REFUERZO DE TEMAS Y CONCEPTOS

ALAN GABRIEL GUTIÉRREZ PRADA

SENA

INSTRUCTOR: ARLE MORALES

CALARCÁ, COLOMBIA

FICHA: 3144886

FEBRERO DE 2025

1. ¿QUÉ ES UN COMPUTADOR?

Un computador es una máquina electrónica capaz de realizar operaciones matemáticas, lógicas, de almacenamiento y comunicación de información. Los computadores tienen diversos componentes como la CPU, la memoria, los dispositivos de entrada y salida que les permiten procesar datos.

Ejemplo: Una computadora de escritorio utilizada para escribir documentos, navegar en internet y jugar.

2. ¿QUÉ ES UN PROGRAMA?

Un programa es un conjunto de instrucciones que le indican a la computadora qué debe hacer para llevar a cabo una tarea específica.

Ejemplo: Un editor de texto como Microsoft Word.

3. LÓGICA DE PROGRAMACIÓN

Es el proceso de estructurar el pensamiento para resolver problemas de manera clara y ordenada. La lógica de programación se basa en descomponer un problema complejo en pasos simples y ordenados.

Ejemplo: Diseñar un flujo de decisión para un cajero automático.

Analogía: Es como si tuvieras que organizar una fiesta. Primero, debes pensar en los detalles más importantes, como la comida, la música, y los invitados. Luego, organizas cada detalle en un orden lógico para que todo salga bien.

4. ALGORITMOS

Un algoritmo es un conjunto de pasos o instrucciones que se siguen para resolver un problema. Los algoritmos pueden escribirse en pseudocódigo antes de implementarlos en un lenguaje de programación.

Ejemplo:

```
Inicio
    Escribir "Ingresa el primer número:"
    Leer num1
    Escribir "Ingresa el segundo número:"
    Leer num2
    suma ← num1 + num2
```

```
    Escribir "La suma es:", suma
Fin
```

5. VARIABLES

Las variables son contenedores que almacenan datos que pueden cambiar durante la ejecución del programa.

Tipos de variables:

- Enteros (números sin decimales)
- Decimales (números con decimales)
- Texto (cadenas de caracteres)

Ejemplo:

```
nombre = "Juan"
edad = 25
altura = 1.75
```

6. TIPOS DE DATOS

Los tipos de datos se refieren al tipo de información que puede almacenar una variable.

- **Enteros (int)**: Números sin decimales (ej. 10, -3).
- **Flotantes (float)**: Números con decimales (ej. 3.14, -2.5).
- **Cadenas (str)**: Texto (ej. "Hola", "Python es genial").
- **Booleanos (bool)**: Valores de verdadero o falso (True, False)

Ejemplo:

```
edad = 30                # Entero
precio = 19.99           # Flotante
nombre = "Ana"           # Cadena
es_estudiante = True     # Booleano
```

7. OPERADORES

Los operadores son símbolos que permiten realizar operaciones matemáticas, lógicas y comparativas.

Operadores aritméticos:

+ (suma)
- (resta)
* (multiplicación)
/ (división)

Operadores lógicos:

AND (y), OR (o), NOT (no)

Ejemplo:

```
suma = 5 + 3      # Resultado: 8

es_mayor = (10 > 5) and (5 > 2)  # True
```

8. ESTRUCTURAS DE CONTROL

Las estructuras de control permiten que el flujo de ejecución de un programa dependa de una condición. Las más comunes son las estructuras condicionales (if, else).

Ejemplo:

```
edad = 18
if edad >= 18:
    print("Eres mayor de edad")
else:
    print("Eres menor de edad")
```

9. BUCLES

Un bucle permite ejecutar repetidamente un bloque de código mientras se cumpla una condición.

Tipos de bucles:

- **Bucle for**: repite el código un número determinado de veces.
- **Bucle while**: repite el código mientras se cumpla una condición.

Ejemplo:

```
for i in range(5):
    print (i)   # Imprime los números del 0 al 4
```

10. ENTRADA Y SALIDA

La entrada es el proceso de leer datos del usuario y la salida es el proceso de mostrar información al usuario.

Ejemplo de entrada y salida:

```
nombre = input ("¿Cuál es tu nombre?")

print ("Hola, {nombre}")
```

11. FUNCIONES BÁSICAS

Las funciones permiten organizar el código en bloques reutilizables, facilitando la lectura y el mantenimiento del programa.

