



ChatGPT: Tu Entrenador de Python

Aprende los Fundamentos en 100 Prompts

Sergio Rojas-Galeano

Sobre el Autor

Sergio Rojas-Galeano es un científico de la computación entusiasta de la Inteligencia Artificial, el Aprendizaje Automático, la Computación Evolutiva, el Procesamiento del Lenguaje Natural, la Ingeniería de Software y la Edición Científica. Con experiencia profesional tanto en posiciones académicas como industriales en Colombia e Inglaterra, está comprometido con la investigación, el avance y la promoción de los estándares éticos en la disciplina de la ciencia de la computación. Actualmente se desempeña como profesor titular en la Universidad Distrital Francisco José de Caldas (Colombia), donde ha enseñado programación e inteligencia computacional durante más de dos décadas. Con un doctorado en Ciencias de la Computación de University College London (UCL) en el Reino Unido (obtenido en 2009), también ha sido investigador visitante en UCL (en 2022) y en la Universidad de Málaga en España (en 2023). Es autor o coautor de más de 70 artículos científicos y anteriormente publicó el libro *Models of Learning and Optimisation for Data Scientists* (LeanPub, 2019) y la versión en inglés de este libro, *ChatGPT: Your Python Tutor* (LeanPub, 2023).

email: sargaleano@gmail.com

Licencia

Este documento está licenciado bajo:



El permiso para utilizar este documento está sujeto a la Licencia; cualquier otro uso está estrictamente prohibido. El contenido de este documento se proporciona "TAL CUAL", sin garantías de ningún tipo, ya sean expresas o implícitas. El autor no asume ninguna responsabilidad o responsabilidad por las consecuencias derivadas del uso de este documento, incluidos, entre otros, errores, omisiones o inexactitudes. El lector asume la responsabilidad total de la aplicación de los conceptos y técnicas descritos en este libro.

© 2023 por Sergio Rojas-Galeano.

Colofón

Este documento fue maquetado utilizando \LaTeX y la clase *kaobook*. El autor elaboró el contenido con la ayuda del sistema de generación de contenido ChatGPT de OpenAI, basado en sus propias ideas, propósito y estilo. El diseño de la portada fue creado por el autor utilizando plantillas y arte disponibles en la plataforma de diseño gráfico Canva. Todos los nombres de productos, logotipos, marcas, marcas comerciales y marcas registradas son propiedad de sus respectivos propietarios. Todos los nombres de empresas, productos y servicios, así como los nombres de celebridades, libros, personajes, bandas, clubes deportivos o películas mencionados en este libro, se utilizan únicamente con fines didácticos y de reconocimiento. El uso de estos nombres o marcas no implica respaldo alguno.

Primera edición – v1.0

Julio de 2023

Bogotá, Colombia

Prefacio

¡Bienvenido a *ChatGPT: Tu Entrenador de Python. Aprende los Fundamentos en 100 Prompts!* Si estás leyendo esto, significa que estás a punto de embarcarte en un emocionante viaje al mundo de la programación. Pero, ¿por qué Python, podrías preguntar? Bueno, Python es uno de los lenguajes de programación más ampliamente utilizados y versátiles, con aplicaciones que van desde el desarrollo web y análisis de datos hasta el aprendizaje automático e inteligencia artificial (IA).

De hecho, en los últimos años ha habido un creciente interés en la IA y sus aplicaciones, incluido el desarrollo de modelos de lenguaje como ChatGPT. Si bien estos modelos admiten cuestionamientos válidos acerca de las implicaciones éticas, los sesgos de datos y la posibilidad de generar contenido falso, vale la pena señalar que en el ámbito de la programación, el contenido generado por ChatGPT consiste principalmente de código que puede evaluarse fácilmente en cuanto a su validez y efectividad simplemente ejecutándolo, con lo cual tales riesgos se ven mitigados en cierta medida. Esto ofrece una ventaja al aminorar los fallos asociados con alucinaciones o información inexacta, ya que la salida se puede verificar a través de la misma máquina, cuando la interpreta. Es así que los modelos de lenguaje como ChatGPT se han convertido en herramientas valiosas para tareas de programación, ofreciendo a los desarrolladores una asistencia provechosa para sus proyectos de codificación.

Esta guía ha sido cuidadosamente diseñada para ayudarte a navegar por los conceptos básicos de la programación en Python con facilidad, usando ChatGPT como tu mentor de confianza. ChatGPT es un potente modelo de lenguaje diseñado para ayudarte con cualquier pregunta o duda que puedas tener mientras aprendes Python. Te proporcionará ejemplos, explicaciones y retroalimentación, haciendo que tu experiencia de aprendizaje sea interactiva y entretenida.

Con ella aprenderás los fundamentos de la programación en Python, incluyendo variables, tipos de datos, operadores, estructuras de control de flujo como condicionales y bucles, contenedores como listas y diccionarios, y conceptos de modularización y reutilización. También enfrentarás desafíos divertidos y emocionantes que pondrán a prueba tus nuevas habilidades.

Pero esta guía es más que una simple colección de ejemplos y ejercicios. Es una invitación a explorar el poder de la programación y su impacto potencial en nuestras vidas, y a comprobar cómo la IA y los modelos de lenguaje como ChatGPT pueden potenciar las habilidades y productividad de los programadores para crear aplicaciones poderosas para un futuro prometedor. A medida que te adentres en el mundo de Python, verás cómo la programación se puede utilizar para resolver problemas prácticos, automatizar tareas tediosas e innovar productos y servicios.

