¡CONVIERTE TUS IDEAS EN SOFTWARE!

ADOLESCENTES PROGRAMADORES

Un libro que te llevará al fascinante mundo de la programación

Parte del proyecto 'NARRATIVAS INTELIGENTES'

Este libro es una co-creación de docentes y estudiantes de nivel superior usando inteligencia artificial generativa con enfoque ético e innovador

Claudia Farina - Deborah Roa - Basilio Segovia - Tómas Balbuena

Instituto Superior de Formación Docente “Gobernador Virasoro”

Argentina

Primera Edición - 2023

ISBN
9798865780694

Proyecto: Narrativas Inteligentes

Instituto Superior de Formación Docente de Ingeniero Valentín Virasoro

TRANSPARENCIA

Este libro es producto de una colaboración enriquecedora y matizada con nuestras ideas, conceptos y conocimiento previo como autores, y la ayuda de ChatGPT, una herramienta de inteligencia artificial. El proceso de creación implicó un diálogo constante, donde las propuestas generadas por ChatGPT surgieron a partir de instrucciones largas, muy detalladas y cuidadosamente diseñadas por este grupo de docentes y estudiantes. Cada contribución de la IA fue meticulosamente revisada y editada para asegurar su alineación con el contenido y los objetivos que nos propusimos, haciéndonos responsable de cada uno de los conceptos aquí vertidos. El enfoque, las ideas centrales y las conclusiones que se presentan son expresamente nuestras, fruto de un ejercicio de autoría que implicó una cuidadosa curación de la información y una profunda reflexión sobre la temática; aunque la inteligencia artificial proporcionó un apoyo muy valioso.

Índice

PRÓLOGO

Queridos estudiantes y apreciados docentes:

Es un honor presentarles este libro que abre las puertas a un emocionante mundo de posibilidades: la programación. En un mundo impulsado por la tecnología, aprender a programar se ha convertido en una habilidad esencial. En estas páginas, nos adentraremos en los fundamentos de la programación, pero, más importante aún, encenderemos la chispa de la curiosidad que les guiará a explorar, crear y contribuir a un futuro impulsado por la innovación.

Este libro ha sido diseñado pensando en ustedes, estudiantes del nivel secundario. Nuestro objetivo es hacer que la programación sea accesible, interesante y divertida. No importa si nunca han escrito una sola línea de código o si ya tienen algo de experiencia; aquí encontrarán un recurso valioso para realizar un viaje en el mundo de la informática.

Sabemos que, como estudiantes, son una fuente inagotable de energía, creatividad y entusiasmo. Aprovechen esas cualidades, porque la programación es una disciplina que celebra la innovación y la resolución de problemas.

Este libro también es una herramienta valiosa para los docentes de informática. La programación puede parecer un campo complejo, pero aquí encontrarán recursos teóricos y prácticos que les ayudarán a transmitir el conocimiento de manera efectiva y envolvente. Queremos ser su aliado en la misión de inspirar a las jóvenes mentes a descubrir su pasión por la programación.

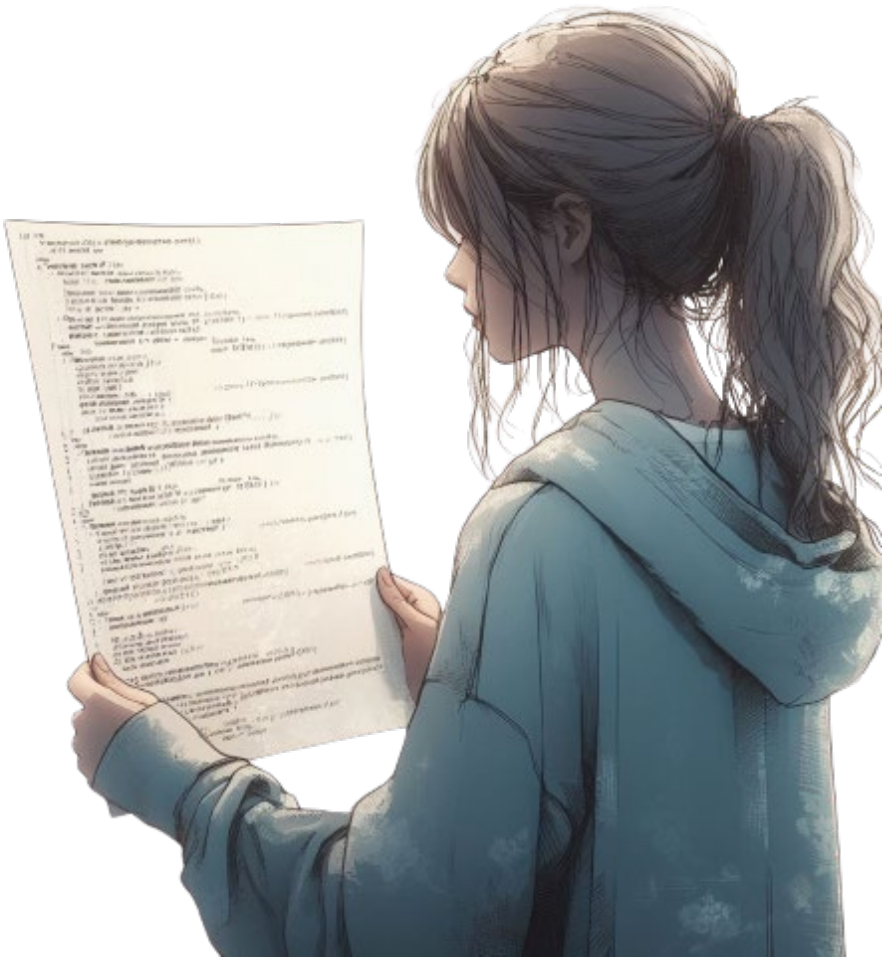
A medida que avancen en este libro, descubrirán la lógica y la magia detrás de la programación. Aprenderán a crear sus propios programas y se sumergirán en conceptos clave como algoritmos, estructuras de datos y buenas prácticas de codificación. Pero más allá de los conceptos técnicos, los animamos a desarrollar habilidades críticas como el pensamiento lógico, la resolución de problemas y la creatividad.

Así que, queridos estudiantes, sumérjanse en este libro con mente abierta y corazón curioso. Exploren, experimenten y, sobre todo, diviértanse mientras aprenden. Y a nuestros dedicados docentes, les agradecemos por su compromiso en guiar a las generaciones futuras en este emocionante viaje.

Sin más preámbulos, ¡comencemos este emocionante viaje juntos!

Capítulo 1

Introducción a la Programación



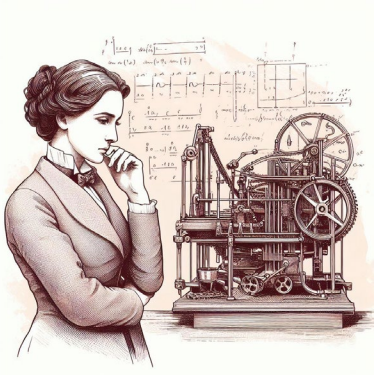
1.1 ¿A qué llamamos Programación?

En este capítulo, vamos a responder la gran pregunta: ¿Qué es la programación? En términos simples, la programación es darle instrucciones a una computadora para que haga cosas que tú quieras que haga. Un buen ejemplo sería enseñarle a un robot a hacer tareas.
Imagina que tienes una amiga robot a la que le quieres pedir que te haga un sándwich. El problema es que tu amiga robot solo entiende un lenguaje especial que se llama "lenguaje de programación". Entonces, para que tu amiga robot te haga un sándwich, necesitas escribirle una lista de instrucciones en ese lenguaje. Este lenguaje se denomina: **Lenguajes de Programación.**
Existen muchos lenguajes de programación diferentes, como Python, JavaScript y Scratch. Cada uno de estos lenguajes tiene sus propias reglas y palabras especiales que la computadora puede entender. Es como hablar diferentes idiomas, pero en lugar de hablar con personas, estamos hablando con las computadoras. Otras palabras importantes que debes conocer son: **Código Fuente y Compilación.**
Cuando escribes esas instrucciones en un archivo, se llama "código fuente". Las computadoras no pueden entender el código fuente directamente. Necesitan que alguien más les traduzca esas instrucciones a un lenguaje que puedan entender. Esa traducción se llama "compilación".

1.2. La Historia de la Programación

¡Viajemos atrás en el tiempo para descubrir cómo comenzó la magia de la programación! La programación es una habilidad increíble que ha evolucionado con el tiempo. En este capítulo, exploraremos los momentos clave en la historia de la programación y cómo ha llegado a ser lo que es hoy.

1.2.1 Los Primeros Pasos



Hace mucho tiempo, en la década de 1800, una mujer matemática llamada Ada Lovelace se convirtió en la primera programadora de la historia. Ella escribió las primeras instrucciones para una máquina que existía solo en papel, la "Máquina Analítica" de Charles Babbage.
Ada Lovelace es considerada una pionera de la programación.

A medida que avanzamos en el tiempo, las computadoras reales comenzaron a desarrollarse en la década de 1940. Los científicos crearon máquinas gigantes que podían realizar cálculos matemáticos complejos. Para darles instrucciones, los programadores tenían que conectar cables y cambiar interruptores manualmente.

A mediados del siglo XX, los programadores comenzaron a desarrollar lenguajes de programación que hacían que la programación fuera más fácil de entender. Uno de los primeros lenguajes de programación fue el "Fortran", diseñado para cálculos científicos. Luego vinieron el "COBOL" para negocios y el "LISP" para inteligencia artificial.

En los años 70 y 80, las computadoras personales comenzaron a aparecer en los hogares. Esto permitió que más personas tuvieran acceso a la programación. Programadores como Bill Gates y Steve Jobs fundaron empresas como Microsoft y Apple, que contribuyeron al auge de la programación y las computadoras personales.

1.2.2. La Era de Internet y las Aplicaciones

A medida que avanzamos hacia los años 90 y 2000, la programación se volvió aún más emocionante con el auge de Internet. Los programadores comenzaron a crear sitios web y aplicaciones que conectaban a personas de todo el mundo. Esto cambió la forma en que vivimos y trabajamos.

La programación ha recorrido un largo camino desde los primeros días de Ada Lovelace. Hoy en día, es una habilidad poderosa que nos permite crear videojuegos, aplicaciones, sitios web y mucho más.

Capítulo 2

Fundamentos de la Codificación

2.1 ¿Qué es un Algoritmo?

En el mundo de la programación, un algoritmo es como una receta o un conjunto de pasos que le dice a una computadora qué hacer. Imagina que estás haciendo una pizza. ¿Qué haces primero? ¿Amasar la masa o poner los ingredientes? Esa es una especie de "receta" para hacer una pizza. Un algoritmo es algo similar, pero para una computadora.
Los algoritmos son importantes porque son la base de todo lo que una computadora hace. Cada vez que juegas a un videojuego, buscas información en Internet o incluso haces una simple suma en una calculadora, estás utilizando algoritmos.

2.1.1 Características de un buen Algoritmo

No todos los algoritmos son iguales. Algunos son más eficientes y rápidos que otros. Un buen algoritmo tiene ciertas características:

1. Claridad: Debe ser fácil de entender. Imagina que estás dando instrucciones a alguien que nunca ha cocinado antes. Las instrucciones deben ser claras y fáciles de seguir.
2. Precisión: Deben ser específicas y sin ambigüedades. No puedes decir "agregue un poco de sal". ¿Cuánto es un poco? Debes decir exactamente cuánta sal agregar.
3. Eficiencia: Un buen algoritmo hace su trabajo de la manera más rápida y eficiente posible. Piensa en la forma más rápida de hacer una tarea y eso es lo que debería hacer tu algoritmo.

2.1.2. Ejemplos de Algoritmos en la Vida Cotidiana

Puedes encontrar algoritmos en muchas partes de tu vida diaria. Por ejemplo, cuando sigues direcciones para llegar a un lugar, cuando sigues una receta de cocina o incluso cuando ordenas tus tareas para hacer la tarea escolar de manera eficiente.

¿Quieres crear tu propio algoritmo?
Primero, piensa en el problema que quieres resolver. Luego, desglosa el problema en pasos simples y ordenados. Asegúrate de que cada paso sea claro y preciso. Luego, ¡prueba tu algoritmo y ajústalo según sea necesario!

Puedes probar realizando el siguiente ejemplo: Imagina que vas a hacer una fiesta para tus amigos. ¿Qué cosas vas a necesitar?, Haz la lista de cosas, y en qué orden. ¿Lista de invitados? ¿Música? ¿Comida? ¿Bebida?, ¿Lugar?, ¿Dinero?, etc. recuerda el orden de prioridades. No puedes comprar la comida, si no has hecho la lista de invitados porque no sabes la cantidad de comida que necesitarás.

