce, podemos armar un diagrama tan complejo como lo necesitemos.

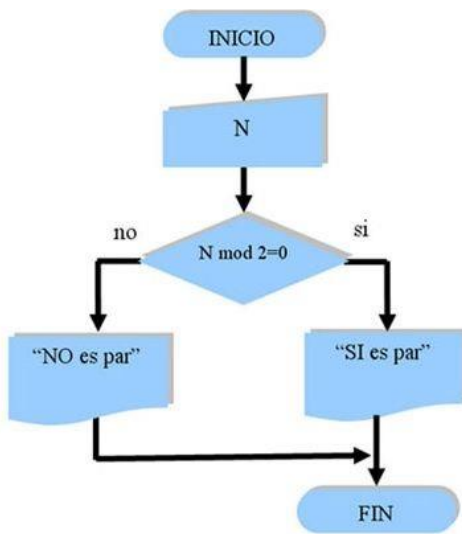
Veamos primero cuales son estos elementos y cómo encarar un problema mediante el uso de estos diagramas.



Fuente:

<https://www.diagramasdeflujo.com/flujograma/diagrama-de-flujo-de-procesos/>

Veamos ahora un ejemplo sencillo para expresar un proceso mediante el uso de un algoritmo o diagrama de flujo que nos permita tomar un número cualquiera y analizar su condición de paridad, es decir, determinar si es par o impar.



Comenzamos ingresando o leyendo un valor numérico. A continuación, lo dividimos sobre dos y verificamos si el resto de dicha división es cero, de ser así, determinamos que el número es par, caso contrario, el número leído será impar.

Este tipo de diagramas son muy útiles para el programador y nos permiten trasladar rápidamente una solución lógica a un lenguaje de programación.

Lenguajes de programación.

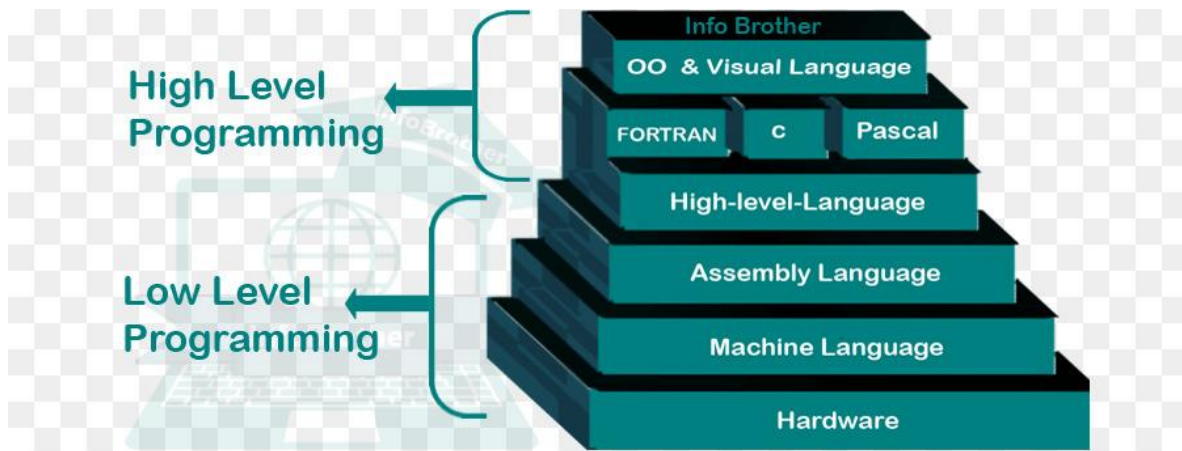
Los dispositivos para los cuales escribimos nuestros programas, como por ejemplo una computadora o un teléfono móvil, solo entienden un lenguaje propio, conocido como lenguaje de máquina.

A los programadores nos resulta algo complejo desarrollar nuestros trabajos en dicho lenguaje, por lo que necesitamos de alguna herramienta, que nos permita plasmar nuestras ideas para luego entregarlas al dispositivo y que ejecute nuestras órdenes en la manera en que nosotros queremos que sean ejecutadas.

Para ello existen los lenguajes de programación.

Los hay de sintaxis algo compleja y difícil de aprender, pero muy eficientes respecto al uso de recursos (lenguajes de bajo nivel) y los hay muy sencillos de entender e implementar, pero en ocasiones, no lo suficientemente rápidos, (lenguajes de alto nivel).

