El futuro digital es de todos

MinTIC

Universidad Industrial de Santander

MisiónTIC 2022

FUNDAMENTOS
DEL LENGUAJE PYTHON

TEMA 1:
INTRODUCCIÓN A LOS CONCEPTOS BÁSICOS DE PROGRAMACIÓN DE COMPUTADORES

1. Conceptos básicos

1.1. ¿Qué es un lenguaje de programación?

En términos generales, un lenguaje de programación es un lenguaje que involucra una serie de instrucciones que producen una serie de salidas o resultados, las cuales son usadas para el desarrollo de aplicaciones de todo tipo.

1.2. Algunos conceptos básicos de programación

Existen conceptos que son comunes a todos los lenguajes de programación independientemente de sus diferencias que pueden ser bastante grandes. Estos que aquí llamamos conceptos básicos son los siguientes:

1.2.1. Algoritmos y secuencias de comandos

para llevar a cabo tareas haciendo uso de los lenguajes de programación, se requiere el uso de una serie de **comandos** o **instrucciones**, que se pueden ver como pasos para realizar una tarea por completo. No basta con saber cuáles son los comandos y los pasos correctos que permiten llevar a cabo la tarea propuesta, sino que también se requiere saber cuál es la secuencia de esos comandos o, en otras palabras, el orden para la ejecución de cada uno de ellos. Existen muchos ejemplos en la vida cotidiana de tareas las cuales en el orden apropiado es esencial para que se tenga un resultado satisfactorio. Como por ejemplo vestirse, bañarse, preparar una receta, etc. Además de estos casos, también existen tareas que para ser llevadas a cabo no interesa el orden de ejecución, independientemente del orden de los pasos que llevan a desarrollar la tarea deseada, siempre se va a obtener el mismo resultado. Para estos casos no existe un orden apropiado, pero sí existe un orden convencional. A la correcta configuración de instrucciones que permite resolver una tarea por completo se le conoce como **algoritmo**. Los algoritmos forman un papel bastante importante en todo lo que tiene que ver con ciencias de la computación, aquí veremos cómo desarrollarlos usando un lenguaje en particular, *Python*.

1.2.2. Estructuras condicionales

En programación es muy recurrente la toma de decisiones, por lo que es de gran importancia establecer condiciones que permitan tomar las mejores decisiones de tal manera que se obtengan los mejores resultados. Este tipo de condiciones permiten realizar tareas basadas en decisiones de Sí o NO, y verdadero o falso. Las condiciones pueden ser usadas para visualizar resultados o para ejecutar una secuencia de comandos. Es como en la vida diaria, que a menudo pensamos cosas como:

- Si son las 7am del día de hoy, entonces voy a desayunar.
- Si amanece lloviendo, entonces no salgo a caminar, de otro modo me quedaré durmiendo.
- Si paso el examen entonces salgo con mis amigos, de otro modo me quedaré en casa estudiando.