2.1.3 Algoritmos en la Programación

En el mundo de la programación, los algoritmos son como el corazón de un programa de computadora. **Son las instrucciones que dicen a la computadora qué hacer**. Los programadores crean algoritmos para realizar tareas específicas, como ordenar una lista de números o buscar información en una base de datos.
Un algoritmo permite organizar los datos y la información de tal modo que al plantearse el mismo es posible pasar después al lenguaje de programación deseado.

Por ejemplo:
Este algoritmo calcula el promedio de tres notas ingresadas por el usuario:

```
INICIO
// Declarar variables
nota1, nota2, nota3, promedio: número real

// Solicitar al usuario que ingrese las tres notas
Escribir "Ingrese la primer nota "
Leer nota1
Escribir "Ingrese la segunda nota "
Leer nota2
Escribir "Ingrese la tercera nota "
Leer nota3

// Calcular el promedio
promedio =(nota1 + nota2 + nota3) / 3

// Mostrar el resultado
Escribir "El promedio de las tres notas es: ", promedio
FIN
```

Te podemos contar que esta estructura se denomina **secuencial**, porque coloca las instrucciones en secuencia, siguiendo un orden, es decir, una a continuación de la otra.
Otro elemento de nuestro algoritmo son las variables: **Las variables** son como cajas de memoria que **te permiten guardar y manipular información en tus algoritmos para resolver problemas**. Puedes nombrarlas como quieras y almacenar diferentes tipos de información, como números, fechas, textos que necesites en tu programa. En el ejemplo anterior, se declaran al comenzar el algoritmo y se usan cuando se necesitan. ¿Las encontraste?, revisa el algoritmo para encontrarlas, en el algoritmo las declaramos como número real para que puedan aceptar decimales.

Este algoritmo verifica si un número ingresado por el usuario es positivo, negativo o cero.

```
INICIO
// Declarar variables
numero: número real

// Solicitar al usuario que ingrese un número
Escribir "Ingrese un número "
Leer numero

// Verificar si el número es positivo, negativo o cero
Si numero > 0 entonces
    Escribir "El número es positivo"
Sino Si numero < 0 entonces
    Escribir "El número es negativo"
Sino
    Escribir "El número es igual a cero"
Fin Si
FIN
```

En este algoritmo, la estructura **condicional** "Si" se utiliza para comprobar (evaluar) si el número ingresado por el usuario es mayor que cero, menor que cero o igual a cero. Dependiendo de la condición que se cumpla, se mostrará un mensaje apropiado en la salida. Los mensajes puestos entre comillas representan mensajes que serán mostrados al usuario, permitiendo al usuario ingresar datos como números, letras o palabras.

Se definió la variable número como número real, para que la misma pueda aceptar números negativos, y se deja un espacio después del mensaje para que el usuario pueda ingresar en ese espacio el número solicitado.

El último tipo de estructura para escribir algoritmos son las estructuras **repetitivas o bucles**. Un bucle "Mientras" o "While" en pseudocódigo, en el ejemplo que verás a continuación, se la utiliza para *calcular la suma de los números del 1 al N, donde N es un número ingresado por el usuario:*

```
INICIO
// Declarar variables
N, suma, contador: número entero.

// Inicializar variables
suma = 0
contador = 1

// Solicitar al usuario que ingrese un numero N
Escribir "Ingrese un numero N: "
Leer N
// Calcular la suma de los números del 1 al N
Mientras contador <= N Hacer
  suma = suma + contador
  contador = contador + 1
Fin Mientras

// Mostrar el resultado
Escribir la suma de los números del 1 al ", N, " es: ",
      suma

FIN
```

El bucle "**Mientras**" se utiliza para repetir una serie de instrucciones mientras se cumpla una condición (en este caso, se repetirá siempre que el contador sea menor o igual a N).

En cada iteración del bucle, se suma el valor del contador a la variable "suma" y se incrementa el contador en 1. Esto se repite hasta que el contador sea mayor que N, momento en el que el bucle se detiene.
Al finalizar el Algoritmo se muestra al usuario el resultado del mismo, En este caso usando una instrucción llamada Escribir, pero dependerá del lenguaje de programación utilizado, la instrucción se llamará de distinta manera.

Otra estructura repetitiva. **Para.**
Enunciado: *"Escribe un algoritmo que imprima los números del 1 al 5 en orden ascendente"*

```
INICIO
// Iniciamos un bucle "PARA" que va desde 1 hasta 5
Para i desde 1 Hasta 5 con paso 1 hacer
     // Imprimimos el valor actual de "i"
     Escribir i
Fin Para
FIN
```

Explicación del algoritmo:

***INICIO**: El programa comienza aquí.*

Para i desde 1 hasta 5 con paso 1 hacer*: Este es el bucle "PARA". La variable "i" se inicia en 1 y se incrementa en 1 en cada iteración (debido a "con paso 1"). El bucle se ejecutará mientras "i" esté en el rango de 1 a 5.*

Escribir i*: Dentro del bucle, imprimimos el valor actual de "i". En la primera iteración, se imprimirá 1, luego 2, 3, 4 y finalmente 5.*

***Fin Para:** Marca el final del bucle "Para".*

***FIN:** El programa ha terminado.*

Este algoritmo utiliza el bucle "PARA" para repetir una acción específica (en este caso, imprimir un número) un número determinado de veces (del 1 al 5 en este caso). Es una forma sencilla de repetir tareas sin tener que escribir el código varias veces.

2.1.4 Ejercicios

¿Qué te parece si ejercitamos? (Al finalizar el libro, encontrarás las respuestas).
Te presentamos 3 enunciados para que puedas ejercitar los algoritmos explicados anteriormente, no te olvides de enunciar las variables, no importa si no se llaman de la misma manera que el ejemplo.

- **Estructura Secuencial:** Calcular el área de un triángulo dado su base y altura. Los valores de la base y la altura son ingresados por el usuario.
- **Estructura Condicional:** Solicitar al usuario que ingrese su edad y determinar si es mayor o menor de edad según las leyes de Argentina. (Mayor de 18 años)
- **Estructura Repetitiva:** Calcular el factorial de un número ingresado por el usuario utilizando un bucle "Mientras".

Si te animas a más, aquí hay dos enunciados que incluyen los tres tipos de estructuras.

- Escribe un programa que solicite al usuario ingresar un número positivo N. Luego, el programa debe calcular e imprimir la suma de todos los números pares desde 1 hasta N. Si el usuario ingresa un número negativo o cero, el programa debe mostrar un mensaje de error.
- Escribe un algoritmo que calcule la suma de los números pares del 1 al 10 e imprima el resultado. Puedes utilizar un bucle “Para” iterar a través de los números del 1 al 10 y una variable para llevar un registro de la suma. En cada iteración, verifica si el número actual es par (divisible por 2) y, si es así, agrégalo a la suma. Al final, muestra el resultado.

2.2. Tipos de Lenguajes de programación

Ya tenemos una idea de los algoritmos, ahora hablaremos de los lenguajes de Programación. Cómo ya lo mencionamos, es fundamental entender que existen diferentes "lenguajes" que las personas y las computadoras utilizan para comunicarse. Los lenguajes de programación son como dialectos especiales que nos permiten dar instrucciones a las computadoras. Aquí hay algunos ejemplos:

- **Python**: Es un lenguaje de programación fácil de aprender y muy poderoso. Es ideal para principiantes y se usa en una amplia variedad de aplicaciones, desde la creación de sitios web hasta la inteligencia artificial.
- **JavaScript**: Se utiliza principalmente en el desarrollo web para hacer que las páginas web sean interactivas. Puedes usarlo para crear juegos y aplicaciones web.
- **C++:** Es un lenguaje más avanzado que se usa en el desarrollo de software de alto rendimiento, como juegos y sistemas operativos.
- **Scratch:** Es genial para principiantes jóvenes. Te permite crear programas arrastrando y soltando bloques en lugar de escribir código.
- **Java:** Se utiliza ampliamente en aplicaciones empresariales y en el desarrollo de aplicaciones para Android.

2.3. Herramientas y Entornos de Desarrollo

Ahora que conocemos algunos lenguajes de programación, es hora de hablar de las herramientas que necesitarás para escribir y ejecutar tus programas. Estas herramientas se llaman "entornos de desarrollo". Aquí hay algunos ejemplos:

- **IDLE (Python):** Es un entorno de desarrollo integrado que se instala junto con Python. Es simple y fácil de usar para escribir y ejecutar programas Python.
- **Visual Studio Code:** Es una herramienta popular y gratuita de Microsoft que se utiliza para muchos lenguajes de programación. Tiene muchas extensiones que hacen que la programación sea más fácil.
- **Eclipse:** Es un entorno de desarrollo muy poderoso que se utiliza principalmente para Java. Es ideal para proyectos más grandes.
- **CodePen:** Es un entorno en línea para escribir y probar código HTML, CSS y JavaScript. Es genial para principiantes que están aprendiendo desarrollo web.
- **Scratch Online:** Si estás usando Scratch, puedes programar directamente en su sitio web. Es fácil de usar y no necesitas instalar nada en tu computadora.

2.3.1. Elige tu Aventura

Cada lenguaje de programación y herramienta de desarrollo tiene sus propias ventajas y desafíos. Tu elección dependerá de lo que quieras crear y tus preferencias personales.

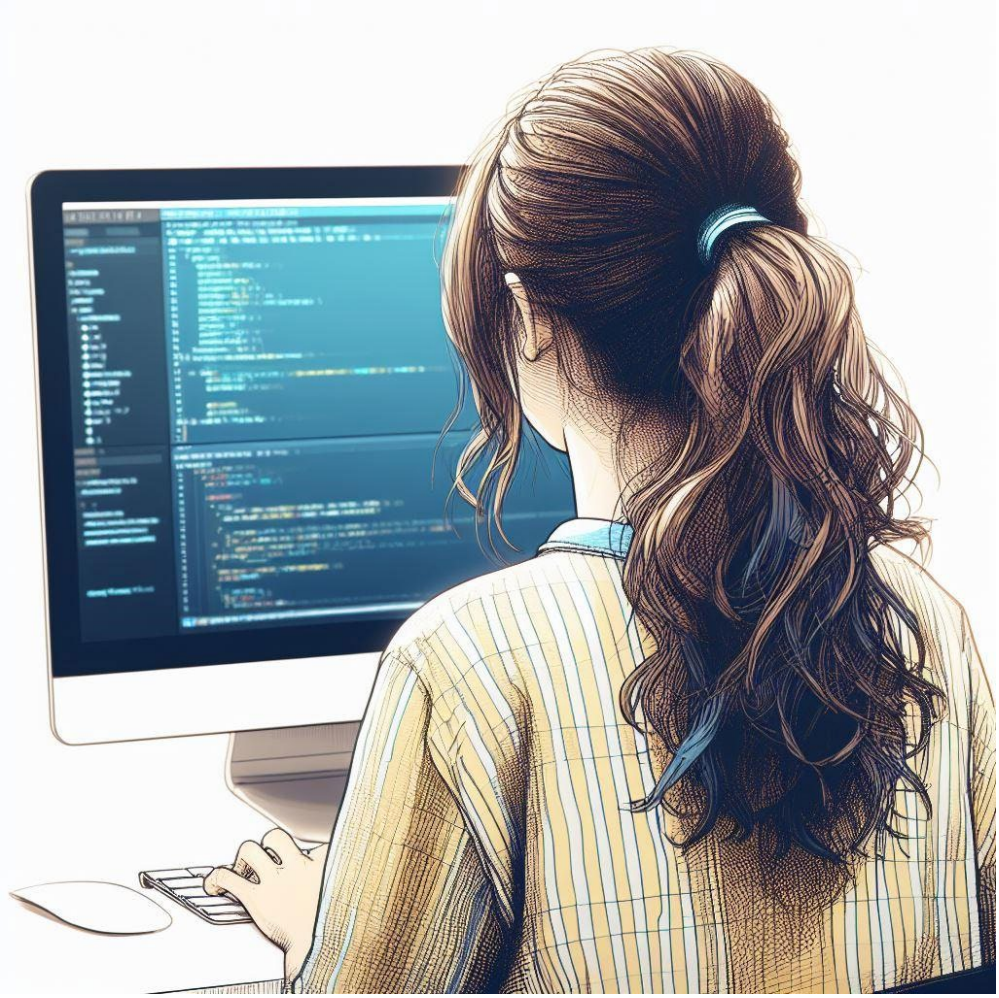
Algunos lenguajes son mejores para ciertas tareas, así que ¡explora y encuentra el que más te guste!, en este libro explicaremos el Lenguaje Python, por ser el más sencillo de aprender y debido a que se utiliza en una gran variedad de aplicaciones.

