



[LCD] MÓDULO 1

Bienvenido al **Módulo 1** del curso **Lenguaje de Ciencia de Datos I.**

INTRODUCCIÓN

Estimado estudiante:

En el presente módulo aprenderás los fundamentos de Python: sus características principales, la sintaxis básica y la filosofía de diseño que lo distingue como un lenguaje claro y versátil. También aprenderás a declarar y manipular variables, comprender los diferentes tipos de datos numéricos y lógicos, y aplicar funciones de entrada y salida para interactuar con el usuario. Además, aprenderás a usar operadores aritméticos, relacionales y lógicos que te permitirán realizar cálculos, tomar decisiones y estructurar programas simples pero funcionales.

Te deseamos una excelente experiencia de aprendizaje en este curso.

¡Muchos éxitos!

 TEMA 1: FUNDAMENTOS DEL LENGUAJE PYTHON

 CIERRE

TEMA 1: FUNDAMENTOS DEL LENGUAJE PYTHON

FUNDAMENTOS DEL LENGUAJE PYTHON

Llegó el momento para que te conviertas en el protagonista de tu aprendizaje.

- Descubre y analiza la información que se presenta en la lección interactiva para transformarla en nuevo conocimiento.

Con Python descubrirás que programar puede ser simple, claro y el primer paso para resolver problemas reales.





Python es un lenguaje de programación reconocido por su claridad y simplicidad, lo que lo convierte en una de las herramientas más poderosas para aprender a programar y resolver problemas reales. En este tema exploraremos sus características esenciales, la sintaxis básica y la manera en que organiza el código. Con estos fundamentos, estarás preparado para comprender cómo funciona un programa en Python y dar tus primeros pasos en la programación.

Características y Sintaxis Básica

Python es un lenguaje de programación interpretado, de alto nivel, multiparadigma y multiplataforma. Su diseño fue concebido a finales de los años 80 por Guido van Rossum y publicado oficialmente en 1991. Desde entonces, se ha convertido en uno de los lenguajes más populares en ámbitos educativos, científicos, empresariales y en el desarrollo de inteligencia artificial.

Principales características

Interpretado

No requiere compilación previa, lo que permite ejecutar el código directamente y agiliza la prueba y depuración.

Tipado dinámico

No es necesario declarar el tipo de dato de una variable, ya que el intérprete lo asigna automáticamente.

Multiparadigma

Soporta programación imperativa, orientada a objetos y funcional, adaptándose a diferentes estilos de desarrollo.

Multiplataforma

Se ejecuta en sistemas operativos como Windows, Linux y macOS sin necesidad de grandes cambios en el código.

Sintaxis clara y legible

Prioriza la simplicidad, de modo que el código se asemeja al lenguaje humano.



Indentación obligatoria

El sangrado en el código determina los bloques, eliminando el uso de llaves {} y mejorando la organización.



Extensible y versátil

Cuenta con miles de librerías y frameworks que amplían sus capacidades en ciencia de datos, inteligencia artificial, desarrollo web y automatización.

Ejemplo de sintaxis

```
print("Hola mundo")
if 5 > 2:
    print("Cinco es mayor que dos")
x = 10
x = "texto"
```



Filosofía de diseño

La filosofía de diseño de Python busca que el código sea comprensible, ordenado y eficiente, facilitando el trabajo colaborativo y la resolución de problemas reales. Algunos de sus principios son:



- **Explicito es mejor que implícito:** el código debe mostrar con claridad lo que hace, evitando ambigüedades.

- **Bello es mejor que feo:** la estética y legibilidad del código son valores que favorecen el aprendizaje y el mantenimiento.
 - **Lo simple es preferible a lo complejo, y lo complejo a lo enrevesado:** la claridad debe guiar las decisiones de programación.
 - **La legibilidad cuenta:** Python está diseñado para que otros programadores comprendan fácilmente lo que se ha escrito.
-
- **Debe haber una —y preferiblemente solo una— manera obvia de hacer las cosas:** esto reduce la confusión y estandariza prácticas.
 - **Los errores nunca deberían pasar silenciosamente:** deben identificarse y manejarse para mejorar la calidad del software.

