Contenido

[**Módulo 1: Fundamentos de Python** 4](#_Toc195315783)

[**1.1 ¿Qué es Python? Instalación y configuración del entorno.** 4](#_Toc195315784)

[**1.2 Primeros pasos: sintaxis, comentarios y ejecución de código.** 4](#_Toc195315785)

[**1.3 Variables: Declaración, tipos de datos básicos (números, cadenas, booleanos)** 5](#_Toc195315786)

[**1.4 Operaciones básicas: Matemáticas** 6](#_Toc195315787)

[**1.5 Concatenación** 6](#_Toc195315788)

[**1.6 Operadores lógicos y de comparacion** 7](#_Toc195315789)

[**1.7 Ejemplo práctico:** 7](#_Toc195315790)

[**1.8 Ejercicio:** 7](#_Toc195315791)

[**Módulo 2: Control de Flujo** 7](#_Toc195315792)

[2.1 Estructuras condicionales: if, elif, else. 8](#_Toc195315793)

[2.2 Bucles: for y while. 8](#_Toc195315794)

[2.3 Operadores de comparación y lógicos. 8](#_Toc195315795)

[2.4 Ejemplo práctico: 8](#_Toc195315796)

[2.5 Ejercicio: 8](#_Toc195315797)

[**Módulo 3: Funciones y Modularidad** 8](#_Toc195315798)

[3.1 ¿Qué es una función? Declaración y uso (def). 8](#_Toc195315799)

[3.2 Parámetros y valores de retorno. 8](#_Toc195315800)

[3.3 Modularidad: Dividir proyectos en archivos manejables. 8](#_Toc195315801)

[3.4 Pruebas unitarias: Introducción a testing con unittest o pytest. 8](#_Toc195315802)

[3.5 Ejemplo práctico: 8](#_Toc195315803)

[3.6 Ejercicio: 8](#_Toc195315804)

[**Módulo 4: Colecciones y Manipulación de Datos** 8](#_Toc195315805)

[4.1 Listas: Creación, manipulación y métodos (append(), pop(), etc.). 8](#_Toc195315806)

[4.2 Tuplas: Uso y propiedades. 8](#_Toc195315807)

[4.3 Diccionarios: Claves, valores y métodos avanzados (keys(), values(), items()). 8](#_Toc195315808)

[4.4 Slicing: Manipulación de listas, cadenas y tuplas. 9](#_Toc195315809)

[4.5 JSON: Introducción y manipulación de datos estructurados. 9](#_Toc195315810)

[4.6 Ejemplo práctico: 9](#_Toc195315811)

[4.7 Ejercicio: 9](#_Toc195315812)

[**Módulo 5: Programación Orientada a Objetos (POO)** 9](#_Toc195315813)

[5.1 Clases y objetos: Definición y uso. 9](#_Toc195315814)

[5.2 Métodos y atributos. 9](#_Toc195315815)

[5.3 Herencia y polimorfismo. 9](#_Toc195315816)

[5.4 Ejemplo práctico: 9](#_Toc195315817)

[5.5 Ejercicio: 9](#_Toc195315818)

[**Módulo 6: Manejo de Errores y Archivos** 9](#_Toc195315819)

[6.1 Excepciones: Captura y manejo con try, except, finally. 9](#_Toc195315820)

[6.2 Archivos: Lectura y escritura (open(), read(), write()). 10](#_Toc195315821)

[6.3 Entornos virtuales con venv: Configuración y gestión de dependencias. 10](#_Toc195315822)

[6.4 Ejemplo práctico: 10](#_Toc195315823)

[6.5 Ejercicio: 10](#_Toc195315824)

[**Módulo 7: Bibliotecas para Análisis de Datos** 10](#_Toc195315825)

[7.1 Pandas: Creación y manipulación de DataFrames. 10](#_Toc195315826)

[7.2 NumPy: Arrays, operaciones matemáticas y manipulación matricial. 10](#_Toc195315827)

[7.3 Ejemplo práctico: 10](#_Toc195315828)

[7.4 Ejercicio: 10](#_Toc195315829)

[**Módulo 8: Visualización de Datos con Matplotlib** 10](#_Toc195315830)

[8.1 Introducción a Matplotlib: ¿Qué es y cómo instalarlo? 10](#_Toc195315831)

