Refuerzo de temas y conceptos

1. *Computador.*

- ***Definición:*** Máquina electrónica capaz de procesar datos mediante instrucciones programadas. Se compone de hardware (componentes físicos) y software (programas que ejecutan tareas específicas).
- **Ejemplo:** Un portátil usado para programación o navegación en internet.
- **Analogía:** Es como un cerebro que procesa información y toma decisiones basadas en los datos recibidos.

2. Programa - Concepto básico de instrucciones para la computadora.

- ***Definición:*** Conjunto de instrucciones escritas en un lenguaje de programación que una computadora interpreta o ejecuta para realizar una tarea específica, como procesar datos o automatizar funciones.
- **Ejemplo:** Un procesador de textos como Microsoft Word.
- **Analogía:** Es como una receta de cocina que indica los pasos a seguir para obtener un resultado.

3. *Lógica de programación - Pensamiento estructurado para resolver problemas.*

- ***Definición:*** Conjunto de principios y técnicas que permiten estructurar y diseñar soluciones eficientes mediante el uso de algoritmos y estructuras de control. Es fundamental para resolver problemas en la programación.
- **Ejemplo:** Decidir cuándo usar un bucle o una condición en un código
- **Analogía:** Es como planear la mejor ruta para llegar a un destino.

4. *Algoritmo - Pasos para resolver un problema (ej. pseudocódigo).*

- ***Definición*:** de pasos finitos y bien definidos que conducen a la solución de un problema. Puede representarse mediante pseudocódigo o diagramas de flujo antes de implementarse en un lenguaje de programación.
- **Ejemplo:** Un algoritmo para ordenar números de menor a mayor.
- **Analogía:** Es como seguir las instrucciones para armar un mueble.

5. *Variable - Concepto y tipos (enteros, decimales, texto).*

- ***Definición:*** Espacio en la memoria de la computadora donde se almacenan datos que pueden cambiar durante la ejecución de un programa. Pueden ser de distintos tipos, como enteros, flotantes, cadenas de texto o booleanos.
- **Ejemplo:** nombre = "Juan" almacena un texto.
- **Analogía:** Es como una caja etiquetada donde guardamos información temporal.

6. *Tipos de datos - Números, cadenas, booleanos.*

- ***Definición:*** Clasificación de la información que puede manejar un programa. Los principales incluyen:

  - ***Enteros (int):*** Números sin decimales (ej. 5, -10).
  - ***Flotantes (float):*** Números con decimales (ej. 3.14, -2.5).
  - ***Cadenas de texto (string):*** Secuencias de caracteres (ej. "Hola").
  - ***Booleanos (bool):*** Valores de verdadero o falso (True, False).

- **Ejemplo:** int edad = 25; almacena un número entero.
- **Analogía:** Es como distintos tipos de contenedores diseñados para ciertos productos.

7. *Operadores - Aritméticos (+, -, *, /), lógicos (AND, OR, NOT).*

- ***Definición:*** Símbolos que realizan operaciones sobre variables o valores. Se dividen en:
  Aritméticos: + (suma), - (resta), * (multiplicación), / (división).
  ***Lógicos*:** AND, OR, NOT (combinan valores booleanos)
  Relacionales: >, <, == (comparaciones entre valores)

- **Ejemplo:** 5 + 3 usa el operador + para sumar.
- **Analogía:** Son como herramientas de cálculo, como una calculadora.

8. Estructuras de control *- Condicionales (if, else).*

- ***Definición:*** Instrucciones que definen el flujo del programa.

Ejemplos:

Condicionales (if, else): Ejecutan diferentes bloques de código según una condición.
Switch: Alternativa a múltiples if-else cuando se comparan valores específicos.

- **Ejemplo:** if (edad >= 18) { permitirAcceso(); }
- **Analogía:** Es como un semáforo que regula el tráfico según la luz.