El viaje de la programación es una aventura emocionante. A medida que aprendas un lenguaje y te familiarices con las herramientas de desarrollo, podrás crear tus propios programas y proyectos. **No temas cometer errores; los errores son oportunidades para aprender.**

Hay una gran comunidad de programadores en línea dispuesta a ayudarte. También hay muchos recursos en línea, tutoriales y cursos para aprender y mejorar tus habilidades de programación. ¡Adelante, no estás solo en este viaje!

Capítulo 3

Aprende un Lenguaje de Programación



3.1 Python: El Lenguaje Amigable

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python se ha vuelto muy popular debido a su sintaxis limpia y legible, lo que lo hace ideal para programadores principiantes y experimentados por igual. Se llama Python en honor a los Monty Python, un grupo de comediantes británicos.

Algunas características clave de Python:

- **Sintaxis legible**: Python se enorgullece de su sintaxis clara y legible, que se asemeja al lenguaje humano. Esto hace que sea más fácil de aprender y de escribir código, lo que resulta en una mayor productividad.
- **Interpretado**: Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea en lugar de compilarlo previamente en un lenguaje de máquina. Esto facilita el desarrollo, la depuración y la experimentación rápida.
- **Multiparadigma**: Python admite múltiples paradigmas de programación, como programación orientada a objetos, programación funcional y programación imperativa, esto les da flexibilidad a los programadores para abordar problemas de diversas maneras.
- **Multiplataforma**: Python es compatible con una variedad de sistemas operativos, como Windows, macOS, Linux y más. Puedes escribir código Python en una plataforma y ejecutarlo en otra sin realizar modificaciones significativas.
- **Gran comunidad y bibliotecas**: Python cuenta con una gran comunidad de desarrolladores que han creado una amplia gama de bibliotecas y módulos que facilitan el desarrollo de aplicaciones. Estas bibliotecas abarcan áreas como la ciencia de datos, desarrollo web, automatización, inteligencia artificial y más.
- **Amplio uso**: Python se utiliza en una variedad de aplicaciones, desde desarrollo web y aplicaciones de escritorio hasta análisis de datos, aprendizaje automático, automatización de tareas y más. Es especialmente popular en campos como la inteligencia artificial y la ciencia de datos debido a sus bibliotecas especializadas, como NumPy, pandas y TensorFlow.

- **Open Source:** Python es un lenguaje de código abierto, lo que significa que su código fuente es accesible para cualquiera que desee verlo o contribuir a su desarrollo. Esto ha impulsado la colaboración y la expansión continua del lenguaje.

Python se ha convertido en un lenguaje de programación versátil y poderoso que es ampliamente utilizado en la industria y en la comunidad de desarrollo de software. Su facilidad de uso y su amplia gama de aplicaciones lo hacen una excelente elección para los programadores.

Ejemplo en Python: **Hola Mundo**

```
print (“¡Hola, mundo!”)
```

En este ejemplo, le estamos diciendo a la computadora que imprima "¡Hola, mundo!" en la pantalla.

3.2 JavaScript: El Lenguaje de la Web

JavaScript obtuvo su nombre debido a una estrategia de marketing de Netscape, la compañía que desarrolló el lenguaje en 1995. En ese momento, Java, un lenguaje de programación desarrollado por Sun Microsystems, estaba ganando popularidad, y Netscape quería capitalizar esa tendencia. A pesar de que JavaScript y Java comparten algunas similitudes sintácticas y conceptuales superficiales, como el uso de la palabra "script" y algunas características de orientación a objetos, son dos lenguajes diferentes.
La elección del nombre "JavaScript" fue en gran medida una maniobra de marketing para aprovechar la popularidad de Java en ese momento. Esto llevó a cierta confusión, ya que las dos tecnologías no están relacionadas en términos de diseño o desarrollo. Java es un lenguaje de programación independiente de la plataforma, mientras que JavaScript está diseñado principalmente para la programación web y se ejecuta en navegadores web. A pesar de la similitud en el nombre, estas dos tecnologías son distintas en muchos aspectos.

JavaScript es un lenguaje de programación ampliamente utilizado en el desarrollo web.

- **Tipado**: JavaScript es un lenguaje de tipado dinámico. Esto significa que no es necesario declarar el tipo de una variable cuando se crea; el tipo se determina en tiempo de ejecución. Por ejemplo, puedes asignar una variable a un número en un momento y a una cadena de texto en otro sin restricciones de tipo.

- **Orientación a objetos:** JavaScript es un lenguaje orientado a objetos, pero a menudo se lo describe como un lenguaje multiparadigma, ya que admite varios enfoques de programación. En JavaScript, todo es un objeto (o puede comportarse como uno), incluyendo funciones.

- **JavaScript** utiliza un modelo de objetos basado en prototipos en lugar de clases tradicionales, como en otros lenguajes orientados a objetos. Esto significa que los objetos pueden heredar propiedades y métodos de otros objetos, lo que proporciona flexibilidad en la programación orientada a objetos.

- **Funciones de primera clase**: En JavaScript, las funciones son ciudadanos de primera clase. Esto significa que las funciones pueden ser asignadas a variables, pasadas como argumentos a otras funciones y retornadas como valores. Este enfoque funcional es una característica poderosa de JavaScript y permite escribir código más modular y reutilizable.

- **Bibliotecas y frameworks**: JavaScript es ampliamente utilizado en el desarrollo web para mejorar la interacción del usuario en sitios y aplicaciones. Para este propósito, se utilizan bibliotecas como jQuery y marcos de trabajo como React, Angular y Vue.js. Estas herramientas facilitan el desarrollo de aplicaciones web dinámicas y altamente interactivas.

- **Ejecución en el navegador**: JavaScript se ejecuta en el navegador web del usuario, lo que permite la creación de páginas web dinámicas. Puede interactuar con el Document Object Model (DOM) para cambiar el contenido de una página en respuesta a eventos del usuario.

- **Amplia comunidad y uso**: JavaScript es uno de los lenguajes de programación más populares y ampliamente utilizados en la actualidad. Su versatilidad y su presencia en el navegador permite su desarrollo en la web.

JavaScript es un lenguaje de programación de tipado dinámico que se utiliza en gran medida para la programación en el navegador web. Es un lenguaje orientado a objetos basado en prototipos y permite la programación funcional. Además, es esencial para el desarrollo web interactivo y cuenta con una amplia comunidad de desarrolladores y una variedad de bibliotecas y frameworks disponibles.

3.3 Javascript en el backend

JavaScript es conocido principalmente por su uso en el lado del cliente, donde se ejecuta en el navegador web para crear interacciones dinámicas en las páginas web. Sin embargo, con la introducción de Node.js, JavaScript también puede utilizarse en el lado del servidor. Node.js es un entorno de tiempo de ejecución de JavaScript de código abierto que permite ejecutar JavaScript en el servidor en lugar de solo en el navegador.
Funciona de manera asincrónica y no bloqueante, lo que significa que puede manejar múltiples conexiones simultáneas sin esperar a que una operación se complete antes de pasar a la siguiente. Esto lo hace muy eficiente para aplicaciones en tiempo real, servidores web y otras tareas de red.
Node.js utiliza el motor V8 de Google Chrome para ejecutar JavaScript en el servidor. Permite a los desarrolladores crear aplicaciones de red escalables y de alto rendimiento utilizando JavaScript en el lado del servidor. Algunos de los casos de uso comunes de Node.js incluyen servidores web, aplicaciones de tiempo real, API y servicios web, entre otros.

Otros frameworks y entornos del lado del servidor que trabajan con JavaScript incluyen:

- **Express.js**: Es un marco de aplicaciones web para Node.js que simplifica la creación de aplicaciones web y APIs.
- **Hapi.js**: Otro marco de aplicaciones web para Node.js que se centra en la construcción de API RESTful.
- **Meteor**: Un framework completo para el desarrollo de aplicaciones web y móviles en tiempo real utilizando JavaScript.
- **NestJS:** Un framework de servidor TypeScript (que es un superset de JavaScript) basado en Node.js que se utiliza para crear aplicaciones escalables y mantenibles.
- **Deno:** Aunque no es un framework, Deno es un entorno de tiempo de ejecución de **JavaScript y TypeScript** del lado del servidor que es una alternativa a Node.js.

Cada uno de estos frameworks y entornos tiene sus propias características y ventajas, lo que permite a los desarrolladores elegir el más adecuado para su proyecto de servidor en JavaScript.

3.4 Instalar Python en Windows

Aquí te explicamos cómo instalar Python en Windows.

1. Descargar Python:
   - Abre tu navegador web (por ejemplo, Google Chrome o Mozilla Firefox).
   - Ve al sitio web oficial de Python en https://www.python.org/downloads/windows/.
   - En la sección "Latest Python Releases," busca la versión más reciente de Python y haz clic en "Download" para Windows. Asegúrate de seleccionar la versión que coincide con la arquitectura de tu computadora (32 o 64 bits).

2. Iniciar el instalador.
   - Una vez que se complete la descarga, ejecuta el archivo descargado. El nombre del archivo podría ser similar a "python-3.x.x.exe," donde "x.x" es la versión de Python.

3. Configuración de la instalación.
   - En la primera pantalla del instalador, asegúrate de marcar la casilla "Add Python 3.x to PATH." Esto permitirá que Python sea accesible desde la línea de comandos.
   - Luego, haz clic en "Install Now" para comenzar la instalación.

4. Progreso de la instalación:
   - Verás una barra de progreso que muestra el avance de la instalación. Espera a que se complete.

5. Instalación completada
   - Una vez que se complete la instalación, verás un mensaje que dice "Setup was successful." Esto significa que Python se ha instalado correctamente en tu computadora.

6.Comprobar la instalación.
   - Para verificar que Python se instaló correctamente, abre el "Símbolo del sistema" (Command Prompt). Puedes hacerlo buscando "cmd" en el menú de inicio o escribiendo "cmd" en la barra de búsqueda.

- En la ventana de comando, escribe `python --version` y presiona Enter. Deberías ver la versión de Python que instalaste, por ejemplo, "Python 3.x.x."

Ya puedes comenzar a escribir y ejecutar programas en Python para aprender programación, solo necesitarás un programa más para ejecutar de la mejor manera los códigos que escribas, a continuación, te explicaremos como instalar el **Visual Studio Code.**

3.5 Instalar Visual Studio Code (VS Code).

Los pasos para poder instalar el Visual Studio Code en Windows, son los siguientes:

1. Descargar Visual Studio Code:
   - Abre tu navegador web y ve al sitio web oficial de Visual Studio Code en https://code.visualstudio.com/.
   - Haz clic en el botón "Download for Windows" para descargar el instalador.

2. Ejecutar el instalador.
   - Una vez que se haya descargado el archivo de instalación, ábrelo haciendo doble clic.
   - El nombre del archivo podría ser "VSCodeUserSetup-x64-x.x.x.exe," donde "x.x.x" representa la versión más reciente.

3. Configuración de la instalación:
   - Aparecerá una ventana de instalación. Asegúrate de que la casilla "Add to PATH" esté marcada para que puedas acceder a VS Code desde la línea de comandos.
   - Haz clic en "Next" para continuar.

4. Selección de componentes.
   - Deja las opciones predeterminadas y haz clic en "Next" nuevamente.

5. Seleccionar directorio de instalación.
   - Puedes optar por cambiar la ubicación de instalación, pero generalmente es mejor dejarla como está. Haz clic en "Next."

6. Selección del menú Inicio.
   - Deja la opción predeterminada y haz clic en "Install."

7. Progreso de la instalación.

- Visual Studio Code se instalará en tu computadora. Espera a que el proceso se complete.

8. Finalización de la instalación.

- Una vez que se complete la instalación, haz clic en "Finish."

9. Ejecutar Visual Studio Code.

- Abre el menú de inicio y busca "Visual Studio Code" o simplemente "Code." Haz clic en el ícono para iniciar VS Code.

Ahora puedes usarlo como un entorno de desarrollo para escribir y ejecutar programas en varios lenguajes, incluyendo Python.

3.6 Sintaxis y estructuras de control

Aquí tienes un compendio de las sintaxis de Python, así como las estructuras de control más comunes en el lenguaje:

Sintaxis de Python:

1. Variables y Asignación:

- Declaración de variables:

```
variable = valor
```

Python es de tipado dinámico, por lo que no es necesario declarar el tipo de variable.