[8.2 Uso de pyplot: Creación de gráficos básicos. 10](#_Toc195315832)

[8.3 Personalización de gráficos: Colores, etiquetas, títulos y estilos. 10](#_Toc195315833)

[8.4 Tipos de gráficos: 11](#_Toc195315834)

[8.5 Ejemplo práctico: 11](#_Toc195315835)

[8.6 Ejercicio: 11](#_Toc195315836)

[**Módulo 9: Frameworks de Desarrollo Web** 11](#_Toc195315837)

[9.1 Flask: Crear aplicaciones web básicas y APIs REST. 11](#_Toc195315838)

[9.2 Django: Crear proyectos robustos con modelos y vistas. 11](#_Toc195315839)

[9.3 Ejemplo práctico: 11](#_Toc195315840)

[9.4 Ejercicio: 11](#_Toc195315841)

[**Módulo 10: Git y Entorno de Trabajo** 11](#_Toc195315842)

[10.1 Git: Iniciar repositorios, comandos básicos (clone, commit, push, pull). 11](#_Toc195315843)

[10.2 Uso de ramas y resolución de conflictos (branch, merge). 11](#_Toc195315844)

[10.3 Configurar requirements.txt en proyectos. 12](#_Toc195315845)

[10.4 Ejemplo práctico: 12](#_Toc195315846)

[10.5 Ejercicio: 12](#_Toc195315847)

[**Módulo 11: Proyecto Final** 12](#_Toc195315848)

[11.1 Ideas de proyectos: 12](#_Toc195315849)

[**Módulo 12: Automatización con Python** 12](#_Toc195315850)

[12.1 Introducción a la automatización: ¿Qué es y por qué es útil? 12](#_Toc195315851)

[12.2 Uso de pyautogui: 12](#_Toc195315852)

[12.3 Web Scraping con BeautifulSoup y requests: 12](#_Toc195315853)

[12.4 Selenium para automatización web: 12](#_Toc195315854)

[12.5 Ejemplo práctico: 13](#_Toc195315855)

[12.6 Ejercicio: 13](#_Toc195315856)

# **Módulo 1: Fundamentos de Python**

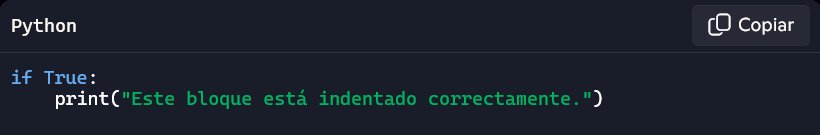
En este módulo se tiene como objetivo aprender lo esencial para empezar a programar en Python.

**1.1 ¿Qué es Python? Instalación y configuración del entorno.**

**¿Qué es Python?** Python es un lenguaje de programación versátil y fácil de aprender. Se utiliza en desarrollo web, análisis de datos, inteligencia artificial, automatización y más. Su sintaxis simple lo hace ideal tanto para principiantes como para expertos.

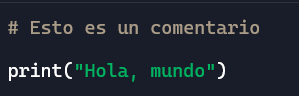
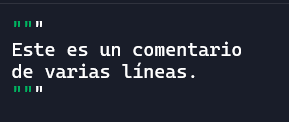
**1.2 Primeros pasos: sintaxis, comentarios y ejecución de código.**

**Sintaxis básica en Python**

* Python utiliza una **sintaxis simple y legible**.
* No necesitas usar ; al final de cada línea.
* ****Los bloques de código se identifican con **indentación** (4 espacios o tabulador).

**Comentarios**

Son útiles para explicar tu código y no se ejecutan, existen dos tipos de comentarios los que se pueden hacer en una sola línea y los que se pueden hacer en multiples líneas.

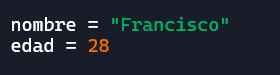
* **Comentarios de una línea:** Usa # al inicio.
* **Comentarios de varias líneas:** Usa comillas triples (""" o ''').

**Ejecucion de codigo**

* Guarda tu archivo, por ejemplo, sintaxis.py.
* Abre la terminal, navega hasta la carpeta donde está el archivo:
  + cd "C:\Users\franc\Documents\Mis proyectos de Python"
* Ejecuta el archivo con: python sintaxis.py

**1.3 Variables: Declaración, tipos de datos básicos (números, cadenas, booleanos)**

**¿Qué es una variable?**

Una **variable** es un nombre que asignamos a un valor en Python. Es como una caja donde puedes guardar datos para usar más adelante.

