Estructuras de Datos y Listas en Python

Introducción y conceptos fundamentales

“Sin requerimientos o diseño programar sólo es el arte de agregar errores a un archivo vacío” .

- Louis Srygley

¿Qué es una estructura de datos?

Una estructura de datos es una forma organizada de almacenar, gestionar y manipular datos eficientemente.

Importancia de las estructuras de datos

Puntos clave:

- Optimización de la eficiencia en programas.
- Manejo de grandes volúmenes de datos.
- Acceso rápido y eficiente a la información.

¿Qué es una lista?

Una lista es una estructura de datos secuencial que almacena elementos ordenados.

Índices en listas

Cada elemento de una lista tiene una posición o índice.

```
lista = ["manzana", "banana", "cereza"]
print(lista[0])  # Salida: manzana
```

Beneficios de usar listas

Puntos clave:

- Organización de la información.
- Acceso rápido mediante índices.
- Capacidad de modificar los datos fácilmente.

Características de las listas

Lista de características clave:

- Ordenadas
- Indexadas
- Mutables
- Homogéneas o heterogéneas

Creación de listas en Python

```
mi_lista = []  # Lista vacía
otra_lista = [1, "Hola", 3.14, True]  # Lista con diferentes tipos
```

Acceso y modificación de listas

```
numeros = [10, 20, 30]
numeros[1] = 25  # Modifica el segundo elemento
print(numeros)  # Salida: [10, 25, 30]
```

Métodos principales de listas

Método	Resultado
`x.append(e)`	Agrega a la lista **x** el elemento **e**
`x.insert(i,e)`	Inserta **e** en la posición **i** de la lista **x**
`x.count(e)`	Conteo de instancias de **e** en la lista **x**
`x.remove(e)`	Elimina el primer elemento **e** de la lista **x**
`x.pop(i)`	Entrega el elemento en la posición `i` en la lista y lo elimina
`x.index(e)`	Entrega la posición del primer elemento **e** en **x**
`x.sort()`	Ordena **x** en forma creciente
`x.reverse()`	Invierte la lista **x**
`x.clear()`	Vaciar la lista **x**
`x.copy()`	Entrega una copia por referencia de una lista de un nivel

Método append()

Agrega un elemento al final de la lista.

```
frutas = ["manzana", "pera"]
frutas.append("naranja")
print(frutas)  # Salida: ['manzana', 'pera', 'naranja']
```

Método extend()

Agrega elementos de otra lista.

```
numeros = [1, 2, 3]
numeros.extend([4, 5, 6])
print(numeros)  # Salida: [1, 2, 3, 4, 5, 6]
```

Método insert()

Inserta un elemento en una posición específica.

```
colores = ["rojo", "azul"]
colores.insert(1, "verde")
print(colores)  # Salida: ['rojo', 'verde', 'azul']
```

Otros métodos importantes

- index(),
- remove()
- pop()
- clear()
- count()
- sort()

- reverse()
- in
- is
- any()
- copy()

Introducción a listas anidadas

Una lista que contiene otras listas como elementos.

```
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

Acceso a elementos en listas anidadas

```
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
```

```
print(matriz[1][1])  # Salida: 5
```

Modificación de listas anidadas

```
matriz[0][0] = 10
print(matriz)  # Salida: [[10, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Iteración sobre listas anidadas

```
for fila in matriz:
    for elemento in fila:
        print(elemento, end=" ")
```

Aplicaciones de listas anidadas

Ejemplos de uso:

- Tablas de datos.
- Matrices matemáticas.
- Representación de estructuras jerárquicas.

EJERCICIO

Crea un programa que gestione el inventario de una tienda con las siguientes opciones:

1. Agregar un producto con nombre y cantidad.
2. Actualizar la cantidad de un producto.
3. Eliminar un producto.
4. Mostrar el inventario.
5. Salir.

EJERCICIO - Análisis de Calificaciones (Usando Listas)

El usuario ingresará nombres de estudiantes y sus calificaciones. Luego, el programa mostrará:

1. Promedio de calificaciones.
2. Mejor y peor calificación.
3. Lista de estudiantes aprobados (nota ≥ 60).