Todos los lenguajes fueron diseñados para un propósito específico, y nosotros como programadores, tenemos que tener presente el objetivo de nuestro desarrollo, entender bien el problema y determinar que lenguaje, en base a sus características, es mejor para nuestra solución.



Dentro de las diversas clasificaciones que usamos para los lenguajes de programación, no es esta (alto o bajo nivel) una de las que más nos interesa.

Pero es importante saber, resumiendo, que hay lenguajes cuya sintaxis es más parecida al lenguaje nativo de las máquinas (bajo nivel) y lenguajes cuya sintaxis es más parecida al lenguaje natural del ser humano (alto nivel).

Otro factor a tener en cuenta a la hora de clasificar a los lenguajes de programación es la forma en que son traducidos, desde el código que nosotros escribimos, al lenguaje de máquina.

Existen dos formas de hacer esta traducción.

La primera es mediante un intérprete, que vaya leyendo línea a línea nuestro programa y a medida que las lee, las traduce para la máquina. A estos lenguajes se los conoce como interpretados, como, por ejemplo, Python y PHP.

La segunda opción, es mediante un compilador, que es un programa encargado de leer todo nuestro código y traducirlo por completo a lenguaje máquina, dejando como resultado, uno o más archivos ejecutables que son directamente ejecutados por el dispositivo, por ejemplo, Java.



Por último, podemos ir un paso más allá y ver en qué lugar será ejecutado nuestro código, ya que hay lenguajes (en especial en el mundo de la programación web) que se ejecutan del lado del cliente, como JAVASCRIPT y lenguajes que se ejecutan del lado del servidor, como PHP, y ambos son interpretados (traducidos a código máquina mediante un programa intérprete).

Entornos de trabajo.

Dentro de las herramientas que utilizaremos como programadores, tenemos editores de texto, IDE y FRAMEWORK.

Cada uno está reservado para un tipo de tarea particular.

Al programar, necesitamos escribir nuestro código de manera simple y organizada, por lo que, además de contar con algún sistema propio de planificación y orden, necesitaremos de algún programa que nos permita escribir código de manera eficiente.

Dentro de esta categoría tenemos a los editores de texto, que van desde simples programas para escribir y entender código, hasta algunos algo más complejos que permiten ejecutar nuestro código, conectarnos con bases de datos y hasta nos marcan los errores de sintaxis, errores de estructura y nos dan sugerencias de qué instrucciones deberíamos utilizar o auto completan lo que vamos escribiendo.

Hay muchos editores de este tipo y la mayoría absolutamente gratuitos, cómo, por ejemplo, VISUAL STUDIO CODE, NOTEPAD++ y SUBLIME TEXT.



Visual Studio Code

Páginas de descarga:

- <https://code.visualstudio.com/download>
- <https://www.sublimetext.com/>
- <https://notepad-plus-plus.org/downloads/>

Todas estas opciones son igualmente útiles y completas a la hora de decidirse, solamente habrá diferencias más ligadas con la comodidad y la experiencia que con las funcionalidades de cada uno.

Otra de las herramientas infaltables, siempre que la necesitemos por el lenguaje que decidamos aprender o utilizar, es el IDE, entorno integral de desarrollo por sus siglas en inglés.

El IDE es un programa complejo que trae todas las funcionalidades necesarias para desarrollar e implementar nuestras creaciones directamente desde dicho IDE y sin necesidad de descargas o herramientas complementarias.

Ejemplos de estos programas son Visual Studio (no confundir con Visual Studio Code) y Android Studio.

Por último, un FRAMEWORK es un conjunto de herramientas y utilidades que, en conjunto, conforman un marco de trabajo que nos permite desarrollar en determinadas tecnologías o entornos sin necesidad de escribir código de más y en donde la mayoría de las necesidades ya se encuentran resueltas, por ejemplo, .NET, ANGULAR y BOOTSTRAP.

Elementos y lógica de programación.

En este módulo conoceremos los elementos fundamentales de la programación desde su concepción teórica, para entender en detalle el comportamiento de cada uno frente a distintas implementaciones.

Conocer cada uno de dichos elementos es la base fundamental para todo programador, ya que nos permite plasmar directamente en código, la lógica de ejecución que vayamos elaborando.