La mayoría del contenido de este libro, incluyendo este prólogo, fue generado inicialmente con la ayuda de ChatGPT. Sin embargo, es importante señalar que el material inicial sirvió como un borrador que luego revisé, ajusté y enriquecí ampliamente. Este proceso fue mediado por mi experiencia educativa, con el objetivo de crear un contenido didáctico e informativo para los lectores. Si bien ChatGPT desempeñó un papel en la generación del material base, el contenido final ha sido cuidadosamente seleccionado y moldeado para garantizar su valor educativo y correctitud.

Para promover la transparencia en este enfoque innovador de escritura académica asistida por IA, he decidido deliberadamente incluir una lista exhaustiva de las 100 indicaciones (o *prompts* como se conocen en Inglés) que concebí y utilicé para alimentar a ChatGPT, asegurando una cobertura completa del alcance temático previsto en esta guía. Esta lista se incorporará al final del libro para una fácil referencia. Además, esta *Lista de Prompts* habilita a los lectores para que puedan eventualmente reproducir una versión similar y personalizada del libro, que se ajuste a sus preferencias y necesidades individuales.

Además, para referir a cada *prompt* utilizaré citaciones numéricas correspondientes a su posición en la lista, en cada lugar del texto donde lo utilice. Por ejemplo, el primer *prompt* utilizado para generar el borrador inicial de este prólogo se cita seguidamente¹.

Por último, he creado un repositorio complementario que aloja una colección de *Jupyter Notebooks* que contienen los ejemplos de código de Python del libro, cuidadosamente organizados por capítulos para una navegación y exploración conveniente:

Link no disponible en la versión de muestra

Además, a partir del Capítulo 4, he agregado una característica llamativa, para los ejemplos de código más interesantes. Se trata de enlaces al sitio web [PythonTutor.com](https://pythontutor.com). Estos enlaces te permiten visualizar la ejecución paso a paso del código, observando dinámicamente el estado de la memoria del programa. Busca estos enlaces en la parte inferior de las listas de código y los recuadros de salida. ¡Es una forma fantástica de mejorar tu comprensión y darle vida al código!

Así que, si estás ansioso por aprender a programar y comenzar a crear programas con Python y ChatGPT, espero que encuentres esta guía útil y placentera de leer.

Sergio Rojas-Galeano
Bogotá, Julio 03, 2023

Índice general

Prefacio	vii
Índice general	ix
Introducción	3
¿Es este el libro adecuado para ti?	3
Acerca de Python	3
El ciclo de programación	4
Las herramientas	5
Uso de Colab	6
Uso de PythonTutor	9
ChatGPT como entrenador de programación	10
Uso de ChatGPT	12
Plantilla de <i>prompts</i> para ChatGPT	14
1 Un Vistazo a lo Esencial	17
1.1 ¡Hola, mundo!	17
1.2 Cadenas de caracteres y formato de cadenas	19
1.3 ¡Manos a la obra!	25
2 Datos y Variables en Acción	29
2.1 Tipos de datos	29
2.2 Entrada/salida básica	32
2.3 Un compañero de programación	34
2.4 ¡Manos a la obra!	38
3 La Sala de Operaciones	41
3.1 Operadores aritméticos	41
3.2 Operadores de comparación	43
3.3 Operadores lógicos	45
3.4 Operadores a nivel de bit	47
3.5 Operadores de asignación	51
3.6 Operadores de identidad	52
3.7 Operadores de pertenencia	54
3.8 Operador de exponenciación	55
3.9 Operadores matemáticos avanzados	56
3.10 ¡Manos a la obra!	58
4 La Torre de Control	61
4.1 Sentencia condicional if/else	62

4.2	Sentencia match/case	66
4.3	Sentencia de bucle for	68
4.4	Sentencia de bucle while	72
4.5	Estructuras de control anidadas	77
4.6	¡Manos a la obra!	84
5	Zona de Contenedores	95
5.1	Listas	95
5.2	Tuplas	103
5.3	Conjuntos	108
5.4	Cadenas de caracteres	112
5.5	Diccionarios	118
5.6	¡Manos a la obra!	132
6	Soluciones Modulares	139
6.1	Modularización y reusabilidad	139
6.2	Funciones	142
6.3	Clases y objetos	150
6.4	¡Manos a la obra!	161
7	Acto Final	169
7.1	¿Qué sigue a continuación?	169
7.2	Lecturas sugeridas	175
	Lista de los 100 <i>prompts</i>	177

Introducción

¿Es este el libro adecuado para ti?

Este libro está diseñado para ayudar a personas como tú, que se están iniciando en la programación, a aprender los conceptos básicos con facilidad, específicamente en el lenguaje Python, uno de los lenguajes de programación más versátiles y ampliamente utilizados.

Pero este libro es más que una guía de programación típica. Es como tener un entrenador dedicado a tu lado, guiándote en el viaje de aprendizaje con ejemplos, explicaciones y retroalimentación. Y ese entrenador resulta ser un modelo de lenguaje de IA: ChatGPT. Con ChatGPT como tu guía, tendrás acceso a una poderosa herramienta que te ayudará a comprender los conceptos de programación esenciales. Recibirás retroalimentación en tiempo real que te ayudará a aprender más rápido y de manera más efectiva, lo que en última instancia te permitirá convertirte en un programador competente.²

Este libro sirve como una guía introductoria a conceptos fundamentales en Python, incluyendo algoritmos, estructuras de control de flujo, contenedores y modularización. Dominar estos fundamentos te proporcionará una base sólida para abordar temas más avanzados en futuros volúmenes. Así que, asimila estos conceptos fundamentales y prepárate para los emocionantes desafíos que te esperan. Mientras tanto, este capítulo presenta las herramientas que se utilizarán a lo largo del libro. Si ya estás familiarizado/a con estas herramientas y deseas comenzar a programar directamente, puedes omitir este capítulo y pasar al Capítulo 1.