2. Impresión:

- Imprimir en la consola:

```
print ("Texto a imprimir")
```

3. Comentarios:

- Comentario de una línea:

```
# Esto es un comentario
```

Comentario multilínea:

```
""" Este es un comentario multilínea """

 o

"""
Este también es un comentario multilínea
"""
```

4. Estructuras de Datos:

```
- Listas: `mi_lista = [1, 2, 3]`
  - Tuplas: `mi_tupla = (1, 2, 3)`
  - Diccionarios: `mi_diccionario = {"clave": "valor"}`
  - Conjuntos: `mi_conjunto = {1, 2, 3}`
```

5. Estructuras de Control:

- Condicionales:

```
    ```python
     if condición:
         # Código si la condición es verdadera
     elif otra_condición:
         # Código si la segunda condición es verdadera
     else:
         # Código si ninguna condición es verdadera
     ```
```

6. Bucles:

- Bucle **`for`** para iterar sobre una secuencia:

```
      ```python
       for elemento in secuencia:
           # Código para cada elemento
       ```
```

- Bucle `**while**` para repetir mientras se cumple una condición:

```python
while condición:
    # Código que se repite
```

7. Funciones:

- Declaración de funciones:

```python
def mi_funcion(parametro1, parametro2):
    # Código de la función
    return resultado
```

8. Manejo de Excepciones:

- Capturar excepciones:

```python
try:
    # Código que podría generar una excepción
except TipoDeExcepción as e:
    # Manejo de excepción
else:
    # Código si no se generó excepción
finally:
    # Código que se ejecutará siempre
```

9. Importación de Módulos:

- Importar módulos:

```python
import modulo
from modulo import función
```

10. Clases y Objetos (OOP - Programación Orientada a Objetos):

- Definición de una clase:

```python
class MiClase:
    def __init__(self, atributo):
        self.atributo = atributo
    def metodo(self):
        # Código del método
```

11. Módulos Estándar:

Python proporciona una amplia gama de módulos estándar para realizar tareas comunes como operaciones matemáticas, manejo de archivos, networking, etc.

Ejemplo en JavaScript: Cambiar el Color de un Botón

```javascript
// Obtener el botón por su identificador
var boton = document.getElementById("miBoton");

// Cambiar el color del botón cuando se hace clic
boton.addEventListener("click", function() {
  boton.style.backgroundColor = "blue";
});
```

En este ejemplo, estamos diciéndole a la computadora que cambie el color de un botón a azul cuando alguien haga clic en él.

Python y JavaScript son dos lenguajes de programación poderosos que te permiten dar vida a tus ideas y crear cosas emocionantes. Recuerda que la práctica es clave, así que no dudes en experimentar y crear tus propios programas. ¡Buena suerte en tu viaje de programación!

3.5 Creación de programas simples

En el **Capítulo 2: Fundamentos de la Codificación** se proponen algunos ejemplos en pseudocódigo. En esta ocasión veremos cómo traducirlos al código Python para que la computadora pueda resolverlo.

1) *Calcula el promedio de tres notas ingresadas por el usuario:*

   El Script quedaría de la siguiente manera:

```
# Declarar variables
nota1, nota2, nota3, promedio = 0.0,0.0,0.0,0.0

# Solicitar al usuario que ingrese las tres notas
nota1 = float(input("Ingrese la primer nota: "))
nota2 = float(input("Ingrese la segunda nota: "))
nota3 = float(input("Ingrese la tercera nota: "))

# Calcular el promedio
promedio = (nota1 + nota2 + nota3) / 3

# Mostrar el resultado
print("El promedio de las tres notas es:", promedio)
```

2) *Este algoritmo verifica si un número ingresado por el usuario es positivo, negativo o cero.*

```
    # Solicitar al usuario que ingrese un número
    numero = float(input("Ingresa un número: "))

    # Verificar si el número es positivo, negativo o cero
    if numero > 0:
            print("El número", numero ," es positivo.")
    elif numero < 0:
            print("El número", numero ," es negativo.")
    else:
            print("El número", numero ,"  es cero.")
```

Te proponemos dos ejercicios en Python que puedes resolver. Los resultados los tendrás en el anexo al final del libro.

Ejercicio 1: Calculadora de IMC (Índice de Masa Corporal)

Crea un programa en Python que calcule el Índice de Masa Corporal (IMC) de una persona. El IMC se calcula mediante la siguiente fórmula:

$$IMC = \frac{peso(kg)}{altura^2(m^2)}$$

Pide al usuario que ingrese su peso en kilogramos y su altura en metros, luego calcula y muestra el IMC. Además, muestra un mensaje que indique si la persona tiene un IMC bajo peso, peso normal, sobrepeso u obesidad, de acuerdo con los rangos establecidos por la Organización Mundial de la Salud.

Ejercicio 2: Adivina el número

Crea un juego en el que la computadora elija un número aleatorio entre 1 y 100, y le pida al usuario que adivine ese número. El programa debe proporcionar pistas al usuario indicando si el número a adivinar es mayor o menor que el número que ha adivinado. El juego debe continuar hasta que el usuario adivine correctamente el número. Al final, muestra cuántos intentos le tomó al usuario adivinar el número.

Capítulo 4

Resolución de Problemas

4.1 Pensamiento Lógico y Resolución de Problemas

Para resolver problemas debes tener en cuenta estos pasos:

- **Introducción al Pensamiento Lógico**: Te permite explicar la importancia de la lógica en la programación y cómo se relaciona con la resolución de problemas.
- **Definición de Problemas:** Podrás entender cómo definir un problema antes de intentar resolverlo, identificando los requisitos y restricciones clave.
- **Algoritmos**: como ya explicamos los algoritmos son secuencias de pasos lógicos para resolver problemas y se aplican en programación.
- **Estructuras de Control:** Tener siempre presente las estructuras de control, como bucles y condicionales, y cómo se utilizan para tomar decisiones y repetir tareas en la programación, te ayudará para simplificar tareas y tomar decisiones.
- **Resolución de Problemas**: Los ejemplos y ejercicios te ayudarán a resolver problemas a través de la programación y en tu vida cotidiana.

4.2. Introducción al Pensamiento Lógico

El pensamiento lógico es la capacidad de razonar y tomar decisiones basadas en la lógica. En programación, esto implica pensar de manera sistemática y estructurada para resolver problemas. Sistemático significa que haces las cosas de manera ordenada, siguiendo un plan o un conjunto de reglas. Por ejemplo, cuando estudias para un examen, sigues un plan que te ayuda a aprender paso a paso, cómo revisar tus apuntes, hacer ejercicios y repasar. No lo haces al azar, sino de una manera organizada.
Organizado se refiere a mantener las cosas en su lugar y estructuradas. Cuando tienes tu mochila o tus útiles escolares organizados, todo está en su sitio y es fácil de encontrar. Lo mismo ocurre en tu vida diaria: si tienes un horario para tus actividades y cumples con él, eso es ser organizado. Los conceptos clave a explorar incluyen:

- **Secuencia Lógica**: Debes comprender la idea de que un programa es una secuencia lógica de instrucciones que se ejecutan en orden. Cada instrucción tiene un propósito específico.

- **Estructuras Lógicas**: Introducir estructuras lógicas como "si-entonces-sino" (condicionales) y bucles (repetición) recuerda que te permiten tomar decisiones y ejecutar tareas repetitivas en un programa.

4.2.1 Definición de Problemas

Antes de abordar cualquier problema en programación, es fundamental definirlo adecuadamente. Aquí se pueden abordar los siguientes aspectos:

- **Requisitos del Problema:** Si tienes un problema que resolver primero debemos descomponerlo en partes, y realizarnos preguntas tales como: ¿Qué debe hacer el programa? ¿Cuáles son los datos de entrada y las salidas esperadas?
- **Restricciones**: Hay cosas que debes definir para que tu programa funcione, se llaman restricciones. Las restricciones pueden incluir límites de tiempo, recursos o reglas específicas. Son como reglas o condiciones que debes seguir al escribir un programa de computadora. Imagina que estás armando un rompecabezas y cada pieza tiene que encajar de una manera específica. Las restricciones son como las reglas que te dicen cómo deben encajar las piezas. Si no sigues estas reglas, el rompecabezas no funcionará correctamente.

  En programación, las restricciones pueden ser cosas como "no puedes usar más de 100 caracteres en este mensaje" o "este número debe ser mayor que otro número". Estas reglas te ayudan a escribir un programa que haga lo que se espera de él y evitan errores o resultados inesperados.

4.2.2 Algoritmos

Cómo ya estudiaste un algoritmo es una serie de pasos lógicos y precisos para resolver un problema.

- **Diseñar Algoritmos**: Ya te enseñamos a diseñar un algoritmo descomponiendo un problema en pasos más pequeños y manejables. Puedes utilizar herramientas como pseudocódigo para describir estos pasos, pero también hay otras herramientas gráficas que puedes utilizar como por ejemplo Diagramas de Flujos de Datos, que es una forma gráfica de diseñar la solución del problema.
- **Eficacia de los Algoritmos**: Siempre debes darle importancia de crear algoritmos eficientes, que resuelvan el problema de la manera más rápida y efectiva posible.

4.2.3 Estructuras de Control

Ya conociste que las estructuras de control son herramientas esenciales para tomar decisiones y controlar el flujo de un programa. Recuerda que éstas se clasifican en:

- **Condicionales:** Las declaraciones condicionales (if-else) se utilizan para ejecutar ciertos bloques de código según una condición.
- **Bucles:** Se utilizan para repetir tareas y procesar datos de manera eficiente. (por ejemplo, for o while)

Resolución de Problemas:
Esta etapa es crucial en el proceso de programación, ya que la planificación adecuada sienta las bases tanto para un desarrollo exitoso del software como su mantenimiento y posterior escalabilidad.

Requisitos del Programa: Antes de escribir una sola línea de código, es fundamental comprender los requisitos del programa. Esto implica:

- **Definir objetivos**: Debes identificar y definir claramente qué debe lograr el programa. ¿Cuál es el propósito de la aplicación? ¿Qué se espera que haga?
- **Recopilación de Datos de Entrada y Salida**: también debes determinar los datos que se ingresarán en el programa y los resultados que se esperan obtener. Esto incluye entender las necesidades del usuario.

Diseño de Algoritmos: Como ya vimos, un algoritmo es una serie de pasos lógicos que resuelven un problema. Por lo general los pasos que debes seguir pueden ser los siguientes:

- **Descomposición del Problema**: Consiste en dividir un problema en subproblemas más pequeños y manejables.
- **Pseudocódigo y Diagramas de Flujo**: Estas son herramientas necesarias para planificar y representar algoritmos antes de la programación real. Esto ayuda a visualizar la lógica del programa.
- **Planificación de Recursos**: Para la programación es fundamental planificar los recursos que necesitarán, como variables, funciones, y estructuras de datos. Esto implica definir las variables que almacenarán información y cómo se organizará la información.

Pruebas y Depuración: La planificación no se limita al diseño del programa; también implica la consideración de pruebas y depuración, ya que los errores son casi siempre inevitables.

- **Pruebas de Unidades**: Debes probar los componentes individuales del programa para asegurarse de que funcionen correctamente.
- **Casos de Prueba:** Luego crear casos de prueba efectivos que cubran diversas situaciones, incluyendo entradas válidas e inválidas, para garantizar la robustez del programa.
- **Depuración Continua**: La depuración es un proceso continuo, por ello debes aprender a identificar y solucionar errores a medida que avanzan en el desarrollo. Para esto, los IDE de programación cuentan con control de errores en tiempo real (JIT), es decir, que te van señalando los errores a medida que escribes el código.
- **Mantenimiento de Código**: El código debe adaptarse a lo largo del tiempo, a nuevas necesidades y corrigiendo errores a medida que surgen. Es un proceso constante, es por ello, que muchos programas tienen nuevas versiones, que mejoran y/o corrigen versiones anteriores.

Capítulo 5

Programación Creativa

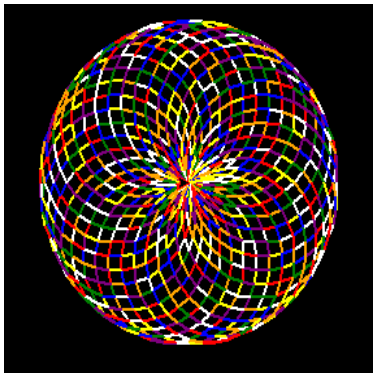
5.1 Arte Generativo con Código

Imagina tener el poder de crear arte único y sorprendente con solo unas pocas líneas de código. Esto es posible gracias al arte generativo con código. Aquí es donde la programación se encuentra con la creatividad. Puedes escribir programas que generen patrones, formas y colores asombrosos. Cada vez que ejecutas el programa, ¡obtienes una obra de arte diferente!