Es imperativo para todos aquellos que quieran iniciarse en el mundo de la programación, entender, probar y aplicar todos los elementos básicos de la programación, comunes a todos los lenguajes.

Como futuros programadores, nuestro objetivo es diseñar y desarrollar distintas soluciones mediante la confección de aplicaciones informáticas.

Una definición que podemos encontrar en primera instancia sobre el desarrollo de una aplicación es: confeccionar, probar y buscar errores en un programa informático, nacido de una necesidad de solución a un determinado problema.

Cuando nos proponemos aprender a desarrollar y programar aplicaciones o sistemas, lo hacemos para cubrir determinadas necesidades, ya sean personales o de terceros, y así obtener un ingreso económico a cambio de nuestro trabajo.

Uno de los pasos fundamentales que debemos efectuar antes de comenzar es aprender la **programación lógica**.

Esto es importante porque, si bien los lenguajes de programación tienen sus particularidades, las soluciones lógicas son analizadas de un solo modo.

De esta manera, conocer este tema claramente nos permitirá migrar a todos los lenguajes que queramos.

Aprender a desarrollar aplicaciones nos ofrece muchas posibilidades, ya que podremos realizar programas en cualquier plataforma, ya sea para la Web, Windows, Linux o Macintosh; incluso, para móviles, televisión inteligente, etc.

El propósito principal es tener la base lógica de programación, y luego elegir cuál es el lenguaje en el que deseamos poner nuestro mayor esfuerzo. Puede ser el que esté latente en el mercado, uno específico de un área (como para los trabajos científicos) o, simplemente, aquel en el que nos sintamos más cómodos para trabajar.

Al adquirir estos conocimientos, podremos tomar cualquier modelo de negocio o problema funcional de una organización, y resolverlo mediante la programación de una aplicación.


Resolver problemas.

Nuestra tarea principal será realizar una aplicación para resolver un problema en particular, o tal vez lo hagamos solo por diversión.


Por ejemplo, podemos crear un programa para llevar en nuestro teléfono móvil una agenda que nos informe los días de estreno de nuestras series favoritas de televisión. También podemos aplicarlo en el trabajo, para agilizar la toma de decisiones y digitalizar la información referida al desempeño de los empleados. Ambos son modelos de negocios distintos que plantean un problema, y nosotros debemos encontrar una solución. Estas necesidades pueden surgir desde distintos ámbitos:

- **Personal:** realizar pequeñas o amplias aplicaciones para un fin que nos beneficie. Por ejemplo: elegir una aplicación que nos indique el consumo de Internet en nuestro teléfono móvil o programar una página web personal.
- **Empresarial:** realizar sistemas informáticos, partes o módulos que tenemos que programar; incluso, arreglar un código que haya sido confeccionado por otro. Por ejemplo: utilizar nuestros conocimientos para mejorar un sistema de inventario o realizar una página web para una organización que cuenta con un módulo de ventas online.

Tengamos en cuenta que el ámbito empresarial es más duro, ya que requiere seguir ciertas pautas y criterios que veremos en los próximos capítulos. En cambio, cuando las metas son personales, podemos dedicarnos a desarrollar de manera **freelance**.



EL PLANEO DE
METAS ES UN PUNTO
EXCLUYENTE EN EL
DESARROLLO DE
APLICACIONES



Las metas empresariales son estrictas y, en general, nos afectan, ya que, por ejemplo, nos imponen un límite de tiempo específico que debemos cumplir.

Dentro del desarrollo de aplicaciones, una meta empresarial que debe influir en nuestros objetivos personales es absorber los conocimientos del grupo de trabajo, para luego aplicarlos a los nuevos desafíos que vayamos afrontando más adelante.

El planteo de metas es un punto excluyente en el desarrollo de aplicaciones, porque tener en claro hacia dónde queremos llegar nos motivará a nivel personal a seguir investigando, buscando y probando.

Al mismo tiempo, nos ayudará a plantearnos los objetivos buscados sobre los desarrollos a realizar. De esta forma, algo que parece tan sencillo como plantearse una meta y conocer los objetivos nos permitirá organizar y optimizar el desarrollo.

Tipos de aplicaciones.