9. Bucles – Repetición (for – while)

- ***Definición:*** Permiten ejecutar un bloque de código repetidamente. Ejemplos:

- **For:** Repetición controlada con un número fijo de iteraciones.
- **While:** Repetición basada en una condición lógica.
- **Ejemplo:** Un `while` que imprime números del 1 al 10.
- **Analogía:** Es como un reloj que repite su ciclo cada 60 segundos.

10. Entrada y salida- Leer datos del usuario y mostrar resultados.

Definición: Procesos para recibir datos (entrada) y mostrar resultados (salida).

- **Ejemplo:** Leer un nombre con input() y mostrarlo con print().
- **Analogía:** Es como una conversación donde alguien habla y otro escucha.

11. Funciones básicas - Reutilización de código

- **Definición:** Una función es un bloque de código que realiza una tarea específica y puede reutilizarse varias veces sin necesidad de reescribirlo.
- **Ejemplo:** En una tienda, un botón de "calcular total" reutiliza la misma fórmula cada vez que se realiza una compra.
- **Analogía:** Es como una receta de cocina: en lugar de escribir los pasos cada vez que quieres cocinar un platillo, sigues la receta y la usas cuando la necesites.

12. Primer lenguaje (ej. Python) - Sintaxis básica y uso

- **Definición:** Un lenguaje de programación es un conjunto de reglas que permiten escribir instrucciones para que la computadora las ejecute. Python es un lenguaje fácil de aprender por su sintaxis sencilla.
- **Ejemplo:** Escribir instrucciones para que una computadora sume dos números y muestre el resultado.
- **Analogía:** Es como aprender un nuevo idioma: primero aprendes la gramática básica antes de poder escribir frases completas.

13. Comentarios en el código - Documentar para claridad

- **Definición:** Los comentarios son anotaciones dentro del código que explican lo que hace una sección del programa, sin afectar su ejecución.
- **Ejemplo:** Un programador deja una nota en su código indicando que una línea específica calcula los impuestos.
- **Analogía:** Es como escribir notas en los márgenes de un libro para recordar puntos clave.

14. Errores comunes - Cómo identificarlos (syntax error, runtime error)

- **Definición:** Son fallos que pueden ocurrir al programar. Un error de sintaxis ocurre cuando el código está mal escrito, mientras que un error de ejecución ocurre cuando algo inesperado sucede durante la ejecución.
- **Ejemplo:** Escribir mal una palabra clave en un lenguaje de programación genera un error de sintaxis. Dividir un número por cero genera un error en tiempo de ejecución.
- **Analogía:** Un error de sintaxis es como escribir una dirección con faltas de ortografía, y un error de ejecución es como intentar abrir una puerta cerrada con llave sin tener la llave correcta.

15. Depuración - Uso básico de herramientas o print statements

- **Definición:** La depuración es el proceso de encontrar y corregir errores en un programa.
- **Ejemplo:** Un programador usa mensajes de salida para verificar si una función está devolviendo el resultado esperado.
- **Analogía:** Es como revisar paso a paso una receta para ver dónde te equivocaste si el platillo no quedó bien.

16. Compiladores vs. intérpretes - Diferencia simple

- **Definición:** Un compilador traduce todo el código de un programa a lenguaje de máquina antes de ejecutarlo, mientras que un intérprete lo traduce y ejecuta línea por línea.
- **Ejemplo:** C usa compiladores, mientras que Python usa intérpretes.
- **Analogía:** Un compilador es como traducir un libro completo antes de leerlo, mientras que un intérprete es como traducir palabra por palabra mientras lees.

17. Cadenas de texto - Manipulación básica (concatenar, longitud)

- **Definición:** Son secuencias de caracteres que pueden manipularse de diversas maneras, como unirlas o medir su longitud.
- **Ejemplo:** Unir el nombre y apellido de una persona para mostrar su nombre completo.
- **Analogía:** Es como unir piezas de un rompecabezas para formar una imagen completa.

