

Programación en Python

Entorno, Variables, Estructuras de Control y Arreglos

Diego Gonzales, Edda Martínez

13 de noviembre de 2025

¿Qué es Python?

- Python es un lenguaje de programación, interpretado, de tipado dinámico, cuya filosofía destaca en una sintaxis que favorezca un código legible.
- Además, es un lenguaje de programación multiparadigma y multiplataforma.

¿Qué es Python?

En otras palabras, Python es:

- **Interpretado:** Porque se ejecuta sin necesidad de ser procesado por el compilador y se detectan los errores en tiempo de ejecución.
- **De tipado dinámico:** Porque las variables se comprueban en tiempo de ejecución.
- **Multiplataforma:** Porque está disponible para plataformas Windows, Linux o MAC.
- **Multiparadigma:** Porque soporta programación funcional, programación imperativa y programación orientada a objetos.

IDE's y Editores

- Cuando se programa en Python, es necesario tener un editor de texto o un entorno de desarrollo integrado (IDE), donde se escribirá el código que se ejecutará más adelante para ver la funcionalidad del programa.
- Existe toda una variedad de editores de texto y entornos de desarrollo (IDE) con los que es posible programar en Python.

Ejemplos de editores de texto



- IDLE, SpyDER, PyCharm, Atom, Jupyter, Anaconda, WINGBEHON

Declaración de variables

- En programación, una variable es un espacio en memoria donde se guardan y recuperan los datos que utiliza un programa.
- Cada variable debe tener un nombre, con el cual se podrá identificar y referirse a ella, durante el desarrollo de un programa.
- En Python, el nombre de una variable, no puede coincidir con los nombres de los comandos asignados a este lenguaje de programación, además, no deberá contener espacios en blanco.

Variables en Python

En Python, los dos tipos de variables más comunes, son:

- Las variables que almacenan números.
- Las variables que almacenan texto.

Variables en Python

En las variables que almacenan números hay dos tipos principales:

- Los números enteros (llamados int).
- Los números decimales o reales (llamados float).

A las variables que almacenan texto, se les denomina strings(str). Es muy importante, que el contenido a almacenar en este tipo de variables, se encuentre entre comillas.

Declaración de una variable

- ➊ Asignar el nombre de la variable.
- ➋ Posteriormente, se crea el espacio en memoria.
- ➌ Se le indica a la variable el dato a guardar.
- ➍ Se asigna el dato que se guardará.
- ➎ Se almacena el dato dentro de la variable.

Python puede identificar el tipo de dato que se quiere almacenar.
Python diferencia mayúsculas de minúsculas.

Librerías matemáticas

- Librerías integradas en Python (no necesitan instalación):
math, statistics, random, decimal, fractions.
- Librerías externas (si necesitan instalación):
numpy, pandas, matplotlib, scipy, sympy.

Estructuras de Control

¿Qué son las estructuras de control?

Son elementos fundamentales que permiten controlar el flujo de ejecución de un programa.

- Control condicional: if-else
- Bucles: for, while
- Operadores condicionales

Importancia

Permiten crear programas dinámicos que toman decisiones y repiten tareas.

Operadores Condicionales

Operadores de Comparación

Operador	Descripción
==	Igual a
!=	Diferente de
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Operadores Lógicos

Operador	Descripción
and	Y lógico
or	O lógico
not	Negación lógica

Estructura if-else

Sintaxis Básica

```
1 if condicion:
2     # codigo si la condicion es True
3 else:
4     # codigo si la condicion es False
```

Ejemplo

```
1 edad = 18
2 if edad >= 18:
3     print("Mayor de edad")
4 else:
5     print("Menor de edad")
```

Estructura if-elif-else

Múltiples Condiciones

```
1 if condicion1:
2     # codigo si condicion1 es True
3 elif condicion2:
4     # codigo si condicion2 es True
5 else:
6     # codigo si ninguna es True
```

Ejemplo Práctico

```
1 nota = 85
2 if nota >= 90:
3     print("A")
4 elif nota >= 80:
5     print("B")
6 else:
7     print("C")
```

Bucle for

Sintaxis Básica

```
1 for variable in secuencia:  
2     # codigo a repetir
```

Ejemplos Comunes

```
1 # Con range()  
2 for i in range(5):  
3     print(i)  
4  
5 # Con lista  
6 frutas = ["manzana", "banana", "naranja"]  
7 for fruta in frutas:  
8     print(fruta)
```

Bucle while

Sintaxis Básica

```
1 while condicion:
2     # codigo a repetir mientras
3     # la condicion sea True
```

Ejemplo Práctico

```
1 contador = 0
2 while contador < 5:
3     print(f"Contador: {contador}")
4     contador += 1
```


Control de Bucles

Palabras Clave

break	Termina el bucle completamente
continue	Salta a la siguiente iteración
pass	No hace nada (marcador de posición)
else	Se ejecuta si el bucle termina normalmente

Ejemplo con break y continue

```
1 for i in range(10):
2     if i == 3:
3         continue    # Salta el 3
4     if i == 7:
5         break        # Termina en 7
6     print(i)
7 else:
8     print("Bucle completado")
```

¿Qué son los arreglos?

Definición

Estructuras de datos que permiten almacenar múltiples valores bajo un mismo nombre de variable.

Arreglos Lineales

- Una sola dimensión
- Secuencia ordenada
- Ejemplos: vectores, listas
- Acceso por posición

Arreglos Lineales: Listas en Python

Características principales

- Colecciones ordenadas y mutables
- Pueden contener diferentes tipos de datos
- Tamaño dinámico (pueden crecer o reducirse)
- Se definen con corchetes cuadrados []

Ventajas

- Flexibilidad en el tipo de datos
- Métodos integrados para manipulación
- Fácil de usar y entender

Operaciones Básicas con Listas

Acceso y Modificación

- **Indexación:** Acceso a elementos por posición
- **Slicing:** Obtención de sub-listas
- **Modificación:** Cambio de valores existentes
- **Longitud:** Determinación del tamaño con `len()`

Búsqueda y Verificación

- Verificar existencia con operador **in**
- Encontrar posición con método **index()**
- Contar ocurrencias con método **count()**

Métodos Esenciales de Listas

Agregar Elementos

- `append()`: Al final
- `insert()`: En posición específica
- `extend()`: Múltiples elementos

Ordenamiento

- `sort()`: Orden ascendente
- `reverse()`: Invertir orden
- `sorted()`: Crear nueva lista ordenada

Eliminar Elementos

- `remove()`: Por valor
- `pop()`: Por posición
- `clear()`: Todos los elementos

Información

- `len()`: Tamaño
- `index()`: Posición
- `count()`: Frecuencia