Listas en Python

1. Introducción a las listas

1.1 ¿Qué son las listas?

Una **lista** es una colección **ordenada** de elementos que pueden ser de cualquier tipo (números, texto, booleanos, incluso otras listas). Las listas son fundamentales en Python porque permiten almacenar múltiples datos en una única variable.

En la vida cotidiana usamos listas constantemente:

- "Mi lista de compras: leche, pan, huevos, queso"
- "Los estudiantes de la clase: Ana, Carlos, María, Juan"
- "Mis calificaciones: 85, 90, 78, 92"

En programación, las listas nos permiten:

- Almacenar múltiples datos relacionados.
- Acceder a datos por su posición.
- Modificar, agregar o eliminar elementos.
- Procesar datos de forma eficiente.

1.2 Crear una lista

La forma más simple de crear una lista es usando corchetes `[]`:

```
# Lista de números
numeros = [1, 2, 3, 4, 5]

# Lista de texto
frutas = ["manzana", "plátano", "naranja", "uva"]

# Lista vacía
lista_vacia = []

# Lista con diferentes tipos
datos_mixtos = [1, "hola", 3.14, True]

# Lista anidada (lista dentro de lista)
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

1.3 Ejemplo introductorio

```
# Crear una lista de estudiantes
estudiantes = ["Ana", "Carlos", "María", "Juan", "Pedro"]

# Acceder al primer elemento
print(estudiantes[^0])  # Ana

# Acceder al último elemento
print(estudiantes[-1])  # Pedro

# Acceder a un elemento en el medio
print(estudiantes[^2])  # María

# Modificar un elemento
estudiantes[^1] = "Miguel"
print(estudiantes)  # ['Ana', 'Miguel', 'María', 'Juan', 'Pedro']

# Agregar un nuevo estudiante
estudiantes.append("Laura")
print(estudiantes)  # ['Ana', 'Miguel', 'María', 'Juan', 'Pedro', 'Laura']

# Saber cuántos estudiantes hay
print(len(estudiantes))  # 6
```

2. Indexación y acceso a elementos

2.1 ¿Qué es el índice?

El **índice** es la posición de un elemento en la lista. **En Python, el índice comienza en 0** (no en 1).

Elemento: "Ana" "Carlos" "María" "Juan" "Pedro" Índice: 0 1 2 3 4 Índice neg: -5 -4 -3 -2 -1

2.2 Acceso por índice positivo

```
frutas = ["manzana", "plátano", "naranja", "uva", "fresa"]

# Acceder al primer elemento (índice 0)
print(frutas[^0])  # manzana

# Acceder al tercer elemento (índice 2)
print(frutas[^2])  # naranja

# Acceder al último elemento (índice 4)
print(frutas[^4])  # fresa
```

2.3 Acceso por índice negativo

Los índices negativos permiten acceder desde el final de la lista:

```
frutas = ["manzana", "plátano", "naranja", "uva", "fresa"]

# Acceder al último elemento (índice -1)
print(frutas[-1])  # fresa

# Acceder al penúltimo elemento (índice -2)
print(frutas[-2])  # uva

# Acceder al primer elemento (índice -5)
print(frutas[-5])  # manzana
```

2.4 Error: Índice fuera de rango

```
frutas = ["manzana", "plátano", "naranja"]

print(frutas[^5])  # ❌ ERROR: IndexError (solo hay 3 elementos, índices 0-2)
```

Para evitar errores, siempre verifica la longitud de la lista:

```
frutas = ["manzana", "plátano", "naranja"]

if len(frutas) > 5:
    print(frutas[^5])
else:
    print("El índice está fuera de rango")
```

3. Operaciones básicas con listas

3.1 Longitud: len()

Obtiene el número de elementos en una lista:

```
frutas = ["manzana", "plátano", "naranja"]
print(len(frutas))  # 3

numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(len(numeros))  # 10

lista_vacia = []
print(len(lista_vacia))  # 0
```

3.2 Agregar elementos: append()

Agrega un elemento al **final** de la lista:

```
frutas = ["manzana", "plátano"]

frutas.append("naranja")
print(frutas)  # ['manzana', 'plátano', 'naranja']

frutas.append("uva")
print(frutas)  # ['manzana', 'plátano', 'naranja', 'uva']
```

3.3 Insertar en posición específica: insert()

Agrega un elemento en una posición específica:

```
frutas = ["manzana", "naranja", "uva"]

