



# **BASES DE PROGRAMACIÓN**



**Parte I**  
**Comunicándonos con**  
**las computadoras**



A continuación exploraremos conceptos clave de la programación. Estos fundamentos son esenciales para comprender el funcionamiento de los programas y son la base para desarrollar soluciones innovadoras en el mundo tecnológico actual. A continuación, examinaremos detalladamente cada uno de estos conceptos

## Programación



La programación es el proceso de crear un conjunto de instrucciones que le indican a la computadora cómo realizar una tarea. Involucra la utilización de algoritmos, que son secuencias de pasos lógicos y precisos diseñados para resolver problemas. Los programadores utilizan su creatividad y lógica para desarrollar soluciones eficientes y efectivas a los desafíos planteados, utilizando diferentes lenguajes de programación y herramientas especializadas.

## Algoritmo

Haz click en la siguiente imagen para ver el video en que se explica que son los algoritmos



## Lenguaje de Programación

Un lenguaje de programación es un conjunto de reglas y símbolos que permiten a los programadores dar instrucciones a una computadora.

Cada lenguaje de programación tiene su propia sintaxis y estructura que dicta cómo deben escribirse las instrucciones.

## Sintaxis

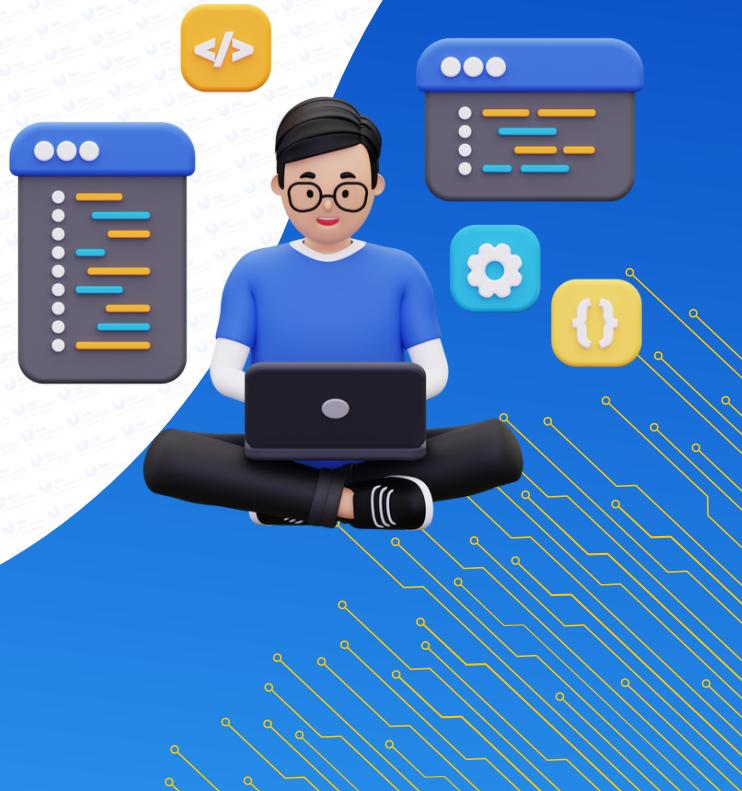
Imagina que estás escribiendo un mensaje a un amigo. Cuando escribes, usas reglas específicas para formar oraciones y palabras. La sintaxis en la programación son las reglas y estructuras que debes seguir al escribir código. Es como la gramática o las reglas de ortografía en el idioma que usas para hablar con la computadora.

Si no sigues las reglas de la sintaxis correctamente, la computadora puede no entender lo que quieras decirle. Es como si escribieras una palabra mal o pusieras las palabras en un orden extraño en tu mensaje, tu amigo podría no entender lo que quieras decir.



## Parte II

# Programming Essentials



## Python



Python es un lenguaje de programación popular y poderoso que se destaca por su simplicidad y versatilidad. Una de sus mayores ventajas es su facilidad de lectura y escritura. Es como un lenguaje que se parece al inglés, lo que lo hace más comprensible y fácil de aprender.

Lo asombroso de Python es que se puede usar para una amplia gama de aplicaciones: desde desarrollar aplicaciones web y móviles hasta análisis de datos, inteligencia artificial, aprendizaje automático y más. Es una herramienta valiosa en campos como la ciencia de datos, la ingeniería de software y la automatización de tareas.