Acerca de Python

Python es un lenguaje de programación versátil y amigable para principiantes. Fue creado a finales de la década de 1980 por un programador holandés llamado Guido van Rossum, quien lo nombró en honor al grupo de comedia británico, Monty Python.³

Hoy en día, Python es utilizado para el desarrollo de software en numerosas organizaciones, desde la NASA hasta Instagram, para equipar desde rovers espaciales hasta filtros fotográficos. De hecho, es uno de los lenguajes de programación más populares en el mundo, y por una buena

Contenido de Capítulo

¿Es este libro adecuado para ti?.....	3
Acerca de Python.....	3
El ciclo de programación.....	4
Las herramientas.....	5
Uso de Colab.....	6
Uso de PythonTutor.....	9
ChatGPT como entrenador.....	10
Uso de ChatGPT.....	12
Plantillas de prompts.....	14

Monty Python

Monty Python es un grupo de comedia británico conocido por su humor único y absurdo. Su programa de televisión, Monty Python's Flying Circus, se emitió desde 1969 hasta 1974 y ganó un culto de seguidores. También crearon varias películas populares, incluyendo Monty Python and the Holy Grail y Life of Brian. Su influencia en la comedia y la cultura popular ha sido significativa y continúa siendo celebrada por fans de todo el mundo.⁴

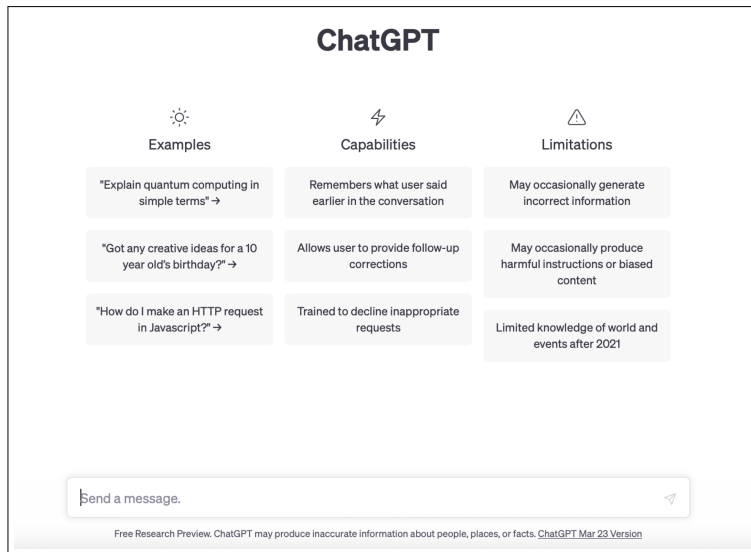


Figura 9: Página principal de ChatGPT.

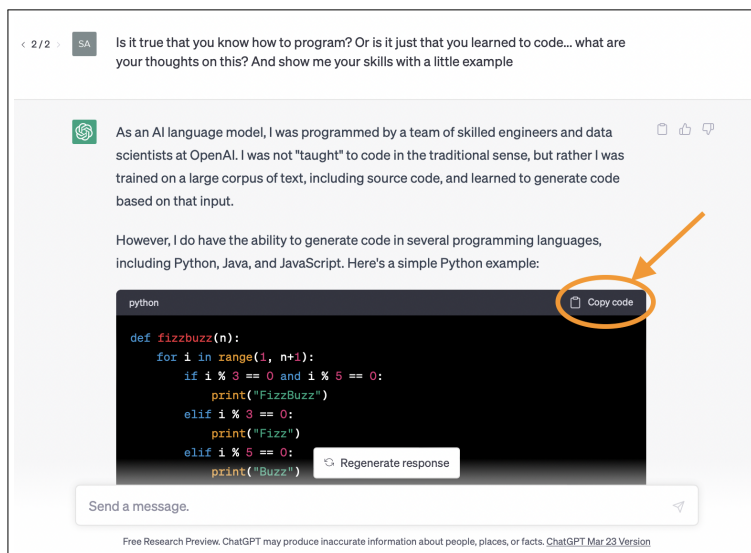


Figura 10: Página de chat de ChatGPT.

- Si la respuesta no responde completamente tu pregunta o necesitas más información, escribe otro *prompt*, envíalo a ChatGPT y continúa la conversación. El historial de conversaciones anteriores se mantiene en el lado izquierdo de la pantalla, por lo que puedes regresar y retomar cualquiera de ellas si lo deseas.

En este libro, utilizaré ChatGPT como se describió anteriormente. Para hacer el material más legible, formatearé los *prompts*, texto explicativo, código y salida de la respuesta usando diferentes cajas de texto de colores. A modo de ilustración, he reproducido la captura de pantalla anterior en el estilo de formato mencionado (traducido al Español):

¿Es cierto que sabes programar? ¿O es solo que aprendiste a codificar... qué piensas al respecto? Y muéstrame tus habilidades con un pequeño ejemplo.

Un Vistazo a lo Esencial

1

1.1 ¡Hola, mundo!

Explica el concepto del programa “¡Hola, mundo!” de una manera sencilla pero colorida, sin utilizar código. Además, incluye una anécdota relacionada para proporcionar a los principiantes en programación una introducción cálida al tema.¹⁴

¡Ah, el legendario programa *¡Hola, mundo!*! Esta es la forma perfecta de comenzar tu viaje en la programación, y es una experiencia que siempre recordarás. Imagínate sentado frente a tu computadora, sintiendo una mezcla de emoción y nerviosismo. ¡Estás a punto de escribir tu primer programa en Python!