En conjunto, estos principios hacen que Python sea no solo un lenguaje técnico, sino también una filosofía de programación que prioriza la comunicación entre personas a través del código.



Gracias a su facilidad de aprendizaje, versatilidad y soporte comunitario, Python es considerado un lenguaje universal para la enseñanza de la programación y para la solución de problemas en sectores como la educación, la industria, la investigación científica y la inteligencia artificial.

CONTINUAR

Variables

En programación

Una variable es un espacio de memoria que sirve para almacenar información y manipularla en el programa. Puede imaginarse como una caja con una etiqueta, donde la etiqueta es el nombre de la variable y el contenido es el valor que se guarda.

En Python

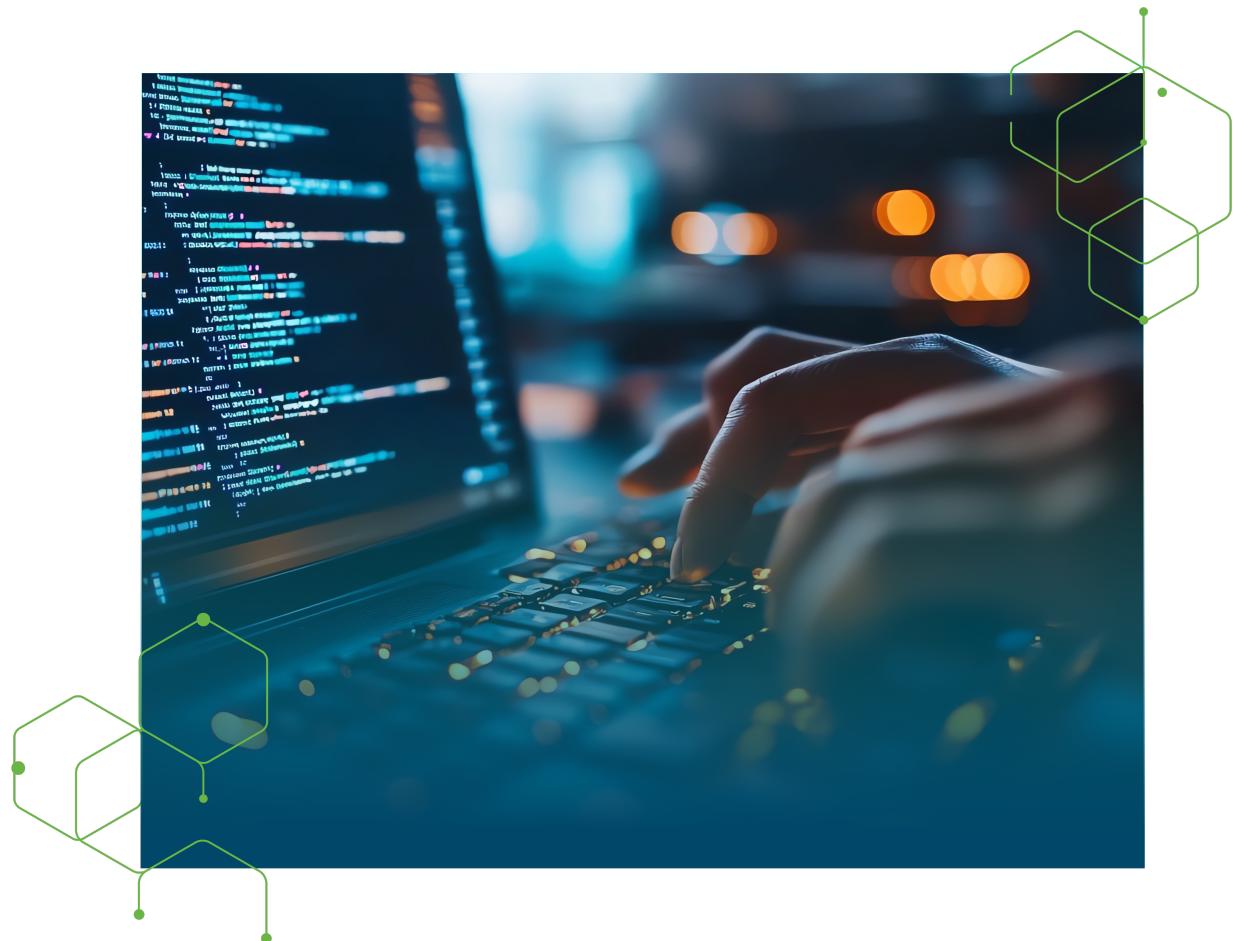
En Python, no es necesario especificar el tipo de dato al crear una variable, porque el lenguaje usa tipado dinámico. Esto significa que el mismo identificador puede cambiar de tipo según el valor que reciba:

```
x = 10      # entero  
x = "Hola"  # texto  
x = 3.5     # decimal
```