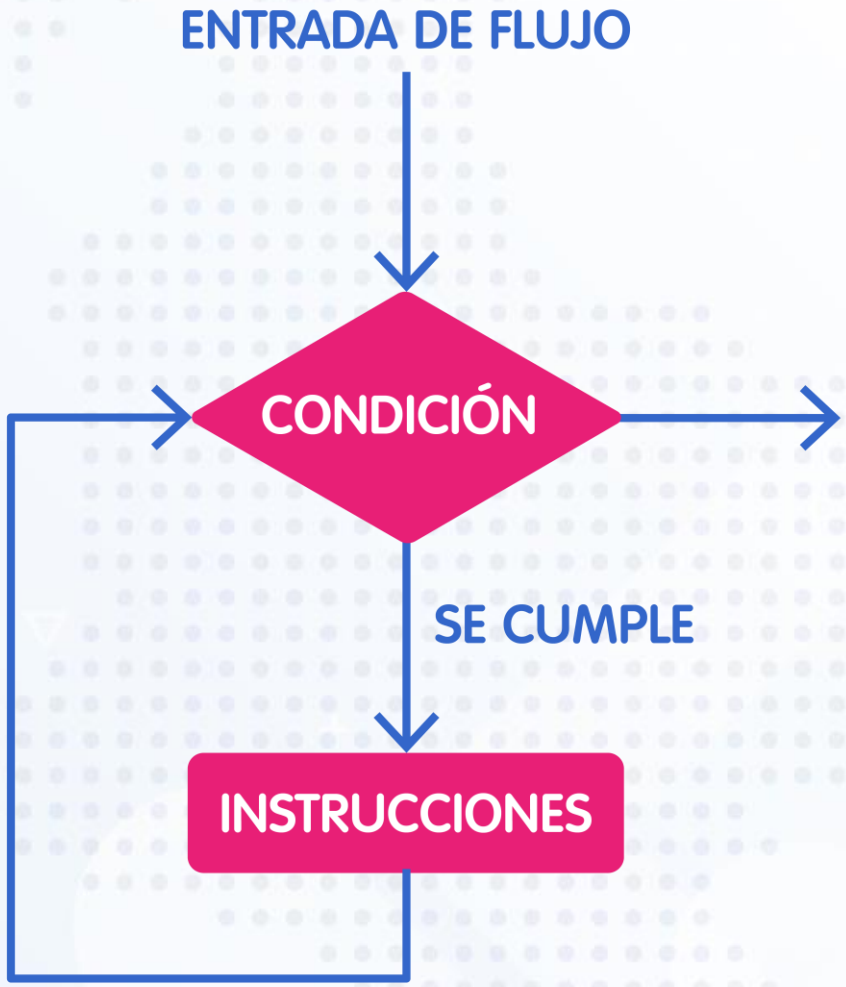
En la mayoría de lenguajes de programación existe una estructura para los condicionales. En estos casos esta estructura toma la forma de: **if** ... **then** ... **else**.

1.2.3. Estructuras de bucle

Las estructuras de bucles son las encargadas de realizar tareas o instrucciones de manera recurrente. Existen muchas tareas en la vida cotidiana que requieren repetir cierto número de veces una instrucción, como por ejemplo contar los primeros *N* números enteros, numerar las páginas de un libro, programar la alarma del celular para que suene cada hora a lo largo del día, etc. De la misma manera en cómo en la vida diaria hacemos tareas repetitivas, en programación también es necesario ejecutar de manera repetitiva ciertas órdenes.

Estas estructuras de bucle pueden ser ejecutadas:

1. Cierto número de veces.
2. Hasta que una condición se cumpla.
3. Todo el tiempo en que cierta condición permanezca verdadero.



Como ejemplo podemos encontrar:

Nos encontramos en una situación de pandemia por lo que, **mientras** exista el COVID-19, todos los días debemos usar una mascarilla.

Mi entrenamiento de atletismo, los días miércoles, consta de llevar a cabo 10 **rondas** de 100 metros planos.

Un restaurante debe preparar un plato **determinado** número de veces.

Este tipo de estructura tiene como ventaja acortar los procesos repetitivos dentro de un código o algoritmo, ya que la tarea que se repite solamente se debe escribir una sola vez dentro de este tipo de estructuras.

1.3. Características de algunos lenguajes de programación

Java

Es un lenguaje de programación orientado a objetos, bastante versátil, seguro y estable, implementado en casi todo tipo de sistemas. Este lenguaje puede ser usado para la creación de muchos aplicativos, desde aplicaciones web hasta programas para ordenadores. También es bastante usado en el entorno de desarrollo *Android* y en el desarrollo de juegos como *Minecraft*. Una característica bastante destacada es que las aplicaciones escritas en este lenguaje pueden ser ejecutadas en casi cualquier sistema independientemente del *hardware*.