18. Hardware básico - CPU, memoria, disco duro

- **Definición:** El hardware básico de un ordenador incluye la CPU (procesador), la memoria RAM (almacenamiento temporal) y el disco duro (almacenamiento permanente).
- **Ejemplo:** Cuando abres un programa, se carga en la memoria RAM y la CPU lo ejecuta.
- **Analogía:** La CPU es como el cerebro, la RAM como una mesa de trabajo temporal y el disco duro como un archivador donde se guardan documentos.

19. Software - Sistema operativo vs. aplicaciones

- **Definición:** El sistema operativo es el software que gestiona el hardware y permite ejecutar aplicaciones, que son programas diseñados para tareas específicas.
- **Ejemplo:** Windows es un sistema operativo, y Microsoft Word es una aplicación.
- **Analogía:** El sistema operativo es como el director de una orquesta, y las aplicaciones son los músicos que tocan diferentes instrumentos.

20. Sistemas operativos - Windows, Linux, funciones básicas

- **Definición:** Un sistema operativo es un software que administra los recursos de la computadora y permite al usuario interactuar con ella. Windows y Linux son ejemplos populares.
- **Ejemplo:** Abrir un explorador de archivos o instalar un programa en Windows o Linux.
- **Analogía:** Es como el gerente de un restaurante que organiza a los empleados y las operaciones para que todo funcione sin problemas.

21. Archivos y carpetas - Organización en el sistema

- **Definición:** Son elementos del sistema de archivos que permiten almacenar y organizar información en una computadora.
- **Ejemplo:** Guardar documentos en carpetas separadas por categorías.
- **Analogía:** Es como organizar papeles en carpetas dentro de un archivador.

22. Terminal o consola - Comandos básicos (cd, dir, ls)

- **Definición:** Es una interfaz de línea de comandos donde se pueden ejecutar instrucciones directamente en el sistema operativo.
- **Ejemplo:** Usar el comando `cd` para cambiar de directorio en la terminal.
- **Analogía:** Es como darle órdenes directas a un asistente personal sin usar una interfaz gráfica.

23. Fundamentos de Desarrollo de Software

- **Definición:** Conjunto de principios y metodologías que guían la creación de software.
- **Ejemplo:** Aplicar el ciclo de vida del software al desarrollar una aplicación.
- **Analogía:** Es como seguir un plan de construcción para diseñar un edificio.

24. Ciclo de vida del software - Idea básica (planificar, diseñar, codificar)

- **Definición:** Son las etapas que sigue un software desde su concepción hasta su implementación y mantenimiento.
- **Ejemplo:** Planificar una app, diseñar su interfaz y luego programarla.
- **Analogía:** Es como construir una casa: primero se diseña el plano, luego se construye y finalmente se mantiene.

25. Requisitos - Qué quiere el usuario

- **Definición:** Son las necesidades y expectativas del usuario que debe cumplir un software.
- **Ejemplo:** Un usuario quiere una app que le ayude a organizar sus tareas diarias.
- **Analogía:** Es como preguntar qué quiere alguien en su pastel antes de hornearlo.

26. Prototipos - Bosquejos simples de software

- **Definición:** Representaciones visuales o funcionales iniciales de un software.
- **Ejemplo:** Un diseñador crea una maqueta de cómo se verá una aplicación antes de programarla.
- **Analogía:** Es como hacer un boceto de una casa antes de construirla.

27. Interfaz de usuario - Concepto de diseño básico

- **Definición:** Es la parte visual con la que los usuarios interactúan en una aplicación.
- **Ejemplo:** Botones, menús y formularios en una app.
- **Analogía:** Es como el tablero de control de un auto que permite al conductor interactuar con el vehículo.

28. Pruebas - Verificar que el programa funcione

- **Definición:** Proceso para asegurarse de que el software funciona correctamente antes de su lanzamiento.
- **Ejemplo:** Probar una aplicación en distintos dispositivos antes de publicarla.
- **Analogía:** Es como probar un automóvil antes de venderlo.

29. Qué es una base de datos - Almacenar información organizada

- **Definición:** Es un sistema que permite almacenar y gestionar grandes volúmenes de información de manera estructurada.
- **Ejemplo:** Un sistema que guarda información de clientes en una tienda.
- **Analogía:** Es como una biblioteca donde cada libro está catalogado para facilitar su búsqueda.