1) *Consigna: Realiza en Python una serie de espirales con colores aleatorios.*

```
import turtle
import random

 # Configurar la pantalla
turtle.bgcolor("black")
turtle.speed(0)
turtle.pensize(2)

# Colores disponibles
colors = ["red", "orange", "yellow", "green", "blue", "purple",
"white"]

# Función para dibujar una espiral de colores
def draw_spiral(sides, length):
    for _ in range(sides):
        color = random.choice(colors)
        turtle.pencolor(color)
        turtle.forward(length)
        turtle.right(360 / sides)

# Dibujar una espiral de 36 lados con longitud inicial de 10
 def draw():
     for _ in range(36):
         draw_spiral(36, 10)
         turtle.right(10)
draw()
# Cerrar la ventana al hacer clic
turtle.exitonclick()
```

Este código utiliza la biblioteca turtle para crear una ventana de dibujo. *Dibuja una serie de espirales con colores aleatorios.* Cada espiral tiene 36 lados y un color aleatorio de la lista de colores disponibles. Puedes experimentar con los valores y agregar más detalles para crear patrones de arte generativo únicos.



5.2 Animaciones y Gráficos Interactivos

¿Alguna vez has soñado con dar vida a tus dibujos y hacer que se muevan? Con la programación, puedes hacerlo realidad. Crear animaciones y gráficos interactivos que respondan a tus acciones, es esencial en el desarrollo de videojuegos y aplicaciones interactivas.

```
import turtle

# Configurar la pantalla
wn = turtle.Screen()
wn.bgcolor("white")

# Crear una tortuga para dibujar
tortuga_dibujo = turtle.Turtle()
tortuga_dibujo.pensize(5)
tortuga_dibujo.speed(0)

# Función para mover la tortuga al hacer clic y arrastrar
def dibujar(x, y):
      tortuga_dibujo.ondrag(None) # Deshabilitar temporalmente
      tortuga_dibujo.setpos(x, y)
      tortuga_dibujo.ondrag(dibujar) #Volver a habilitar

# Configurar la tortuga para seguir el cursor
tortuga_dibujo.ondrag(dibujar)

# Mantener la ventana abierta
wn.mainloop()
```

Este programa crea una ventana en blanco donde puedes dibujar haciendo clic y arrastrando el mouse. La tortuga sigue el cursor y dibuja a medida que te mueves por la pantalla. Puedes personalizar el programa ajustando el color de la línea, el grosor del lápiz o cualquier otra característica que desees. ¡Es una forma simple y divertida de crear una animación interactiva en Python!

5.3 Creación de Música y Sonido

La programación no se trata solo de imágenes, también puedes crear música y sonido. Exploraremos cómo usar el código para componer música, generar efectos de sonido y crear paisajes sonoros envolventes

En este caso, reproduciremos una serie de notas para crear una melodía simple:

```
import winsound, time

# Definir frecuencias de las notas (en Hz)
C = 261.63 # Do
D = 293.66 # Re
E = 329.63 # Mi
F = 349.23 # Fa
G = 392.00 # Sol
A = 440.00 # La
B = 493.88 # Si

# Duración de cada nota en milisegundos
duracion = 500 # 500 ms (medio segundo)

# Reproducir una melodía simple
melodia = [C, D, E, F, G, A, B]

for nota in melodia:
    winsound.Beep(int(nota), duracion)
    time.sleep(0.2) # Pausa de 200 ms entre notas
```

Este código utiliza la **función winsound.Beep** para reproducir notas musicales en un sistema Windows. Cada nota se especifica en Hz (frecuencia) y se reproduce durante un período de tiempo definido (duración). Después de cada nota, hay una breve pausa para separar las notas en la melodía.

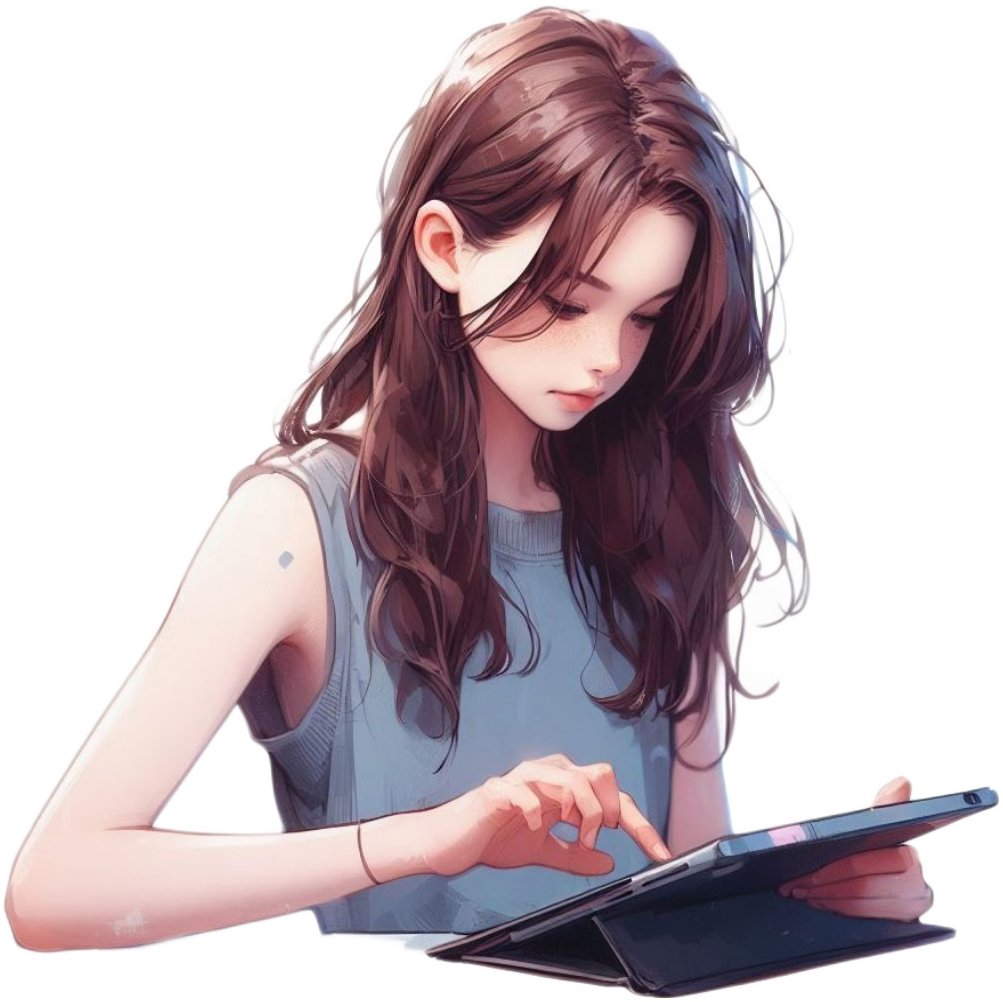
Puedes ajustar las frecuencias y las duraciones de las notas para crear tu propia melodía.

Recuerda que siempre encontrarás recursos en línea, tutoriales y comunidades donde puedes obtener inspiración y ayuda. No importa tu nivel de habilidad, siempre hay oportunidades para crecer como artista y programador.

Con la programación, puedes transformar tus ideas en realidad, explorar tu creatividad y sorprender al mundo con tus creaciones. ¡Así que adelante, comienza tu viaje y descubre el poder de la programación en el mundo del arte, la animación y la música!

Capítulo 6

Seguridad en la Programación



6.1 Seguridad Informática: Protegiendo tu trabajo y actuando responsablemente

Vivimos en un mundo tecnológico, y aprender programación es genial, pero también debemos saber cómo mantener nuestras creaciones seguras y ser buenos ciudadanos digitales. En esta parte, nos enfocaremos en cómo mantener nuestras creaciones seguras y proteger los datos de las personas.

6.2 Conceptos de Seguridad Informática

1. Confidencialidad: Piensa en esto como un secreto bien guardado. En el mundo de la programación, significa asegurarte de que solo las personas autorizadas puedan ver y acceder a cierta información. Es como mantener tus contraseñas y datos personales en una caja fuerte digital.

2. Integridad: Este concepto trata de mantener las cosas tal como son. Cuando programas, significa asegurarte de que los datos no sean alterados o corrompidos por personas no autorizadas. Es como proteger tus deberes escolares para que nadie pueda cambiar tus respuestas.

3. Disponibilidad: Piensa en esto cómo asegurarte de que tus juegos favoritos están siempre listos para jugar. En programación, implica garantizar que las aplicaciones y los recursos estén disponibles cuando los necesites, sin caídas ni problemas técnicos.

6.3 Protegiendo tu Código desde el Principio

En el mundo de la tecnología y la programación, es importante no solo aprender a crear programas, sino también entender cómo mantenerlos seguros y respetar la ética. Aquí te presentamos algunos consejos esenciales, especialmente si estás empezando:

1. **Validación de Entradas**: Asegurarnos de que la información que entra en nuestro programa sea correcta y segura.
2. **Protección contra XSS:** Evitar que se ejecuten programas maliciosos que puedan dañar nuestra aplicación.
3. **Contraseñas fuertes:** Motivar a las personas a usar contraseñas fuertes y guardarlas en un lugar seguro.

4. **Control de acceso:** Permitir solo a las personas adecuadas usar nuestro programa. Decidir qué pueden hacer las personas una vez que las identificamos a través de permisos de usuarios.
5. **Protección de Datos**: Mantener seguros los datos personales de las personas con el cifrado.
6. **Actualizaciones y Parches**: Mantener todo actualizado, instalando las últimas correcciones de seguridad.
7. **Aprender Siempre:** Aprender siempre sobre nuevas amenazas y cómo protegerse.
8. **Seguridad desde el Inicio:** Pensar en la seguridad desde el principio al crear tu programa. Hacer copias de seguridad para no perder datos importantes.
9. **Mensajes de Error Seguros:** No muestres mensajes de error que revelen información importante. Usa mensajes genéricos para mantener a los curiosos fuera.

Ejemplo de Validación de entrada y mensaje de error:

Aquí tienes un ejemplo sencillo en Python que demuestra cómo validar la entrada de un número entero. En este caso, estamos asegurándonos de que el usuario ingrese while True:

```
def validar_entero():
 while True:
   try:
      entrada = input("Por favor,ingresa un número entero: ")

      numero_entero = int(entrada) # convierte a entero

      # imprime el número y termina el bucle
      print(“Has ingresado un número entero:(numero_entero)”)
   break # sale del bucle while

   except ValueError:   # muestra el error de conversión

       print("Eso no parece ser un número entero válido.
Inténtalo de nuevo.")
```

Este código utiliza un bucle while para solicitar al usuario que ingrese un número entero. Luego, intenta convertir la entrada en un número entero utilizando **int (entrada)**. Si la conversión es exitosa, imprime el número y sale del bucle. Si ocurre un error de conversión (por ejemplo, si el usuario ingresa texto en lugar de un número), muestra un mensaje de error y vuelve a solicitar la entrada. De esta manera, se asegura de que solo se aceptan números enteros como entrada.

6.4 Ética en la Programación

La ética es sobre hacer lo correcto. Aquí aprenderemos cómo ser buenos ciudadanos digitales cuando creamos tecnología. Algunas cosas importantes son:

- No ser injustos o hacer discriminación en nuestros programas.
- Ser honestos sobre cómo funciona lo que creamos y cómo tomamos decisiones.
- Ser responsables por lo que hacemos.
- Asegurarnos de que todos, incluyendo las personas con discapacidad, puedan usar lo que creamos.
- Pensar en cómo nuestra tecnología afecta al medio ambiente.
- Mantener a las personas seguras y proteger nuestra tecnología de ataques.

En resumen, cuando programamos, no solo creamos cosas, también debemos cuidarlas y actuar de manera responsable. La seguridad, la protección de datos y la ética son muy importantes, ¡así que asegurémonos de aprender sobre ellas y ser buenos programadores y ciudadanos digitales!

Capítulo 7

Programación para el Futuro.

7.1 Programación para el futuro: ¡Tu Aventura en el Mundo Digital!

La programación es como una varita mágica del siglo XXI que nos permite dar vida a nuestras ideas y resolver problemas. A medida que avanzamos hacia un mundo más digital, la programación se vuelve fundamental. Puedes:

- **Ser Creativo sin Límites:** Programar te permite crear aplicaciones, juegos y sitios web, ¡tu imaginación es el único límite!
- **Resolver Problemas Importantes:** La programación te da superpoderes para abordar problemas desde el cambio climático hasta la medicina.
- **Aprender Divirtiéndote:** Es una forma emocionante de aprender. Puedes crear proyectos educativos y aprender mientras programas.