Ejemplo:

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

JavaScript

Es el lenguaje que más se usa en el entorno de desarrollo web. Es un lenguaje jerarquizado, bien estructurado y de fácil lectura. Este lenguaje de programación está orientado a objetos y se puede utilizar del lado del cliente con herramientas como HTML y PHP o del lado del servidor en herramientas como nodejs. Si se desea usar del lado del cliente, este funciona directamente desde el navegador, por lo que lo único que se requiere es incluir *scripts* desarrollados en *JavaScript* dentro del desarrollo de páginas web para que el navegador los interprete.

Ejemplo:

```
<!DOCTYPE HTML>
<html>
<body>
    <p>Contenido antes del script...</p>
    <script>
        alert( 'Hello, World!' );
    </script>
    <p>Contenido después de script...</p>
</body>
</html>
```

Go

Como comentamos anteriormente, este lenguaje fue creado por Google y desde su lanzamiento se ha ido haciendo cada vez más popular. Este es un lenguaje procedural (dividido en una serie de procedimientos), busca ser tan rápido como C++, pero ser de fácil aprendizaje como *Python*. Usando este lenguaje se pueden crear distintos tipos de aplicaciones, principalmente aplicaciones web.

Ejemplo:

```
package main
import “fmt”
func main() {
    fmt.Println(“Hello, World!”)
}
```

Kotlin

Este lenguaje fue desarrollado en Rusia y actualmente es bastante usado para el desarrollo de aplicaciones *Android*, aplicaciones web, aplicaciones científicas, y aplicaciones multiplataforma (funcionalidad tanto en *Android* como en *iOS*). Este lenguaje es completamente compatible con Java, ya que su finalidad era que pudiese reemplazar a Java.

Ejemplo:

```
fun main(args: Array<String>) {
    println("Hello, World!")
}
```

Python

Es un lenguaje bastante amigable, de fácil aprendizaje y lectura. Se usa bastante en áreas como finanzas, estadística, aprendizaje de máquinas, inteligencia artificial y ciencia de datos.

Ejemplo:

```
print("Hello, World!")
```

C

Es un lenguaje de programación de bajo nivel creado para diseñar sistemas operativos. El núcleo de Linux se encuentra desarrollado en C. Es bastante rápido y eficiente, también utilizado en el ámbito científico.

Ejemplo:

```
#include <stdio.h>
int main() {
    printf("Hello, World!");    return 0;
}
```

C ++

Es un lenguaje derivado del lenguaje de programación C. Incluye programación orientada a objetos por lo que es más sencillo de usar que C. Se usa también para crear sistemas operativos como es el caso de parte de Windows y Linux, usado además para el desarrollo de videojuegos y muchos otros tipos de aplicaciones.

Ejemplo:

```
#include <iostream>
int main() {
    std::cout << "Hello World!";
    return 0;
}
```

Ruby

Ejemplo:

```
puts "Hello World!"
```

Lenguaje de programación dinámico, enfocado en la simplicidad y productividad; de fácil lectura y escritura. Es usado principalmente para desarrollar aplicaciones web. Parte del código de aplicaciones como *Twitter*, *GitHub*, *Airbnb,* es desarrollado usando Ruby. Para un mejor desempeño en cuanto al desarrollo de aplicaciones web, Ruby es utilizado como lenguaje base dentro del framework Ruby on Rails.

2. Evolución de los lenguajes de programación

Existen una gran variedad de lenguajes de programación, desde lenguajes viejos hasta lenguajes nuevos, desde aquellos que se encuentran a un nivel más cercano a la máquina, hasta aquellos que no. Todos estos lenguajes han ido evolucionando con el tiempo y se han ido adaptando a diferentes tipos de dispositivos. Vemos que hoy en día casi todas las cosas funcionan porque alguien las programó para realizar ciertas tareas específicas, las aplicaciones de computadores, las aplicaciones en los televisores, las aplicaciones móviles, las aplicaciones que se encuentran en la nube, portales web, etc.