# Insertar "plátano" en la posición 1
frutas.insert(1, "plátano")
print(frutas)  # ['manzana', 'plátano', 'naranja', 'uva']

# Insertar "fresa" al principio (posición 0)
frutas.insert(0, "fresa")
print(frutas)  # ['fresa', 'manzana', 'plátano', 'naranja', 'uva']
```

3.4 Eliminar por índice: pop()

Elimina un elemento en una posición específica y lo **retorna**:

```
frutas = ["manzana", "plátano", "naranja", "uva"]

# Eliminar el último elemento (índice -1 por defecto)
fruta_eliminada = frutas.pop()
print(fruta_eliminada)  # uva
print(frutas)  # ['manzana', 'plátano', 'naranja']

# Eliminar el elemento en la posición 1
fruta_eliminada = frutas.pop(1)
print(fruta_eliminada)  # plátano
print(frutas)  # ['manzana', 'naranja']
```

3.5 Eliminar por valor: remove()

Elimina la **primera ocurrencia** de un valor:

```
frutas = ["manzana", "plátano", "naranja", "plátano"]

# Eliminar "plátano" (elimina la primera ocurrencia)
frutas.remove("plátano")
print(frutas)  # ['manzana', 'naranja', 'plátano']

# Intentar eliminar algo que no existe
frutas.remove("sandía")  # ❌ ERROR: ValueError
```

3.6 Buscar el índice: index()

Encuentra el índice de la **primera ocurrencia** de un valor:

```
frutas = ["manzana", "plátano", "naranja", "plátano"]

# Encontrar el índice de "plátano"
indice = frutas.index("plátano")
print(indice)  # 1 (primera ocurrencia)

# Intentar encontrar algo que no existe
indice = frutas.index("sandía")  # ❌ ERROR: ValueError
```

3.7 Verificar si existe: in

Comprueba si un elemento está en la lista:

```
frutas = ["manzana", "plátano", "naranja"]

if "plátano" in frutas:
    print("El plátano está en la lista")

if "sandía" not in frutas:
    print("La sandía NO está en la lista")
```

3.8 Contar ocurrencias: count()

Cuenta cuántas veces aparece un valor:

```
numeros = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

print(numeros.count(1))  # 1
print(numeros.count(2))  # 2
print(numeros.count(3))  # 3
print(numeros.count(4))  # 4
print(numeros.count(5))  # 0 (no existe)
```

4. Modificación de listas

4.1 Cambiar elementos por índice

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

# Cambiar el segundo elemento (índice 1)
numeros[^1] = 25
print(numeros)  # [10, 25, 30, 40, 50]

# Cambiar el último elemento
numeros[-1] = 55
print(numeros)  # [10, 25, 30, 40, 55]
```

4.2 Cambiar múltiples elementos: slicing

El **slicing** permite acceder a un rango de elementos:

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Obtener elementos del índice 2 al 5 (5 no incluido)
print(numeros[2:5])  # [3, 4, 5]

# Obtener primeros 5 elementos
print(numeros[:5])  # [1, 2, 3, 4, 5]

# Obtener últimos 3 elementos
print(numeros[-3:])  # [8, 9, 10]

# Obtener todos excepto el último
print(numeros[:-1])  # [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Obtener con paso (cada 2 elementos)
print(numeros[::2])  # [1, 3, 5, 7, 9]

# Invertir la lista
print(numeros[::-1])  # [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

4.3 Asignar a un slice

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

# Reemplazar elementos del índice 1 al 3
numeros[1:3] = [20, 30]
print(numeros)  # [1, 20, 30, 4, 5]
```

```
# Reemplazar con diferente cantidad de elementos
numeros = [1, 2, 3, 4, 5]
numeros[1:4] = [20, 30]
print(numeros)  # [1, 20, 30, 5]
```

4.4 Limpiar una lista: clear()

```
frutas = ["manzana", "plátano", "naranja"]

frutas.clear()
print(frutas)  # []
print(len(frutas))  # 0
```

4.5 Copiar una lista

```
# Referencia (comparte la misma lista)
lista1 = [1, 2, 3]
lista2 = lista1
lista2[^0] = 10
print(lista1)  # [10, 2, 3] ← Cambió también lista1!