30. Internet - Cómo funciona a nivel básico.

- **Definición:** La Internet es una red global de computadoras interconectadas que permite el intercambio de información a través de protocolos de comunicación, como el TCP/IP.
- **Ejemplo:** Cuando entras a YouTube y ves un video, tu computadora se conecta a los servidores de YouTube, descarga el video en pequeñas partes y lo reproduce.
- **Analogía:** Imagina que la Internet es como una gran red de autopistas que conecta ciudades (dispositivos), permitiendo que la información viaje de un lugar a otro, como los autos transportan personas y mercancías.

31. *Direcciones IP - Identificadores simples*.

- **Definición:** Una dirección IP (Internet Protocol) es un número único que identifica un dispositivo en la red. Puede ser estática (fija) o dinámica (cambia con el tiempo).
- **Ejemplo:** El servidor de Google tiene una dirección IP como 8.8.8.8, que permite que cualquier dispositivo en la red se conecte a él.
- **Analogía:** Es como el número de teléfono de una persona: cada usuario tiene uno único, lo que permite que otros lo contacten directamente.

32. Navegadores - Qué hacen y cómo.

- **Definición:** Son programas que permiten acceder a información en la web interpretando y mostrando páginas HTML. Algunos ejemplos populares son Google Chrome, Mozilla Firefox y Safari.
- **Ejemplo:** Cuando buscas en Google, el navegador interpreta la página web y muestra los resultados de búsqueda.
- **Analogía:** Es como un lector de libros digitales que convierte texto codificado en un formato legible y presentable.

33. Cliente y servidor - Interacción básica.

- **Definición:** En una red, un cliente solicita información o servicios a un servidor, que los procesa y responde.
- **Ejemplo:** Cuando abres Facebook en tu teléfono, tu dispositivo (cliente) envía una solicitud a los servidores de Facebook, que devuelven tu feed de noticias.
- **Analogía:** Es como pedir comida en un restaurante: tú (cliente) haces el pedido y el mesero (servidor) te trae el platillo desde la cocina.

34. Seguridad inicial - Contraseñas y riesgos.

- **Definición:** La seguridad informática básica incluye prácticas como el uso de contraseñas seguras, autenticación en dos pasos y evitar compartir datos personales en sitios no confiables.
- **Ejemplo:** En lugar de usar "123456" como contraseña, una segura sería "M7x!pQ$9" con letras, números y símbolos.

- **Analogía:** Es como tener una puerta con varias cerraduras en tu casa para evitar robos.

35. HTML - Estructura de una página web.

- **Definición:** Es un lenguaje de marcado que define la estructura de las páginas web usando etiquetas como <h1> para títulos y <p> para párrafos.
- **Ejemplo:**

  html

  ```
  <h1>Hola, mundo</h1>
  <p>Este es un párrafo de ejemplo.</p>
  ```

- **Analogía:** HTML es como el esqueleto de un edificio: define la estructura, pero no su apariencia.

36. CSS - Estilo básico (colores, fuentes).

- **Definición:** Es un lenguaje que define la apariencia de una página web, controlando colores, fuentes y diseño.
- **Ejemplo:**

  css

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

- **Analogía:** CSS es como la pintura y decoración de una casa que le da un aspecto único.

37. JavaScript introductorio - Interactividad simple (alertas).

- **Definición:** Lenguaje de programación que permite agregar dinamismo e interactividad a una página web.
- **Ejemplo:**

  javascript

  ```
  alert("Hola, mundo!");
  ```

- **Analogía:** Es como un control remoto que permite interactuar con un televisor.

38. Páginas estáticas - Crear algo visible.

- **Definición:** Son sitios web cuyo contenido no cambia para cada usuario, sino que es el mismo para todos.
- **Ejemplo:** Un portafolio personal con información fija.
- **Analogía:** Es como un cartel publicitario que siempre muestra el mismo mensaje.