Ejemplo de uso:

```
function saludar(nombre) {
    console.log (`Hola, ${nombre}`);
}

saludar("Juan");
```

12. PRIMER LENGUAJE (EJ. PYTHON)

Python es un lenguaje de programación de alto nivel que es sencillo y versátil.

Ejemplo de sintaxis básica:

```
print("Hola Mundo")
```

Este código muestra el texto "Hola, mundo" en pantalla.

Analogía: Un lenguaje de programación es como un idioma en el que te comunicas con la computadora para que haga lo que necesitas. Python es como el español simple y claro de la programación.

13. COMENTARIOS EN EL CÓDIGO

Son anotaciones dentro del código que ayudan a explicar su funcionamiento sin afectar la ejecución del programa. Se utilizan para que otros programadores (o tú mismo en el futuro) comprendan mejor el código.

Ejemplo:

```
# Esto es un comentario
print("Hola Mundo")  # Este código imprime un mensaje
```

Analogía: Es como dejar notas adhesivas en un libro para recordar información importante.

14. ERRORES COMUNES – COMO IDENTIFICARLOS

Definición: Son problemas que pueden ocurrir al escribir o ejecutar código. Los más comunes son:

- **Errores de sintaxis:** Ocurren cuando se escribe mal una instrucción.
- **Errores en tiempo de ejecución:** Se producen cuando el programa intenta hacer algo inválido (ej. dividir entre cero).
- **Errores lógicos:** Suceden cuando el programa no hace lo esperado debido a un problema en la lógica.

Ejemplo de error de sintaxis:

```
print "Hola"   # Falta paréntesis, debe ser print("Hola")
```

Analogía: Es como escribir mal una palabra en un mensaje de texto; el destinatario podría no entender lo que querías decir.

Cómo identificarlos:

- **Errores de sintaxis:**
    - El compilador o intérprete muestra mensajes de error indicando dónde está el problema.
    - Aparecen advertencias en editores de código como VS Code o PyCharm.
- **Errores en tiempo de ejecución:**
    - El programa se detiene abruptamente y muestra un mensaje de error en la consola.
    - Se pueden manejar usando try-except en lenguajes como Python.

- **Errores lógicos:**
    - El programa se ejecuta sin errores, pero los resultados no son los esperados.
    - Se detectan probando el código con diferentes casos y depurándolo paso a paso.

15. DEPURACIÓN

Definición: Es el proceso de encontrar y corregir errores en un programa. Se puede hacer manualmente con mensajes de depuración (print()) o con herramientas especializadas como depuradores.

Ejemplo de depuración con print():

```
x = 10
print("Valor de x:", x)  # Ayuda a verificar el valor de x
```

Analogía: Es como revisar una receta si un platillo no salió bien; vuelves a cada paso para encontrar qué salió mal.

16. COMPILADORES VS. INTÉRPRETES

- Un **compilador** traduce todo el código de un programa a lenguaje máquina antes de ejecutarlo (ej. C, Java).
- Un **intérprete** traduce y ejecuta línea por línea (ej. Python, JavaScript).

Ejemplo:

Python usa un **intérprete**, mientras que C usa un **compilador**.

Analogía:

- Un compilador es como traducir todo un libro antes de leerlo.
- Un intérprete es como un traductor simultáneo que traduce en tiempo real.

17. CADENAS DE TEXTO

Son secuencias de caracteres usadas para representar palabras, frases o cualquier tipo de texto.

Ejemplo:

```
mensaje = "Hola, mundo"
print(len(mensaje))  # Muestra la longitud de la cadena
```

Analogía: Es como una tira de letras en una máquina de escribir; cada letra tiene una posición y se pueden manipular como una secuencia.

18. HARDWARE BÁSICO

Son los componentes físicos de una computadora. Los principales son:

- **CPU (Procesador):** Es el cerebro del computador.
- **Memoria RAM:** Guarda datos temporales para que el sistema funcione rápido.
- **Disco duro/SSD:** Almacena información permanentemente.

Analogía:

- La CPU es como el cerebro.
- La RAM es como una mesa de trabajo donde colocas lo que necesitas en el momento.
- El disco duro es como un archivador donde guardas documentos para usarlos después.

19. SOFTWARE

Son los programas y sistemas que permiten que el hardware funcione. Se divide en dos tipos:

- **Sistema operativo:** Controla el hardware y permite ejecutar programas (ej. Windows, Linux).
- **Aplicaciones:** Programas específicos como navegadores, editores de texto, juegos, etc.