Reglas para nombrar variables

- No pueden iniciar con números.
- No deben contener espacios (se recomienda usar guion bajo).
- No se pueden usar palabras reservadas del lenguaje (ej.: for, if, class).
- Se recomienda emplear el estilo snake_case: todas las letras en minúscula y separadas por guiones bajos.



Tipos de datos que pueden almacenar



En Python, es posible almacenar números, texto, colecciones y valores lógicos, lo que permite construir programas flexibles y adaptados a distintas necesidades.

- Números: enteros (int), decimales (float), complejos (complex).
- Texto: cadenas (str).
- Colecciones: listas, tuplas, diccionarios y conjuntos.

- Valores lógicos: True y False (booleanos).

```
nombre = "Ana"    # cadena
edad = 20    # entero
nota = 18.7   # decimal
aprobado = True   # booleano
cursos = ["Python", "Álgebra"]  # lista
alumno = {"nombre": "Ana", "edad": 20} # diccionario
```



Cuadro comparativo: Buenas y malas prácticas al nombrar variables

Nombrar correctamente las variables es un aspecto esencial en la programación, ya que mejora la legibilidad del código y facilita su mantenimiento. Este cuadro comparativo muestra ejemplos de buenas y malas prácticas al asignar nombres, destacando la importancia de utilizar identificadores descriptivos, coherentes y fáciles de entender en lugar de abreviaciones confusas o poco significativas.

Buenas prácticas	Malas prácticas
precio_producto	pp

total_suma	ts1
promedio_notas	x
nombre_estudiante	NE
cantidad_maxima	c
numero_de_clientes	numcli
es_aprobado	flag1



El manejo de variables es esencial porque permite organizar, reutilizar y transformar información dentro de los programas.

CONTINUAR

Tipos de Datos Numéricos y Lógicos

En Python, los tipos de datos numéricos permiten representar y manipular valores matemáticos, mientras que los datos lógicos o booleanos permiten tomar decisiones.

Tipos de datos numéricos

En Python, los datos numéricos permiten realizar cálculos matemáticos y representar distintos valores según la necesidad del programa. La siguiente tabla resume los principales tipos de números, su descripción y ejemplos prácticos que muestran cómo se usan.

Tipo	Descripción	Ejemplo
int	Enteros: números positivos o negativos sin decimales.	edad = 25 temperatura = -3
float	Números con decimales, usados para cálculos más precisos.	pi = 3.1416 promedio = 15.7

complex	Números complejos con parte real e imaginaria.	$z = 2 + 3j$ <code>z.real → 2.0</code> <code>z.imag → 3.0</code>
----------------	--	--

Datos lógicos o booleanos

Además de los números, Python emplea valores lógicos para la toma de decisiones. Los datos booleanos permiten establecer condiciones verdaderas o falsas, y con ellos se controla el flujo de un programa. La siguiente tabla presenta sus características y ejemplos de aplicación.