En el mercado informático actual, nos encontramos con diferentes soportes de hardware que albergan variados tipos de aplicaciones, ya sea exclusivas de Internet, del sistema operativo o de un aplicativo en particular.

Así como antes comenzamos a formar el concepto de desarrollo de una aplicación, ahora vamos a reforzarlo haciendo un repaso de las aplicaciones existentes, de modo de tener una idea gráfica de qué podemos considerar para nuestro trabajo.

Aplicaciones web

Las aplicaciones web son herramientas muy comunes en organizaciones que desean ampliar las fronteras de sus modelos de negocios o, simplemente, alcanzar la autogestión para empleados, alumnos, docentes, etcétera.

Hay una amplia variedad de sitios web destinados a distintos rubros, como puede ser el automotriz, en donde es posible personalizar o armar autos a gusto, elegir colores, definir agregados, etc.

Esto nos demuestra la variedad de trabajo que se puede realizar en los desarrollos para la Web.

Aplicaciones de escritorio

Las aplicaciones de escritorio son aquellas que funcionan sobre un sistema operativo de PC (computadora personal) o notebook.

Los desarrollos en este ámbito también son enormes, y podemos encontrarnos con algunos muy costosos, utilizados por grandes empresas; y con otros gratuitos y útiles que pueden servirnos para diferentes tareas. Veremos que muchos de estos programas cuentan con un tipo de distribución llamado **trial**.

Se trata de una instalación de prueba, generalmente por un máximo de 30 días a partir de su instalación, que suele tener funcionalidades limitadas. Otras versiones de prueba gratuitas pueden ser **shareware** o **freeware**, que podemos instalar y utilizar en los equipos que queramos.

Aplicaciones móviles

Son aplicaciones que se utilizan en equipos móviles, como teléfonos celulares o tabletas. Suelen ser muy similares a las de escritorio, ya que permiten realizar las mismas tareas, aunque el ingreso de datos es táctil o por voz.

Para visualizar algunos ejemplos, podemos visitar la página del mercado de Android, donde hay una infinidad de opciones gratuitas y pagas: <https://play.google.com/store>.

Entrada – Proceso – Salida.

La **entrada** es el ingreso o comando de datos que vamos a realizar sobre un dispositivo, como, por ejemplo: tocar la pantalla, escribir, mover el puntero del mouse, hacer el movimiento con un joystick, etc.

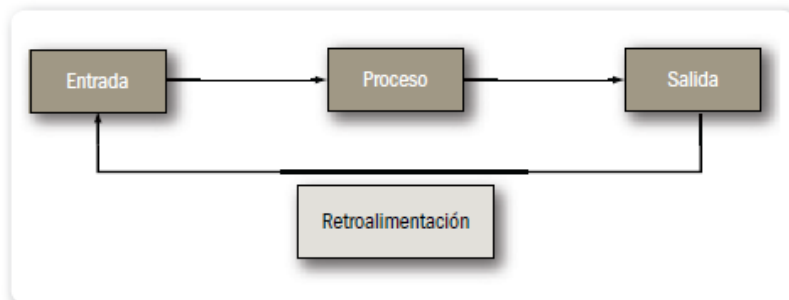
Por lo tanto, toda entrada se hará por medio de un dispositivo, como puede ser una pantalla táctil, un teclado, una webcam o un mouse.

El **proceso** es el trabajo, la interpretación y el cálculo de la información ingresada. Esta información puede ser un movimiento del mouse, una tecla pulsada, datos para calcular enviados, y otros.

Fundamentalmente, en el proceso ya entran en juego el procesador y la memoria de un dispositivo.

La **salida** es el resultado de las acciones que se efectúan sobre la información. Por lo tanto, si pulsamos el botón del mouse, se ejecutará una aplicación (pulsar el botón Enviar de un correo), se realizará una acción en un juego (como disparar), se devolverá el resultado de un cálculo, se ejecutará un video, y otras opciones más.

Este proceso de retroalimentación nos dará los mismos resultados, presionando ya sea uno o varios botones del teclado.