A continuación, se muestra la evolución de los lenguajes de programación, desde el primer lenguaje hasta los lenguajes que conocemos hoy en día.

1801: Se inventó el **telar programable**, por Joseph Marie Jacquard. Este telar recibía unas tarjetas programables que permitían automatizar procesos, de tal manera que se podía generar diferentes tejidos.

1843: Ada Lovelace inventó el primer lenguaje de programación para una máquina analítica creada por Charles Babbage.

1950: Se desarrolló el lenguaje **Ensamblador**, el cual es un lenguaje de bajo nivel en donde hay una correspondencia bastante fuerte entre las instrucciones del lenguaje y las instrucciones del código de máquina de la arquitectura.

1957: John Backus crea FORmulation TRANslation (**FORTRAN**). Este lenguaje fue creado para la computación científica de alto nivel, matemática y estadística.

1987: Larry Wall crea Perl (Practical Extraction Report Language).

1991: **Python** es creado por Guido Van Rossum para la solución de problemas en el lenguaje ABC.

1991: Alan Cooper desarrolla para Microsoft el lenguaje **Visual Basic**. En su tiempo tuvo un gran impacto por su fácil entendimiento.

1993: Ruby es creado por Yukihiro Matsumoto. **Ruby** es creado utilizando partes de otros lenguajes de programación como Perl, Smalltalk, Eiffel, Ada y Lisp.

1995: **PHP** es creado por Rasmus Lerdoff. El propósito principal de PHP era el reemplazo de algunos scripts desarrollados en **Perl** que era usado para mantener un sitio web. Actualmente este lenguaje hace parte de muchas páginas web.

En este mismo año es creado **Java** por parte de unos desarrolladores de Sun Microsystems, liderados por James Gosling. Actualmente este lenguaje está presente desde aplicaciones de escritorio hasta aplicaciones web y móvil.

Además de estos dos lenguajes, fue creado **JavaScript** por Brendan Eich, el cual en sus inicios se llamó Mocha. Actualmente es bastante usado en el desarrollo web, y muchos frameworks lo tienen integrado.

2000: El diseñador de los lenguajes **Turbo Pascal** y **Delphi**, Anders Hejlsberg, crea **C#**. Su fin era ser un lenguaje de programación orientado a objetos de uso general y que fuese sencillo.

2004: Es creado **Scala**, un lenguaje de programación moderno que fue diseñado para ser un lenguaje escalable.

2006: Se crea **Scratch**, un lenguaje de programación visual, desarrollado en el MIT. Su fin es que niños, jóvenes y adultos, que no posean idea alguna de programación, puedan aprender a programar de manera visual.

2009: Google desarrolla **Go** o **Golang**. Este lenguaje se encuentra inspirado en **C**, pero es mucho más complejo y su objetivo principal es la seguridad.

2012: Es creado **Kotlin** por **JetBrains**. Es un lenguaje de programación que corre sobre la máquina virtual de Java y se ha hecho bastante popular en el entorno de desarrollo Android.

