GUÍA ACADÉMICA

FUNDAMENTOS DE PROGRAMACIÓN CON PYTHON

LIC. VERÓNICA SANCHEZ MSc

Imagen generada con DALL-E, inteligencia artificial de OpenAI

INSTITUTO SUPERIOR TECNOLÓGICO
EL LIBERTADOR

dTC
DOCTRINA
TECH

Fundamentos de Programación con Python - Guía Académica

Fundamentos de Programación con Python - Guía Académica

1era Edición 2024

Sánchez Aguiar, Verónica Elizabeth

Editorial: Doctrinatech S.A.S, Quito - Ecuador 2024

ISBN: 978-9942-7128-2-0

Área: UMJ - Programación funcional

Materia: 005 - Programación. programas. datos de computadores

Tipo de Contenido: Computación y sistemas

Páginas: 100

INTRODUCCIÓN

La Guía Académica de Programación en Python, diseñada para la carrera de Electrónica del Instituto El Libertador, es un recurso educativo integral que combina teoría, práctica y evaluación en un formato accesible y orientado al aprendizaje activo, ofrece a los estudiantes una inmersión en la programación, enfocándose en aplicaciones directamente relacionadas con la electrónica. Nace como respuesta a la necesidad de formar profesionales que no solo dominen los conceptos fundamentales de la electrónica, sino que además cuenten con competencias en programación, una habilidad cada vez más demandada en el campo tecnológico actual. Python, por su versatilidad, claridad y amplia adopción en áreas como la automatización, el procesamiento de señales y el control de dispositivos, se presenta como el lenguaje ideal para iniciar a los estudiantes en el desarrollo de soluciones computacionales aplicadas a su especialidad.

El material ha sido diseñado bajo un enfoque pedagógico que equilibra el aprendizaje conceptual con la aplicación práctica, siguiendo una progresión gradual de dificultad que permite al estudiante construir su conocimiento de manera sólida, cada unidad temática está estructurada para introducir los fundamentos teóricos de la programación, reforzarlos con ejemplos concretos —muchos de ellos vinculados al contexto de la electrónica—, y culminar con actividades de práctica y autoevaluación de esta forma el estudiante no solo aprende a programar, sino que comprende cómo esa programación puede integrarse con sistemas electrónicos.

La guía también incorpora actividades contextualizadas, orientadas a resolver problemas reales mediante código, por ejemplo, los estudiantes podrán escribir programas que simulan el encendido de LEDs según condiciones lógicas, representar señales digitales, calcular parámetros eléctricos a partir de entradas del usuario, o registrar mediciones simuladas de sensores para analizarlas en tiempo real. Estas tareas promueven el razonamiento lógico, el pensamiento algorítmico y la comprensión de las aplicaciones del software en el hardware electrónico.

Finalmente, este recurso también sirve como base para el trabajo autónomo, permitiendo que los estudiantes refuercen lo aprendido en el aula, desarrollen pequeños proyectos personales y adquieran una mayor autonomía en su proceso formativo, se espera que los estudiantes no solo hayan adquirido competencias en programación básica con Python, sino que estén en capacidad de integrar dichas habilidades en sus prácticas y proyectos dentro del campo de la electrónica.

🧠 ¿Qué es Programar?

Vivimos rodeados de tecnología, teléfonos inteligentes, semáforos automatizados, electrodomésticos inteligentes, cajeros automáticos, sistemas de control industrial, entre muchos otros ejemplos, todos estos dispositivos tienen algo en común: funcionan gracias a instrucciones previamente escritas por una persona, esas instrucciones forman lo que se conoce como programas, y el proceso de crearlos se llama ***programar***.

Ilustración 1

Aprendiendo Programación en Python Aplicada a la Electrónica

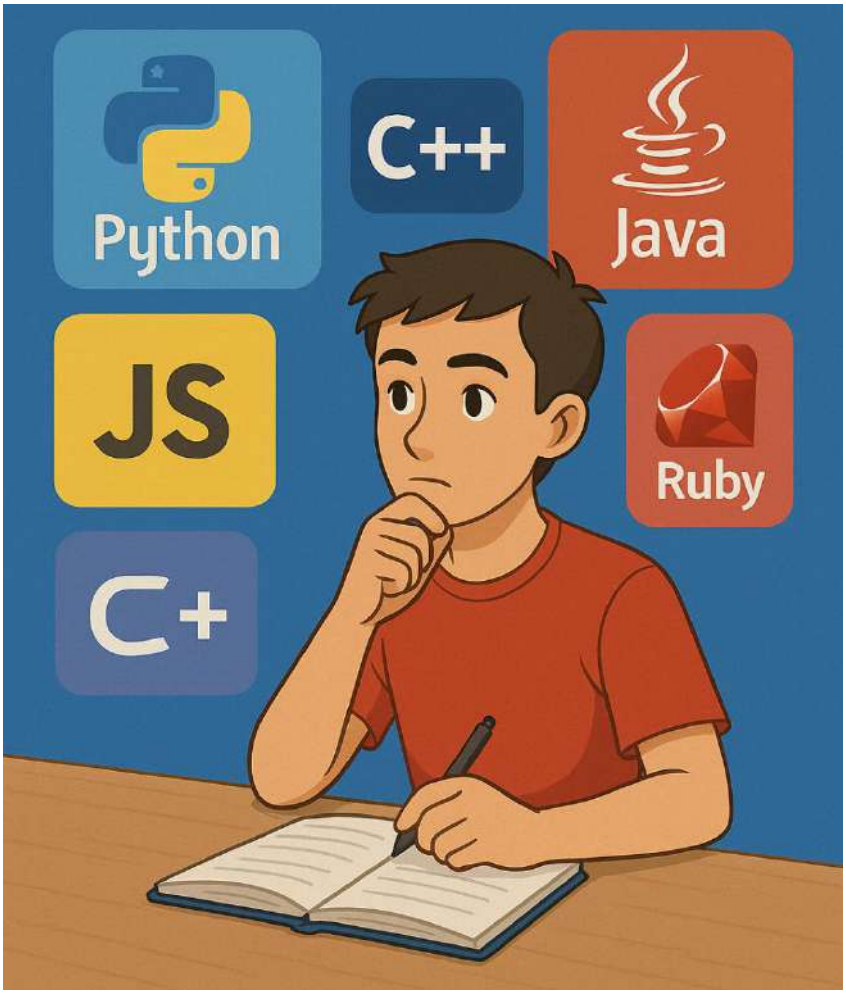
Programar consiste entonces, en términos generales, en dar instrucciones precisas a una computadora para que realice una tarea, estas instrucciones no se dan de forma verbal ni en lenguaje natural, sino utilizando lenguajes de programación, que son estructuras formales y lógicas que permiten comunicarnos con los dispositivos de forma eficiente. Lejos de ser únicamente una actividad técnica, la programación es también una forma de pensamiento que combina creatividad, lógica, resolución de problemas y capacidad de abstracción, uno de los conceptos clave que subyace al acto de programar es el pensamiento computacional.

Este pensamiento es la habilidad de descomponer un problema complejo en partes más pequeñas y manejables, identificar patrones que se repiten, abstraer lo esencial del problema dejando de lado lo accesorio, y diseñar una solución paso a paso. Estas habilidades no sólo se aplican al mundo de la informática, sino también a la vida cotidiana, al trabajo científico, a la ingeniería, a la educación y más, por ello, aprender a programar no es solo aprender a escribir código: es también aprender a pensar de manera lógica y estructurada.

Para programar, necesitamos utilizar lenguajes de programación, algunos de los más conocidos son Python, Java, C++, JavaScript y Ruby, cada uno tiene características propias y se utiliza para distintos fines, aunque todos permiten expresar ideas y algoritmos que luego son ejecutados por la computadora. Python, en particular, ha ganado una enorme popularidad por su simplicidad, claridad y versatilidad, se utiliza tanto en la enseñanza como en el desarrollo de aplicaciones web, análisis de datos, inteligencia artificial, automatización de procesos y muchas otras áreas.

Ilustración 2

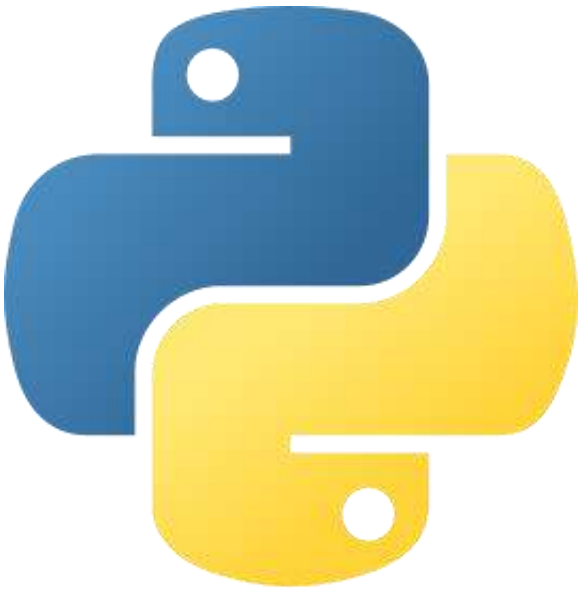
Lenguajes de Programación: Herramientas para Crear con la Computadora



Python ofrece una gran ventaja para los principiantes porque permite escribir programas útiles con muy pocas líneas de código, por ejemplo, un programa tan simple como pedir el nombre al usuario y saludarlo puede escribirse en dos líneas: una para solicitar el nombre con la función input() y otra para mostrar el saludo con la función print(), esto permite enfocarse en la lógica y la estructura del pensamiento, sin quedar atrapado en una sintaxis complicada.