El programa *¡Hola, mundo!* es el programa más sencillo que puedes escribir, pero es un evento significativo. Es el equivalente al primer llanto de un recién nacido o a la primera nota de un músico en un nuevo instrumento. Este programa es la base de todos los lenguajes de programación, y es el punto de partida perfecto para cualquiera que quiera aprender a programar.

Ahora, podrías preguntarte, “¿Por qué cominos querría escribir un programa que simplemente diga: *¡Hola, mundo!*?” Bueno, no se trata del mensaje en sí, sino del proceso de escribir el programa. Este programa sencillo es una excelente manera de familiarizarte con un nuevo lenguaje, superar el miedo a comenzar algo nuevo y ganar la confianza para seguir aprendiendo y explorando.

*Pero, ¿por qué el mensaje *¡Hola, mundo!*? ¿Quién lo inventó? ¿Cuál es su significado?¹⁵*

El origen del programa *¡Hola, mundo!* es algo incierto, pero se cree que se originó en los Bell Labs a principios de la década de 1970. La primera instancia registrada de la frase utilizada en un contexto de programación fue en un libro de 1974 sobre el lenguaje de programación C escrito por Brian Kernighan y Dennis Ritchie.

En cuanto al significado detrás del mensaje, simplemente es una forma para que los programadores prueben que su programa está funcionando correctamente y puede mostrar una salida. Es una tradición y un rito de iniciación para los principiantes en el mundo de la programación.

Bueno, suficiente de palabras. Vamos a sumergirnos en el código y comenzar la experiencia de entrenamiento con ChatGPT.

¡En sus marcas, listos... código!

Escribe el clásico programa '¡Hola, mundo!' en Python y explícalo. Proporciona también la salida. Utilizaré Colab.¹⁶

Aquí está el clásico programa *¡Hola, mundo!* en Python:

```
1 print("¡Hola, mundo!")
```

La función `print()` es una función incorporada en Python que te permite mostrar texto u otros datos en la consola, en este caso el texto (también conocido como *cadena*) "¡Hola, mundo!". La salida debería verse así:

```
iHola, mundo!
```

Para ejecutar este programa en Google Colab, sigue estos pasos:

1. Haz clic en el botón *Copy code* en ChatGPT.
2. Pégallo en la primera celda de tu cuaderno en Colab.
3. Haz clic en el botón ▷ junto a la celda o presiona *Shift + Enter*.

Explica qué es un error de sintaxis y muestra 3 posibles errores de sintaxis en el programa '¡Hola, mundo!' con sus respectivas salidas.¹⁷

Cuando escribes un programa, debes seguir las reglas de sintaxis correcta del lenguaje de programación. Un error de sintaxis ocurre cuando el código que escribes no cumple con dichas reglas, como omitir un signo de puntuación, utilizar mayúsculas incorrectas para una palabra clave o escribir un operador incorrecto. Estos errores evitan que el código se interprete correctamente. A continuación tres posibles errores de sintaxis que podrían haber ocurrido en el programa *¡Hola, mundo!*:

1. Paréntesis faltante:

```
print("iHola, mundo!"
File "<stdin>", line 1
print("iHola, mundo!"
                        ^
SyntaxError: unexpected EOF while parsing
```

El mensaje de error nos dice que se encontró el final inesperado del archivo (EOF) mientras se analizaba el código.

2. Palabra clave mal escrita:

```
prnt("iHOLA, mundo!")
File "<stdin>", line 1
  prnt("iHOLA, mundo!")
    ^
NameError: name 'prnt' is not defined
```

1.3 ¡Manos a la obra!

*Me gustaría poner en práctica lo que acabo de aprender sobre print() y el formateo de cadenas. Por favor, proporciona cuatro ejercicios de código en los que deba corregir los errores para que se ejecuten correctamente.*²⁵

Aquí tienes cuatro ejercicios en los que debes corregir los errores para que se ejecuten correctamente:

Ejercicio 1:

```
1 print(¡Hola, Mundo!)
```

Ejercicio 2:

```
1 mensaje = "Tengo {} manzanas"
2 print(mensaje+format(5))
```

Ejercicio 3:

```
1 nombre = input("¿Cuál es tu nombre? ")
2 edad = input("¿Cuántos años tienes? ")
3 print("Tu nombre es {} y tienes {} años."format(nombre, edad))
```

Ejercicio 4:

```
1 numero = 42
2 print("La respuesta es {:.10}".format(numero))
```

*Me gustaría poner en práctica lo que acabo de aprender sobre print() y el formateo de cadenas. Esta vez, sugiere algunos ejercicios que se puedan resolver utilizando solo print() y el formateo de cadenas.*²⁶

Ejercicio 1:

Imprime un recibo de compra como el que se muestra a continuación.

```
-----
|          RECIBO          |
| Artículo    | Cantidad | Precio|
|-----|-----|-----|
| Manzanas    | 3        | $1.50 |
| Bananas     | 2        | $0.80 |
| Naranjas    | 4        | $2.00 |
|-----|-----|-----|
```

Datos y Variables en Acción2

2.1 Tipos de datos

¿Qué es exactamente un programa? ¿Cómo se relaciona con los datos?³⁰

Un programa es un conjunto de instrucciones que le dice a una computadora qué hacer. Es una secuencia de declaraciones que son ejecutadas por la computadora para realizar una tarea específica. Los programas se escriben en un lenguaje de programación, que es un conjunto de reglas de sintaxis que define cómo deben escribirse tales instrucciones.

