

# Conferencia 03

Variables en Serie

# Objetivo de Hoy

Entender cómo agrupar y gestionar datos.

- Dominar **Listas y Tuplas**.
- Comprender conceptos clave: **Mutabilidad y Referencia**.

## ¿Por Qué Agrupar Datos?

Imagina las notas de 1000 alumnos...

**Ineficiente:** `nota1, nota2, nota3, ...`

**Eficiente:** Una sola “caja” que contenga todas las notas. `[nota1, nota2, nota3, ...]`

# Listas: La Navaja Suiza

Una colección **ordenada** y **MODIFICABLE** de elementos.

# Creación de Listas [ ]

Se crean usando corchetes [ ].

- `notas = [8.5, 9.0, 7.2, 10.0]`
- `nombres = ["Ana", "Juan", "Yosvany"]`
- `mixta = ["Hola", 100, True]`
- `vacía = []`

## Ejemplo de Código

## Acceso: El Índice

Cada elemento tiene una posición. La numeración empieza en 0

Elemento	“Ana”	“Juan”	“Yosvany”
Índice	0	1	2

- `nombres[0]` devuelve "Ana".
- `nombres[2]` devuelve "Yosvany".

# Índices Negativos

Un atajo para contar desde el final.

- `nombres[-1]` devuelve el último elemento: "Yosvany".
- `nombres[-2]` devuelve el penúltimo: "Juan".



## Operador **in** y **del**

- **in**: ¿Está este elemento en la lista?
  - `"Ana" in nombres -> True`
- **del**: Elimina un elemento por su índice.
  - `del nombres[1] -> Elimina a “Juan”.`

## Ejemplo de Código

# Slicing: Rebanadas de tu Lista

Extraer una porción de la lista con la sintaxis `[inicio:fin:paso]`.

- `fin` **NO** se incluye.
- `[ :3 ]` -> Los primeros 3.
- `[ -3: ]` -> Los últimos 3.
- `[ : ]` -> Todos (una copia).
- `[ ::-1 ]` -> ¡La lista invertida!

## Ejemplo de Código

# Tuplas

Una colección **ordenada** e **INMUTABLE** (no se puede cambiar).

# Creación de Tuplas ( )

Se crean usando paréntesis ( ).

- `punto_3d = (10, 20, 5)`
- `dias_semana = ("Lunes", "Martes", ...)`
- **Ojo:** Una tupla de un solo elemento necesita una coma: `(42,)`

## Ejemplo de Código

## Tuplas vs. Listas: Lo Común

Las tuplas se comportan igual que las listas para:

- Acceso por índice: `mi_tupla[0]`, `mi_tupla[-1]`
- Operador `in`: `"Lunes" in dias_semana`
- Función `len()`: `len(dias_semana)`
- Slicing: `dias_semana[:5]`



# Descomposición

Asignar sus valores a variables de una sola vez.

```
coordenadas = (10, 20, 5)
```

```
x, y, z = coordenadas
```



- x es 10
- y es 20
- z es 5

# Usos de la Descomposición

- Intercambio de variables: `a, b = b, a`
- Recoger restos: `primero, *resto = (1, 2, 3, 4)`
  - `primero` es 1
  - `resto` es `[2, 3, 4]` (una lista!)

## Ejemplo de Código

# El Concepto Clave: Mutabilidad

- **Mutable:** Se puede cambiar después de su creación.
-  **Listas**
- **Inmutable:** NO se puede cambiar.
-  **Tuplas, Números, Strings.**

## Ejemplo de código

## Referencia vs. Copia 🏠

Este es uno de los conceptos más importantes.

`lista_b = lista_a` **NO** crea una copia.

Crea una nueva “etiqueta” que apunta a la **misma dirección** en memoria.

## La Sorpresa...

Si modificas `lista_b`, ¡también modificas `lista_a`!

Es como tener dos llaves para la misma casa. Si uno pinta la puerta, el otro la verá pintada.

## Ejemplo de Código



# ¿Cómo Crear una Copia Real?

Para tener dos listas independientes, usa:

1. **Método** `.copy()` `lista_b = lista_a.copy()`
2. **Slicing** `[:]` `lista_b = lista_a[:]`

Ahora son dos “casas” diferentes.

## Ejemplo de Código

Demostración en vivo: Creando copias correctas con `.copy()` y `[:]`.

# Resumen Final

- Listas `[]`:
  - Ordenadas y **Mutable**s. Tu herramienta principal.
- Tuplas `()`:
  - Ordenadas e **Inmutable**s. Para datos constantes.
- Mutabilidad:
  - La gran diferencia entre ambas.
- Referencias:
  - `b = a` no es una copia para muchos *tipos*.
  - Usa `.copy()` o `[:]` para duplicar.