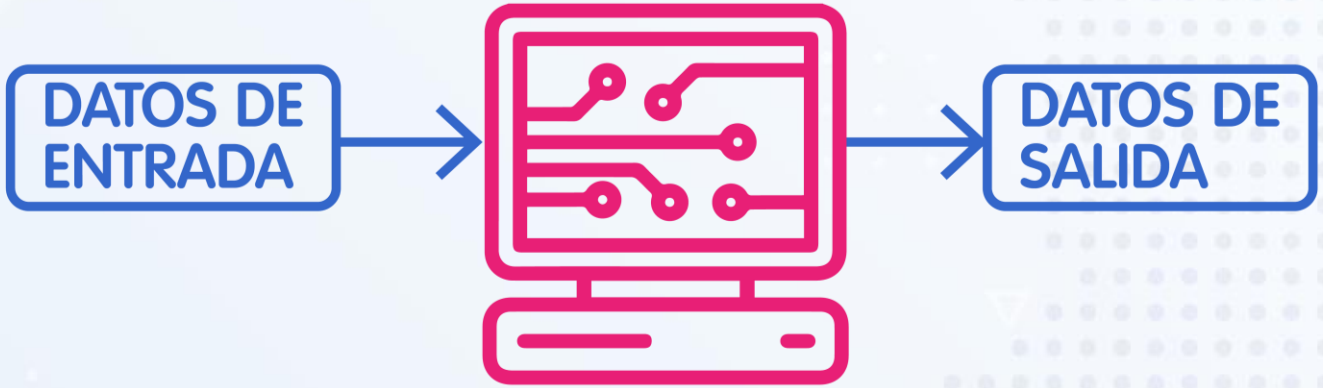
2013: Apple Inc. crea **Swift**, un lenguaje de programación pensando en agilizar la programación. Este lenguaje de programación es hoy en día clave para el desarrollo de iOS.

3. Composición de un sistema computacional

Las computadoras u ordenadores se pueden definir como **procesadores de datos**. Este concepto es demasiado general y hace que este tipo de máquinas se vean como una caja negra, esto si no se tiene conocimiento de cómo ellas procesan la información (Fig. 1). Dentro de este concepto de procesadores de datos también entran las calculadoras, los celulares, los relojes inteligentes, etc. Lo que quiere decir que a un amplio número de dispositivos se les puede llamar procesadores de datos. Como por ejemplo:

- Se introduce en una computadora una imagen y esta es capaz de procesar esa información y visualizarla en la pantalla.
- Se le pasan unos números a una calculadora para que esta realice determinada operación.

Esta definición es demasiado general, de tal manera que no se sabe si el sistema del que se habla es un sistema de propósito general o de propósito específico.



Existen sistemas, como por ejemplo, un procesador de datos, programado específicamente para el monitoreo de los niveles de agua de un río, lo que quiere decir que fue diseñado para una tarea específica. Pero también existen otros tipos de procesadores de datos, como es el caso de una computadora, que por el contrario es un sistema de propósito general, que permite realizar una enorme cantidad de tareas.

Las computadoras además de que son procesadores de datos también son sistemas programables, por lo que también se les conoce como procesadores de datos programables. A este tipo de sistemas se les puede pasar un conjunto de instrucciones escritas en determinado lenguaje que entienda la máquina, lo que constituye una aplicación, de tal manera que pueda llevar a cabo cualquier tarea deseada sobre determinados datos de entrada (Fig. 2). Los datos de salida van a depender de dos factores, uno, los datos de entrada y dos, de la aplicación que los procese. Se pueden generar distintas salidas con los mismos datos de entrada, esto cambiando la aplicación; pero también se pueden generar distintos datos de salida usando la misma aplicación; esto cambiando los datos de entrada.

Una computadora u ordenador es una máquina que constituye un sistema computacional (Fig. 3). Los sistemas computacionales son sistemas que permiten almacenar y procesar información y se encuentran conformados por un conjunto de elementos funcionales que se interrelacionan entre sí. Estos elementos son:

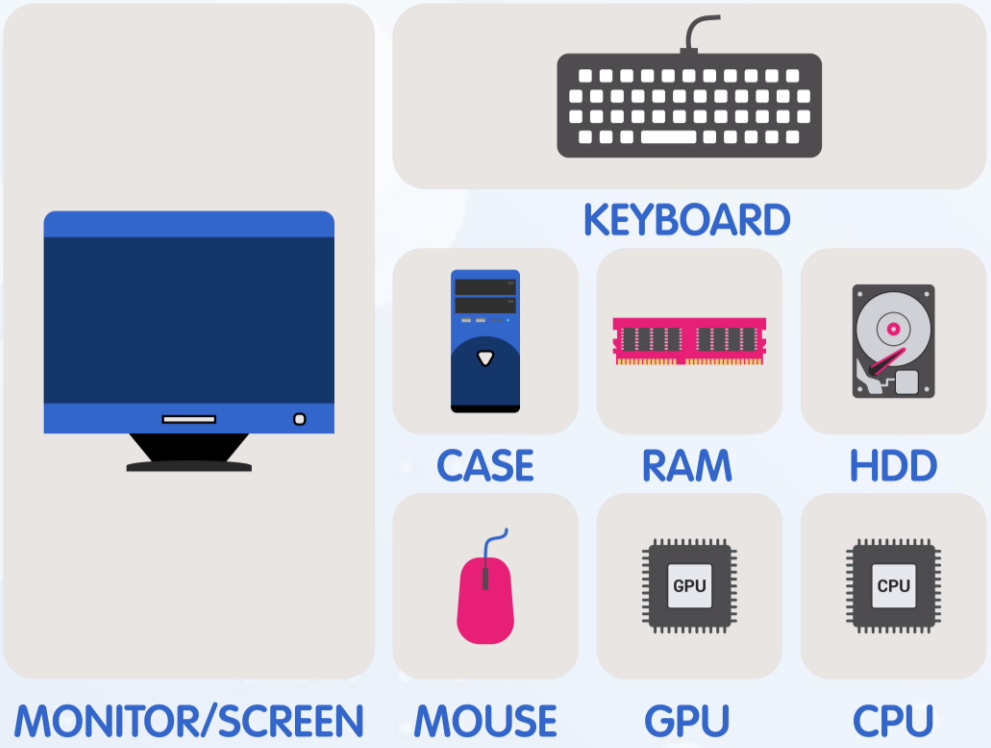
•El *hardware* (computadora)

•El sistema operativo (Windows, Linux, iOS)

•Las aplicaciones (Word, Blender, Photoshop, Safari, Firefox)

•Los usuarios

3.1. El hardware



El *hardware* de una computadora se puede decir que lo conforman todas sus partes físicas, entre las cuales se encuentran, la Unidad Central de Procesos (CPU), la memoria RAM, la placa madre, discos de almacenamiento, y los periféricos como el ratón, el teclado, la cámara, el micrófono, etc.

La Unidad Central de Procesos (CPU) ha pasado por un proceso evolutivo bastante importante; tanto así, que hoy en día la mayoría de personas carga una en su bolsillo. Esta constituye el núcleo fundamental de una computadora, cuya función es la de procesar información y de ejecutar todas las instrucciones que comprometen el sistema operativo y sus aplicaciones, de lo cual hablaremos más adelante.

En la actualidad se usa el modelo de Von Neumann para estudiar el interior de las computadoras, el cual permite definir como esta lleva a cabo el procesamiento. Este modelo describe una computadora a partir de 4 subsistemas:

1. ***1.La memoria***, es la zona en donde se almacena tanto las aplicaciones como sus datos durante el procesamiento.

2. ***1.La Unidad Aritmética Lógica (ALU),*** es un circuito digital encargado de realizar operaciones aritméticas como suma, resta y operaciones lógicas como por ejemplo encontrar el menor entre dos elementos de ciertos datos.

3. ***La unidad de Control (CU),*** es una componente de la CPU cuya tarea es la de direccionar las operaciones del procesador.

4. ***1.Entrada/Salida,*** compuesta por dos subsistemas, uno de entrada y otro de salida. El subsistema de entrada acepta los datos y las aplicaciones que son ingresados al sistema desde el exterior; el subsistema de salida envía el resultado del procesamiento de los datos hacia el exterior. Un dispositivo de entrada/salida puede ser un disco duro o una memoria USB, si los datos son leídos entonces este es un dispositivo de entrada, pero si los datos obtenidos como resultado después del procesamiento son almacenados en este dispositivo entonces se dice que es de salida.

