

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.

Lista de compras		
<i>Vegetales</i>	<i>Frutas</i>	<i>Lácteos</i>
Brocoli Patatas Zanahoria Apio Zapallo Nabo Choco Tomate Poro Espárragos Coliflor Culantro Perejil Yuca Camote Cebolla	Lechuga Pepino Beterraga Valinitas Pimiento Rocoto arvejas Huacatay Col Limones Hierbabuena Espinaca Albahaca Rabanito Palta Ajos	Plátano Papaya Granada Naranjas Mandarina Piña Aguaymanto Arañados Manzanas kiwi Pitahaya camu camu Aguaje Mefocotón Mango Melón
	Uvas Fresas Maracuya Guanabana Chirimoya Lúcuma Nispero Moras frambuesas Pera Ciruela Sandía Cereza Toronja Higo	Queso fresco Queso edam Queso suizo Queso mozzarella Queso ahumado Queso parmesano Yogurt natural Yogurt de vainilla Yogurt de fresa Yogurt de aguaymanto Leche Leche deslactosada Leche en polvo Leche condensada Mantequilla Crema de leche
<i>Carnes</i>	<i>Fideos</i>	<i>Condimentos</i>
Gallina Pollo Carne d'Chanco Churrasco Osobuco Cuadril Bistec Patita de Vaca Sangrecita Hotdog Jamonada Atún Huevos	Cabello d'angel Sernola Caracoles Fideos de colores Tallarín grueso Tallarín delgado	Pimienta Caldo de gallina Caldo Carne
<i>Papeles y Bolsas</i>	<i>Panes</i>	
Papel toalla Bolsa de papel Papel higiénico Papel kraft	Bolsas pequeñas Bolsas medianas Bolsas grandes Bolsas de basura	Pan Integral Pan blanco Petipanes tostadas
<i>Hongos</i>	<i>Frutos secos</i>	<i>Otros</i>
Champiñones Shitake Hongos de tallarines	Pistacho Albaricoques Nueces	Aguaymanto deshidratado Pecanas

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
<code>x.append(e)</code>	Agrega a la lista <code>x</code> el elemento <code>e</code>
<code>x.insert(i,e)</code>	Inserta <code>e</code> en la posición <code>i</code> de la lista <code>x</code>
<code>x.count(e)</code>	Conteo de instancias de <code>e</code> en la lista <code>x</code>
<code>x.remove(e)</code>	Elimina el primer elemento <code>e</code> de la lista <code>x</code>
<code>x.pop(i)</code>	Entrega el elemento en la posición <code>i</code> en la lista y lo elimina
<code>x.index(e)</code>	Entrega la posición del primer elemento <code>e</code> en <code>x</code>
<code>x.sort()</code>	Ordena <code>x</code> en forma creciente
<code>x.reverse()</code>	Invierte la lista <code>x</code>
<code>x.clear()</code>	Vaciar la lista <code>x</code>
<code>x.copy()</code>	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).