Los datos son cualquier tipo de información que puede ser almacenada y procesada por una computadora. Pueden ser números, texto, imágenes, audio o video. Los programas utilizan los datos para realizar sus tareas. Por ejemplo, un programa que calcula el promedio de un conjunto de números necesitará leer los números desde un archivo de datos o pedir al usuario que los ingrese.

La relación entre los programas y los datos es que los programas se pueden utilizar para crear, almacenar, recuperar y manipular datos. También se pueden utilizar para analizar datos y generar nueva información.

Aquí hay algunos ejemplos de cómo se relacionan los programas y los datos:

- ▶ Un procesador de texto utiliza datos (texto) para crear documentos.
- ▶ Una hoja de cálculo utiliza datos (números) para calcular valores.
- ▶ Una base de datos utiliza datos (registros) para almacenar información.
- ▶ Un navegador web utiliza datos (páginas web) para mostrar información en una pantalla.
- ▶ Un juego utiliza datos (personajes, niveles, etc.) para crear una experiencia interactiva.

Entendido. Explica con ejemplos y salidas en pantalla cómo se representa los datos en Python. Proporciona ejemplos de personajes de cómics para entender mejor el concepto.³¹

En Python, los datos se representan asignando valores a variables. Estos valores pueden ser de diferentes tipos, como cadenas de texto, enteros y booleanos. Aquí tienes algunos ejemplos:

```
1 # Asigna una cadena de texto a una variable
2 heroe = "Spider-Man"
3 print("Personaje:", heroe)
4
```

La Sala de Operaciones3

Enumera todos los operadores de Python y proporciona una breve descripción para cada uno.⁴²

Aquí tienes una lista de los diferentes tipos de operadores en Python y sus definiciones:

1. **Operadores aritméticos:** se utilizan para realizar operaciones matemáticas, como la suma, resta, multiplicación, división, etc.
2. **Operadores de comparación:** se utilizan para comparar valores y devolver un valor booleano (Verdadero o Falso), como igual a, no igual a, mayor que, menor que, etc.
3. **Operadores lógicos:** se utilizan para evaluar la verdad o falsedad de combinaciones de proposiciones o condiciones.
4. **Operadores a nivel de bits:** se utilizan para realizar operaciones en bits individuales de números binarios.
5. **Operadores de asignación:** se utilizan para asignar valores a variables, incluso operándolas con operadores aritméticos.
6. **Operadores de identidad:** se utilizan para comparar la ubicación en memoria de dos objetos.
7. **Operadores de pertenencia:** se utilizan para comprobar si un valor pertenece o no pertenece en una secuencia.
8. **Operador de exponenciación:** se utiliza para elevar un número a una potencia.

Así que demos un vistazo más de cerca a cada uno de ellos.

3.1 Operadores aritméticos

Explica los detalles de los operadores aritméticos en Python en un tono divertido e incluye un ejemplo de código para ilustrar el uso de todos los operadores. El ejemplo debe comparar estadísticas entretenidas sobre las películas de Marvel y X-Men y su recaudo en taquilla.⁴³

En Python, los operadores aritméticos se utilizan para realizar operaciones matemáticas básicas, como la suma, resta, multiplicación y división. Estos operadores se utilizan para manipular tipos de datos numéricos como enteros y números de punto flotante.

Seguidamente se presentan los operadores aritméticos clave en Python:

- **Suma:** El signo más (+) se utiliza para la suma de dos valores. Suma el segundo valor al primero. Por ejemplo, $2 + 2$ es igual a 4.
- **Resta:** El signo menos (-) se utiliza para la resta. Resta el segundo valor del primer valor. Por ejemplo, $5 - 2$ es igual a 3.

```

Real Madrid (70) y Barcelona (65) tienen la misma cantidad de puntos: False

Real Madrid (76) y Barcelona (85) tienen una cantidad diferente de goles anotados:
    True

Real Madrid (39) ha recibido menos goles que Barcelona (38): False

La diferencia de goles del Real Madrid (37) es menor o igual a la diferencia de
    goles del Barcelona (47): True

Real Madrid (70) tiene más puntos que Barcelona (65): True

Barcelona (38) ha recibido más goles que el Real Madrid (39): False

```

Como ilustra este ejemplo, estos operadores de comparación son muy útiles en la programación de Python, ya que nos permiten verificar las relaciones entre diferentes valores y tomar decisiones basadas en esas relaciones.

3.3 Operadores lógicos

Explica el concepto de los operadores lógicos en Python en un tono informal. Después de definir cada operador, incluye un ejemplo de código para ilustrar el uso de todos ellos. Para este fin, usa una tabla de platos de comida rápida indicando su idoneidad para vegetarianos y veganos.⁴⁶

Los operadores lógicos en Python son herramientas que nos permiten combinar y manipular condiciones booleanas que representan afirmaciones verdaderas o falsas. Estos operadores se utilizan principalmente con variables booleanas, que solo pueden ser **True** o **False**.

Los operadores lógicos más comúnmente utilizados en Python son **and** (y), **or** (o) y **not** (no). El operador **and** devuelve **True** solo si ambos operandos son **True**, mientras que el operador **or** devuelve **True** si al menos uno de los operandos es **True**. El operador **not** se utiliza para invertir el valor booleano de su operando.

Consideremos la siguiente tabla de platos de comida rápida que indica su idoneidad para vegetarianos y veganos:

Plato	¿Vegetariano?	¿Vegano?
Hamburguesa con queso	No	No
Pizza Margherita	Sí	No
Ensalada de jardín	Sí	Sí
Helado	Sí	No

La Torre de Control4

Enumera todas las estructuras de control en programación y proporciona una breve descripción de cada una.⁵⁹

En programación, existen varias estructuras de control que te permiten controlar el flujo de ejecución en tu código. Estas estructuras de control se pueden clasificar en tres tipos principales: estructuras de ramificación, estructuras de repetición y estructuras anidadas.