En el modelo de Von Neumann los programas deben almacenarse en memoria, al igual que sus datos, por lo que ambos deben estar en el mismo formato, formato binario, que comprende una secuencia de ceros y unos.

3.2. El sistema operativo

El sistema operativo es el sistema de *software* encargado de manejar el *hardware* que compone un ordenador, los recursos de *software* y provee servicios en común para los programas o aplicaciones. En términos generales, el sistema operativo es un intermediario entre las aplicaciones y el *hardware* que componen el ordenador.

Windows

iOS

Los sistemas operativos están presentes en una serie de dispositivos tales como ordenadores personales, ordenadores portátiles, servidores, consolas de videojuegos, celulares y relojes inteligentes. Entre los sistemas operativos encontrados en ordenadores personales se destacan Microsoft Windows, macOS de Apple Inc., y la gran variedad de distribuciones de Linux. Entre los dispositivos móviles, como lo son los teléfonos móviles y los relojes inteligentes, encontramos sistemas como Android y Apple iOS. Entre los servidores predominan los sistemas Linux, pero también se usa la edición de servidor de Microsoft, Windows Server.

3.3. Las aplicaciones

Las aplicaciones son *software* diseñados a partir de un conjunto de instrucciones para llevar a cabo una tarea específica. Estas se almacenan en la memoria de la computadora, al igual que los datos. Las aplicaciones se pueden encontrar en todo tipo de *hardware* como son los ordenadores personales, dispositivos móviles, servidores, etc., y son desarrolladas típicamente para ser usadas por un usuario final. Entre las aplicaciones existentes podemos encontrar procesadores de texto, programas de hojas de cálculo, navegadores web, reproductores de medios, juegos de video, editores de imágenes/videos, etc.

Las aplicaciones se pueden clasificar:

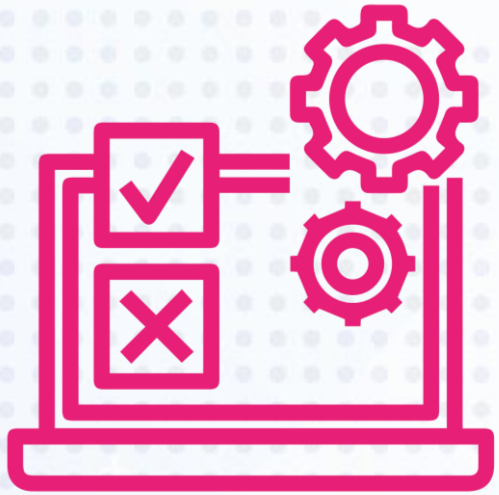
- Por derecho de propiedad y derechos de uso, en donde encontramos *software* de código cerrado vs. *software* de código abierto y *software* libre vs *software* privativo.

- Por lenguaje de programación, en donde encontramos aplicaciones web, escritas en HTML, JavaScript y otras tecnologías web, y aplicaciones nativas, escritas en cualquier tipo de lenguaje disponible en el sistema operativo para el cual esté diseñada.

- Por propósito y salida, en donde se pueden encontrar aplicaciones verticales enfocadas en las necesidades de un angosto rango de industrias, y aplicaciones horizontales las cuales son de propósito general, usadas en un amplio rango de industrias.

3.4. Los usuarios

Los usuarios son las personas que finalmente usan el *software* desarrollado o la aplicación. Estos usuarios típicamente no poseen los conocimientos técnicos del producto, pero estas aplicaciones son desarrolladas de tal manera que su uso sea lo más amigable posible para ellos.



Material de estudio complementario

History of programming languages
Arquitectura von Neumann

Referencias Bibliográfica

Horowitz, E. (1984). The Evolution of Programming Languages. In *Fundamentals of Programming Languages* (pp. 1-31). Springer, Berlin, Heidelberg.