La programación es como dar vida a tus ideas en el mundo digital. Hoy en día, la tecnología está en todas partes y aprender a programar es como aprender un nuevo idioma con el poder de crear cosas sorprendentes.

7.2 Opciones de Carreras en Tecnología y Programación

En Argentina, puedes encontrar diversas carreras técnicas, universitarias y de grado relacionadas con la tecnología y la programación. Algunos ejemplos incluyen:

- **Carreras Técnicas o tecnicaturas:** Te preparan para roles prácticos en desarrollo de software, redes de computadoras, ciberseguridad, análisis de sistemas, soporte de infraestructura informática y más.
- **Licenciaturas Universitarias:** Ofrecen una formación más amplia en informática, ciencias de la computación, telecomunicaciones y más.
- **Carreras de Grado:** Estas incluyen ingenierías en informática, sistemas de información, robótica y muchas otras opciones.

La elección de la carrera depende de tus intereses personales y objetivos profesionales.

7. 3 Proyectos de Código Abierto y Contribuciones

La comunidad de código abierto es una gran familia global de programadores que trabajan juntos para crear software gratuito y accesible para todos. Unirte a proyectos de código abierto te permite aprender, construir tu portafolio y hacer un cambio positivo en el mundo.

Puedes contribuir en áreas como desarrollo de sistemas operativos, navegadores web, inteligencia artificial y más.
Algunos de los proyectos de código abierto populares en los que puedes contribuir, son: Linux, Mozilla Firefox, Apache HTTP Server, WordPress, Python, Git, Docker, Kubernetes, TensorFlow, Ruby on Rails, FreeCodeCamp y proyectos humanitarios. No es necesario ser un experto para contribuir, ya que muchos proyectos valoran la participación y el aprendizaje continuo. Contribuir a proyectos de código abierto es una excelente manera de adquirir experiencia, mejorar tus habilidades de programación y unirte a una comunidad global de desarrolladores.

7.4 Tecnológicas Emergentes

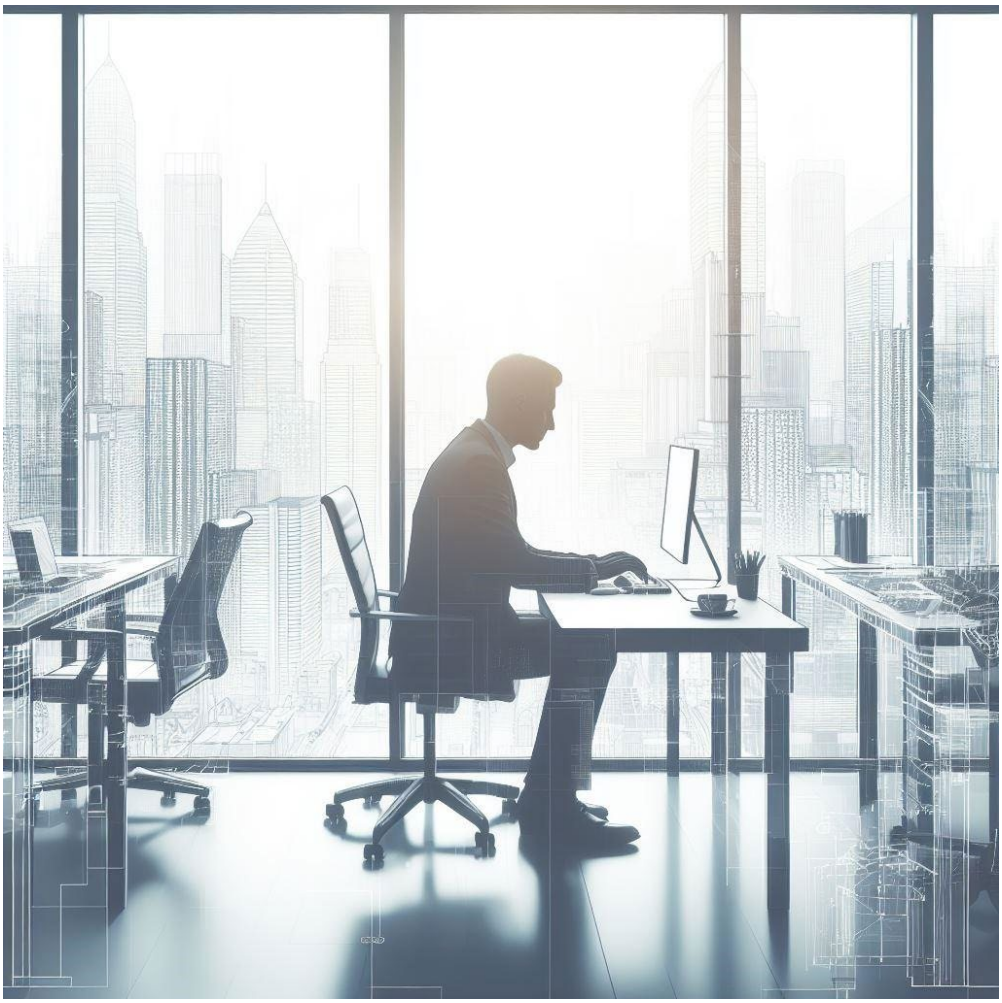
La tecnología nunca se detiene, y siempre hay algo nuevo en el horizonte. Estas son algunas tendencias emocionantes a seguir:

1. **Inteligencia Artificial (IA)**: La IA está revolucionando la forma en que interactuamos con la tecnología, desde asistentes virtuales hasta vehículos autónomos.
2. **Realidad Virtual (RV) y Realidad Aumentada (RA)**: La RV y la RA están transformando la forma en que experimentamos el mundo, desde juegos inmersivos hasta aplicaciones médicas.
3. **Ciberseguridad**: Con el aumento de la tecnología, la ciberseguridad se vuelve cada vez más importante para proteger nuestros datos y privacidad.

La programación es la llave para un futuro emocionante en el mundo digital. Puedes ser un creador, un solucionador de problemas y un líder en tecnología. ¡Prepárate para un emocionante viaje en el mundo de la tecnología y la programación! ¡El futuro es tuyo para programar!

Capítulo 8

Historia Inspiradoras



8.1 Historias inspiradoras: La Chispa que Inicia el Fuego

En el mundo de la programación de aplicaciones, la inspiración puede surgir de diversas fuentes. Aquí te presento algunas estrategias clave:

1. **Identifica Problemas Cotidianos:** Encuentra situaciones diarias problemáticas y crea aplicaciones para resolverlas, cómo organizar tareas o elegir películas.
2. **Observa Tendencias Actuales**: Mantente al día con las tendencias tecnológicas y sociales, como la realidad aumentada o la inteligencia artificial, para crear apps innovadoras.
3. **Conecta con tus Pasiones**: Desarrolla aplicaciones relacionadas con tus intereses y pasiones para comprender mejor las necesidades de los usuarios.
4. **Mejora lo existente:** Examina aplicaciones existentes y busca formas de mejorarlas, desde interfaces más sencillas hasta soluciones para problemas comunes.
5. **Recopila Comentarios**: Pide opiniones a amigos y comunidades en línea para inspirarte en problemas no resueltos.
6. **Piensa en el Futuro**: Anticipa las necesidades futuras de los usuarios considerando la evolución tecnológica.
7. **Combina Ideas:** Experimenta combinando conceptos de diferentes áreas para generar ideas creativas.
8. **Hackatones:** Participa en concursos de programación para generar ideas bajo presión.
9. **Lleva un Cuaderno de Ideas**: Registra todas tus ideas, incluso las pequeñas, que pueden crecer con el tiempo.
10. **Crea Juegos Interactivos**: Si te gustan los videojuegos, desarrolla tus propios juegos, como rompecabezas o aventuras.
11. **Educa Divirtiendo**: Diseña aplicaciones educativas atractivas, como apps para aprender idiomas o ciencias de manera interactiva.
12. **Soluciones para la Comunidad:** Considera cómo tu aplicación puede beneficiar a tu comunidad local, desde conectar voluntarios con proyectos locales hasta promover la sostenibilidad ambiental.

Recuerda que la programación es la herramienta que te permitirá dar vida a tus ideas. ¡Diviértete creando aplicaciones que hagan la vida más fácil o emocionante para ti y otros adolescentes!

8.2 Proyectos Innovadores Liderados por Jóvenes:

Los proyectos innovadores liderados por jóvenes destacan por su creatividad y capacidad para abordar problemas de la sociedad utilizando la tecnología y la programación. Aquí te presento una breve síntesis de algunos ejemplos notables:

- **Samaira Mehta - CoderBunnyz**: Samaira Mehta, a la edad de 10 años, creó "CoderBunnyz", un juego de mesa que enseña conceptos de programación a niños. Su enfoque lúdico ha inspirado a jóvenes a explorar la programación y la tecnología.
- **Gitanjali Rao - Detector de Plomo en el Agua**: Gitanjali Rao, nombrada la "Niña del Año" por TIME en 2020, desarrolló un dispositivo llamado "Tethys" que detecta plomo en el agua potable, abordando problemas de salud pública.
- **Jóvenes contra el Cambio Climático**: Grupos de adolescentes lideran iniciativas para combatir el cambio climático utilizando la programación y la recopilación de datos. Sus aplicaciones monitorean la calidad del aire, sensores de temperatura y alertas tempranas para eventos climáticos extremos.
- Estudiantes argentinos de 15 y 17 años del Colegio Mendocino Tomás Alva Edison ganaron el **2do lugar de la FIRST Global Challenge** o mejor conocida como “El Mundial de Robótica”

Estos jóvenes demuestran cómo la creatividad y la pasión pueden ser impulsoras poderosas para cambiar el mundo a través de la tecnología y la programación, resolviendo problemas importantes y dejando huella en sus respectivos campos.

8.3 Proyectos volvieron millonarios a sus fundadores

En Argentina:

- **MercadoLibre:** Fundado por Marcos Galperin en 1999, es uno de los principales marketplaces en línea de América Latina, que ofrece una amplia gama de productos y servicios.
- **Globant:** Fundada por Martín Migoya, Guibert Englebienne, Martín Umaran y Néstor Nocetti en 2003, se especializa en servicios de desarrollo de software y soluciones digitales.
- **Despegar:** Roberto Souviron y Matías Mute cofundaron Despegar.com en 1999, una plataforma líder en la industria de viajes en América Latina.
- **OLX Argentina:** Alec Oxenford y Fabrice Grinda lanzaron OLX en 2006, una plataforma de comercio en línea global con presencia significativa en Argentina.
- **Rappi:** Los emprendedores Simón Borrero, Sebastián Mejía y Felipe Villamarín son los fundadores de Rappi, una aplicación de entrega a domicilio que ofrece una amplia gama de servicios.
- **Etermax:** Maximo Cavazzani fundó Etermax y es conocida por crear el popular juego de palabras **"Preguntados"** y otros juegos exitosos.

En el Mundo:

- **Apple:** Fundada por Steve Jobs, Steve Wozniak y Ronald Wayne, es conocida por productos como el iPhone y el iPad, y ha revolucionado la tecnología.
- **Microsoft:** Fundada por Bill Gates y Paul Allen, es una de las principales empresas de software en el mundo.
- **Google:** Fundada por Larry Page y Sergey Brin, es líder en motores de búsqueda y tecnología.
- **Facebook (Meta Platforms, Inc.):** Fundada por Mark Zuckerberg, ha transformado las redes sociales y se enfoca en la realidad virtual y aumentada.
- **Amazon:** Fundada por Jeff Bezos, es una gigante del comercio electrónico y la nube.
- **Netflix:** Fundada por Reed Hastings y Marc Randolph, lidera la transmisión en línea.
- **SpaceX:** Fundada por Elon Musk, ha revolucionado la industria aeroespacial.

- **Uber:** Fundada por Travis Kalanick y Garrett Camp, ha cambiado la movilidad urbana.

Estos emprendimientos, ya sea en Argentina o en todo el mundo, son ejemplos inspiradores de cómo las mentes jóvenes y apasionadas pueden transformar industrias y cambiar el mundo con la tecnología y la programación.

Capítulo 9

Recursos y Herramientas de la Programación

9.1. Entornos de Desarrollo (IDEs) y Editores de Texto Recomendados

¿Qué es un IDE?

Un **Entorno de Desarrollo Integrado (IDE)** es una herramienta que reúne diversas funcionalidades para facilitar la escritura, edición, depuración y gestión de código. Para jóvenes programadores, es esencial contar con un IDE que sea intuitivo y eficiente.