1. Estructuras de Ramificación (Condicionales):

- **Sentencia If/Else:** Ejecuta un bloque de código denotado `if`, si una cierta condición es verdadera. También puede incluir una declaración opcional `else` para proporcionar un bloque de código alternativo que se ejecutará si la condición es falsa.
- **Sentencia Match/case:** Evalúa una expresión y la compara con varios casos, ejecutando el bloque de código asociado con el caso coincidente.

2. Estructuras de Repetición (Bucles):

- **Bucle For:** Ejecuta un bloque de código un número específico de veces. Se utiliza con frecuencia cuando se conoce de antemano el número de iteraciones.
- **Bucle While:** Ejecuta un bloque de código repetidamente siempre que se cumpla una condición especificada. Es útil cuando el número de iteraciones no se conoce de antemano.
- **Bucle Do-While:** Similar al bucle **While**, pero garantiza que el bloque de código se ejecute al menos una vez antes de verificar el cumplimiento de la condición.

3. Estructuras Anidadas:

- **Condicionales Anidados:** Permite tener una sentencia `if/else` dentro de otra sentencia `if/else`. Proporciona una ramificación más compleja basada en múltiples condiciones.
- **Bucles Anidados:** Implica ejecutar un bucle dentro de otro bucle. Esto permite un control más intrincado sobre tareas repetitivas, con diferentes niveles de repetición.

Estas estructuras de control te brindan la capacidad de tomar decisiones, repetir acciones y concebir una lógica de programa más intrincada. Al combinar estas estructuras de manera efectiva, puedes crear programas potentes y flexibles para cumplir con diversos requisitos.

Así que veamos más de cerca cada una de ellas.

4.1 Sentencia condicional if/else

*Explicame todo lo que necesito saber sobre las sentencias if/else en Python de una manera divertida y creativa, con ejemplos didácticos.*⁶⁰

¡Claro! Imagina que eres un detective que resuelve casos en una ciudad llena de crimen. Tu trabajo consiste en tomar decisiones basadas en las pistas y pruebas que recopilas. Eso es precisamente de lo que tratan las instrucciones if/else: ¡tomar decisiones en tu código!

Aquí tienes un ejemplo didáctico: supongamos que eres un detective investigando un crimen misterioso. Tienes un sospechoso y necesitas determinar si es culpable o inocente. Puedes usar una instrucción if/else para ayudarte. La estructura general de esta instrucción condicional es:

```
1 if condicion:
2     # Código a ejecutar si la condición es verdadera
3 else:
4     # Código a ejecutar si la condición es falsa
```

Entonces, codifiquemos el ejemplo propuesto:

```
1 # Pide al usuario que ingrese su nombre
2 sospechoso = input("Ingresa tu nombre: ")
3
4 # Verifica si el nombre ingresado coincide con "Jack el Destripador"
5 if sospechoso == "Jack el Destripador":
6     # Si el nombre coincide, imprime un mensaje de arresto
7     print(f"¡Estás arrestado, {sospechoso}! ¡Eres culpable!")
8 else:
9     # Si el nombre no coincide, imprime un mensaje de inocencia
10    print(f"¡Puedes irte tranquilo, {sospechoso}! ¡Eres inocente!")
```

↖ Ejecuta paso a paso este código en PythonTutor: <https://tinyurl.com/ykdth9bd>

Aquí tienes dos ejecuciones diferentes de este programa:

```
Ingresa tu nombre: Sherlock Holmes
¡Puedes irte tranquilo, Sherlock Holmes! ¡Eres inocente!
```

```
Ingresa tu nombre: Jack el Destripador
¡Estás arrestado, Jack el Destripador! ¡Eres culpable!
```

Además, también puedes usar instrucciones if/elif/else para manejar más de dos escenarios diferentes. Estos escenarios pueden definirse combinando varias condiciones mediante

Zona de Contenedores

5

Explica qué es una estructura de contenedor, luego enumera todas las estructuras de contenedor disponibles en Python y proporciona una breve descripción para cada una, utilizando analogías de la vida real y un tono amigable.⁷⁶

Una estructura de contenedor en Python es una estructura de datos capaz de contener múltiples valores u objetos, organizados bajo un solo nombre. Permite organizar y almacenar datos de manera estructurada. Cada estructura de contenedor tiene sus propias características únicas y es adecuada para diferentes propósitos.

1. **Listas:** Piensa en una bolsa de compras donde puedes poner varios artículos. Es una colección ordenada de elementos que pueden ser de diferentes tipos.
2. **Tuplas:** Imagina un sobre sellado que contiene múltiples piezas de información. Una tupla es similar a una lista, pero es inmutable, lo que significa que no puede modificarse una vez creada.
3. **Conjuntos:** Visualiza un cajón lleno de elementos únicos. Un conjunto es una colección desordenada de elementos únicos.
4. **Cadenas de caracteres:** Considera una cadena como una oración o una secuencia de caracteres. Es un contenedor inmutable que contiene una colección de símbolos.
5. **Diccionarios:** Imagina un diccionario en la vida real donde se almacenan palabras y sus definiciones. Un diccionario es una estructura de datos de pares clave-valor.

Estas estructuras de contenedores brindan flexibilidad para organizar y manipular datos según tus necesidades específicas. Al elegir la estructura de contenedor adecuada, puedes almacenar y procesar datos de manera eficiente en tus programas de Python.