Ilustración 3

Logotipo de Python



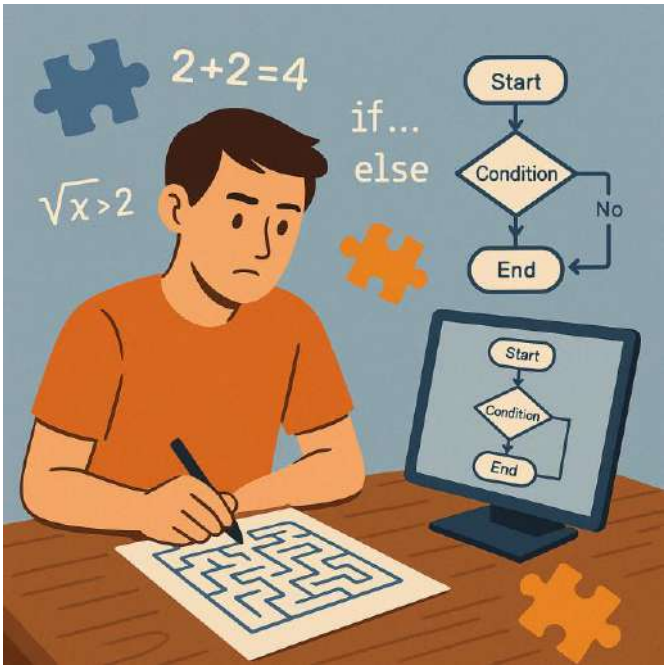
Programar es mucho más que escribir líneas de código, es una forma de pensar, de resolver problemas y de construir soluciones que pueden tener un impacto real, es también una herramienta educativa poderosa que permite desarrollar habilidades transversales como el razonamiento lógico, la atención al detalle, la creatividad y la perseverancia. Al iniciar esta guía, el lector comenzará un camino que no solo lo llevará a entender cómo funciona Python, sino también a pensar como un programador: alguien que no se rinde ante los problemas, sino que los descompone, los analiza y los resuelve paso a paso.

> *Con práctica, dedicación y curiosidad, aprender a programar puede convertirse no solo en una competencia útil, sino también en una fuente de satisfacción personal y profesional.*

🧠 Pensamiento Lógico y Algoritmos

En el estudio de la programación y especialmente en su aplicación dentro del campo de la electrónica, el desarrollo del pensamiento lógico representa una habilidad fundamental, antes de aprender cualquier lenguaje de programación, incluso antes de escribir una sola línea de código, es necesario aprender a pensar como un programador. Esto implica razonar de forma estructurada, descomponer problemas complejos en partes más simples, y encontrar soluciones paso a paso, esta capacidad de razonamiento ordenado se conoce como pensamiento lógico, y es el cimiento sobre el cual se construye todo el aprendizaje posterior en programación.

Ilustración 4
Pensamiento Lógico



El pensamiento lógico se manifiesta cuando una persona puede identificar relaciones de causa y efecto, aplicar reglas de manera sistemática y evaluar posibles soluciones para elegir la más adecuada, en el contexto de la electrónica, este tipo de pensamiento se refleja al analizar circuitos, interpretar señales, o diseñar soluciones automatizadas. Por ejemplo, al trabajar con un sistema de control de temperatura, un técnico en electrónica debe considerar condiciones como la temperatura actual, el umbral deseado, y la acción a tomar (encender o apagar un ventilador). Estas decisiones, aunque puedan parecer simples, se basan en estructuras lógicas que luego se pueden traducir fácilmente a un lenguaje de programación como Python.

Para que una computadora pueda ejecutar una tarea, necesita una secuencia clara de instrucciones, esta secuencia se conoce como algoritmo que no es otra cosa que un conjunto de pasos finitos, ordenados y definidos, que permiten resolver un problema o realizar una tarea, son independientes del lenguaje de programación; pueden escribirse en lenguaje natural, como una receta de cocina, o representarse gráficamente mediante diagramas de flujo o pseudocódigo.

Por ejemplo, si deseamos calcular el valor promedio de tres tensiones medidas en un circuito, el algoritmo podría ser el siguiente:

1. Leer el primer valor de voltaje
2. Leer el segundo valor
3. Leer el tercer valor,
4. Sumar los tres valores
5. Dividir el total entre tres
6. Mostrar el resultado.

Cada uno de estos pasos puede luego transformarse en una instrucción programable en Python, manteniendo la misma lógica subyacente, el desarrollo de algoritmos ayuda a entrenar el pensamiento secuencial, a visualizar el comportamiento esperado de un sistema y a prever posibles errores o condiciones especiales que puedan surgir. Esto es especialmente útil en aplicaciones electrónicas, donde un fallo en la lógica de control puede afectar el funcionamiento de un dispositivo o provocar resultados inesperados.

¿Por qué Python?

En el mundo de la programación, existen decenas de lenguajes que han sido diseñados para distintos propósitos, algunos son ideales para el desarrollo de aplicaciones web, otros para la creación de videojuegos, otros para el procesamiento de sistemas embebidos, y muchos para fines generales, ante tantas opciones, surge una pregunta lógica:

¿Por qué elegir Python como lenguaje base en la formación de los estudiantes de electrónica?

La respuesta se encuentra en la combinación de accesibilidad, versatilidad y poder que ofrece Python, este lenguaje fue creado con una premisa fundamental: ser fácil de leer y de aprender, sin sacrificar funcionalidad.

Su sintaxis es clara, directa y muy cercana al lenguaje humano, lo que permite a los estudiantes enfocarse en la lógica y el diseño de las soluciones, en lugar de perderse en detalles técnicos o sintácticos complejos.

Por ejemplo, para mostrar un saludo en pantalla, en Python basta con escribir print ("Hola, mundo"), sin necesidad de definir estructuras ni configuraciones previas.

Ilustración 5

Python como lenguaje de propósito general

Python es un lenguaje de propósito general, lo que significa que puede usarse en una amplia variedad de contextos: desde automatización industrial hasta inteligencia artificial, desde análisis de datos hasta control de dispositivos, esta característica lo convierte en una herramienta ideal para los estudiantes de electrónica, quienes eventualmente necesitarán interactuar con sensores, microcontroladores, sistemas de adquisición de datos, o redes de comunicación.

Con Python, es posible escribir scripts que controlen dispositivos a través de puertos seriales, procesen señales digitales, o gestionen grandes volúmenes de información en tiempo real, uno de los mayores beneficios es su ecosistema. Existen miles de bibliotecas disponibles que extienden sus capacidades, muchas de las cuales están orientadas directamente a la ingeniería, por ejemplo, bibliotecas como NumPy y SciPy permiten realizar cálculos matemáticos avanzados, matplotlib permite graficar datos con facilidad, y pyserial permite la comunicación entre Python y dispositivos electrónicos mediante puertos de comunicación.

> *Esto significa que los estudiantes no solo aprenden a programar, sino que también pueden aplicar de inmediato esos conocimientos en proyectos relacionados con su carrera*.

Python es además un lenguaje multiplataforma: funciona en Windows, macOS y Linux, y se puede ejecutar tanto en computadoras de escritorio como en dispositivos pequeños como la Raspberry Pi, esta portabilidad lo hace especialmente útil en entornos educativos, donde se valora el acceso desde distintos equipos o laboratorios, al ser de código abierto, no requiere licencias comerciales, lo cual lo convierte en una herramienta económicamente accesible para instituciones educativas y estudiantes por igual.

ACTIVIDAD 1

Objetivo:	Desarrollar el pensamiento lógico mediante la identificación y estructuración de pasos ordenados para resolver un problema cotidiano, como base para el diseño de algoritmos y futuras implementaciones en Python.
Descripción:	Antes de programar, es fundamental aprender a descomponer un problema en pasos claros, precisos y lógicos, esta actividad busca que los estudiantes comprendan qué es un algoritmo desde una perspectiva no técnica, utilizando una situación de la vida diaria. A través del análisis de una tarea común, los estudiantes aprenderán a pensar como programadores: ordenando acciones, identificando decisiones y estructurando secuencias.
Instrucciones:	El docente propondrá uno o varios problemas de la vida cotidiana, por ejemplo: • Preparar una taza de café. • Encender una lámpara desde dos interruptores. • Verificar si un cargador de celular está funcionando correctamente. • Encender un ventilador cuando la temperatura supere un umbral. En grupos o de forma individual, los estudiantes deberán: • Describir el problema en sus propias palabras. • Listar los elementos involucrados (objetos, personas, condiciones, decisiones). • Escribir paso a paso las instrucciones necesarias para resolver el problema, utilizando un lenguaje claro y lógico. • Identificar si hay decisiones que requieren condiciones (por ejemplo, “si la taza está vacía, llenar con agua”). • Representar el procedimiento con un pseudocódigo o un diagrama de flujo básico. Una vez completada la tarea, los estudiantes compartirán sus soluciones con la clase y se analizarán las distintas formas de representar un mismo proceso.
Resultados esperados:	• Los estudiantes comprenderán la importancia de detallar las instrucciones. • Aprenderán a trabajar con lógica secuencial y condicional. • Reconocerán la necesidad de ser precisos para que una máquina pueda seguir las instrucciones sin ambigüedades.
Relación con la asignatura	Esta actividad sienta las bases para pensar en automatización y control de dispositivos, tal como se realiza en sistemas embebidos, domótica o control con sensores. Entender el comportamiento lógico de un proceso cotidiano facilitará la transición hacia la programación con Python y su aplicación en situaciones reales del campo electrónico.