Recomendaciones para Jóvenes Programadores

- **Visual Studio Code (VS Code)**: Este es un IDE ligero y muy popular. Ofrece resaltado de sintaxis, autocompletado y una amplia variedad de extensiones para personalizar tu entorno de desarrollo.

- **Thonny**: Ideal para principiantes en Python, Thonny proporciona un entorno simple y limpio para escribir y ejecutar código Python.

9.2 Control de Versiones y Sistemas de Gestión de Código Fuente

Organizando tu Código con Control de Versiones

El control de versiones es crucial para llevar un registro de los cambios en tu código a medida que trabajas en un proyecto. **Git** es una herramienta poderosa para esto, y **GitHub** es una plataforma que facilita la colaboración en proyectos de código abierto.

Colaborando en Proyectos

Vamos a aprender cómo crear un repositorio en GitHub y cómo clonarlo en tu máquina local. Veremos cómo hacer cambios en tu código, hacer commit y realizar push para compartir tus contribuciones.

Colaborar en proyectos a través de plataformas como GitHub es una habilidad esencial para cualquier programador. A continuación, te guiaré a través de los pasos para crear un repositorio en GitHub, clonarlo en tu máquina local y contribuir al proyecto.

9.2.1 Creando un repositorio en GitHub. Paso a Paso.

1. Inicia Sesión en tu Cuenta de GitHub
   - Ve a [GitHub](https://github.com/) y accede a tu cuenta.

2. Crea un Nuevo Repositorio

- Haz clic en el botón "New" (Nuevo) en la esquina derecha de tu panel de control.
- Ingresa un nombre para tu repositorio, una descripción opcional y elige si deseas que sea público o privado (dependiendo de tus necesidades).

3. Configura tu Repositorio

- Puedes elegir agregar un archivo README, un archivo de licencia y un archivo **`.gitignore`** según tus preferencias.

4. Crea el Repositorio

- Haz clic en el botón "Create repository" (Crear repositorio) para crear tu nuevo proyecto en GitHub.

Paso 2: Clonando el Repositorio en tu Máquina Local

1. Obtén el Enlace del Repositorio

- En la página del repositorio en GitHub, haz clic en el botón verde "Code" (Código) y copia el enlace HTTPS.

2. Abre tu Terminal o Consola

- Abre la terminal o consola en tu máquina y navega al directorio donde deseas clonar el repositorio.

3. Clona el Repositorio

- Ejecuta el siguiente comando, reemplazando `[enlace]` con el enlace que copiaste:

```
git clone [enlace]
```

- Esto descargará una copia completa del repositorio en tu máquina local.

Paso 3: Haciendo Cambios y Contribuciones

1. Abre tu Editor de Código

- Utiliza tu IDE o editor de texto preferido para abrir los archivos del proyecto.

2. Realiza los Cambios Necesarios

- Edita los archivos según las contribuciones que desees hacer.

3. Añade tus Cambios al Área de Staging

- En la terminal, ejecuta los siguientes comandos para añadir tus cambios al área de staging:

```
git add
```

4. Realiza un Commit de tus Cambios

- Después de añadir los cambios, ejecuta el siguiente comando para realizar un commit con un mensaje descriptivo:

```
git commit -m "Mensaje descriptivo de tus cambios"
```

5. Envía tus Contribuciones a GitHub

- Finalmente, envía tus cambios a GitHub ejecutando:

```
git push origin main
```

Esto subirá tus contribuciones al repositorio en línea.
¡Ahora has colaborado con éxito en un proyecto en GitHub! Recuerda que este es un flujo de trabajo básico y hay muchas otras técnicas y prácticas avanzadas que puedes aprender a medida que te vuelvas más experimentado. ¡Disfruta colaborando en proyectos y aprendiendo de otros programadores!

9.3 Herramientas de Depuración y Prueba: Detectando y Corrigiendo Errores

Inevitablemente, te encontrarás con errores en tu código. Aprender a identificar y corregir estos errores es una habilidad crucial para cualquier programador.
Imagina que estás escribiendo un cuento en una hoja de papel, pero en lugar de palabras, estás escribiendo instrucciones para que una computadora las siga. Python y otros programas son muy buenos para seguir esas instrucciones, pero a veces cometemos errores.
Cuando escribimos código en Python, la computadora trata de entenderlo, pero si cometemos un error, cómo escribir una palabra incorrecta u olvidar un paso importante, se confunde y no puede seguir adelante. Es como si escribieras una palabra que no existe en tu cuento y alguien se detiene al leer porque no sabe qué significa.
Python y otros programas tienen una especie de "detective de errores" que revisa el código. Este detective es muy inteligente y puede señalar dónde están los errores. Por ejemplo, si escribiste mal una palabra, el detective de errores te dirá dónde y te dará una pista para corregirla. También te dirá si te olvidaste de algún paso importante.

Estos detectores de errores son muy útiles porque te ayudan a encontrar y arreglar los problemas en tu código antes de que lo ejecutes.

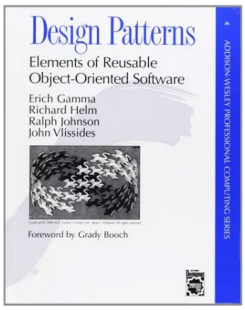
9.4. Bibliografía Recomendada

En este capítulo, te proporcionaremos una lista de libros esenciales que te ayudarán a fortalecer tus habilidades de programación, independientemente de tu nivel de experiencia. Además, te ofreceremos recomendaciones específicas para diferentes lenguajes de programación y áreas de desarrollo.

Libros Esenciales para Programadores Principiantes

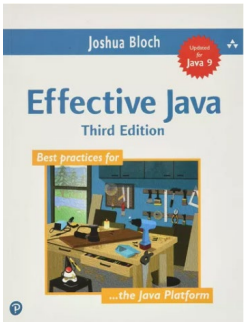
1. "**Python Crash Course**" de Eric Matthes: Una excelente introducción a la programación usando Python, que incluye ejercicios prácticos y proyectos interesantes.
2. "**JavaScript and JQuery: Interactive Front-End Web Development**" de Jon Duckett: Ideal para aquellos interesados en el desarrollo web, este libro cubre los fundamentos de JavaScript y cómo interactúa con HTML y CSS.
3. "**Head First Java**" de Kathy Sierra y Bert Bates: Una guía amigable para aprender Java, un lenguaje ampliamente utilizado en el desarrollo de aplicaciones.

Libros Recomendados para Programadores Intermedios



1. "**Clean Code: A Handbook of Agile Software Craftsmanship**" de Robert C. Martin: Enseña cómo escribir código limpio y bien estructurado, un aspecto fundamental para cualquier programador.
2. "**Design Patterns: Elements of Reusable Object-Oriented Software**" de Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, y Grady Booch: Explora patrones de diseño que te ayudarán a escribir código más eficiente y reutilizable.
3. "**Introduction to the Theory of Computation**" de Michael Sipser: Este libro es una excelente introducción a la teoría de la computación y la teoría de la complejidad, temas cruciales para programadores avanzados.

Libros Recomendados para Programadores Avanzados



1. "**Effective Java**" de Joshua Bloch: Ofrece consejos y técnicas avanzadas para escribir código Java de alta calidad y eficiencia.
2. "**Design Patterns: Elements of Reusable Object-Oriented Software**" de Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, y Grady Booch: Aunque es recomendado para programadores intermedios, es una lectura esencial para aquellos que deseen profundizar en patrones de diseño.
3. "**Clean Architecture: A Craftsman's Guide to Software Structure and Design**" de Robert C. Martin: Explora cómo organizar y estructurar sistemas de software a gran escala.

9.4.1 Recomendaciones Específicas por Lenguaje y Área de Desarrollo

Para Desarrollo Web Frontend (HTML, CSS, JavaScript):

- "Eloquent JavaScript" de Marijn Haverbeke
- "You Don't Know JS" de Kyle Simpson

Para Desarrollo Web Backend (Node.js, Express, Ruby on Rails, Django, etc.):

- "Node.js Design Patterns" de Mario Casciaro y Luciano Mammino
- "Django for Beginners" de William S. Vincent

Para Desarrollo de Aplicaciones Móviles (Android, iOS):

- "Kotlin Programming: The Big Nerd Ranch Guide" de Josh Skeen y David Greenhalgh
- "iOS Programming: The Big Nerd Ranch Guide" de Christian Keur y Aaron Hillegass

Para Ciencia de Datos y Análisis de Datos:

- "Python for Data Science For Dummies" de John Paul Mueller
- "The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling" de Ralph Kimball y Margy Ross

9.5. Sitios Web y Comunidades de Programación

Te presentamos una selección de plataformas de aprendizaje en línea, foros y comunidades, así como blogs y sitios web de referencia en programación. Estos recursos te brindarán valiosa información, soporte y oportunidades para expandir tus conocimientos y habilidades.

Plataformas de Aprendizaje en Línea

1. Coursera

Descripción: Coursera ofrece una amplia gama de cursos en línea de universidades y organizaciones reconocidas a nivel mundial. Los cursos abarcan una variedad de temas, incluida la programación y la ciencia de la computación.

Recomendaciones: "Machine Learning" de Andrew Ng, "Python for Everybody" de Charles Severance.

2. Udemy

Descripción: Udemy es una plataforma de aprendizaje en línea que ofrece una amplia variedad de cursos sobre programación y desarrollo de software, impartidos por expertos en la industria.

Recomendaciones: "The Web Developer Bootcamp 2023" de Colt Steele, "Complete Python Bootcamp: Go from zero to hero in Python 3" de Jose Portilla.

3.*edX

Descripción: edX es una plataforma de cursos en línea fundada por Harvard y el MIT. Ofrece una amplia gama de cursos gratuitos y de pago en diversas disciplinas, incluida la programación.

Recomendaciones: "CS50's Introduction to Computer Science" de Harvard University, "Python for Data Science" de UC San Diego.

9.6 Foros y Comunidades

1. Stack Overflow

   Descripción: Stack Overflow es una comunidad de preguntas y respuestas donde programadores de todo el mundo comparten conocimientos y resuelven problemas.

   Recomendación: Únete a discusiones relevantes, plantea preguntas específicas y participa en la comunidad para aprender de otros y mejorar tus habilidades.

2. Reddit (r/learnprogramming, r/programming)

   Descripción: Reddit es una plataforma de discusión en línea con numerosos subreddits dedicados a la programación. r/learnprogramming es ideal para principiantes, mientras que r/programming ofrece discusiones más avanzadas.

Recomendación: Únete a la comunidad, comparte tus dudas y participa en debates para aprender de otros programadores.

3. GitHub
Descripción: GitHub es una plataforma de desarrollo colaborativo que aloja proyectos de código abierto. Es un lugar excelente para explorar código de otros y contribuir a proyectos interesantes.
Recomendación: Explora proyectos en GitHub que te interesen y considera contribuir con correcciones o nuevas funcionalidades.

9.6.1 Blogs y Sitios Web de Referencia en Programación

1. Mozilla Developer Network (MDN)
Descripción: MDN proporciona documentación completa sobre tecnologías web, incluyendo HTML, CSS y JavaScript. Es una referencia esencial para desarrolladores web.

2. FreeCodeCamp
Descripción: FreeCodeCamp ofrece tutoriales y proyectos prácticos en desarrollo web y ciencia de datos. También tiene una comunidad activa y un foro de ayuda.

3. Dev.to
Descripción: Dev.to es una plataforma social para desarrolladores que ofrece artículos, tutoriales y discusiones sobre una amplia variedad de temas de programación y tecnología.
Al explorar estos recursos, asegúrate de participar activamente, hacer preguntas y contribuir a la comunidad. Esto te permitirá aprender de otros y establecer conexiones valiosas en el mundo de la programación. ¡Disfruta del viaje de aprendizaje!

Anexos

Ejercicios: Resolución

Capítulo 2: Fundamentos de la Codificación.