Ejemplo:
Microsoft Word es un software de aplicación, mientras que Windows es un sistema operativo.

Analogía:

- El hardware es como un cuerpo humano.
- El software es como la mente y los conocimientos que le permiten funcionar.

20. SISTEMAS OPERATIVOS

Son programas esenciales que gestionan los recursos del hardware y permiten a los usuarios interactuar con la computadora. Los más comunes son:

- **Windows:** Popular en PC.
- **macOS:** Usado en computadoras Apple.
- **Linux:** Sistema de código abierto utilizado en servidores y desarrollo.

Ejemplo:
Windows permite abrir archivos, navegar en internet y usar aplicaciones.

Analogía: Un sistema operativo es como el gerente de una empresa: organiza los recursos para que todo funcione correctamente.

21. ARCHIVOS Y CARPETAS

- **Archivos:** Son unidades de almacenamiento de información (documentos, imágenes, programas).
- **Carpetas:** Son contenedores que organizan archivos dentro del sistema.

Ejemplo:
Un archivo documento.txt dentro de la carpeta Mis Documentos.

Analogía: Un archivo es como una hoja de papel con información, y una carpeta es como un archivador donde organizamos varios documentos.

22. TERMINAL O CONSOLA

Definición: Es una interfaz que permite escribir comandos directamente para interactuar con el sistema operativo.

Ejemplo de comandos básicos:

- cd (cambiar directorio).
- ls o dir (listar archivos en la carpeta actual).

Ejemplo en Windows:

cd Escritorio
dir

Analogía: Es como dar órdenes directas a un asistente en vez de usar botones en una interfaz gráfica.

23. FUNDAMENTOS DE DESARROLLO DE SOFTWARE

Es el conjunto de principios y técnicas utilizadas para diseñar, crear y mantener programas de computadora. Incluye la planificación, escritura de código, pruebas y mantenimiento.

Ejemplo: Crear una aplicación móvil que permita a los usuarios pedir comida en línea.

Analogía: Desarrollar software es como construir una casa: primero se diseña el plano, luego se construye y finalmente se realizan ajustes y mantenimiento.

24. CICLO DE VIDA DEL SOFTWARE

Son las etapas por las que pasa un software desde su concepción hasta su retiro. Las fases principales son:

1. **Planificación:** Se define qué se quiere hacer.
2. **Análisis y diseño:** Se estructura cómo funcionará.
3. **Desarrollo:** Se programa el software.
4. **Pruebas:** Se verifica que funcione correctamente.
5. **Implementación:** Se lanza para los usuarios.
6. **Mantenimiento:** Se hacen mejoras y correcciones.

Ejemplo: Un equipo de desarrollo crea una app de clima. Primero investigan qué funciones incluir, luego diseñan la interfaz, programan las funciones, prueban errores, la lanzan en la tienda de apps y después publican actualizaciones.

Analogía: Es como fabricar un automóvil: primero se diseña, luego se ensambla, se prueba, se pone en venta y con el tiempo se le da mantenimiento.

25. REQUISITOS

Son las características y funciones que el software debe cumplir según lo que necesitan los usuarios.

Ejemplo: Un sistema de ventas puede requerir opciones para agregar productos al carrito, generar facturas y aceptar pagos en línea.

Analogía: Son como los ingredientes en una receta: definen qué se necesita para hacer el platillo correctamente.

26. PROTOTIPOS

Son modelos iniciales de una aplicación o programa que ayudan a visualizar cómo funcionará antes de su desarrollo completo.

Ejemplo: Un diseñador crea un prototipo de una tienda en línea con botones y pantallas antes de programarla.

Analogía: Es como hacer un boceto antes de pintar un cuadro.

27. INTERFAZ DE USUARIO

Es la parte visual de un software con la que los usuarios interactúan. Debe ser clara, intuitiva y fácil de usar.

Ejemplo: La pantalla de inicio de Facebook con el botón de "Me gusta", la barra de búsqueda y el feed de publicaciones.

Analogía: Es como el tablero de control de un automóvil: debe ser fácil de entender y usar.

28. PRUEBAS

Son procedimientos para verificar que un software funcione correctamente y no tenga errores antes de su lanzamiento.

Ejemplo: Un equipo de testers revisa si una aplicación bancaria procesa pagos correctamente y no permite transferencias incorrectas.

Analogía: Es como revisar un auto antes de venderlo, asegurándose de que el motor y los frenos funcionen bien.