39. Hosting básico - Subir una página a la web.

- **Definición:** Es el servicio que permite almacenar y hacer accesibles sitios web en Internet.
- **Ejemplo:** Publicar una página en GitHub Pages o Netlify.
- **Analogía:** Es como alquilar un local en un centro comercial para abrir una tienda.

40. Editores de código - VS Code, uso básico.

- **Definición:** Son herramientas que permiten escribir y editar código de forma organizada.
- **Ejemplo:** Usar Visual Studio Code para programar en JavaScript.
- **Analogía:** Como un cuaderno especial para escribir fórmulas matemáticas.

41. Control de versiones - Qué es Git (concepto inicial).

- **Definición:** Sistema que permite rastrear cambios en archivos de código y revertirlos si es necesario.
- **Ejemplo:** Usar `git commit` para guardar una versión del código antes de hacer cambios importantes.
- **Analogía:** Como la función de "deshacer" en un editor de texto, pero más avanzada.

42. Repositorios - Idea de GitHub.

- **Definición:** Un repositorio es un espacio donde se almacena código usando herramientas como GitHub.

- **Ejemplo:** Un repositorio de código abierto donde los desarrolladores colaboran.
- **Analogía:** Es como una biblioteca digital donde cada libro es un proyecto de código.

43. Línea de comandos - Comandos útiles para programar.

- **Definición:** Interfaz de texto para interactuar con el sistema operativo.
- **Ejemplo:** `cd carpeta` para cambiar de directorio.
- **Analogía:** Como usar atajos de teclado en lugar de menús con el mouse.

44. Entornos de desarrollo - Instalación de Python o similar.

- **Definición:** Son herramientas y configuraciones necesarias para programar en un lenguaje específico.
- **Ejemplo:** Instalar Python y usar Jupyter Notebook para análisis de datos.
- **Analogía:** Como tener una cocina equipada con los utensilios adecuados para preparar un platillo.

45. Metodología ágil - Idea de iteraciones cortas.

- **Definición:** Es un enfoque para desarrollar software de manera incremental, dividiendo el trabajo en ciclos cortos llamados "sprints" para mejorar continuamente.
- **Ejemplo:** En un equipo de desarrollo, cada dos semanas se entregan pequeñas partes funcionales de una aplicación en lugar de hacer todo el producto de una vez.
- **Analogía:** Es como construir una casa por habitaciones, asegurando que cada una esté bien hecha antes de avanzar a la siguiente.

46. Documentación - Escribir cómo funciona el código.

- **Definición:** Escribir descripciones y guías sobre el código para que otros desarrolladores (o uno mismo en el futuro) puedan entenderlo fácilmente.
- **Ejemplo:** Un archivo `README.md` en un proyecto de GitHub que explica cómo instalar y usar el software.
- **Analogía:** Es como el manual de usuario de un electrodoméstico, explicando cómo usarlo y solucionar problemas.

47. Resolución de problemas - Dividir en partes.

- **Definición:** Técnica para analizar un problema grande dividiéndolo en partes más pequeñas y manejables.
- **Ejemplo:** Si una página web no carga correctamente, se puede dividir el problema en: revisar la conexión a Internet, verificar el código HTML/CSS y comprobar el servidor.
- **Analogía:** Es como armar un rompecabezas, donde ensamblar pequeñas secciones facilita ver la imagen completa.

48. Comunicación - Explicar ideas técnicas.

- **Definición:** La habilidad de transmitir conceptos de programación de manera clara a colegas, clientes o usuarios finales.
- **Ejemplo:** Un desarrollador explica en términos simples a un cliente cómo funcionará su aplicación móvil.
- **Analogía:** Es como traducir un idioma difícil en palabras comprensibles para todos.

49. Pensamiento crítico - Evaluar soluciones.

- **Definición:** Analizar las diferentes opciones para resolver un problema y elegir la más eficiente.
- **Ejemplo:** Un programador evalúa si es mejor usar un bucle `for` o una función recursiva para resolver un problema.
- **Analogía:** Como elegir la mejor ruta en un mapa considerando el tráfico y la distancia.