🔧💻 Instalación y entorno de trabajo

Antes de comenzar a programar con Python, es fundamental contar con un entorno de trabajo adecuado que permita escribir, ejecutar y depurar código de manera eficiente, esta sección guiará al estudiante en el proceso de instalación del lenguaje Python y en la elección de herramientas simples y funcionales para su uso educativo, como IDLE o editores ligeros. Tener un entorno configurado correctamente no solo facilita el aprendizaje, sino que permite concentrarse en la lógica y estructura del programa, sin distracciones técnicas.

1. Descargar Python 3 desde el sitio oficial

Para comenzar, accede al sitio web oficial de Python:

https://www.python.org/downloads/

En la página principal aparecerá un botón con la última versión recomendada para tu sistema operativo (Windows, macOS o Linux). Haz clic en el botón “Download Python 3.x.x” para descargar el instalador.

Ilustración 6

Descarga de python3

Asegúrate de que la versión sea Python 3, no Python 2, ya que es la versión moderna, segura y compatible con las herramientas que utilizaremos.

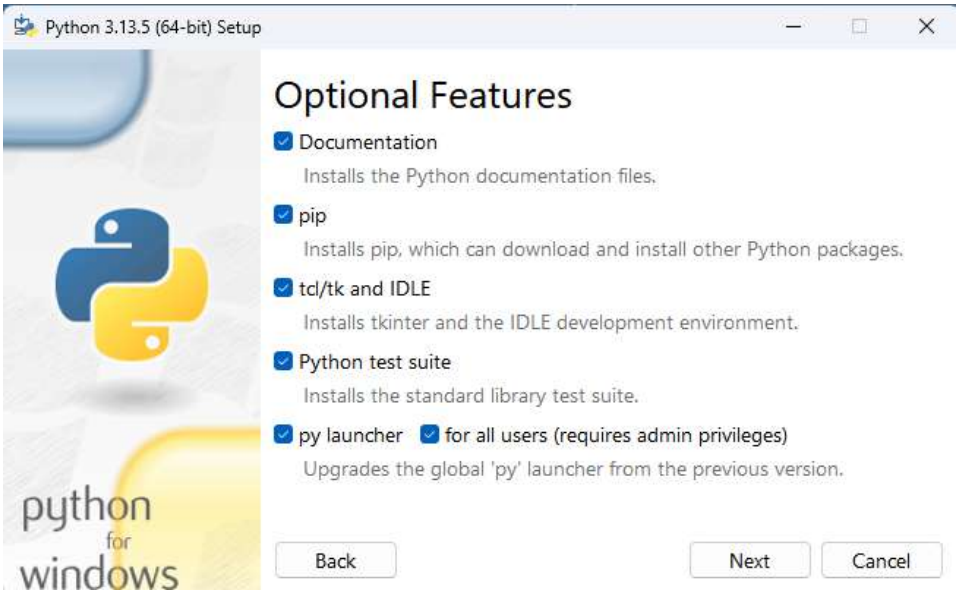
2. Instalar Python en tu equipo

Una vez descargado el archivo, ábrelo para iniciar la instalación.

Muy importante: antes de hacer clic en “Install Now”, activa la opción “Add Python to PATH” (esto permite usar Python desde la terminal).

Ilustración 7

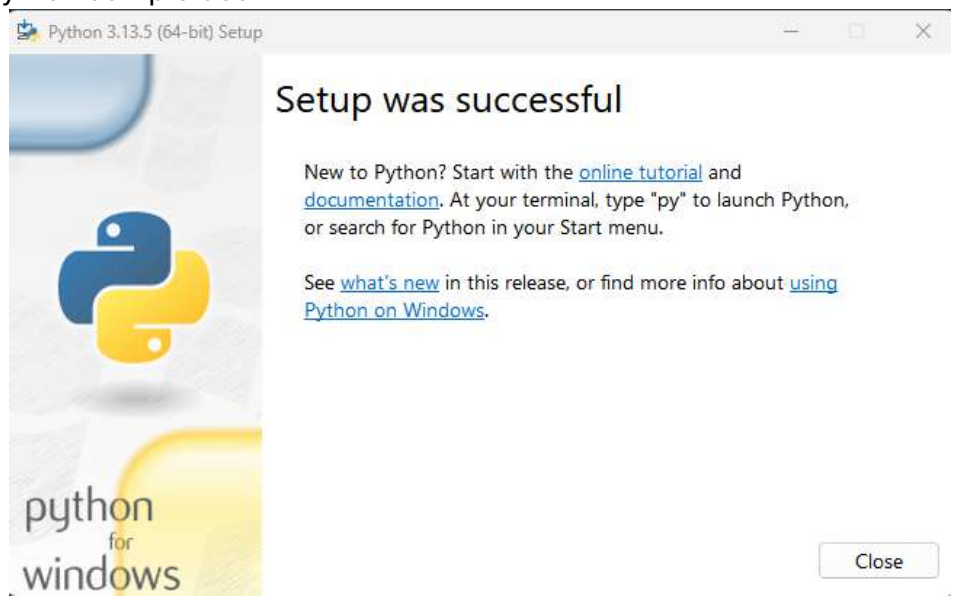
Instalación de Python3



Luego haz clic en “Install Now” y espera a que finalice el proceso. Al completar, verás una pantalla con el mensaje “Setup was successful”.

Ilustración 8

Instalación de Python completada



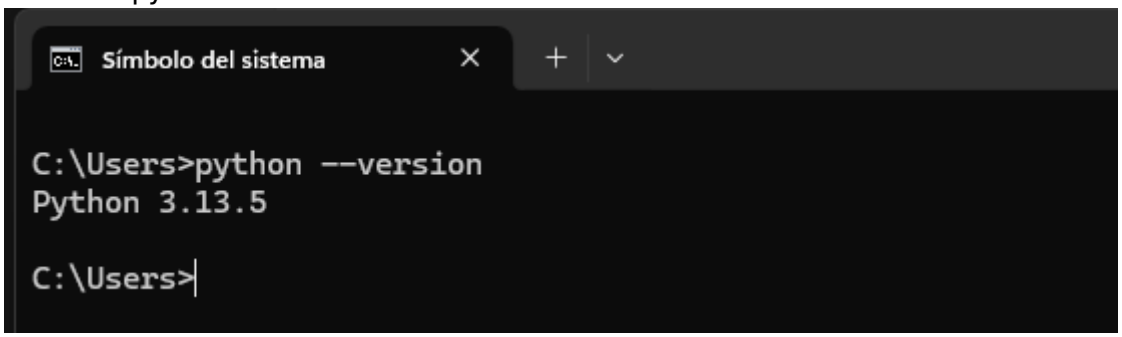
3. Verificar la instalación

Para comprobar que Python se ha instalado correctamente, abre la terminal de tu sistema:

En Windows: busca “CMD” o “Símbolo del sistema”. En Linux/macOS: abre la terminal.

Escribe el siguiente comando: `python --version`

Ilustración 9
Verificar versión de python



4. Instalar Visual Studio Code (VS Code)

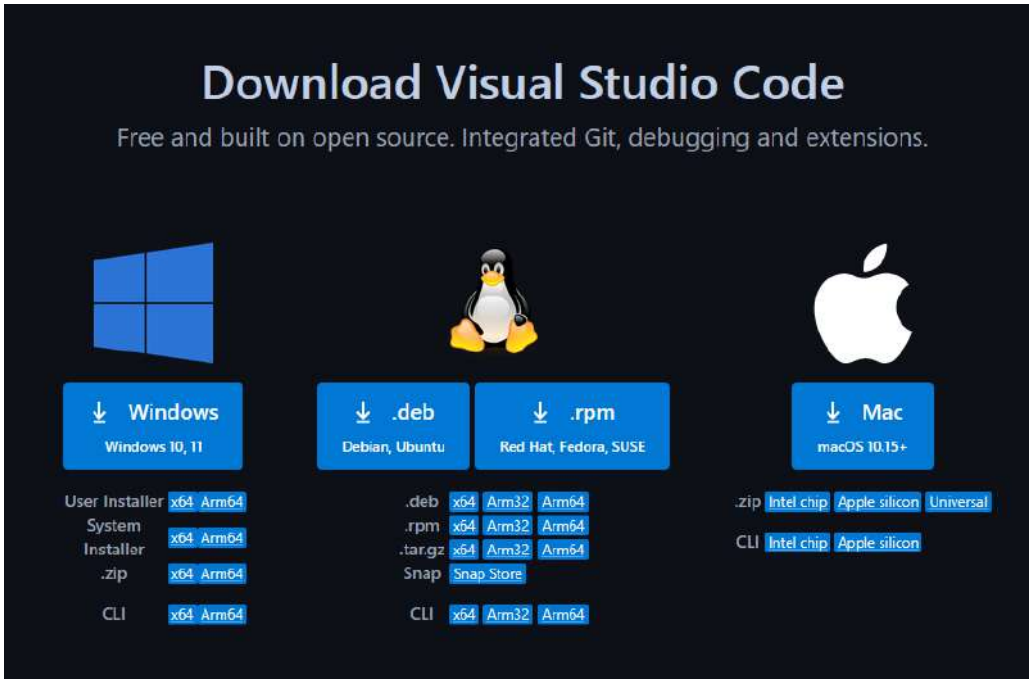
Accede al sitio oficial de Visual Studio Code:

https://code.visualstudio.com/

Haz clic en el botón “Download for Windows” (o el sistema que uses) y ejecuta el instalador.