**Tipos de datos básicos en Python:**

1. **Enteros (int):** Números como 1, 10, -5
2. **Decimales (float):** Números con punto decimal como 3.14, -0.5



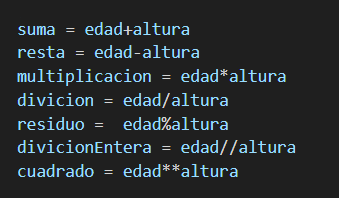
1. **Texto (str):** Secuencias de caracteres dentro de comillas ("" o '').
2. **Booleanos (bool):** Valores verdaderos o falsos (True o False).



**1.4 Operaciones básicas: Matemáticas**

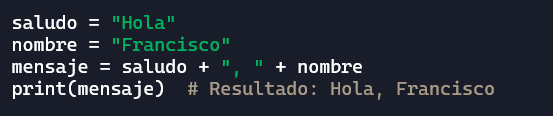
**Operaciones matemáticas:**

Se pueden realizar operaciones matemáticas como suma, resta, multiplicación, división, residuo, división entera y cuadrado

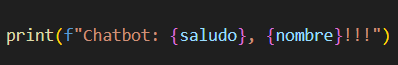
****

## **1.5 Concatenación**

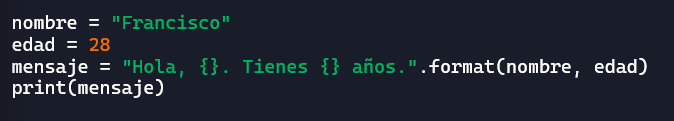
Se puede concatenar cadenas de texto usando el signo + dentro de una cadena



Se pueden formatear cadenas agregando f al inicio y antes de las comas dobles para agregar variables dentro del texto



Se puede usar el método format() para colocar valores , se hace uso de una llave por cada valor a agregar

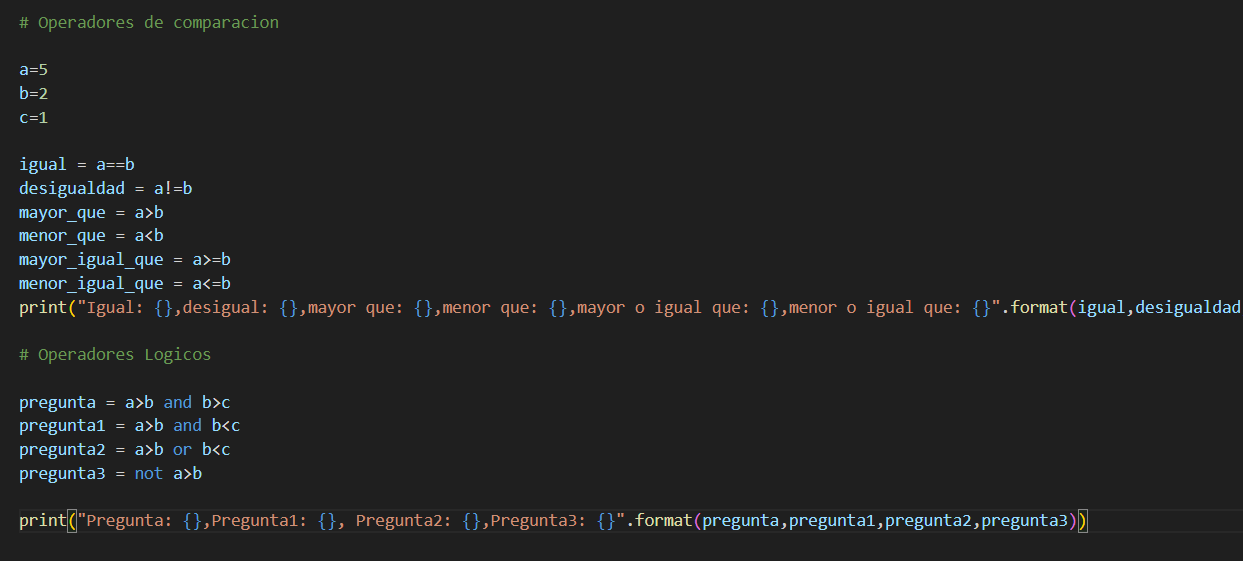


## **1.6 Operadores lógicos y de comparacion**

Los **operadores lógicos** se utilizan para realizar comparaciones entre valores o evaluar condiciones en Python

