

# **Taller Investigativo**

## Arrays en Python



# 1. ¿Qué es un array o lista en Python?

- Investiga qué son los arrays (o listas) en Python y para qué se utilizan.

En Python, un array comúnmente se representa mediante una lista (list). Es una colección ordenada y modificable de elementos que puede contener distintos tipos de datos (números, cadenas, booleanos, etc.). Se utilizan para almacenar y manipular conjuntos de datos, realizar operaciones de agregación, búsqueda y filtrado, y como estructuras de datos en algoritmos y funciones.

- ¿Cómo se declara una lista vacía?

```
mi_lista_vacia = []
```

- ¿Cómo se crea una lista con valores iniciales?

```
mi_lista = [1, 2, 3, 4, 5]
```

Ejemplo práctico:

```
mi_lista = [1, 2, 3, 4, 5]
print(mi_lista)
```

# 2. ¿Cómo accedemos a los elementos de una lista?

- ¿Cómo se accede al primer elemento de una lista?

```
mi_lista[0]
```

- **¿Qué significa usar un índice negativo?**

Permite acceder a los elementos desde el final de la lista.

-1 es el último, -2 el penúltimo, etc.

- **¿Qué pasa si intento acceder a un índice que no existe?**

Se genera un error: IndexError.

### Ejemplo práctico:

```
numeros = [10, 20, 30, 40]
```

```
print(numeros[0]) # Primer elemento: 10
```

```
print(numeros[-1]) # Último elemento: 40
```

## 3. ¿Qué es el "slicing" o rebanado de listas?

El slicing permite obtener sublistas seleccionando rangos de índices.

- **¿Qué significa "slicing" en listas?**

Slicing en listas significa extraer una sublista de elementos de una lista original, especificando un rango de índices. Permite obtener una porción específica de la lista.

- **¿Cómo se obtiene una sublista usando slicing?**

Se obtiene especificando un rango de índices en la lista original, utilizando la sintaxis `lista[inicio:fin]`, donde `inicio` es el índice inicial y `fin` es el índice final (exclusive).

- **¿Qué significa dejar vacío el inicio o el final en el slicing?**

`lista[:fin]`: Inicia desde el principio de la lista hasta el índice `fin`.

`lista[inicio:]`: Inicia desde el índice `inicio` hasta el final de la lista.

### Ejemplo práctico:

```
datos = [10, 20, 30, 40, 50]
```

```
print(datos[1:4]) # [20, 30, 40]
```

```
print(datos[:3]) # [10, 20, 30]
```

```
print(datos[2:]) # [30, 40, 50]
```

## 4. ¿Cómo modificamos los elementos de una lista?

- ¿Cómo se cambia el valor de un elemento de la lista?

Modificar un elemento:

```
lista[índice] = nuevo_valor
```

- ¿Qué pasa si modificamos un índice que no existe?

Da un error IndexError.

Ejemplo práctico:

```
lista = [10, 20, 30, 40]
lista[2] = 99
print(lista) # [10, 20, 99, 40]
```

## 5. ¿Cómo agregamos nuevos elementos a una lista?

- ¿Cómo se agrega un elemento al final de la lista?

Agregar al final con .append():

```
lista.append(valor)
```

- ¿Cómo se inserta un elemento en una posición específica?

Insertar en posición específica con .insert():

```
lista.insert(posición, valor)
```

- **¿Cómo se combinan dos listas en una sola?**

Combinar listas con + o .extend():

lista += otra\_lista

Ejemplo práctico:

```
lista = [10, 20, 30]
lista.append(40)      # [10, 20, 30, 40]
lista.insert(1, 15)    # [10, 15, 20, 30, 40]
lista += [50, 60]     # [10, 15, 20, 30, 40, 50, 60]
print(lista)
```

## 6. ¿Cómo eliminamos elementos de una lista?

- **Cómo se elimina un valor específico de una lista?**

Eliminar un valor con .remove(valor):

lista.remove(valor)

- **¿Qué hace el método pop()?**

Eliminar por índice con .pop():

.pop() elimina el último.

.pop(i) elimina el índice i.