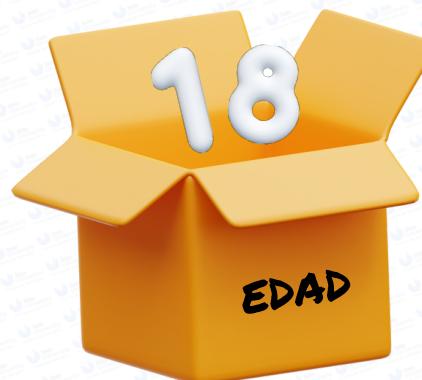
## Variables

Imagina que tienes una caja etiquetada con un nombre. Puedes poner diferentes objetos dentro de esta caja y cada vez que necesites uno de esos objetos, solo tienes que mirar dentro de la caja con ese nombre.

Una variable en programación es como esa caja. Tiene un nombre que le asignas y puedes guardar diferentes tipos de información dentro de ella, como números, palabras, o cualquier otro tipo de dato que necesites para tu programa.

Por ejemplo, podrías tener una variable llamada 'edad' donde guardas la edad de una persona. En otra variable llamada 'nombre', podrías guardar el nombre de esa persona. Así, cuando necesites acceder a la edad o al nombre en tu programa, solo tienes que referirte a esas variables por su nombre.

Las variables en programación te permiten almacenar información de manera organizada y acceder a ella fácilmente cuando la necesitas, como si estuvieras buscando algo dentro de una caja con un nombre específico.



## Tipos de Datos

Los tipos de datos son las categorías o clasificaciones que se utilizan para representar diferentes tipos de información. Cada tipo de dato tiene características específicas y se utiliza para almacenar un tipo particular de información en un programa.

### Número (Number)

Representa valores numéricos como enteros o decimales.

```
● ● ●  
# Ejemplo de números enteros  
edad = 25  
cantidad_de_productos= 10  
# Ejemplo de números decimales  
precio = 15.99  
promedio = 7.5
```

### String (Cadena de caracteres)

Representa texto o palabras.

```
● ● ●  
# Ejemplo de strings  
nombre = "Juan"  
mensaje = 'Hola Mundo!'  
direccion = "Calle Principal 123"
```

### Booleano (Boolean)

Representa valores lógicos de verdadero (True) o falso (False).

```
● ● ●  
# Ejemplo de booleanos  
es_mayor_de_edad= True  
tiene_descuento= False  
esta_encendido = True
```

### Lista (List)

Representa una colección ordenada de elementos.

```
● ● ●  
# Ejemplo de listas  
numeros = [1, 2, 3, 4, 5]  
nombres = ['Ana', 'Juan', 'María']  
mezcla = [10, 'texto', True, 3.14]
```

## Operadores Aritméticos

Los operadores aritméticos en programación son símbolos especiales que se utilizan para realizar operaciones matemáticas en un programa.

### Suma (+)

Se utiliza para sumar dos valores

```
●●●  
a = 5  
b = 3  
suma = a + b  
print(suma) # Resultado: 8
```

### Resta (-)

Se utiliza para restar un valor de otro.

```
●●●  
a = 10  
b = 4  
resta = a - b  
print(resta) # Resultado: 6
```

### Multiplicación (\*)

Se utiliza para multiplicar dos valores.

```
●●●  
a = 6  
b = 3  
multiplicacion = a * b  
# Resultado: 18  
print(multiplicacion)
```

### División (/)

Se utiliza para dividir un valor entre otro.

```
●●●  
a = 15  
b = 5  
division = a / b  
# Resultado: 3.0  
print(division)
```

### Módulo (%)

Devuelve el resto de la división entre dos números.

```
●●●  
a = 17  
b = 4  
resto = a % b  
print(resto) # Resultado: 1
```

### Potenciación (\*\*)

Se utiliza para elevar un número a una potencia.

```
●●●  
a = 2  
b = 3  
potencia = a ** b  
print(potencia) # Resultado: 8
```

## Operadores Relacionales

Los operadores relacionales se utilizan para comparar valores y devolver un resultado booleano (verdadero o falso) según la relación entre los valores.

