Colecciones

Librerías estándar y manejo de colecciones

Las colecciones son fundamentales para el **almacenamiento y manipulación eficiente de datos en aplicaciones.**

Sirven para **modelar relaciones entre objetos y facilitar operaciones de almacenamiento y manipulación de datos** en aplicaciones de *software.*

Cada lenguaje ofrece estructuras y métodos específicos para esta tarea.

Colecciones en JavaScript

En JavaScript, las colecciones se manejan principalmente con la **clase `Array`**.

Se representan utilizando corchetes `[]`, donde **cada elemento tiene un índice numérico.**

Es posible tener un ***array* de números o de objetos.**

Operaciones comunes con *arrays*	
Métodos	**Descripción**
▪ `push` ▪ `pop` ▪ `shift` ▪ `unshift`	Agregar y quitar elementos.
▪ `forEach` ▪ `map` ▪ `filter`	Métodos de iteración para procesar elementos del *array*.

Ejemplo

- Se define un ***array* `numeros`** que contiene una secuencia de números del 1 al 5.
- Mediante el **método `push`**, se agrega el número 6 al final del *array.*
- Con el **método `pop`**, se elimina el último elemento del *array* (6).
- **Se itera sobre el array con `forEach`**, y se imprime cada número en la consola.

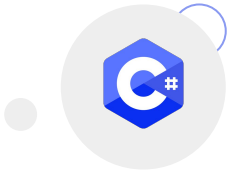
```
 // Ejemplo de colecciones en JavaScript
// Definición de un array de números
let numeros = [1, 2, 3, 4, 5];

// Agregar un elemento al final del array
numeros.push(6);

// Eliminar el último elemento del array
numeros.pop();

// Iterar sobre el array utilizando forEach
numeros.forEach(numero => { console.log(numero);
```

Colecciones en C# (.NET)

En C#, las colecciones se manejan principalmente con la clase **`List<T>`**.

Esta clase permite **almacenar elementos del tipo específico `T`** de manera dinámica.

A estas clases de colecciones se las conoce como ***genéricas.***



Uso de métodos para manejar listas	
Métodos	**Descripción**
▪ **`Add`** ▪ **`Remove`** ▪ **`Contains`**	Agregar, eliminar y verificar la existencia de elementos.
▪ **`ForEach`** ▪ **`FindAll`**	Métodos de iteración para procesar elementos de la lista.

Ejemplo

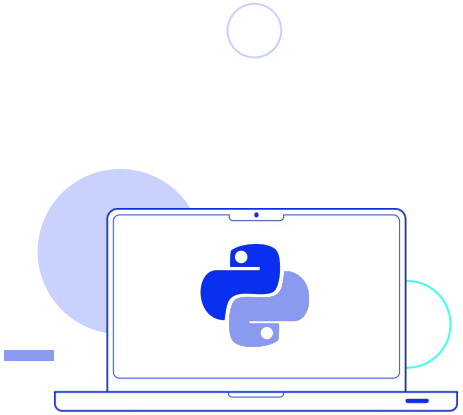
- Se define una lista **personas** de objetos de tipo **Persona**, donde cada objeto tiene propiedades como **Nombre** y **Edad**.
- Se utiliza el método **Add** para agregar objetos a la lista.

```
List<Persona> personas = new List<Persona>();

// Agregar objetos a la lista personas.Add(new
Persona("Juan", 25)); personas.Add(new
Persona("María", 30)); personas.Add(new
Persona("Pedro", 28));

// Iterar sobre la lista utilizando foreach
foreach (Persona persona in personas) {
Console.WriteLine($"Nombre: {persona.Nombre},
Edad: {persona.Edad}"); }
```

Colecciones en Python

Las listas se representan utilizando **corchetes [ ]**.

- Una lista en Python puede **contener elementos de diferentes tipos de datos** y permite la modificación de sus elementos.
- Las listas en Python son **estructuras de datos dinámicas y versátiles**, adecuadas para almacenar colecciones de elementos de manera ordenada y **accesible por índice**.



Ejemplo

- Se define una clase **Persona** con propiedades **nombre** y **edad**.
- Se crea una lista **personas** que contiene objetos de tipo **Persona**.
- Se utiliza un bucle **for** para iterar sobre la lista e imprimir cada objeto **Persona**.
- Se filtran personas mayores de 25 años.

```
# Creación de una lista de objetos de tipo Persona
personas = [
    Persona("Juan", 25),
    Persona("María", 30),
    Persona("Pedro", 28)
]

# Iteración sobre la lista
print("Listado de personas:")
for persona in personas:
    print(persona)

# Filtrar personas mayores de 25 años
mayores = [persona for persona in personas if
persona.edad > 25]
```

Colecciones en Java

En Java, las listas se representan utilizando la **interfaz `List`** y sus implementaciones como **`ArrayList`**, **`LinkedList`**, entre otras.

Una lista en Java:

- Es una estructura de datos que permite **almacenar elementos ordenados y acceder a ellos por índice.**
- Proporciona **métodos** para agregar, eliminar, y modificar elementos de manera dinámica.
- Es fundamental para el manejo de colecciones de datos en aplicaciones Java. Facilita operaciones como **búsqueda, inserción y ordenamiento de elementos.**

> Las listas son parte del **paquete *java.util*.**

- Permite almacenar colecciones ordenadas de **objetos del mismo tipo.**
- Es **dinámica,** lo que significa que puede crecer o reducirse según se agreguen o eliminen elementos.

Ejemplo

```
// Creación de una lista de personas
List<Persona> personas = new ArrayList<>();

// Agregar personas a la lista
personas.add(new Persona("Juan", 25));
personas.add(new Persona("María", 30));
personas.add(new Persona("Pedro", 28));

// Iterar sobre la lista
for (Persona persona : personas) {
     System.out.println(persona);
}
```