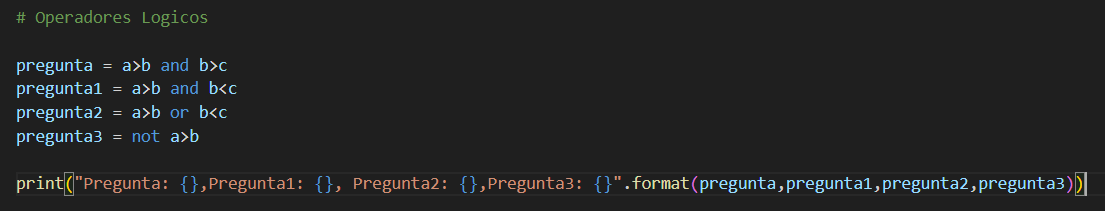
**Operadores de comparación:**

* == igualdad
* != desigualdad
* > mayor que
* < menor que
* >= mayor o igual que
* <= menor o igual que

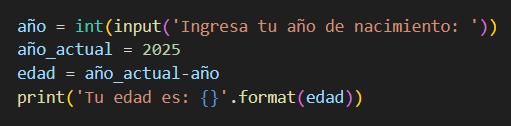


**Operadores lógicos:**

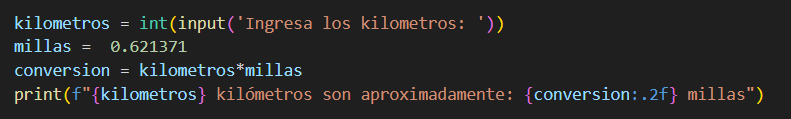
* And es verdadero si todas las condiciones son verdaderas
* Or es verdadero si al menos una condición es verdadera
* Not invierte el valor lógico de True a False y viceversa



## **1.7 Ejemplo práctico:**

* Crear un programa que calcule la edad del usuario a partir de su fecha de nacimiento.

## **1.8 Ejercicio:**

* Escribir un programa que convierta kilómetros a millas.

# **Módulo 2: Control de Flujo**

En este módulo se tiene como objetivo manejar la lógica en programas.

2.1 Estructuras condicionales: if, elif, else.

2.2 Bucles: for y while.

2.3 Operadores de comparación y lógicos.

2.4 Ejemplo práctico:

* Crear una calculadora que evalúe operaciones básicas según las opciones del usuario

## 2.5 Ejercicio:

* Escribir un programa que imprima los números primos en un rango dado.

# **Módulo 3: Funciones y Modularidad**

En este módulo se introduce la reutilización de código.

3.1 ¿Qué es una función? Declaración y uso (def).

3.2 Parámetros y valores de retorno.

3.3 Modularidad: Dividir proyectos en archivos manejables.

3.4 Pruebas unitarias: Introducción a testing con unittest o pytest.

3.5 Ejemplo práctico:

* Crear una función que calcule el área de un círculo.

3.6 Ejercicio:

* Implementar una función que convierta grados Celsius a Fahrenheit y escribir pruebas unitarias.

# **Módulo 4: Colecciones y Manipulación de Datos**

En este módulo se trabaja con estructuras de datos avanzadas.

4.1 Listas: Creación, manipulación y métodos (append(), pop(), etc.).

4.2 Tuplas: Uso y propiedades.

4.3 Diccionarios: Claves, valores y métodos avanzados (keys(), values(), items()).

4.4 Slicing: Manipulación de listas, cadenas y tuplas.

4.5 JSON: Introducción y manipulación de datos estructurados.

4.6 Ejemplo práctico:

* Crear un diccionario para gestionar inventario de productos.

## 4.7 Ejercicio:

* Trabajar con datos en formato JSON y gestionar contactos telefónicos.

# **Módulo 5: Programación Orientada a Objetos (POO)**

En este módulo se comprenden los principios básicos de la POO.

5.1 Clases y objetos: Definición y uso.

5.2 Métodos y atributos.

5.3 Herencia y polimorfismo.

5.4 Ejemplo práctico:

* Crear una clase para representar cuentas bancarias y operaciones de depósito/retiro.