Ilustración 10

Descarga de Visual Studio Code



Durante la instalación, se recomienda marcar las casillas:

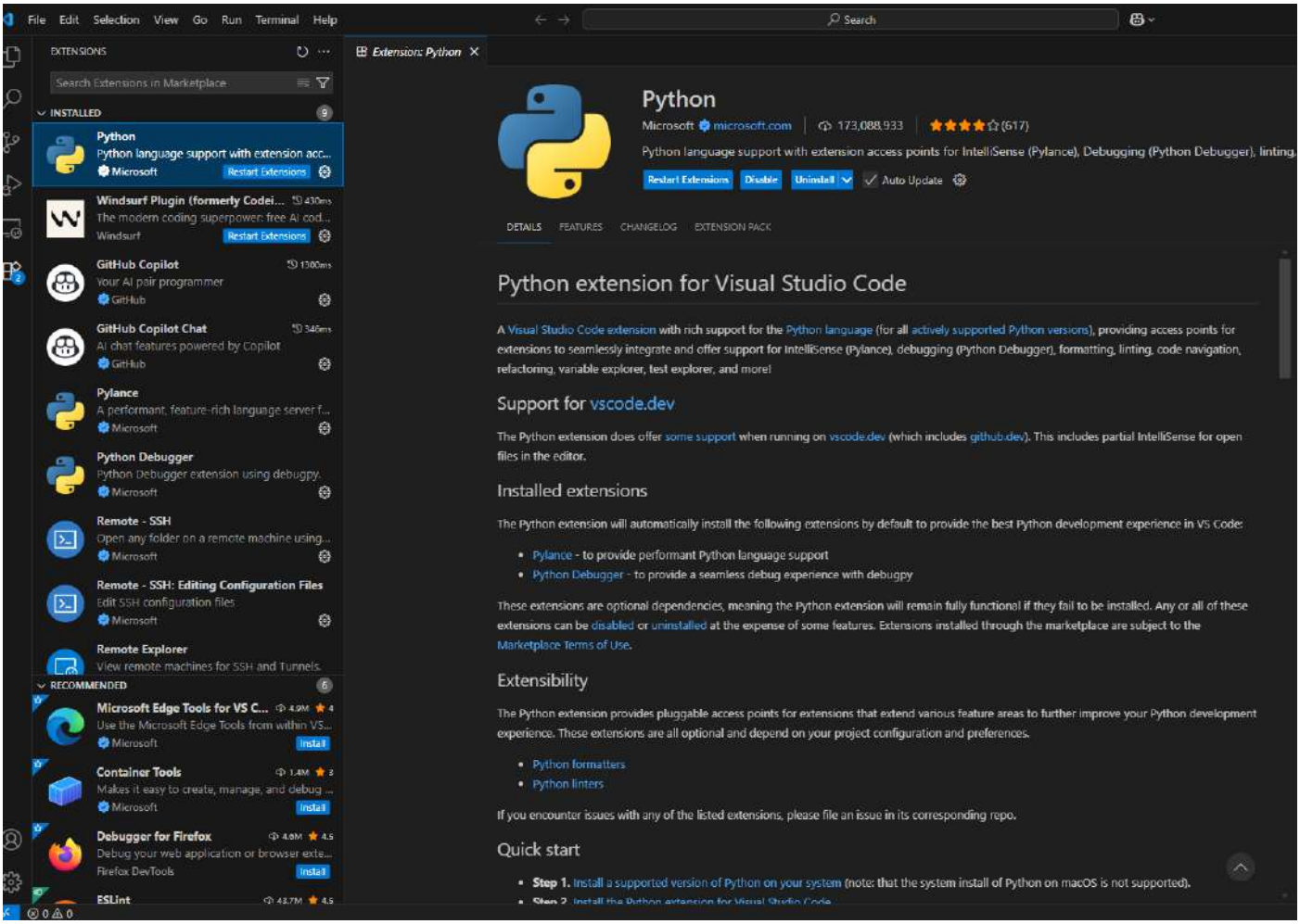
- ✓ “Add to PATH”
- ✓ “Open with Code” (clic derecho)
- ✓ “Register Code as editor for supported file types”

Esto facilitará la apertura y uso de VS Code desde cualquier carpeta.

5. Instalar la extensión de Python en VS Code

Abre Visual Studio Code. En la barra lateral izquierda, haz clic en el ícono de extensiones o presiona Ctrl + Shift + X.

Ilustración 11
Instalación de la extensión de Python en Visual Studio Code



En la barra de búsqueda, escribe Python y selecciona la extensión oficial desarrollada por Microsoft (es la más popular y tiene millones de descargas). Haz clic en “Instalar”. Esto agregará soporte para código Python, resaltado de sintaxis, autocompletado, depuración, y más.

6. Probar el entorno

Crea un nuevo archivo en VS Code y guárdalo con extensión .py, por ejemplo: hola.py.

Escribe el siguiente código de prueba:

```
print("¡Hola, mundo!")
```

Ilustración 12

Hola mundo en Python

Guarda el archivo y presiona Ctrl + F5 para ejecutarlo. Si todo está bien configurado, verás el mensaje en la consola de salida.

ACTIVIDAD 2

Objetivo:	Comprobar que Python y Visual Studio Code están correctamente instalados, y familiarizar al estudiante con el entorno de desarrollo mediante la creación de un primer programa funcional.
Descripción:	Esta actividad marca el primer contacto del estudiante con la programación real, a través de una tarea guiada y contextualizada, el estudiante desarrollará confianza en el entorno de trabajo y reconocerá el flujo básico de desarrollo de software: escribir código, guardar, ejecutar y corregir.

Instrucciones:	1. Abrir VS Code
	2. Inicia Visual Studio Code desde el escritorio o el menú de inicio.
	3. Crear una nueva carpeta para tus proyectos
	4. En tu escritorio o en tus documentos, crea una carpeta llamada ProyectosPython. Desde VS Code, ve a Archivo > Abrir carpeta y selecciona esa carpeta.
	5. Crear un archivo Python nuevo
	6. Haz clic en Archivo > Nuevo archivo y guárdalo como primer_programa.py.
	7. Escribir tu primer programa
	8. En el archivo, escribe el siguiente código:
	
	`nombre = input("¿Cuál es tu nombre? ")`
	`print("Hola, " + nombre + ". Bienvenido a Python.")`
	
	9. Ejecutar el programa(Presiona Ctrl + F5)o haz clic en el botón de "Ejecutar" (▶) en la esquina superior derecha.
	Si aparece una terminal en la parte inferior, sigue las instrucciones y escribe tu nombre.
	10. Comprobar resultados
	Asegúrate de que la salida del programa sea correcta, por ejemplo:
	
	¿Cuál es tu nombre? Juan
	Hola, Juan. Bienvenido a Python.
	
	11. Guardar y cerrar
	Guarda el archivo y ciérralo. Este programa servirá como base para tus próximos ejercicios.

Resultados esperados:	• Usar el entorno de Visual Studio Code para crear y ejecutar archivos Python. • Interactuar con el usuario mediante entrada (input()) y salida (print()). • Comprender cómo se estructura un programa básico y cómo se ejecuta paso a paso.

ACTIVIDAD 2.1

Modifica el programa para simular la lectura de un sensor de voltaje:

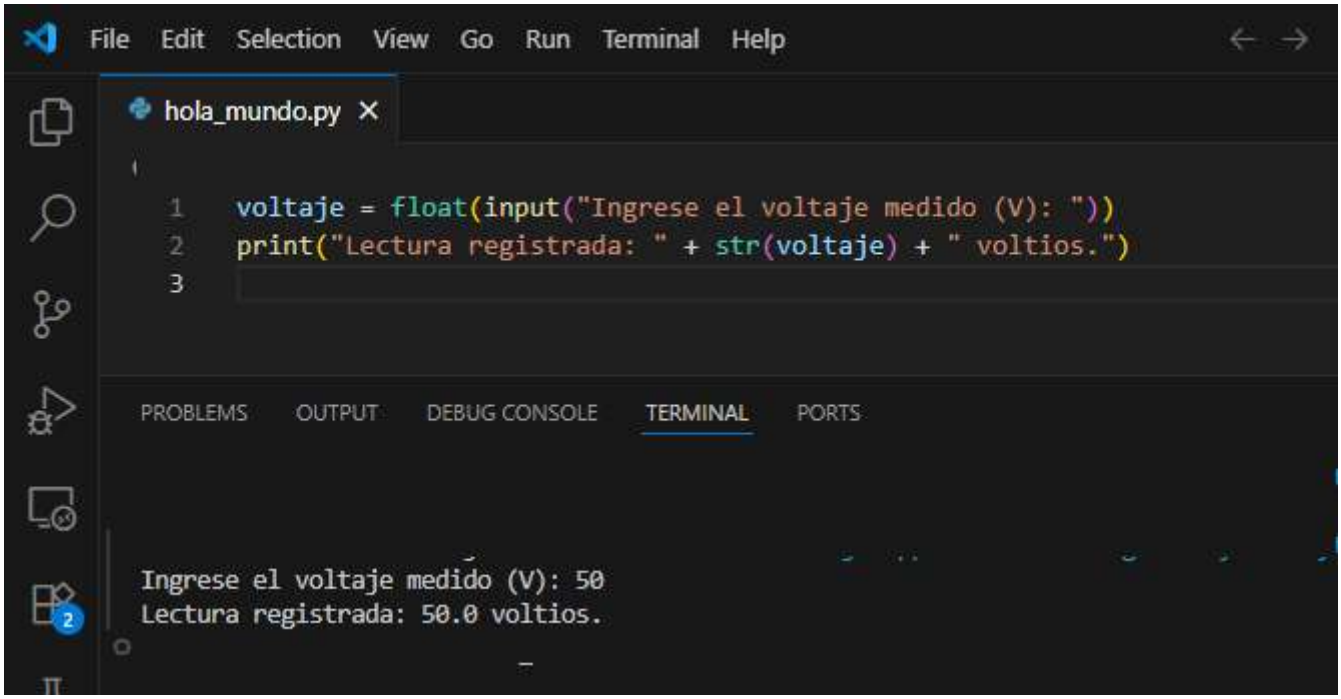
```
voltaje = float(input("Ingrese el voltaje medido (V): "))

print("Lectura registrada: " + str(voltaje) + " voltios.")
```

Esto servirá como base para ejercicios futuros donde se trabajará con datos de sensores reales.