- **¿Cómo se elimina un elemento usando del?**

Eliminar con del:

del lista[i]

Ejemplo práctico:

```
lista = [10, 20, 30, 40, 50]
lista.remove(30)    # [10, 20, 40, 50]
lista.pop()         # [10, 20, 40]
del lista[1]         # [10, 40]
print(lista)
```

# **7. ¿Cómo buscamos elementos dentro de una lista?**

- **¿Cómo se verifica si un elemento está presente en una lista?**

Verificar si un valor está presente:

valor in lista

- **¿Cómo encontrar el índice de un elemento?**

Obtener índice con .index(valor):

lista.index(valor)

- **¿Cómo contar cuántas veces aparece un valor en la lista?**

Contar apariciones con .count(valor):

lista.count(valor)

**Ejemplo práctico:**

```
lista = [10, 20, 30, 40, 50]
print(20 in lista)      # True
print(lista.index(30))  # 2
print(lista.count(20)) # 1
```

# **8. ¿Cómo ordenamos los elementos de una lista?**

- **¿Cómo se ordena una lista de manera ascendente?**

Orden ascendente con .sort():

lista.sort()

- **¿Cómo se ordena en orden descendente?**

Orden descendente:

```
lista.sort(reverse=True)
```

- **¿Qué diferencia hay entre sort() y sorted()?**

sort() vs sorted():

sort() modifica la lista original.

sorted() devuelve una nueva lista ordenada.

#### Ejemplo práctico:

```
Ordenar la lista [40, 10, 30, 20]
```

```
# Lista original
```

```
lista = [40, 10, 30, 20]
```

```
# Orden ascendente (modifica la lista original)
```

```
lista.sort()
```

```
print("Orden ascendente:", lista)
```

```
# Orden descendente (modifica la lista original)
```

```
lista.sort(reverse=True)
```

```
print("Orden descendente:", lista)
```

```
# Nueva lista ordenada sin modificar la original
```

```
lista_original = [40, 10, 30, 20]
```

```
lista_ordenada = sorted(lista_original)
```

```
print("Lista original:", lista_original)
```

```
print("Lista ordenada (sin modificar original):", lista_ordenada)
```

# 9. ¿Cómo invertimos el orden de los elementos de una lista?

- ¿Cómo invertir una lista usando reverse()?

Usando .reverse():

Invierte la lista en el lugar (modifica la lista original).

python

Copiar código

lista.reverse()

- ¿Cómo invertir una lista usando slicing?

Usando slicing:

Crea una nueva lista invertida sin modificar la original.

python

Copiar código

lista\_invertida = lista[::-1]

## Ejemplo práctico:

python

Copiar código

lista = [10, 20, 30, 40]

```
# Método 1: reverse()
```

```
lista1 = lista.copy()
```

```
lista1.reverse()
```

```
print("Con reverse():", lista1)
```

```
# Método 2: slicing
```

```
lista2 = lista[::-1]
```

```
print("Con slicing:", lista2)
```

# 10. ¿Cómo hacemos una copia de una lista?

- ¿Cómo copiar una lista usando slicing?

Con slicing:

python

Copiar código

```
copia1 = lista[:]
```

- ¿Cómo copiarla usando list()?

Con list():

python

Copiar código

```
copia2 = list(lista)
```

- ¿Cómo copiarla usando copy()?

Con .copy():

python

Copiar código

```
copia3 = lista.copy()
```

**Ejemplo práctico:**

python

Copiar código

```
lista_original = [10, 20, 30]
```

```
copia1 = lista_original[:]
```

```
copia2 = list(lista_original)
```

```
copia3 = lista_original.copy()
```

```
print(copia1)
```

```
print(copia2)
```

```
print(copia3)
```

# 11. ¿Cómo comprobamos si una lista está vacía?

- ¿Cómo podemos saber si una lista no tiene elementos?

Usar una condición if not lista:

```
python
Copiar código
if not lista:
    print("La lista está vacía")
```

Ejemplo práctico:

```
python
Copiar código
lista_vacia = []

if not lista_vacia:
    print("La lista está vacía")
else:
    print("La lista tiene elementos")
```

# 12. ¿Cómo pedir varios datos al usuario y almacenarlos en una lista?

- ¿Cómo pedimos al usuario la cantidad de datos que quiere ingresar?

```
python
Copiar código
cantidad = int(input("¿Cuántos datos quieres ingresar?"))
```

- **¿Cómo almacenamos esos datos en una lista usando un ciclo for?**

python

Copiar código

```
datos = [] # Lista vacía
```

```
for i in range(cantidad):
```

```
    dato = input("Ingrese el dato {i + 1}: ")
```

```
    datos.append(dato) # Agrega cada dato a la lista
```

```
print("Lista final de datos:", datos)
```

### Ejemplo práctico:

```
lista_datos = []
```

```
cantidad = int(input("¿Cuántos elementos quieres ingresar? "))
```

```
for i in range(cantidad):
```

```
    elemento = input("Ingrese el elemento {i+1}: ")
```

```
    lista_datos.append(elemento)
```

```
print("Lista final:", lista_datos)
```