## 5.5 Ejercicio:

* Implementar una clase para gestionar estudiantes con métodos avanzados (calcular promedio, etc.).

# **Módulo 6: Manejo de Errores y Archivos**

En este módulo se gestionan excepciones y se aprende a trabajar con archivos.

6.1 Excepciones: Captura y manejo con try, except, finally.

6.2 Archivos: Lectura y escritura (open(), read(), write()).

6.3 Entornos virtuales con venv: Configuración y gestión de dependencias.

6.4 Ejemplo práctico:

* Crear un programa que lea un archivo CSV y analice datos.

## 6.5 Ejercicio:

* Implementar un sistema de registro diario que guarde datos en un archivo.

# **Módulo 7: Bibliotecas para Análisis de Datos**

En este módulo se utilizan herramientas avanzadas.

7.1 Pandas: Creación y manipulación de DataFrames.

7.2 NumPy: Arrays, operaciones matemáticas y manipulación matricial.

7.3 Ejemplo práctico:

* Analizar datos de ventas usando Pandas y crear gráficos básicos

## 7.4 Ejercicio:

* Realizar operaciones matriciales y cálculos estadísticos usando NumPy.

# **Módulo 8: Visualización de Datos con Matplotlib**

En este módulo se introduce la creación de gráficos y visualizaciones con Matplotlib.

8.1 Introducción a Matplotlib: ¿Qué es y cómo instalarlo?

8.2 Uso de pyplot: Creación de gráficos básicos.

8.3 Personalización de gráficos: Colores, etiquetas, títulos y estilos.

8.4 Tipos de gráficos:

* Gráficos de líneas.
* Gráficos de barras.
* Gráficos de dispersión.
* Histogramas.

8.5 Ejemplo práctico:

* Crear un gráfico de barras para mostrar las ventas mensuales de un negocio

8.6 Ejercicio:

* Crear un gráfico de dispersión que muestre la relación entre dos conjuntos de datos (por ejemplo, horas de estudio y calificaciones).

# **Módulo 9: Frameworks de Desarrollo Web**

En este módulo se introduce el desarrollo web.

9.1 Flask: Crear aplicaciones web básicas y APIs REST.

9.2 Django: Crear proyectos robustos con modelos y vistas.

9.3 Ejemplo práctico:

Crear un formulario web y una API básica con Flask

## 9.4 Ejercicio:

* Desarrollar un sistema CRUD (Crear, Leer, Actualizar, Eliminar) con Django.

# **Módulo 10: Git y Entorno de Trabajo**

En este módulo se aprenden herramientas esenciales para la gestión de proyectos.

10.1 Git: Iniciar repositorios, comandos básicos (clone, commit, push, pull).

10.2 Uso de ramas y resolución de conflictos (branch, merge).

10.3 Configurar requirements.txt en proyectos.

10.4 Ejemplo práctico:

* Crear y colaborar en un proyecto usando GitHub

## 10.5 Ejercicio:

* Subir un proyecto a GitHub y trabajar en equipo.

# **Módulo 11: Proyecto Final**

En este módulo se consolidan los conocimientos con un proyecto completo.

11.1 Ideas de proyectos:

* Chatbot básico: Responder preguntas predefinidas.
* Gestión de inventarios: Registrar, buscar y actualizar datos.
* Análisis de datos: Visualización y estadísticas con Pandas.

# **Módulo 12: Automatización con Python**

En este módulo aprenderás cómo usar Python para automatizar tareas.

12.1 Introducción a la automatización: ¿Qué es y por qué es útil?

12.2 Uso de pyautogui:

* Controlar el mouse y el teclado con Python.
* Automatizar clics y escritura de texto

## 12.3 Web Scraping con BeautifulSoup y requests:

* Extraer información de sitios web.
* Manipular HTML para obtener datos específicos.

## 12.4 Selenium para automatización web:

* Simular interacción con un navegador.
* Completar formularios automáticamente.
* Descargar archivos de forma automática.

## 12.5 Ejemplo práctico:

* Crear un script que abra varias páginas web y extraiga información útil

## 12.6 Ejercicio:

* Diseñar un bot que complete formularios en línea automáticamente.