De manera que, demos un vistazo más de cerca a cada una de ellas.

5.1 Listas

Explica todo lo que necesito saber sobre listas en Python de una manera divertida y creativa, usando ejemplos lúdicos.⁷⁷

¡Claro! ¡Vamos a embarcarnos en una aventura divertida en el mundo de las listas de Python! Imagina que eres un mago con una mochila mágica que puede contener todo tipo de objetos fantásticos. En Python, las listas son como tu mochila mágica, donde puedes guardar, agregar, eliminar y recuperar elementos a tu conveniencia.

¡Súper! También mencionaste que los diccionarios nos permiten explorar relaciones y conexiones complejas entre los elementos dentro del diccionario. ¿Qué significa eso? ¿Pueden los diccionarios ayudarme a determinar relaciones como “los enemigos de mis aliados son mis enemigos”, “los aliados de mis aliados son mis amigos”, “los héroes con el mismo superpoder que el mío son mis amigos” o “mostrar una tabla de superhéroes ordenados por año de estreno”? Por favor, proporciona algunos ejemplos.⁹¹

¡Claro que sí! Aquí tienes cuatro ejemplos de cómo se pueden cruzar y relacionar datos utilizando diccionarios.

► Ejemplo 1: Los enemigos de mis aliados son mis enemigos.

```

1 # Construye un diccionario para almacenar enemigos extendidos
2 aliados_enemigos = {}
3 for hero, detalles in superheroes.items():
4     aliados = detalles["aliados"]
5     enemigos = detalles["enemigos"]
6     for aliado in [hero] + aliados:
7         # Comprueba si el héroe aún no está presente en aliados_enemigos
8         if hero not in aliados_enemigos:
9             aliados_enemigos[hero] = []
10        # Comprueba si el aliado existe en el diccionario superheroes
11        if aliado in superheroes:
12            # Ampliar la lista de enemigos del héroe con los enemigos del aliado
13            aliados_enemigos[hero].extend(superheroes[aliado]["enemigos"])
14
15 # Imprime los enemigos originales
16 print('-'*80)
17 print(f'{"Enemigos Originales":^80}')
18 print('-'*80)
19 for hero, detalles in superheroes.items():
20     print(f"{hero:^10} -> {detalles['enemigos']}")
21
22 # Imprime los enemigos extendidos
23 print('-'*80)
24 print(f'{"Los Enemigos de Mis Aliados Son Mis Enemigos":^80}')
25 print('-'*80)
26 for aliado, enemigos in aliados_enemigos.items():
27     print(f"{aliado:^10} -> {enemigos}")
28

```

↖ Ejecuta paso a paso este código en PythonTutor: <https://tinyurl.com/yckcbr4k>

6.1 Modularización y reusabilidad

Explica brevemente el concepto de modularización y reusabilidad, así como por qué son útiles en proyectos de programación. Luego, utilizando analogías del mundo real, explica los diversos enfoques para implementar estas ideas de manera clara y amigable.⁹⁴

La reusabilidad y la modularización son como técnicas avanzadas en el desarrollo de software, especialmente para proyectos grandes y complejos. Vienen con una serie de beneficios que facilitan la vida de los programadores.

Cuando escribes código reutilizable, no tienes que empezar desde cero cada vez. Puedes ahorrar mucho tiempo y esfuerzo utilizando soluciones existentes en lugar de reinventar la rueda. Y con la modularización, descompones tu código en piezas más pequeñas y manejables. Es como tener piezas de un rompecabezas que encajan perfectamente. Esto hace que sea más fácil de entender, mantener y corregir errores que surjan. Además, puedes reutilizar esas piezas modulares en diferentes partes de tu proyecto.

Así que, cuando se trata de abordar esos proyectos más grandes, adoptar la reusabilidad y la modularización es clave. Te ayuda a trabajar de manera más inteligente, colaborar sin problemas y mantener tu código robusto a largo plazo. En Python, tienes un montón de técnicas y prácticas interesantes para poner en marcha estas ideas:

- 1. Funciones:** Las funciones son como herramientas especializadas en una cocina. Encapsulan tareas específicas y pueden reutilizarse en diferentes recetas. De manera similar, en la programación, las funciones encapsulan operaciones específicas para solucionar tareas puntuales y detalladas. Al utilizar funciones, los programadores pueden escribir código modular que puede reutilizarse fácilmente, lo que facilita el manejo de proyectos más grandes y complejos.
- 2. Clases y Programación Orientada a Objetos (POO):** Las clases en POO son como trabajadores especializados en una organización. Cada trabajador tiene un rol y experiencia específicos. De manera similar, las clases encapsulan datos y comportamientos en objetos, promoviendo la especialización del código. Al aprovechar las clases y objetos, los programadores pueden crear un código modular y escalable, lo que facilita el manejo de la complejidad en proyectos más grandes.
- 3. Módulos y Paquetes:** Los módulos y paquetes son como cajas de herramientas con compartimentos etiquetados para diferentes herramientas. Proporcionan una forma de organizar y empaquetar funciones y clases relacionadas, promoviendo la reutilización del código. Al usar módulos y paquetes, los programadores pueden

Al llamar a la función `obtener_fecha_hora_notificacion()` varias veces en un bucle, puedes simular la recepción de notificaciones en diferentes momentos, reutilizando efectivamente el código sin necesidad de repetir las mismas líneas de código. De nuevo, así se promueve la reutilización del código, eliminando redundancias y garantizando un código más eficiente y conciso.

```
=== Obtener notificación ===
Nueva notificación recibida a las: 23/jul/2023 02:59:47
Nueva notificación recibida a las: 23/jul/2023 02:59:55
Nueva notificación recibida a las: 23/jul/2023 03:00:01
```