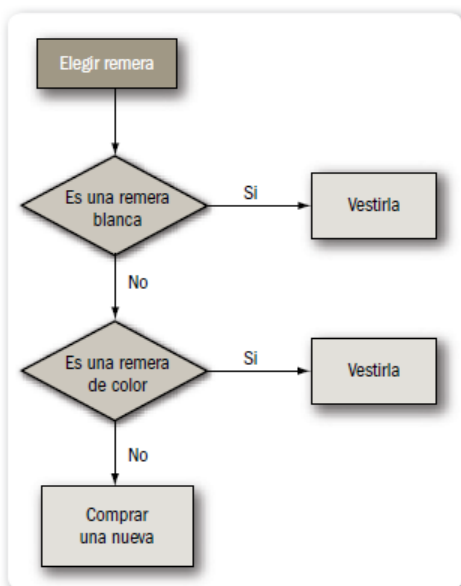


Algoritmo.

Si bien encontraremos múltiples definiciones de lo que es un algoritmo, nosotros trabajaremos con la genérica que toma la RAE, en la que se hace referencia a un conjunto ordenado y finito de operaciones que permite hallar la solución de un problema.

Nosotros, como seres humanos, tenemos incorporado un “**algoritmo**” de decisiones.

Por ejemplo, si deseamos vestir una remera, realizamos un proceso de selección de cuál o tal queremos, y terminamos por hacer la selección deseada. En un conjunto ordenado y finito de operaciones, podríamos representar, a través de un algoritmo, este proceso de selección y solución.



De esta manera, podemos definir el algoritmo como una serie de pasos ordenados que debemos seguir para lograr, finalmente, la resolución de una situación o problema.

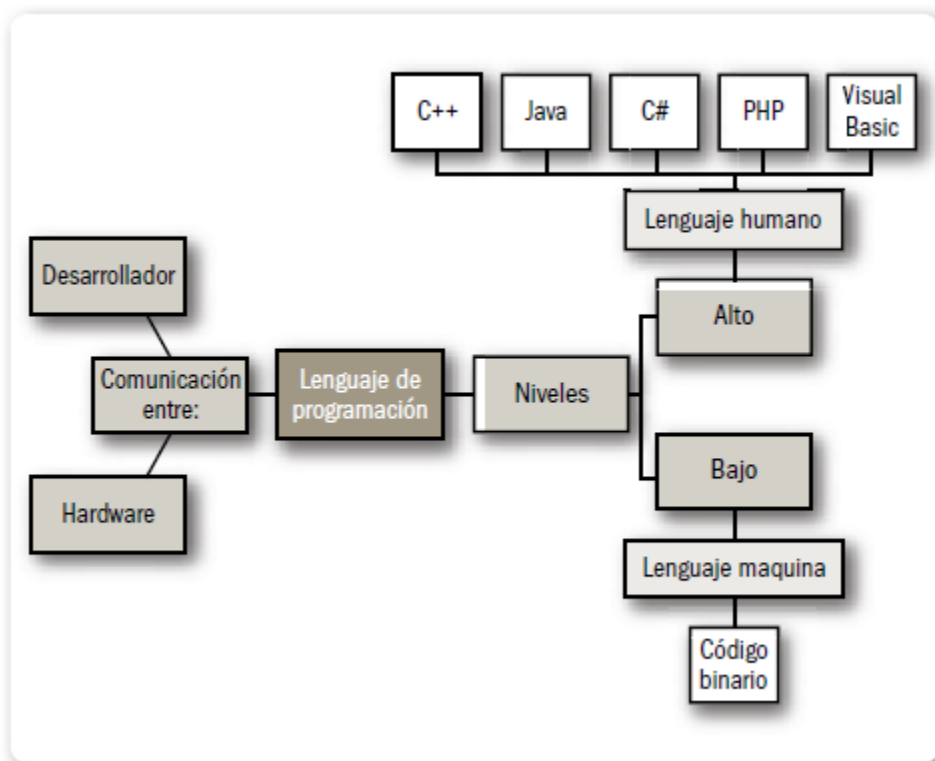
En el desarrollo, para poder ejecutar una aplicación, tenemos que traducir esto a sentencias ordenadas de código que se ejecuten línea a línea.

Lenguaje de programación.

Anteriormente presentamos la comunicación que existe entre un software y el hardware.

Ahora vamos a conocer la comunicación que debemos establecer nosotros, como desarrolladores, frente a nuestro hardware, para lograr que este ejecute las tareas o procesos que deseamos.

Para este fin, necesitaremos como herramienta primordial un **lenguaje de programación**.