29. ¿QUÉ ES UNA BASE DE DATOS?

Es un sistema que permite almacenar, organizar y gestionar información de forma estructurada para que pueda ser consultada fácilmente.

Ejemplo: Una tienda usa una base de datos para guardar información de clientes, productos y ventas.

Analogía: Es como una biblioteca digital en la que puedes buscar información en segundos.

30. INTERNET - CÓMO FUNCIONA A NIVEL BÁSICO

Es una red mundial de computadoras interconectadas que permite la transmisión de información mediante protocolos como HTTP y TCP/IP.

Ejemplo: Cuando visitas una página web, tu computadora solicita información a un servidor y recibe la respuesta para mostrar el contenido.

Analogía: Es como una oficina de correos digital donde las computadoras envían y reciben mensajes constantemente.

31. DIRECCIONES IP

Son identificadores numéricos únicos que se asignan a cada dispositivo conectado a Internet para diferenciarlo de los demás.

Ejemplo: La dirección IP 192.168.1.1 puede ser la de un router en una red local.

Analogía: Es como la dirección de tu casa en una ciudad: permite que la información llegue a tu dispositivo correctamente.

32. NAVEGADORES

Son programas que permiten acceder y visualizar páginas de internet. Ejemplos: Google Chrome, Firefox, Safari.

Ejemplo: Usar Chrome para buscar recetas en Google.

Analogía: Es como una ventana al mundo digital que te permite explorar la web.

33. CLIENTE Y SERVIDOR

- **Cliente:** Es el dispositivo o programa que solicita información (ej. un navegador web).
- **Servidor:** Es la computadora que almacena y entrega la información solicitada.

Ejemplo: Cuando abres YouTube en tu navegador (cliente), este solicita un video al servidor de YouTube, que lo envía para que puedas verlo.

Analogía: Es como pedir comida en un restaurante: el mesero (cliente) hace el pedido y la cocina (servidor) lo prepara y entrega.

34. SEGURIDAD INICIAL

Son las prácticas básicas para proteger información y dispositivos contra ataques o accesos no autorizados.

Ejemplo: Usar contraseñas seguras, activar la autenticación en dos pasos y no abrir correos sospechosos.

Analogía: Es como cerrar con llave tu casa y poner una alarma para evitar robos.

35. HTML

Definición: Es un lenguaje de marcado que se usa para definir la estructura y contenido de las páginas web.

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi página</title>
  </head>
  <body>
    <h1>¡Hola, mundo!</h1>
  </body>
</html>
```

Esto crea una página con un título y un encabezado.

Analogía: HTML es como el esqueleto de un edificio: define su estructura.

36. CSS

Es un lenguaje que se usa para darle estilo a las páginas web, cambiando colores, fuentes y diseños.

Ejemplo:

```
h1 {
  color: blue;
  font-size: 24px;
}
```

Esto hace que los títulos (h1) sean azules y más grandes.

Analogía: Es como la pintura y la decoración de una casa después de construirla con HTML.

37. JAVASCRIPT INTRODUCTORIO

Es un lenguaje de programación que permite agregar interactividad a las páginas web.

Ejemplo:

```
alert("¡Bienvenido a mi página!");
```

Esto muestra una alerta en la pantalla del usuario.

Analogía: Es como los motores y mecanismos en un juguete que hacen que se mueva.

38. PÁGINAS ESTÁTICAS

Son sitios web cuyo contenido no cambia para cada usuario y se muestra tal como fue escrito en el código HTML. No tienen interactividad dinámica sin actualizar la página.

Ejemplo: Un sitio web de una empresa que solo muestra información sobre sus productos y contacto.

Analogía: Es como un folleto impreso: el contenido es siempre el mismo, sin importar quién lo lea.

39. HOSTING BÁSICO

Es el servicio que permite almacenar y hacer accesible una página web en Internet. Un servidor web aloja los archivos del sitio y los entrega a los usuarios cuando visitan la página.

Ejemplo: Usar GitHub Pages o Netlify para subir un sitio web estático.

Analogía: Es como alquilar un espacio en un centro comercial para abrir una tienda: necesitas un lugar donde mostrar tu contenido.

40. EDITORES DE CÓDIGO

Son programas diseñados para escribir y editar código de manera eficiente. Algunos incluyen resaltado de sintaxis, autocompletado y depuración.

Ejemplo: VS Code, Sublime Text y Notepad++.

Analogía: Es como un procesador de texto, pero especializado en escribir código.

41. CONTROL DE VERSIONES

Es un sistema que permite hacer seguimiento de los cambios en el código de un proyecto, facilitando la colaboración y recuperación de versiones anteriores.

Ejemplo: Git es un sistema de control de versiones usado en proyectos de software.

Analogía: Es como guardar copias de seguridad de un documento cada vez que lo editas, para poder volver a versiones anteriores si es necesario.

42. REPOSITORIOS

Son espacios donde se almacena código fuente y su historial de cambios. Pueden estar en local (en la computadora) o en la nube (como GitHub o GitLab).

Ejemplo: Un proyecto en GitHub con el código de una aplicación.

Analogía: Es como una biblioteca donde se guardan diferentes versiones de un libro para poder consultarlas cuando sea necesario.

43. LÍNEA DE COMANDOS

Es una interfaz que permite escribir comandos en lugar de usar un entorno gráfico, facilitando tareas avanzadas y automatización.

Ejemplo de comandos básicos:

- **cd** (cambiar directorio).
- **mkdir** (crear una carpeta).
- **ls** o **dir** (listar archivos).

Analogía: Es como hablar directamente con un asistente en lugar de usar botones en un menú.

44. ENTORNOS DE DESARROLLO

Son herramientas y configuraciones necesarias para programar en un lenguaje específico. Pueden incluir un editor de código, un compilador o intérprete y un depurador.

Ejemplo: Instalar Python y configurar VS Code para programar en él.

Analogía: Es como tener un escritorio de trabajo bien organizado con todo lo necesario para hacer una tarea.

45. METODOLOGÍA ÁGIL

Definición: Es un enfoque de desarrollo de software que se basa en la entrega rápida de funcionalidades, el trabajo en equipo y la adaptación a cambios.

Ejemplo: Scrum es una metodología ágil donde los equipos trabajan en ciclos cortos llamados "sprints".

Analogía: Es como construir una casa en etapas: primero haces los cimientos, luego las paredes, luego el techo, y en cada fase puedes hacer ajustes si es necesario.

46. DOCUMENTACIÓN

La documentación es el conjunto de explicaciones y notas que describen cómo funciona un programa o sistema, también pueden ser archivos o comentarios dentro del código que explican su propósito, funciones y cómo usarlo. Es crucial para que otros programadores o incluso tú mismo en el futuro entiendan el código.

Ejemplo: Comentando el código o creando un manual para el usuario puede facilitar la comprensión del software y su mantenimiento.

```
def sumar(a, b):
    """Esta función suma dos números y devuelve el resultado."""
    return a + b
```

Analogía: Es como un manual de instrucciones para que otros puedan entender y usar el código fácilmente.

47. RESOLUCIÓN DE PROBLEMAS

La resolución de problemas en programación es el proceso de dividir un problema complejo en partes más pequeñas y manejables. Este enfoque permite encontrar soluciones más efectivas.

Ejemplo: Si necesitas hacer una aplicación que calcule la suma de varios números, primero divídelos en pasos: leer los números, sumarlos, y luego mostrar el resultado.

48. COMUNICACIÓN

La comunicación en el contexto de la tecnología es crucial. Es necesario ser capaz de explicar ideas técnicas de manera clara, tanto a los usuarios como a otros desarrolladores.

Ejemplo: Cuando trabajas en un proyecto con un equipo, deberás explicar tu enfoque sobre el código o las decisiones que tomaste de forma que todos puedan entenderlo.

49. PENSAMIENTO CRÍTICO

El pensamiento crítico en programación es la capacidad de evaluar diferentes soluciones a un problema, sopesando las ventajas y desventajas de cada una, y eligiendo la mejor opción.

Ejemplo: Un programador debe explicar a un diseñador cómo debe funcionar una página web.

Analogía: Es como si estuvieras eligiendo una ruta para ir a un destino: evalúas diferentes caminos, consideras cuál es más rápido o seguro y tomas la mejor decisión.

50. ÉTICA EN TI

La ética en tecnología se refiere al uso responsable de la tecnología. Es importante desarrollar software que respete la privacidad, la seguridad y los derechos de las personas.

Ejemplo: Cuando desarrollas una aplicación, debes asegurarte de que los datos de los usuarios sean tratados con cuidado, sin vulnerar su privacidad.

51. PRIVACIDAD

La privacidad en TI es la protección de la información personal y la seguridad de los datos del usuario. Es esencial asegurarse de que la información sensible no sea divulgada sin el consentimiento adecuado.

Ejemplo: Asegúrate de que una página web que almacena contraseñas las encripte para evitar que los datos de los usuarios sean robados.

52. PERSISTENCIA

En programación (y en la vida), la persistencia es la capacidad de seguir intentando y aprendiendo de los errores en lugar de rendirse. Los errores son parte del proceso de desarrollo y permiten mejorar las habilidades.

Ejemplo: Un programador que recibe un error al ejecutar su código investiga, prueba diferentes soluciones y eventualmente lo corrige.

Analogía: Es como aprender a andar en bicicleta: al principio te caes varias veces, pero sigues intentándolo hasta lograrlo.

53. PROYECTO SIMPLE

Definición: Un proyecto sencillo es una buena forma de practicar conceptos básicos de programación y aplicar lo aprendido en un contexto real.

Ejemplo: Una calculadora en Python:

```
def sumar(a, b):
    return a + b

num1 = float(input("Ingresa el primer número: "))
num2 = float(input("Ingresa el segundo número: "))
```

```
print("La suma es:", sumar(num1, num2))
```

Analogía: Es como practicar con ejercicios pequeños antes de intentar un proyecto grande, similar a aprender a cocinar con recetas sencillas antes de preparar un plato complejo.

54. REUTILIZACIÓN DE CÓDIGO

La reutilización de código permite evitar escribir lo mismo varias veces, aprovechando funciones o módulos ya creados. Esto mejora la eficiencia y facilita el mantenimiento.

Ejemplo: Puedes escribir una función para sumar dos números y luego usarla varias veces en tu programa sin tener que escribir la misma lógica nuevamente.

Analogía: Es como usar una plantilla para hacer documentos en lugar de escribir todo desde cero cada vez.

55. INTELIGENCIA ARTIFICIAL

La inteligencia artificial (IA) es una rama de la informática que busca crear máquinas o programas capaces de realizar tareas que normalmente requieren inteligencia humana, como aprender de los datos.

Ejemplo: Sistemas como Siri o Alexa usan IA para entender comandos de voz y responder a preguntas.

56. TIPOS DE ARCHIVOS: DOC, PNG

Definición: Existen diferentes tipos de archivos según su contenido y propósito. Algunos comunes son:

- **.doc o .docx:** Documentos de texto (Microsoft Word).
- **.png:** Imágenes con fondo transparente.
- **.jpg:** Imágenes comprimidas.
- **.pdf:** Documentos de solo lectura.
- **.mp3:** Archivos de audio.
- **.mp4:** Archivos de video.

Ejemplo: Guardar un informe en .docx para poder editarlo más tarde o en .pdf si solo se necesita leerlo sin cambios.

Analogía: Es como diferentes tipos de papel: algunos son para escribir (documentos), otros para imprimir fotos (imágenes) y otros para escuchar música (archivos de audio).

57. APLICACIONES MÓVILES

Las aplicaciones móviles son programas diseñados para ejecutarse en dispositivos móviles como smartphones o tabletas. Están diseñadas para facilitar tareas o proporcionar entretenimiento.

Ejemplo de aplicación móvil: WhatsApp es una aplicación móvil utilizada para enviar mensajes instantáneos entre usuarios.

58. VIDEOJUEGOS

Los videojuegos son programas interactivos que se juegan en computadoras, consolas o dispositivos móviles. El desarrollo de videojuegos implica tanto la programación como el diseño gráfico.

Ejemplo: Un videojuego como "Minecraft" permite a los jugadores explorar, crear y sobrevivir en un mundo virtual.

59. IMPACTO DEL SOFTWARE

El software tiene un gran impacto en cómo interactuamos con el mundo. Desde aplicaciones que nos ayudan a gestionar nuestras finanzas hasta sistemas que permiten que las empresas operen de manera más eficiente.

Ejemplo: Aplicaciones de navegación como Google Maps han cambiado la forma en que las personas viajan y encuentran lugares.

Analogía: Es como la electricidad en el siglo XX: algo que antes era un lujo se ha vuelto una necesidad para la sociedad.

60. APRENDIZAJE CONTINUO

La tecnología cambia constantemente, por lo que es fundamental seguir aprendiendo nuevas herramientas, lenguajes y metodologías para mantenerse actualizado.

Ejemplo: Un programador que empezó con Python puede aprender JavaScript para expandir sus habilidades.

Analogía: Es como entrenar para una competencia: si dejas de practicar, pierdes habilidades y te quedas atrás.