50. Ética en TI - Uso responsable de la tecnología.

- **Definición:** Aplicar principios morales en el uso y desarrollo de tecnología, asegurando que se use para el bien común.
- **Ejemplo:** Un programador que decide no desarrollar software de espionaje sin el consentimiento del usuario.
- **Analogía:** Como un médico que sigue un código ético para no dañar a sus pacientes.

51. Privacidad - Proteger datos básicos.

- **Definición:** Proteger la información personal y evitar que terceros la accedan sin permiso.
- **Ejemplo:** No compartir contraseñas y usar autenticación en dos pasos.
- **Analogía:** Como cerrar con llave tu diario personal para que nadie más lo lea.

52. Persistencia - Lidiar con errores y fracasos.

- **Definición:** La capacidad de seguir adelante y mejorar tras enfrentar problemas o fallos en el desarrollo de software.
- **Ejemplo:** Un desarrollador intenta varias soluciones hasta encontrar la correcta para un error de código.
- **Analogía:** Como aprender a andar en bicicleta cayéndose varias veces hasta lograrlo.

53. Proyecto simple - Calculadora o lista de tareas.

- **Definición:** Desarrollo de un programa sencillo para aplicar conocimientos básicos de programación.
- **Ejemplo:** Crear una calculadora básica en JavaScript.
- **Analogía:** Como hacer un experimento científico simple para entender mejor una teoría.

54. Reutilización de código - Usar funciones ya hechas.

- **Definición:** Aprovechar código ya existente en lugar de escribirlo desde cero para mejorar la eficiencia.
- **Ejemplo:** Usar una librería como `lodash` en JavaScript en lugar de escribir funciones repetitivas.
- **Analogía:** Como usar piezas de Lego en lugar de tallar bloques de madera desde cero.

55. Inteligencia artificial - Qué es en términos simples.

- **Definición:** Programas y sistemas diseñados para simular inteligencia humana, tomando decisiones y aprendiendo de datos.

- **Ejemplo:** Un chatbot que responde preguntas en una tienda en línea.
- **Analogía:** Como un asistente virtual que aprende de tus hábitos para darte mejores recomendaciones.

56. Tipos de archivos: doc, pnh.

- **Definición:** Diferentes formatos de archivos que se utilizan para almacenar información.
- **Ejemplo:** `.docx` para documentos de texto, `.png` para imágenes.
- **Analogía:** Como diferentes tipos de cajas donde cada una está diseñada para un contenido específico.

57. Aplicaciones móviles - Ejemplo de su uso.

- **Definición:** Programas diseñados para ejecutarse en teléfonos y tabletas.
- **Ejemplo:** WhatsApp, una aplicación de mensajería instantánea.
- **Analogía:** Como tener una navaja suiza digital que ofrece diferentes herramientas en tu bolsillo.

58. Videojuegos - Introducción al desarrollo básico.

- **Definición:** Creación de software interactivo con gráficos, sonido y mecánicas de juego.
- **Ejemplo:** Un juego en Unity donde el jugador debe esquivar obstáculos.
- **Analogía:** Como escribir una historia interactiva donde el usuario decide qué sucede.

59. Impacto del software - Cómo cambia el mundo.

- **Definición:** La influencia que tienen los programas y aplicaciones en la vida cotidiana, la economía y la sociedad.
- **Ejemplo:** Aplicaciones como Uber han cambiado la forma en que las personas se transportan.
- **Analogía:** Como la electricidad en su momento: un avance que transforma la manera en que vivimos.

60. Aprendizaje continuo - Importancia de seguir estudiando.

- **Definición:** La necesidad de actualizar constantemente los conocimientos en tecnología para mantenerse relevante en el campo.
- **Ejemplo:** Un desarrollador que aprende nuevos lenguajes de programación como Python o Rust para mejorar sus habilidades.
- **Analogía:** Como un médico que sigue estudiando para conocer los últimos tratamientos y técnicas en su área.