Existen muchos lenguajes de programación que nos permiten desarrollar, por medio de un código (protocolo), sentencias algorítmicas que luego son traducidas a lenguaje máquina.

Estos cumplen la función de intermediarios entre el desarrollador y el hardware.

Teniendo en cuenta esta diversidad, veremos que hay dos grupos generales. Por un lado, se encuentran los lenguajes más próximos a la arquitectura del hardware, denominados lenguajes de bajo nivel (son más rígidos y complicados de aprender). Por otro lado, están aquellos más cercanos a los programadores y usuarios, denominados lenguajes de alto nivel (son más comprensibles para el lenguaje humano).

En distintos escritos se consideran lenguajes de **bajo nivel** a algunos como: FORTRAN, ASSEMBLER y C. Como lenguajes de **alto nivel** podemos mencionar: Visual Basic, Visual C++ y Python.

Si bien podemos encontrar categorizaciones más finas al respecto, que describan diferentes tipos de lenguajes, recordemos que, en términos generales, siempre se habla de lenguajes de alto nivel y de bajo nivel.

▼ LENGUAJE	▼ DESCRIPCIÓN
Máquina	Código interpretado directamente por el procesador. Las invocaciones a memoria, como los procesos aritmético-lógicos, son posiciones literales de conmutadores físicos del hardware en su representación booleana. Estos lenguajes son literales de tareas. Ver su ejemplo en la Figura 10.
Bajo nivel	Instrucciones que ensamblan los grupos de conmutadores necesarios para expresar una mínima lógica aritmética; están íntimamente vinculados al hardware. Estos lenguajes están orientados a procesos compuestos de tareas, y la cantidad de instrucciones depende de cómo haya sido diseñada la arquitectura del hardware. Como norma general, están disponibles a nivel firmware, CMOS o chipset. Ver su ejemplo en la Figura 11.
Medio nivel	Son aquellos que, basándose en los juegos de instrucciones disponibles (chipset), permiten el uso de funciones a nivel aritmético; pero a nivel lógico dependen de literales en ensamblador. Estos lenguajes están orientados a procedimientos compuestos de procesos. Este tema se verá a continuación, en el ejemplo Código C y Basic.
Alto nivel	Permiten mayor flexibilidad al desarrollador (a la hora de abstraerse o de ser literal), y un camino bidireccional entre el lenguaje máquina y una expresión casi oral entre la escritura del programa y su posterior compilación. Estos lenguajes están orientados a objetos. Ver su ejemplo en la Figura 12.

Ciclo de vida (etapas) de una resolución de un problema.

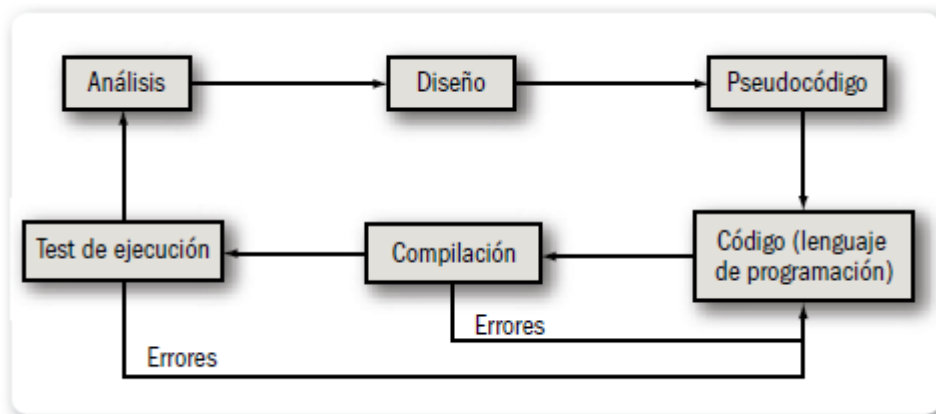
Ahora que conocemos las herramientas involucradas en el desarrollo de aplicaciones, es conveniente revisar qué tareas generales debemos considerar para llevar a cabo esta labor.

Como seres humanos, tenemos incorporada intuitivamente la resolución de problemas cotidianos gracias a nuestra experiencia, y para intentar afrontar un inconveniente, solemos hacer un proceso rápido de selección e intentamos buscar la opción más favorable.

En el ámbito laboral, y más aún en el desarrollo de aplicaciones, debemos ser cautelosos al momento de resolver alguna tarea o proceso.