Estructura Secuencial: *Calcular el área de un triángulo dado su base y altura. Los valores de la base y la altura son ingresados por el usuario.*

```
INICIO
base, altura, área: número real // Declarar variables
// Solicitar al usuario que ingrese la base
Escribir "Ingrese la base del triángulo, "
Leer base

Escribir "Ingrese la altura del triángulo, " //la altura
Leer altura
área = (base * altura) /2 // Calcular el área

// Mostrar el resultado
Escribir "El área del Triángulo es, área"
FIN
```

Estructura Condicional: *Solicitar al usuario que ingrese su edad y determinar si es mayor o menor de edad según las leyes de Argentina.*

```
INICIO
edad: número entero // Declarar variables

Escribir "Ingrese su edad, "
Leer edad

Si edad >= 18 entonces // Verificar si es mayor o menor
  Escribir "Usted es mayor de edad"
Sino
  Escribir "Usted es menor de edad"
Fin Si
FIN
```

Estructura Repetitiva: *Calcular el factorial de un número ingresado por el usuario utilizando un bucle "Mientras ingresa un número positivo N. El programa debe calcular e imprimir la suma de todos los números pares desde 1 hasta N. Si el usuario ingresa un número negativo o cero, el programa debe mostrar un mensaje de error.*

```
INICIO
base, altura, área: número real // Declarar variables
Escribir "Ingrese la base del triángulo, " // Ingrese la base
Leer base

Escribir "Ingrese la altura del triángulo, " // altura
Leer altura

área = (base * altura) /2 // Calcular el área

Escribir "El área del Triángulo es, área" // resultado
FIN
```

Estructura Condicional: *Solicitar al usuario que ingrese su edad y determinar si es mayor o menor de edad según las leyes de Argentina.*

```
INICIO
numero, factorial, contador: número
factorial =1 // Inicializar variables
contador = 1

Escribir "Ingrese un número entero positivo: "// ingresar
     número
Leer numero
Mientras contador <= numero Hacer //Calcular factorial
    factorial = factorial * contador
     contador = contador + 1
Fin Mientras

Escribir "El factorial de ", numero, " es: ", factorial
FIN
```

¿Te animas a más?: Algoritmos en los que utilizamos todas las estructuras de control

Escribe un programa que solicite al usuario ingresar un número positivo N. Luego, el programa debe calcular e imprimir la suma de todos los números pares desde 1 hasta N. Si el usuario ingresa un número negativo o cero, el programa debe mostrar un mensaje de error.

```
INICIO
N, suma, contador: número entero // Declarar variables

suma = 0 // Inicializar variables

Escribir "Ingrese un número positivo N: "
Leer N

Si N <= 0 Entonces //Verificar si N es positivo
    Escribir "Error: El número debe ser positivo"
Sino

//Calcular la suma de números pares desde 1 hasta N
Para contador desde 2 hasta N paso 2 Hacer
     suma = suma + contador
Fin Para

Escribir "La suma de los números enteros pares desde 1 hasta
      ", N, "es: ", suma
FIN
```

Este algoritmo utiliza un bucle "PARA" para iterar a través de los números del 1 al 10. En cada iteración, verifica si el número actual es par (usando la operación MOD para verificar si el resultado de la división por 2 es igual a 0). Si el número es par, se agrega a la variable "suma". Al final, se muestra el resultado de la suma de los números pares.

```
INICIO
// Inicializamos una variable
suma = 0

// Iniciamos un bucle que va desde 1 hasta 10
Para i desde 1 Hasta 10 con Paso 1 Hacer
// Verificamos si el número actual (i) es par
Si i MOD 2 = = 0 Entonces
      // Si es par, lo agregamos a la suma
      Suma = Suma + i
Fin Si
Fin Para

Escribir "La suma del 1 al 10 es: ", suma
FIN
```

Capítulo 3 - Aprende un Lenguaje de Programación. Resoluciones.

Ejercicio 1: Calculadora de IMC

```python
peso = float(input("Ingresa tu peso en kilogramos: "))# el usuario ingresa kg
altura = float(input("Ingresa tu altura en metros: ")) # altura

imc = peso / (altura ** 2) # Calcular el IMC

print(f"Tu IMC es: {imc:.2f}")# Mostrar el IMC

if imc < 18.5: # Clasificar el IMC
    print("Tienes bajo peso.")
elif imc < 24.9:
    print("Tienes un peso normal.")
elif imc < 29.9:
    print("Tienes sobrepeso.")
else:
    print("Tienes obesidad.")
```

Ejercicio 2: Adivina el número

```
import random

# Generar un número aleatorio entre 1 y 100
numero_secreto = random.randint(1, 100)

intentos = 0
adivinado = False

print("¡Adivina el número entre 1 y 100!")

while not adivinado:
    intentos += 1
    guess = int(input("Ingresa tu suposición: "))

    if guess == numero_secreto:
        adivinado = True
    elif guess < numero_secreto:
        print("El número secreto es mayor. Intenta de nuevo.")
    else:
        print("El número secreto es menor. Intenta de nuevo.")

print(f"¡Felicidades! Adivinaste el número en {intentos}
intentos.")
```

PROMPTS a ChatGPT

- *Compórtate como si fueras un experto en programación, y debes escribir para adolescentes que transitan la educación secundaria, cómo explicarías en un libro el siguiente tema: ¿Qué es la programación?*
- *Puedes colocarte el rol de un experto en programación y explicar a estudiantes del nivel secundario de argentina, los conceptos básicos de algoritmo, el texto deberá ser creado para escribir un libro dirigido a estos estudiantes.*
- *En el mismo papel de experto en Programación, y de manera que los estudiantes de secundaria entiendan ¿cómo explicarías la historia de la programación? en un libro*
- *Como experto en programación ¿podrías darme un ejemplo de un algoritmo sencillo para que los estudiantes de secundaria de argentina aprendan a programar?*
- *Me darías otro ejemplo de algoritmo sencillo donde se incluya una estructura condicional para estudiantes del nivel secundario*
- *¿Podrías darme un ejemplo sencillo utilizando una estructura repetitiva en un algoritmo para estudiantes de secundaria?*
- *Me darías tres enunciados sencillos, diferentes a los anteriormente planteados que permitan desarrollar un algoritmo utilizando las tres estructuras de control,*
- *Ahora en el mismo rol de experto en Programación y dirigido a estudiantes del nivel secundario de adolescentes de argentina, podrías explicar sobre los tipos de lenguaje de programación y herramientas y entornos de desarrollo, para incluirlo en un libro orientado a dichos estudiantes.*
- *Como un experto en programación y siempre dirigiéndote a estudiantes del nivel secundario que viven en Argentina, que podrías explicar sobre Arte generativo con código, Animaciones y gráficos interactivos, Creación de música y sonido, esta información puede ser incluida en un libro dirigido a dichos estudiantes.*
- *me darías un ejemplo sencillo del Capítulo 1: Arte Generativo con Código, usando el lenguaje Python*
- *Me darías un ejemplo sencillo en el lenguaje Python sobre Capítulo 2: Animaciones y Gráficos Interactivos*
- *Como experto en programación, me darías un código sencillo en python que explique el Capítulo 3: Creación de Música y Sonido*
- *Escribe un prólogo para un libro de 60 páginas dirigido a los estudiantes del nivel secundario de Argentina, el libro se refiere a Programación y los primeros pasos en la misma, intenta inspirar en los jóvenes la curiosidad y ayudar a los docentes de informática con el contenido teórico y práctico.*

- *” como docente programador escribe una introducción al lenguaje python y javascript con un nivel de lenguaje accesible a alumnos de secundaria. “*
- *Escribe el script en python para el siguiente enunciado: verifica si un número ingresado por el usuario es positivo, negativo o cero.*
- *Desarrolla los siguientes puntos relacionados con aprendizaje en programación que serán incluidos en un libro para alumnos de secundaria. Los puntos son: Pensamiento lógico y resolución de problemas. - Cómo planificar y diseñar programas. - Depuración de código.*
- *Escribe un paso a paso de como instalar Python en Windows para alumnos de secundaria.*
- *escribe el script en python para el siguiente algoritmo: INICIO // Declarar variables nota1, nota2, nota3, promedio: número real // Solicitar al usuario que ingrese las tres notas Escribir “Ingrese la primer nota ” Leer nota1 Escribir “Ingrese la segunda nota” Leer nota2 Escribir “Ingrese la tercera nota ” Leer nota3 // Calcular el promedio promedio =(nota1 + nota2 + nota3) / 3 // Mostrar el resultado Escribir “El promedio de las tres notas es: “, promedio FIN*
- *nota1 no debería ser float también?*
- *Haz un compendio de las sintaxis de python y cuáles son las estructuras de control*
- *Compórtate como un experto y desarrolla contenidos sobre Conceptos de Seguridad Informática*
- *Compórtate como un experto y desarrolla contenido para las siguientes temáticas Seguridad en la Programación - Conceptos de seguridad informática. - Protección de datos y privacidad en línea. - Ética en la programación.*
- *Compórtate como un docente experto y sintetiza este texto en lenguaje para principiantes: Seguridad en la Programación……*
- *Comportarse como Docente experto, como si fueras Steve Jobs, para desarrollar un ensayo, en un tono amable y accesible, con una extensión de 3 apartados. Debe estar destinada a adolescentes, demostrando conocimientos en: Educación y pedagogía, Tecnología y programación, Tecnologías de la información y la comunicación. Demostrando habilidades en: Conocimientos en tecnología, Habilidades de programación. sobre estos temas: Programación para el Futuro Carreras en tecnología y programación. Proyectos de código abierto y contribuciones. Tendencias tecnológicas emergentes.*
- *Compórtate como un experto y lista Carreras técnicas y de grado en Tecnología y Programación*

- *Sintetiza este texto para que sea accesible a estudiantes de nivel secundario Programación para el Futuro: ¡Tu Aventura en el Mundo Digital!.....*
- *Compórtate como un experto y dame ideas de cómo generar ideas para programar aplicaciones*
- *Compórtate como un experto y relata la fundación de los emprendimientos tecnológicos y de programación más importantes y sus fundadores.*
- *Compórtate como un experto y describe la fundación de emprendimientos tecnológicos más importantes de argentina y sus fundadores*
- *Compórtate como un experto y sintetiza este texto Historia inspiradoras. La chispa que inicia el fuego.*
- *Si tuviéramos los siguientes títulos: Recursos y herramientas de la programación, bibliografía recomendada, sitios web y comunidades de programación, consejos para seguir aprendiendo y mejorando. cómo los desarrollarías en un libro, teniendo en cuenta que eres un experto en programación.*
- *Capítulo 2: Recursos y Herramientas de la Programación*
  Entornos de desarrollo (IDEs) y editores de texto recomendados.
  Control de versiones y sistemas de gestión de código fuente. Herramientas de depuración y prueba. Ahora en el mismo papel te pido que desarrolles esas secciones
- *Capítulo 3: Bibliografía Recomendada*
  Libros esenciales para programadores principiantes, intermedios y avanzados.
  Recomendaciones específicas para diferentes lenguajes de programación y áreas de desarrollo.
- *Capítulo 4: Sitios Web y Comunidades de Programación*
  Plataformas de aprendizaje en línea (Coursera, Udemy, edX, etc.).
  Foros y comunidades (Stack Overflow, Reddit, GitHub, etc.).
  Blogs y sitios web de referencia en programación.
- *Siguiendo en el rol de experto en programación, podrías desarrollar esto: Colaborando en Proyectos*
  Vamos a aprender cómo crear un repositorio en GitHub y cómo clonarlo en tu máquina local. Veremos cómo hacer cambios en tu código, hacer commit y realizar push para compartir tus contribuciones.
- *Asumiendo el rol de programador, podrías explicar cómo python u otros programas detectan automáticamente los errores, pero recuerda que debe ser entendible para adolescentes.*

ADOLESCENTES PROGRAMADORES.
¡CONVIERTE TUS IDEAS EN SOFTWARE!

Adolescentes programadores. ¡Convierte tus ideas en software!

¿Eres un adolescente curioso, apasionado por la tecnología y con ganas de dar vida a tus ideas creativas? ¡Has llegado al lugar correcto! "Adolescentes Programadores: Convierte tus ideas en software" es tu guía esencial para entrar en el emocionante mundo de la programación.

Este libro está diseñado específicamente para jóvenes con un deseo de aprender a programar. Descubrirás cómo convertir tus ideas en aplicaciones reales, explorarás los fundamentos de la codificación y te aventurarás en emocionantes proyectos de programación creativa. Desde la creación de juegos interactivos hasta la protección de tus creaciones con seguridad informática y ética en la programación, esta aventura te preparará para el futuro digital.

Aprenderás a:

·Explorar la historia de la programación y entender su evolución.
·Dominar los algoritmos y su papel en la resolución de problemas cotidianos.
·Elegir y trabajar con lenguajes de programación populares como Python y JavaScript.
·Crear programas simples y comprender las estructuras de control.
·Desarrollar tu creatividad mediante el arte generativo, animaciones y música con código.
·Conocer conceptos de seguridad informática para proteger tu trabajo y actuar de manera responsable.
·Explorar opciones de carreras en tecnología y programación.
·Descubrir proyectos innovadores liderados por jóvenes y cómo se volvieron millonarios.
·Acceder a recursos y herramientas esenciales para la programación.

Este libro es tu brújula en el vasto mundo de la programación. ¡Empieza tu viaje hoy y convierte tus sueños digitales en realidad!.