Ilustración 13

Programa para simular la lectura de un sensor de voltaje



Variables y Tipos de Datos

En programación, una de las tareas fundamentales es almacenar y manipular información, para ello utilizamos las variables, que son espacios en la memoria de la computadora donde podemos guardar datos con un nombre que los identifique. Así como en electrónica etiquetamos los componentes de un circuito para reconocer su función, en programación etiquetamos los datos mediante nombres de variables para poder usarlos, modificarlos y analizarlos dentro de nuestros programas.

Ilustración 14
Variables

¿Qué es una variable?

Una variable es una etiqueta simbólica que representa un valor, este valor puede cambiar durante la ejecución del programa, por eso se llama “variable”. Cada vez que asignamos un nuevo valor a esa etiqueta, la información anterior se reemplaza. En Python, crear una variable es muy sencillo, no es necesario declarar su tipo previamente, como ocurre en otros lenguajes como C. Basta con escribir el nombre de la variable, el signo =, y el valor que se desea asignar, ejemplo:

```
voltaje = 5.0
sensor_activado = True
nombre_equipo = "Multímetro Digital"
```

En este ejemplo tenemos tres variables:

- voltaje almacena un número decimal.
- sensor_activado almacena un valor lógico (booleano).
- nombre_equipo almacena una cadena de texto.

Reglas para nombrar variables

Al igual que en electrónica debemos seguir normas para etiquetar componentes o conexiones, en programación también hay reglas para nombrar variables:

- Deben comenzar con una letra o guion bajo (_), nunca con un número.
- No deben contener espacios ni símbolos especiales (solo se permite el guion bajo _).
- No deben usar palabras reservadas del lenguaje (como print, input, if, etc.).
- Se recomienda usar nombres descriptivos y en minúsculas, por ejemplo: temperatura_interna, estado_rele.

Python distingue entre mayúsculas y minúsculas, por lo tanto, **voltaje** y **Voltaje** serían variables distintas.

Tipos de datos básicos en Python

Python reconoce automáticamente el tipo de dato que almacena cada variable, a continuación, se presentan los tipos más comunes que usaremos en esta guía:

- Enteros (int): Números sin decimales. Ejemplo: corriente = 12
- Decimales (float): Números con punto decimal. Ejemplo: resistencia = 47.8
- Cadenas de texto (str): Texto encerrado entre comillas. Ejemplo: estado = "Activo"
- Booleanos (bool): Verdadero o falso. Ejemplo: alarma_encendida = False

Python también permite combinar variables en expresiones, por ejemplo:

```
voltaje = 12.0
corriente = 2.0
potencia = voltaje * corriente
print("La potencia es:", potencia, "W")
```

Este código realiza un cálculo eléctrico sencillo con variables, demostrando cómo Python puede ser útil en análisis de circuitos.

Conversión de tipos

En algunos casos, necesitamos convertir un tipo de dato a otro. Por ejemplo, si usamos la función input(), Python guarda el dato como texto, y debemos convertirlo a número si queremos hacer operaciones:

```
entrada = input("Ingrese la tensión (V): ")
tension = float(entrada)
```

También podemos convertir un número a cadena para mostrarlo junto a texto:

```
mensaje = "Tensión medida: " + str(tension) + " V"
print(mensaje)
```

Las variables y los tipos de datos son la base para cualquier programa, en el contexto de la electrónica, nos permiten almacenar mediciones, estados de dispositivos, parámetros de configuración, y resultados de cálculos. Dominar su uso es esencial para construir aplicaciones que interactúen con el mundo físico a través de sensores, actuadores y microcontroladores.

Tabla 1

Tipos de datos en Python

Tipo de Dato	Nombre	Descripción	Ejemplo
int	Entero	Números sin decimales.	resistencias = 10
float	Decimal (flotante)	Números con parte decimal.	voltaje = 3.7
str	Cadena de texto	Secuencia de caracteres alfanuméricos.	nombre = "Sensor LM35"
bool	Booleano	Representa un valor lógico: verdadero o falso.	activo = True
list	Lista	Colección ordenada y modificable de elementos.	datos = [2.5, 3.1, 4.0]
dict	Diccionario	Colección de pares clave:valor.	{"voltaje": 5, "corr": 2}
NoneType	Nulo	Representa la ausencia de valor o un estado vacío.	valor = None

Elaborado por: Verónica Sánchez

Python detecta automáticamente el tipo de dato cuando asignamos un valor, es posible convertir datos entre tipos usando funciones como int(), float(), str().

ACTIVIDAD 3

Objetivo:	Practicar el uso de variables y tipos de datos en Python (enteros, decimales, cadenas y booleanos) mediante la simulación de la configuración de un sensor electrónico y el almacenamiento de su información básica. **¿Qué se busca lograr?** • Comprender cómo se usan variables de distintos tipos. • Utilizar funciones básicas como input(), float(), int(), str(). • Aplicar lógica para convertir texto en valores booleanos. • Relacionar la programación con el contexto real de la electrónica (sensores, parámetros técnicos, validación de datos).
Descripción:	Los dispositivos electrónicos modernos, como sensores y módulos de comunicación, requieren configurar ciertos parámetros para funcionar correctamente. Esta actividad simula el registro de esos parámetros a través de un programa interactivo en Python.
Instrucciones:	El estudiante debe desarrollar un programa en Python que solicite al usuario los siguientes datos relacionados con un sensor electrónico: • **Nombre del sensor** (str) • **Voltaje de operación en voltios** (float) • **Rango de lectura en unidades (por ejemplo, °C o ppm)** (int) El programa deberá mostrar un resumen de la configuración ingresada.
Resultados esperados:	Ingrese el nombre del sensor: Sensor de gas MQ-2 Voltaje de operación (V): 5.0 Rango de lectura (ppm): 3000 --- CONFIGURACIÓN DEL SENSOR --- Nombre: Sensor de gas MQ-2 Voltaje: 5.0 V Rango: 3000 ppm

⚙️ Operadores y Expresiones

En programación, así como en electrónica usamos componentes para construir circuitos funcionales, utilizamos operadores para construir expresiones que manipulan, comparan y evalúan datos. Las expresiones permiten realizar cálculos, tomar decisiones, y ejecutar acciones en función de resultados. En Python, existen diferentes tipos de operadores, cada uno con su propósito específico se describen a continuación:

➕ Operadores Aritméticos

Los operadores aritméticos permiten realizar operaciones matemáticas básicas sobre variables numéricas, muy útiles en electrónica para calcular potencia, resistencias equivalentes, caídas de voltaje, entre otros.

Tabla 2

Operadores Aritméticos

Operador	Operación	Ejemplo	Resultado
+	Suma	3 + 2	5
-	Resta	5 - 1	4
*	Multiplicación	2 * 3	6
/	División (float)	6 / 4	1.5
//	División entera	6 // 4	1
%	Módulo (residuo)	7 % 3	1
**	Potenciación	2 ** 3	8

Elaborado por: Verónica Sánchez

⚖️ Operadores Relacionales (de comparación)

Tabla 3

Operadores relacionales

Operador	Descripción	Ejemplo	Resultado
==	Igual que	5 == 5	True
!=	Distinto de	5 != 3	True
>	Mayor que	7 > 5	True
<	Menor que	3 < 2	False
>=	Mayor o igual que	5 >= 5	True

Elaborado por: Verónica Sánchez

🔁 Operadores Lógicos

Los operadores lógicos combinan expresiones relacionales. Se utilizan mucho en sistemas de control, donde deben cumplirse una o más condiciones para activar un dispositivo.

Tabla 4

Operadores Lógicos

Operador	Significado	Ejemplo	Resultado
and	Y lógico	True and False	False
or	O lógico	True or False	True
not	Negación	not True	False

Elaborado por: Verónica Sánchez

🧠 Evaluación de expresiones

Una expresión es cualquier combinación de variables, operadores y valores que puede ser evaluada por el programa para obtener un resultado. Python sigue un orden lógico para evaluar expresiones, similar a las reglas matemáticas:

1. Paréntesis ()
2. Potencias **
3. Multiplicación, división *, /, //, %
4. Suma y resta +, -
5. Comparaciones ==, !=, >, <, >=, <=
6. Lógicos not, and, or

Comprender los operadores y cómo se combinan en expresiones es fundamental para tomar decisiones automáticas dentro de un programa. En electrónica, esta lógica se traduce en cómo responde un sistema a los datos que recibe de sensores o entradas digitales. El correcto uso de operadores permite controlar, calcular y automatizar procesos con eficiencia y precisión.