### Operador Igualdad (==)

Comprueba si dos valores son iguales.

```
● ● ●  
a = 5  
b = 7  
# False, ya que 5 no es igual a 7  
resultado = (a == b)
```

### Operador Distinto (!=)

Comprueba si dos valores son diferentes.

```
● ● ●  
a = 10  
b = 10  
# False, ya que 10 es igual a 10  
resultado = (a != b)
```

### Operador Mayor que (>)

Comprueba si un valor es mayor que otro.

```
● ● ●  
edad = 18  
edad_minima = 21  
# False, ya que 18 no es mayor que 21  
resultado = (edad > edad_minima)
```

### Operador Menor que (<)

Comprueba si un valor es menor que otro.

```
● ● ●  
calificacion = 75  
calificacion_aprobatoria= 70  
# False, ya que 75 no es menor que 70  
resultado = (calificacion < calificacion_aprobatoria)
```

### Operador Mayor o igual que (>=)

Comprueba si un valor es mayor o igual que otro.

```
● ● ●  
numero1 = 15  
numero2 = 10  
# True, ya que 15 es mayor o igual que 10  
resultado = (numero1 >= numero2)
```

### Operador Menor o igual que (<=):

Comprueba si un valor es menor o igual que otro.

```
● ● ●  
cantidad_productos= 100  
cantidad_maxima = 150  
# True, ya que 100 es menor o igual que 150  
resultado = (cantidad_productos<= cantidad_maxima)
```

## Operadores Logicos

Los operadores lógicos se utilizan para realizar operaciones lógicas entre expresiones booleanas (que devuelven verdadero o falso) y para combinar o modificar los resultados lógicos.

### and

Devuelve verdadero si ambas expresiones son verdaderas, de lo contrario, devuelve falso.

```
●●●  
edad = 18  
tiene_licencia = True  
  
edad == 18 and tiene_licencia #resultado True  
edad < 18 and tiene_licencia #resultado False
```

### or

Devuelve verdadero si al menos una de las expresiones es verdadera, devuelve falso si ambas son falsas.

```
●●●  
edad = 18  
tiene_licencia = False  
  
edad == 18 or tiene_licencia #resultado True  
edad < 18 or tiene_licencia #resultado False
```

### not

Devuelve el valor opuesto de la expresión

```
●●●  
tiene_licencia = True  
  
not tiene_licencia #resultado False
```

## Listas

Las listas nos permiten almacenar una colección ordenada de elementos. Aquí te presento las operaciones básicas que puedes realizar con listas

### Creación de listas

Puedes crear una lista simplemente colocando elementos entre corchetes [ ]

```
lista_numeros = [1, 2, 3, 4, 5]
lista_strings = ['manzana', 'naranja', 'plátano']
lista_mixta = [1, 'dos', True, 3.14]
```

### Acceso a elementos:

Puedes acceder a los elementos de una lista utilizando su índice. Los índices comienzan desde 0

```
lista = ['a', 'b', 'c', 'd', 'e']
primer_elemento = lista[0] # Accede al primer elemento ('a')
tercer_elemento = lista[2] # Accede al tercer elemento ('c')
```

### Modificación de elementos

Puedes cambiar o actualizar elementos de una lista utilizando su índice

```
lista = [10, 20, 30, 40, 50]
lista[2] = 35 # Modifica el tercer elemento de la lista
```

## Añadir elementos

Puedes agregar elementos al final de una lista utilizando el método `append()`

```
●●●  
lista = [1, 2, 3]  
lista.append(4) # Agrega el número 4 al final de la lista
```

## Eliminar elementos

Puedes eliminar elementos de una lista utilizando el método `remove()` o `pop()`

```
●●●  
lista = ['manzana', 'naranja', 'plátano']  
lista.remove('naranja') # Elimina 'naranja' de la lista  
elemento_eliminado= lista.pop(0) # Elimina y devuelve el elemento en el índice 0
```

## Longitud de la lista

Puedes obtener la cantidad de elementos de una lista utilizando la función `len()`

```
●●●  
lista = [10, 20, 30, 40, 50]  
cantidad_elementos= len(lista) # Devuelve 5
```