Tipo	Descripción	Ejemplo
bool	Representan valores de verdad: True o False.	<code>es_mayor = True</code> <code>es_menor = False</code>
Condiciones	Permiten controlar el flujo de ejecución en los programas.	<code>edad = 20</code> <code>if edad >= 18:</code> <code>print('Mayor de edad')</code>

Los tipos de datos numéricos y lógicos son la base de los algoritmos. Gracias a ellos, Python puede ejecutar cálculos y evaluar condiciones para controlar el flujo de un programa.

CONTINUAR

Entrada/Salida y Operadores

En Python, los programas pueden interactuar con el usuario a través de funciones de entrada y salida. La función `input()` permite capturar datos y `print()` mostrarlos.

Operadores aritméticos

Los operadores aritméticos permiten realizar operaciones matemáticas básicas en Python. Gracias a ellos podemos sumar, restar, multiplicar, dividir y calcular valores con facilidad dentro de un programa.

Operador	Descripción	Ejemplo
+	Suma	$7 + 2 \rightarrow 9$
-	Resta	$7 - 2 \rightarrow 5$
*	Multiplicación	$7 * 2 \rightarrow 14$
/	División	$7 / 2 \rightarrow 3.5$
//	División entera	$7 // 2 \rightarrow 3$
%	Módulo (residuo)	$7 \% 2 \rightarrow 1$
**	Exponente	$7 ** 2 \rightarrow 49$

Operadores relacionales

Los operadores relacionales sirven para comparar valores. Sus resultados siempre son verdaderos o falsos, y son esenciales para

establecer condiciones en la ejecución de un programa.

Operador	Descripción	Ejemplo
>	Mayor que	$5 > 3 \rightarrow \text{True}$
<	Menor que	$5 < 3 \rightarrow \text{False}$
\geq	Mayor o igual	$5 \geq 5 \rightarrow \text{True}$
\leq	Menor o igual	$3 \leq 5 \rightarrow \text{True}$
\equiv	Igual a	$5 \equiv 3 \rightarrow \text{False}$
\neq	Diferente de	$5 \neq 3 \rightarrow \text{True}$

Operadores lógicos

Los operadores lógicos permiten combinar condiciones y evaluar expresiones más complejas. Son fundamentales para la toma de decisiones, ya que determinan si una acción debe ejecutarse o no según varias reglas.

Operador	Descripción	Ejemplo
and	Verdadero si ambas condiciones son verdaderas	True and False → False
or	Verdadero si al menos una condición es verdadera	True or False → True
not	Invierte el valor lógico	not True → False

Las funciones de entrada y salida, junto con los operadores, permiten crear programas dinámicos e interactivos. Gracias a ellos, Python puede recibir información del usuario, procesarla mediante cálculos y condiciones, y mostrar resultados claros y útiles.

CONTINUAR

El estudio de los fundamentos de Python permite comprender la base sobre la cual se construyen programas más complejos. Gracias a su sintaxis clara, su flexibilidad y sus principios de diseño, este lenguaje ofrece un entorno accesible para aprender programación y resolver problemas reales. Con el dominio de variables, tipos de datos y operadores, se adquieren herramientas esenciales para avanzar hacia aplicaciones prácticas en ciencia de datos y desarrollo de software.



CONTINUAR

Para finalizar el tema, recordemos algunas ideas claves.

1

Características del lenguaje: Python es interpretado, multiparadigma y multiplataforma, con una sintaxis clara y legible que prioriza la simplicidad.

2

Variables y datos: permiten almacenar y manipular información de manera flexible gracias al tipado dinámico.

3

Tipos de datos numéricos y lógicos: constituyen la base para realizar cálculos y tomar decisiones en los programas.

4

Entrada, salida y operadores: facilitan la interacción con el usuario y el procesamiento de información mediante operaciones aritméticas, relaciones y lógicas.

CONTINUAR

CIERRE