Por eso, nos será de gran utilidad aplicar una herramienta del lenguaje de programación que nos permita confeccionar un programa y, así, resolver dicha situación.

Si bien este esquema nos será útil para la resolución de un desarrollo sencillo, en caso de trabajar con sistemas amplios, deberemos incluir ciertas técnicas de ingeniería del software.



En el proceso que vemos en el gráfico de la figura, puede suceder que debamos retroceder y volver a analizar o replantear algunas de las acciones. Revisemos los pasos expuestos en el esquema (y en los próximos capítulos veremos cómo se desarrolla una aplicación basada en él).

Los siguientes aspectos son pasos que seguiremos como desarrolladores para resolver una situación:

- Analizar el problema que vamos a resolver.
- Diseñar una solución.

- Traducir la solución a pseudocódigo.
- Implementar en un lenguaje de programación todo lo analizado.
- Compilar el programa.
- Realizar pruebas de ejecución.
- Corregir los errores que haya.

¿Qué es la programación?

La programación es el proceso de diseñar, escribir, probar y mantener el código fuente de un software. En términos más simples, se trata de crear instrucciones que le dicen a una computadora qué hacer.

A través de la programación, podemos crear todo tipo de software, desde aplicaciones móviles hasta juegos y sistemas de gestión de bases de datos. El objetivo de la programación es resolver problemas de manera eficiente y automatizada. En lugar de realizar una tarea repetitiva manualmente, podemos escribir un programa que la haga por nosotros. Esto no solo nos ahorra tiempo y esfuerzo, sino que también reduce el riesgo de errores humanos.

La programación se basa en un lenguaje de programación, que es un conjunto de reglas sintácticas y semánticas que permiten al programador comunicarse con la computadora. Hay muchos lenguajes de programación diferentes, cada uno con sus propias ventajas y desventajas dependiendo del uso que se le quiera dar.

Algunos de los lenguajes más populares incluyen Java, Python, JavaScript, C++, C# y Ruby.

Importancia de la programación.

La programación se ha convertido en una habilidad esencial en la era digital actual. Es una herramienta poderosa para solucionar problemas y automatizar tareas en diferentes campos, desde el entretenimiento hasta la industria, la ciencia y la medicina. En esta sección, exploraremos la importancia de la programación en nuestra sociedad actual.

Automatización de tareas

La programación permite la automatización de tareas, lo que ahorra tiempo y esfuerzo. En lugar de realizar tareas repetitivas manualmente, la programación nos permite escribir código que realiza estas tareas

automáticamente. Por ejemplo, en lugar de enviar correos electrónicos manualmente a cada cliente, se puede programar una aplicación que lo haga automáticamente.

Mejora de la eficiencia

La programación también puede mejorar la eficiencia en diferentes procesos. Las empresas pueden crear aplicaciones personalizadas que se adapten a sus necesidades, mejorando su productividad y eficiencia. Por ejemplo, los sistemas de gestión de inventario pueden ser programados para enviar alertas cuando los niveles de inventario bajan a cierto punto, lo que permite una gestión de inventario más eficiente.

Innovación y avance tecnológico

La programación es la base de la innovación tecnológica y ha sido la clave para muchos avances en diferentes campos, desde la medicina hasta la exploración espacial. Los programadores crean soluciones innovadoras que mejoran la vida de las personas y cambian el mundo. La programación también es esencial en el desarrollo de inteligencia artificial, la tecnología blockchain, y otras tecnologías emergentes que están transformando la manera en que vivimos y trabajamos.

Facilita la toma de decisiones

La programación también puede facilitar la toma de decisiones en diferentes ámbitos. Los científicos pueden programar modelos para predecir el clima, los economistas pueden programar modelos para predecir el comportamiento del mercado, y los médicos pueden utilizar algoritmos de aprendizaje automático para diagnosticar enfermedades. La programación nos permite obtener información y conocimientos que pueden ser utilizados para tomar decisiones informadas y precisas.

Creación de empleo

La programación es una habilidad altamente demandada en la economía actual, lo que significa que hay una gran cantidad de oportunidades laborales para aquellos que la poseen. La programación también es una habilidad útil para aquellos que buscan emprender, ya que pueden crear y lanzar sus propias aplicaciones y servicios.