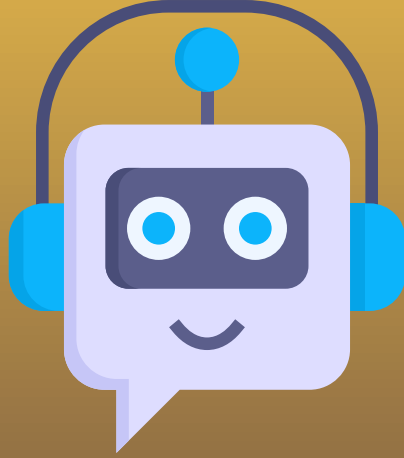
↖ Ejecuta paso a paso este código en PythonTutor: <https://tinyurl.com/ye29dv4m>

- **Valores de Retorno:** A continuación, se presentan algunos ejemplos para ilustrar este concepto.

```
1 # Ejemplo 1: Obtener Información del Usuario
2 def obtener_informacion_usuario():
3     # Simula la obtención de información del usuario desde una plataforma de redes
4     # sociales
5     nombre = "Harry Potter"
6     edad = 17
7     return nombre, edad
8
9 # Ejemplo 2: Obtener Foto de Perfil
10 def obtener_foto_perfil():
11     # Simula la obtención de la foto de perfil del usuario
12     url_foto = "https://twitter.com/Official_HPWiki/photo"
13     return url_foto
14
15 # Ejemplo 3: Obtener Cantidad de Amigos
16 def obtener_cantidad_amigos():
17     # Simula la obtención del número de amigos en la red del usuario
18     amigos = ["Hermione", "Ron", "Neville", "Luna"]
19     return len(amigos)
20
21 # Función Principal
22 def main():
23     # Obtener datos del perfil del usuario
24     nombre, edad = obtener_informacion_usuario()
25     url_foto = obtener_foto_perfil()
26     cantidad_amigos = obtener_cantidad_amigos()
```

Lista de los 100 *Prompts*

1. “ChatGPT: Tu Entrenador de Python. Aprende los Fundamentos en 100 Prompts” es el título de un libro que estoy escribiendo para enseñar los fundamentos de Python. Proporciona una versión preliminar de un prólogo atractivo y provocador. Enfatiza el impacto que la IA, y los modelos de lenguaje en particular, están teniendo en muchas áreas, incluida la programación, y cómo se espera que aprovechen las habilidades y la productividad de los programadores para crear aplicaciones poderosas en un futuro cercano, mencionando de pasada que los problemas relacionados con el contenido falso y la generación de información errónea que se han detectado en estos modelos de lenguaje se alivian en el área de la programación porque el código generado se puede validar simplemente ejecutándolo. También comenta que el contenido generado inicialmente ha sido cuidadosamente revisado y enriquecido basado en mi experiencia educativa.
2. Explica con un tono colorido que el propósito de este libro es introducir a un principiante en el mundo de la programación, en particular en el lenguaje Python, y ayudarlo a aprender los conceptos básicos con ejemplos, explicaciones y retroalimentación, al igual que un profesor guía a un estudiante en su viaje de aprendizaje. Solo que aquí, ese profesor es ChatGPT. Enfatiza las nuevas posibilidades y ventajas de utilizar desarrollos modernos en IA para ayudar en el proceso de aprendizaje y desarrollo de programadores.
3. Proporciona una descripción ingeniosa del lenguaje de programación Python que sea adecuada para un estudiante universitario de primer año. Haga algunas notas sobre la historia del lenguaje y su versión actual, especula sobre su futuro y sugiere enlaces útiles a organizaciones o sitios web donde el lector pueda obtener más información técnica.
4. Escribe una breve nota sobre Monty Python.
5. Describe brevemente las diferencias clave entre las versiones Python 2.x y Python 3.x. Destaca las ventajas de la versión más reciente.
6. Crea un esquema del ciclo estándar de desarrollo de programas, utilizando el paquete LaTeX smartdiagram con el tipo de diagrama circular.
7. Explica a un lector universitario de primer año el ciclo estándar de desarrollo de programas, incluyendo los pasos de escribir (codificar), ejecutar, probar y corregir (depurar). Hazlo con una lista con viñetas.
8. Aclara que Python es un lenguaje de programación interpretado; explica cómo difiere de un lenguaje compilado (ventajas y desventajas). Menciona que hay una variedad de IDEs disponibles, pero en este libro, en lugar de lidiar con problemas de instalación de software, utilizaremos entornos listos para usar que satisfacen las necesidades de este libro, para que podamos centrarnos en conceptos y algoritmos. Se utilizarán Jupiter Notebooks, Colaboratory, PythonTutor y ChatGPT como herramientas. Haz una lista con viñetas de estas herramientas, incluyendo una breve descripción, el sitio web y los pasos necesarios para abrir una cuenta si se requiere.
9. Describe detalladamente cómo crear un nuevo cuaderno Jupyter en Google Colab.
10. Explica los dos tipos de celdas en Jupyter Notebook.
11. Explica en detalle cómo usar PythonTutor.



Esta guía ha sido cuidadosamente diseñada para ayudarte a navegar los conceptos básicos de la programación en Python con facilidad, usando ChatGPT como tu confiable entrenador.

ChatGPT es un potente modelo de lenguaje capaz de ayudarte a resolver preguntas o dudas que surjan mientras aprendes Python.

Te proporcionará ejemplos, explicaciones y retroalimentación, haciendo que tu experiencia de aprendizaje sea interactiva y estimulante.