ACTIVIDAD 4

Objetivo:	Aplicar operadores aritméticos, relacionales y lógicos para crear un programa que evalúe si un sistema de carga de batería está operando en condiciones seguras.
Descripción:	En sistemas electrónicos que involucran baterías (como en fuentes portátiles, UPS o sistemas solares), es crucial monitorear tensión, corriente, y temperatura. Un microcontrolador o sistema embebido puede realizar este análisis, en esta actividad, el estudiante programará una versión simplificada en Python.
Instrucciones:	Desarrolla un programa que solicite al usuario los siguientes datos: • **Tensión de entrada** (en voltios, float) • **Corriente de carga** (en amperios, float) • **Temperatura del sistema** (en grados Celsius, int) Luego, el programa debe evaluar si las condiciones son seguras para continuar cargando la batería. Las condiciones son: 1. La **tensión** debe estar entre 4.5 V y 5.5 V. 2. La **corriente** debe ser menor a 2.0 A. 3. La **temperatura** debe estar entre 15 °C y 45 °C. El programa mostrará un mensaje como: • ✅ "Condiciones normales: la carga puede continuar." • ⚠️ "Advertencia: fuera de rango, detener carga."

Resultados esperados:	✓ Comprender y aplicar operadores relacionales y lógicos en una situación técnica.
	✓ Evaluar múltiples condiciones usando expresiones compuestas.
	✓ Relacionar conceptos de programación con monitoreo y control electrónico.
	
	tension = float(input("Ingrese la tensión (V): "))
	corriente = float(input("Ingrese la corriente (A): "))
	temperatura = int(input("Ingrese la temperatura (°C): "))
	condicion_tension = 4.5 <= tension <= 5.5
	condicion_corriente = corriente < 2.0
	condicion_temperatura = 15 <= temperatura <= 45
	if condicion_tension and condicion_corriente and condicion_temperatura:
	print("✅ Condiciones normales: la carga puede continuar.")
	else:
	print("⚠️ Advertencia: fuera de rango, detener carga.")

⌨️ Entrada y Salida de Datos

Todo programa interactivo necesita comunicarse con el usuario, ya sea para recibir datos de entrada o para mostrar resultados. En Python, esta interacción se realiza principalmente a través de dos funciones: input() para recibir datos del usuario, y print() para mostrar mensajes o resultados por pantalla. Entender cómo funcionan estas dos herramientas es fundamental, ya que son la base de todo programa que requiere interacción, control, monitoreo o retroalimentación de valores, como ocurre frecuentemente en proyectos electrónicos.

🎛️ La función input(): Recibir datos del usuario

La función input() se utiliza para leer información que el usuario escribe desde el teclado. Python siempre recibe esta entrada como una cadena de texto (str), incluso si parece un número. Por ello, si necesitamos usar ese valor en una operación matemática, debemos convertirlo usando int() o float().

Ejemplo

```
voltaje = float(input("Ingrese el voltaje medido (V): "))

print("Tensión ingresada:", voltaje)
```

En este caso, usamos float() para transformar la cadena ingresada en un número decimal que pueda utilizarse en cálculos posteriores.

🖨️ La función print(): Mostrar información

La función print() sirve para mostrar texto, variables o resultados por pantalla. Puede recibir varios elementos separados por comas, y Python los mostrará en orden.

Ejemplo:

```
sensor = "DHT11"

print("Sensor seleccionado:", sensor)
```

Se puede combinar texto fijo con variables para presentar información útil al usuario.

🎯 Formato de salida con f-strings

Una forma moderna y muy clara de mostrar texto junto con valores es mediante f-strings (cadenas formateadas), disponibles en Python 3.6 en adelante. Se colocan una letra f antes de la cadena, y las variables se escriben directamente dentro de las llaves {}.

Ejemplo:

```
sensor = "LM35"

temperatura = 36.5

print (f"Sensor: {sensor} | Temperatura: {temperatura} °C")
```

Esto es equivalente a concatenar cadenas, pero mucho más legible y fácil de mantener, especialmente cuando se deben mostrar múltiples variables técnicas.

🧠 **Aplicación técnica**

En un sistema de monitoreo, es frecuente que un microcontrolador reciba entradas (de sensores, botones o comandos) y devuelva información por pantalla o red. Simular ese proceso en Python, con input() y print(), ayuda a los estudiantes a comprender cómo funcionan las interfaces de control en electrónica y cómo estructurar correctamente la comunicación humano–máquina.

Dominar el uso de input() y print() es esencial para crear programas que reciban datos y generen resultados útiles. En proyectos de electrónica, estas funciones permiten simular la interacción con sensores, módulos o usuarios, lo cual es clave para el desarrollo de soluciones automatizadas, el uso de f-strings mejora la presentación y claridad de la información mostrada, haciendo que los programas sean más profesionales y fáciles de entender.

ACTIVIDAD 5

Objetivo:	Aplicar las funciones de entrada y salida (input(), print()) y utilizar **f-strings** para presentar datos técnicos de manera clara y estructurada, como si se tratara de un sistema real de monitoreo electrónico.
Descripción:	En sistemas electrónicos es común ingresar datos manuales de medición o recibirlos desde sensores para su procesamiento. Esta actividad simula ese proceso a través de un programa en Python que recibe parámetros eléctricos y genera un **informe por consola**, útil para interpretar los resultados en tiempo real.
Instrucciones:	Desarrolla un programa que solicite al usuario los siguientes datos: • Nombre del **dispositivo electrónico** (por ejemplo, "Fuente regulada", "Sensor de temperatura"). • **Voltaje** de operación (en voltios, float) • **Corriente** de operación (en amperios, float) Luego, el programa debe mostrar un **reporte técnico** con formato claro, usando **f-strings**, mostrando los datos como se haría en una interfaz de diagnóstico.
Resultados esperados:	✓ Utilizar correctamente input() para recibir diferentes tipos de datos. ✓ Mostrar información estructurada con print() y f-strings. ✓ Simular un escenario técnico real donde se reportan parámetros eléctricos. ✓ Comprender la utilidad de la presentación clara de datos en sistemas electrónicos.
Código de ejemplo:	nombre = input("Ingrese el nombre del dispositivo: ") voltaje = float(input("Ingrese el voltaje de operación (V): ")) corriente = float(input("Ingrese la corriente de operación (A): ")) estado_texto = input("¿El dispositivo está activo? (sí/no): ") estado = "ACTIVO" if estado_texto.lower() == "sí" else "INACTIVO" print("\n--- REPORTE DEL DISPOSITIVO ---") print(f"Dispositivo: {nombre}") print(f"Voltaje: {voltaje} V") print(f"Corriente: {corriente} A") print(f"Estado: {estado}")

EJERCICIOS PROPUESTOS

🔤 **Variables y Tipos de Datos**

1. **Crea un programa que almacene tu nombre, edad y carrera y los imprima con print().**

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

2. Convierte una entrada de texto a número entero y realiza una operación con él.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

3. Solicita el nombre de un componente electrónico y su código (“LM317”) y muéstralo.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

4. Pide el número de resistencias en un circuito y calcula la resistencia total (asume serie).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

5. Declara una variable tipo booleano llamada sensor_activo y muestra su valor.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

6. Lee dos valores de corriente y almacénalos en variables; luego calcula el promedio.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

7. Define una lista con tres nombres de sensores y muestra el segundo.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

8. Declara un diccionario con "voltaje", "corriente" , "potencia" y muestra su contenido.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

9. Muestra un mensaje de bienvenida utilizando solo variables tipo cadena.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

➕ Operadores Aritméticos

1. Solicita dos tensiones y calcula la suma y diferencia entre ellas.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

2. Calcula la potencia entregada por una fuente a partir de voltaje y corriente.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

3. Calcula el valor de una resistencia equivalente en paralelo de dos resistencias dadas.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

4. Usa % para determinar si un número ingresado es par o impar.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

5. Solicita la frecuencia de una señal y calcula su periodo.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

6. Calcula el voltaje de salida de un divisor de tensión con dos resistencias.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

7. Solicita el tiempo y velocidad de un motor y calcula la distancia recorrida.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

8. Usa ** para elevar una corriente al cuadrado.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

9. Calcula la capacitancia total de tres capacitores en serie.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

10. Calcula el valor promedio de tres mediciones de temperatura.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

🔁 MÓDULO 3: CONTROL DE FLUJO

Condicionales

En programación, las estructuras condicionales permiten que un programa tome decisiones, en lugar de ejecutar todas las instrucciones en orden, una condición evalúa si algo es verdadero o falso, y en función de eso, se ejecutan ciertos bloques de código. Esta lógica es esencial para automatizar procesos, monitorear estados o responder a eventos en sistemas electrónicos, como activar una alarma si la temperatura supera cierto límite, o encender un ventilador si un sensor detecta sobrecalentamiento.

🔍 if, elif, else

En Python, las instrucciones condicionales se escriben utilizando las palabras clave if (si), elif (sino si) y else (sino).

La estructura básica es la siguiente:

```
if condición:
    # Bloque si la condición es verdadera
elif otra_condición:
    # Bloque si la segunda condición es verdadera
else:
    # Bloque si ninguna de las anteriores se cumple
```

Ilustración 15

Estructura IF

Ejemplo:

```
temperatura = float(input("Ingrese la temperatura actual (°C): "))

if temperatura > 70:

    print("⚠️ Alerta: Sobrecalentamiento")

elif temperatura >= 50:

    print("🔧 Activar ventilador")

else:

    print("✅ Temperatura estable")
```

Actividad: Evaluador de edad y permisos

En este ejemplo, se evalúa si una temperatura supera ciertos umbrales. Solo se ejecuta uno de los tres bloques, según cuál condición sea verdadera primero.

🧱 Comparaciones anidadas

Una condición anidada es aquella que se encuentra dentro de otra. Se utiliza cuando se necesita evaluar más de una condición dentro de un mismo escenario.

```
voltaje = float(input("Ingrese voltaje de entrada (V): "))

if voltaje >= 0:

    if voltaje <= 5:

        print("✅ Voltaje en rango aceptable")

    else:

        print("⚠️ Voltaje demasiado alto")

else:

    print("⚠️ Voltaje negativo no permitido")
```

Este tipo de estructura es útil cuando se desea analizar rangos con más detalle o cuando una condición principal determina si deben evaluarse otras más específicas.

Las estructuras condicionales son comunes en controladores de sistemas electrónicos: encender relés, mostrar mensajes de error, cortar el suministro de energía, enviar señales, entre otros. También permiten interpretar datos de sensores de forma lógica para que un sistema pueda tomar decisiones automáticas.

Las instrucciones condicionales permiten que un programa responda al entorno, algo indispensable en sistemas electrónicos. Gracias a if, elif y else, los programas en Python pueden comportarse de forma inteligente, activando o desactivando funciones según el estado de las variables. Las condiciones anidadas permiten mayor precisión en las decisiones, algo esencial en sistemas críticos donde los valores deben estar dentro de rangos estrictos.

ACTIVIDAD 6

Objetivo:	Aplicar estructuras condicionales (if, elif, else) para simular un sistema de control de ventilación en función de la temperatura medida por un sensor.
Descripción:	En sistemas electrónicos como fuentes de alimentación, inversores o gabinetes de servidores, es común usar sensores de temperatura para controlar automáticamente ventiladores. En esta actividad, se simulará la lógica de ese sistema en Python, usando condiciones para decidir qué acción tomar.
Instrucciones:	Desarrolla un programa en Python que solicite al usuario ingresar la temperatura actual del sistema en grados Celsius. El programa debe tomar decisiones basadas en el siguiente comportamiento: • Si la temperatura es menor a 30 °C, mostrar el mensaje: "Temperatura normal. Ventilador apagado." • Si la temperatura está entre 30 y 50 °C inclusive, mostrar: "Temperatura moderada. Ventilador a velocidad media." • Si la temperatura es mayor a 50 °C, mostrar: "Temperatura alta. Ventilador a máxima velocidad."
Resultados esperados:	✓ Comprender y aplicar estructuras condicionales básicas (if, elif, else). ✓ Relacionar lógica de programación con sistemas de control en electrónica. ✓ Identificar la importancia del orden en las condiciones.

Código de ejemplo:	temperatura = float(input("Ingrese la temperatura actual (°C): "))
	if temperatura < 30:
	  print("Temperatura normal. Ventilador apagado.")
	elif temperatura <= 50:
	  print("Temperatura moderada. Ventilador a velocidad media.")
	else:
	  print("Temperatura alta. Ventilador a máxima velocidad.")

Extensión opcional:	Agrega un mensaje adicional si la temperatura ingresada es negativa o no realista (por ejemplo, menor a -10 °C), usando una condición adicional:

EJERCICIOS PROPUESTOS

1. Solicita una temperatura y muestra si está por debajo o por encima de 25 °C.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

2. Ingresa una tensión y muestra si está “dentro del rango”, “por debajo del mínimo” o “por encima del máximo”.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

3. Verifica si la corriente ingresada es exactamente 0 A (circuito abierto) o distinta.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

4. Pide una señal lógica (1 o 0) y muestra si el dispositivo está encendido o apagado.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

5. Solicita el estado de un pulsador (presionado/no presionado) y muestra la acción correspondiente.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

6. Determina si una resistencia medida cumple con el valor nominal esperado.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

7. Compara dos lecturas de voltaje y muestra cuál es mayor o si son iguales.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

8. Ingresa un valor booleano (como texto) y evalúa si el sistema debe activarse.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

9. Evalúa si una frecuencia está en el rango aceptable de 45 a 65 Hz.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

10. Detecta si la entrada de un pin digital es alta, baja o flotante (valor inválido)

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

11. Simula el control de un sistema trifásico: si una de las fases está fuera de rango, mostrar "Fase defectuosa".

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

12. Evalúa si un sistema puede arrancar: se requiere que la tensión esté en 220–240 V y que la temperatura esté por debajo de 40 °C.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

13. Ingresa dos mediciones y evalúa si ambas están dentro de tolerancia; si solo una lo está, mostrar advertencia parcial.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

14. Simula un sistema de control de carga que solo permite continuar si se cumplen tres condiciones: tensión correcta, corriente permitida y temperatura estable.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

🔁 Modulo 4: Ciclos (Estructuras Repetitivas)

En programación, un ciclo es una estructura que permite repetir una o más instrucciones varias veces de forma automática. Así como en electrónica se repiten lecturas de sensores, ciclos de carga, o señales PWM, en programación los ciclos son fundamentales para automatizar tareas que se repiten. Gracias a ellos, podemos recorrer listas de datos, realizar mediciones múltiples, o repetir acciones hasta que se cumpla una condición. En Python, los dos tipos principales de ciclos son while y for.

1️⃣ Ciclo while: ejecución basada en una condición

El ciclo while repite un bloque de código mientras una condición sea verdadera. Es útil cuando no sabemos cuántas veces se repetirá una acción, pero tenemos una condición lógica de salida.

Ejemplo

```
voltaje = 0
while voltaje < 5.0:
    voltaje += 0.5
    print(f"Voltaje aumentado a: {voltaje} V")
```

Este ejemplo simula un incremento de voltaje hasta llegar al umbral de 5.0 V, como podría hacerse en una rampa de alimentación.

2️⃣ Ciclo for: ejecución sobre una secuencia

El ciclo for se usa para recorrer elementos de una secuencia (como listas, cadenas o rangos). Es útil cuando sí sabemos cuántas veces se va a repetir la acción o cuando trabajamos con colecciones de datos.

```
sensores = ["DHT11", "LM35", "MQ-135"]
for sensor in sensores:
    print(f"Activando sensor: {sensor}")
```

Este código simula la activación de diferentes sensores en un sistema de monitoreo ambiental.

🔧 Control dentro de los ciclos

Python permite modificar el flujo del ciclo con las siguientes instrucciones:

- break: interrumpe el ciclo inmediatamente.
- continue: salta al siguiente ciclo sin terminar el actual.
- else: se ejecuta si el ciclo finaliza sin un break.

Ejemplo con break:

```
for i in range(10):
    if i == 5:
        break
    print("Contador:", i)
```

Ejemplo con continue:

```
for i in range(5):
    if i == 2:
        continue
    print("Valor:", i)
```

Ejemplo: medición repetida de temperatura

```
for i in range(3):
    temperatura = float(input(f"Ingrese temperatura {i + 1}: "))
    print(f"Medición {i + 1}: {temperatura} °C")
```

Los ciclos permiten que un programa sea dinámico, repetitivo y autónomo, como los sistemas electrónicos reales. Gracias a estructuras como for y while, podemos recorrer datos, monitorear señales o automatizar procesos repetitivos sin escribir las mismas instrucciones muchas veces. Dominar el uso de ciclos es esencial para la creación de programas eficientes y adaptables en el campo de la electrónica.

ACTIVIDAD 7

Objetivo:	Aplicar ciclos for y while en la simulación de un sistema de monitoreo que lee múltiples valores de temperatura, calcula el promedio y emite alertas si se detecta sobrecalentamiento.
Descripción:	En un sistema de control térmico (como una fuente regulada o un sistema de ventilación), se requiere realizar lecturas repetidas de temperatura para identificar posibles riesgos. Esta actividad simula dicho comportamiento mediante un programa en Python.
Instrucciones:	• Solicita al usuario cuántas temperaturas desea ingresar (por ejemplo, 5). • Utiliza un ciclo for para pedir ese número de temperaturas y guardarlas. • Durante cada lectura, muestra la temperatura ingresada y si es normal (< 45 °C) o alta (≥ 45 °C). Al final, muestra: • El número total de temperaturas ingresadas. • Cuántas fueron consideradas “altas”. • El promedio de todas las temperaturas.
Resultados esperados:	✓ Usar ciclos for para repetir una tarea una cantidad definida de veces. ✓ Aplicar condicionales dentro de ciclos. ✓ Relacionar lectura de datos y lógica de evaluación con sistemas de monitoreo en electrónica. ✓ Calcular promedios y contar ocurrencias durante un ciclo.

Código de ejemplo:	n = int(input("¿Cuántas lecturas de temperatura realizará? "))
	altas = 0
	suma = 0
	for i in range(n):
	  temp = float(input(f"Ingrese temperatura {i+1}: "))
	  suma += temp
	  if temp >= 45:
	  print("Temperatura alta")
	  altas += 1
	  else:
	  print("Temperatura normal")
	promedio = suma / n
	print("\nResumen:")
	print(f"Total de lecturas: {n}")
	print(f"Temperaturas altas: {altas}")
	print(f"Temperatura promedio: {promedio:.2f} °C")

EJERCICIOS PROPUESTOS

1. Mostrar 5 veces el mensaje "Bienvenido al sistema de monitoreo".

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

2. Usar un ciclo while para contar de 1 a 10.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

3. Usar for con range() para mostrar los primeros 5 números pares.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

4. Mostrar los números del 10 al 1 en orden descendente usando while.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

5. Imprimir los valores de una lista de sensores: ["LM35", "DHT11", "MQ-2"].

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

6. Solicitar 3 temperaturas y mostrarlas una por una usando for.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

7. Leer 4 valores de voltaje y mostrar su suma total.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

8. Calcular el promedio de 5 mediciones de corriente introducidas por el usuario.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

9. Usar un ciclo para imprimir los múltiplos de 5 entre 0 y 50.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

10. Contar de 1 a N, donde N es ingresado por el usuario.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

11. Simular una subida de voltaje desde 0 V hasta 5 V en pasos de 0.5 V usando while.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

12. Leer datos de temperatura hasta que el usuario escriba un número negativo.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

13. Contar cuántas veces se ingresó una temperatura mayor a 35 °C (N lecturas).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

14. Mostrar los caracteres de la palabra "Osciloscopio" uno por uno.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

15. Simular la activación secuencial de 4 relés numerados del 1 al 4.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

16. Sumar los valores de resistencia ingresados hasta que el total supere 100 ohm.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

17. Mostrar solo las temperaturas mayores a 30 °C de una lista ingresada por el usuario.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

18. Leer 5 mediciones de corriente y contar cuántas superan 2A.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

19. Simular el parpadeo de un LED alternando mensajes de encendido/apagado 6 veces.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

20. Generar una tabla de conversión de grados Celsius a Fahrenheit de 0 a 100 cada 10 grados.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

21. Leer voltajes hasta que uno supere 12 V (usar break).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

22. Saltar la impresión de valores inferiores a 1 A usando continue.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

23. Simular 5 ciclos de prueba donde se mide voltaje y corriente y se muestra la potencia calculada.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

24. Mostrar solo los valores válidos de un lote (descartar negativos con continue).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

25. Contar cuántos sensores están activos (True) en una lista de estados.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

26. Verificar si hay algún valor crítico en una serie de temperaturas (> 70 °C) y detener el ciclo.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

27. Mostrar una cuenta regresiva de 10 a 0 y luego imprimir "Sistema iniciado".

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

28. Simular la lectura continua de un sensor que se detiene cuando la lectura es igual a cero.

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

29. Crear un sistema que lea valores hasta que se ingresen tres consecutivos en el rango seguro (20–30 °C).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

30. Evaluar una lista de valores de tensión e identificar cuántos están en zona de riesgo (fuera de 4.5–5.5 V).

Análisis del problema, ¿Qué datos necesito?	
¿Qué tipo de datos se usarán?	☐ **int** ☐ **float** ☐ **str** ☐ **bool** ☐ **list** ☐ **dict**
¿Qué operación o condición se debe realizar?	
Código Python propuesto	# Escribir aquí el código completo
Resultado esperado por pantalla:	
Verificación y observaciones	**¿Funcionó correctamente?** ☐ **Sí** ☐ **No (¿Por qué?)** _____ _____ **¿Qué aprendiste con este ejercicio?** _____ _____

ANEXOS

GLOSARIO BÁSICO DE PROGRAMACIÓN

TÉRMINO	DEFINICIÓN
ALGORITMO	Conjunto ordenado y finito de instrucciones que permiten resolver un problema o realizar una tarea.
CÓDIGO FUENTE	Conjunto de instrucciones escritas en un lenguaje de programación (como Python) que una computadora puede interpretar.
COMPILAR / INTERPRETAR	Proceso mediante el cual el código fuente es traducido y ejecutado por la computadora. Python es un lenguaje interpretado.
SINTAXIS	Conjunto de reglas que definen la estructura correcta de las instrucciones en un lenguaje de programación.
VARIABLE	Espacio en la memoria para almacenar un valor que puede cambiar durante la ejecución del programa. Ejemplo: nombre = "Ana"
CONSTANTE	Valor que no cambia durante la ejecución del programa. Python no tiene constantes como otros lenguajes, pero se usa por convención: PI = 3.1416
TIPO DE DATO	Clasificación de los datos según su naturaleza. En Python: int (entero), float (decimal), str (cadena), bool (booleano).
OPERADOR	Símbolo que indica una operación a realizar. Ejemplos: +, -, ==, and, not.
ENTRADA (INPUT)	Información que el usuario proporciona al programa. En Python se utiliza la función input().
SALIDA (OUTPUT)	Información que el programa muestra al usuario. En Python se usa la función print().
CONDICIONAL	Estructura que permite ejecutar código dependiendo de una condición. Se usa if, elif, else.
BUCLE / CICLO	Instrucción que repite un bloque de código. En Python: while y for.
FUNCIÓN	Bloque de código reutilizable que realiza una tarea específica. Se define con def. Ejemplo: def saludar(): print("Hola")
PARÁMETRO	Variable que se usa dentro de una función para recibir valores externos. Ejemplo: def sumar(a, b): return a + b
RETORNO	Valor que devuelve una función al ser ejecutada. Se usa la palabra return.

LISTA	Colección ordenada y modificable de elementos. Ejemplo: frutas = ["manzana", "banana"]
DICCIONARIO	Colección no ordenada de pares clave-valor. Ejemplo: persona = {"nombre": "Luis", "edad": 20}
ERROR / EXCEPCIÓN	Problema que ocurre durante la ejecución del programa. Algunos errores pueden prevenirse con try y except.
DEPURACIÓN	Proceso de encontrar y corregir errores en el código.
INDENTACIÓN	Espacios al inicio de una línea de código que indican estructura. En Python es obligatoria para definir bloques.

TABLAS DE OPERADORES

🔢 1. Operadores Aritméticos

Operador	Descripción	Ejemplo	Resultado
+	Suma	5 + 2	7
-	Resta	5 - 2	3
*	Multiplicación	5 * 2	10
/	División (float)	5 / 2	2.5
//	División entera	5 // 2	2
%	Módulo (residuo)	5 % 2	1
**	Potenciación	5 ** 2	25

🧮 2. Operadores de Comparación (Relacionales)

Operador	Descripción	Ejemplo	Resultado
==	Igual a	5 == 5	True
!=	Diferente de	5 != 3	True
>	Mayor que	5 > 3	True
<	Menor que	5 < 3	False
>=	Mayor o igual que	5 >= 5	True
<=	Menor o igual que	5 <= 4	False

🔁 3. Operadores Lógicos

Operador	Descripción	Ejemplo	Resultado
and	Y lógico	True and False	False
or	O lógico	True or False	True
not	Negación lógica	not True	False

📝 4. Operadores de Asignación

Operador	Descripción	Ejemplo	Equivalente a
=	Asignación	x = 5	x = 5
+=	Suma y asigna	x += 2	x = x + 2
-=	Resta y asigna	x -= 2	x = x - 2
***=**	Multiplica y asigna	x *= 3	x = x * 3
/=	Divide y asigna	x /= 2	x = x / 2
//=	División entera y asigna	x //= 2	x = x // 2
%=	Módulo y asigna	x %= 3	x = x % 3
****=**	Potencia y asigna	x **= 2	x = x ** 2

Enlaces de práctica y recursos online

🧪 Práctica Interactiva

Recurso	Descripción	Enlace
Python Tutor	Visualizador de código paso a paso para comprender cómo se ejecuta el programa. Ideal para principiantes.	https://pythontutor.com
Replit	Entorno en línea para programar en Python sin necesidad de instalar nada. Permite guardar y compartir tus proyectos.	https://replit.com
Programiz	Tutoriales y ejercicios prácticos de Python desde lo más básico hasta temas intermedios.	https://www.programiz.com/python-programming
W3Schools Python	Plataforma con ejemplos, teoría y ejercicios prácticos en línea.	https://www.w3schools.com/python/
Exercism.io	Plataforma gratuita con ejercicios prácticos y mentoría opcional.	https://exercism.io/tracks/python

📚 Documentación Oficial y Manuales

Recurso	Descripción	Enlace
Python.org Docs	Documentación oficial del lenguaje Python. Ideal para consulta técnica.	**https://docs.python.org/3/**
Cheat Sheet de Python	Resumen rápido de sintaxis y estructuras comunes de Python.	**https://www.pythoncheatsheet.org**

👥 Comunidades y Ayuda

Recurso	Descripción	Enlace
Stack Overflow	Foro global para programadores. Puedes buscar preguntas similares o hacer las tuyas.	**https://stackoverflow.com/questions/tagged/python**
r/learnpython (Reddit)	Comunidad en inglés dedicada al aprendizaje de Python.	**https://www.reddit.com/r/learnpython/**
Comunidad Python Ecuador	Grupo local de usuarios Python. Ideal para conectarse con desarrolladores del país.	**https://python.ec**

🎮 Aprender Jugando

Recurso	Descripción	Enlace
CodeCombat	Aprende Python jugando en un entorno de aventuras. Ideal para jóvenes y principiantes.	**https://codecombat.com**
CheckiO	Resuelve desafíos de lógica en forma de juegos.	**https://checkio.org**

ISBN: 978-9942-7128-2-0