# Copia (crea una nueva lista)
lista1 = [1, 2, 3]
lista2 = lista1.copy()
lista2[^0] = 10
print(lista1)  # [1, 2, 3] ← No cambió
print(lista2)  # [10, 2, 3]
```

5. Métodos de ordenamiento y organización

5.1 Ordenar: sort()

Ordena la lista en el lugar (modifica la lista original):

```
numeros = [5, 2, 8, 1, 9]

# Ordenar de menor a mayor
numeros.sort()
print(numeros)  # [1, 2, 5, 8, 9]

# Ordenar de mayor a menor
numeros.sort(reverse=True)
print(numeros)  # [9, 8, 5, 2, 1]
```

```
# Ordenar strings alfabéticamente
frutas = ["naranja", "manzana", "uva", "plátano"]

frutas.sort()
print(frutas)  # ['manzana', 'naranja', 'plátano', 'uva']

# Ordenar alfabéticamente inverso
frutas.sort(reverse=True)
print(frutas)  # ['uva', 'plátano', 'naranja', 'manzana']
```

5.2 Ordenar sin modificar: sorted()

Retorna una **nueva lista ordenada** sin modificar la original:

```
numeros = [5, 2, 8, 1, 9]

# Crear una copia ordenada
numeros_ordenados = sorted(numeros)
print(numeros_ordenados)  # [1, 2, 5, 8, 9]
print(numeros)  # [5, 2, 8, 1, 9] ← No cambió
```

5.3 Invertir: reverse()

Invierte el orden de la lista:

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

numeros.reverse()
print(numeros)  # [5, 4, 3, 2, 1]
```

6. Operaciones avanzadas

6.1 Concatenar listas: +

```
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]

# Concatenar (crea una nueva lista)
lista3 = lista1 + lista2
print(lista3)  # [1, 2, 3, 4, 5, 6]
```

```
print(lista1)  # [1, 2, 3] ← No cambió
print(lista2)  # [4, 5, 6] ← No cambió
```

6.2 Repetir lista: *

```
numeros = [1, 2, 3]

# Repetir la lista 3 veces
numeros_repetidos = numeros * 3
print(numeros_repetidos)  # [1, 2, 3, 1, 2, 3, 1, 2, 3]

# Crear lista de ceros
ceros = [^0] * 5
print(ceros)  # [0, 0, 0, 0, 0]
```

6.3 Extender: extend()

Agrega múltiples elementos de otra lista:

```
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]

# extend() agrega todos los elementos
lista1.extend(lista2)
print(lista1)  # [1, 2, 3, 4, 5, 6]

# Diferencia con append()
lista1 = [1, 2, 3]
lista1.append(lista2)  # Agrega la lista completa como un elemento
print(lista1)  # [1, 2, 3, [4, 5, 6]]
```

6.4 Máximo y mínimo

```
numeros = [5, 2, 8, 1, 9, 3]

print(max(numeros))  # 9
print(min(numeros))  # 1

# También funciona con strings
frutas = ["naranja", "manzana", "uva", "plátano"]
print(max(frutas))  # uva (orden alfabético)
print(min(frutas))  # manzana
```

6.5 Suma: sum()

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

print(sum(numeros))  # 15

# Con valor inicial
print(sum(numeros, 10))  # 25 (15 + 10)
```

7. Iteración sobre listas

7.1 Bucle for simple

```
frutas = ["manzana", "plátano", "naranja"]

for fruta in frutas:
    print(fruta)
```

Salida:

```
manzana
plátano
naranja
```

7.2 Con enumerate() (índice + elemento)

```
frutas = ["manzana", "plátano", "naranja"]

for indice, fruta in enumerate(frutas):
    print(f"{indice}: {fruta}")
```

Salida:

```
0: manzana
1: plátano
2: naranja
```

7.3 Con índice manual

```
frutas = ["manzana", "plátano", "naranja"]
```

```
for i in range(len(frutas)):
    print(f"{i}: {frutas[i]}")
```

7.4 Iterar múltiples listas: zip()

```
nombres = ["Ana", "Carlos", "María"]
edades = [20, 25, 22]

for nombre, edad in zip(nombres, edades):
    print(f"{nombre} tiene {edad} años")
```

Salida:

```
Ana tiene 20 años
Carlos tiene 25 años
María tiene 22 años
```

8. Listas anidadas

8.1 Crear listas anidadas

```
# Matriz 3x3
matriz = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Acceder a elementos
print(matriz[^0])      # [1, 2, 3]
print(matriz[^0][^0])   # 1
print(matriz[^1][^2])   # 6
print(matriz[^2][^1])   # 8
```

8.2 Modificar listas anidadas

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

# Cambiar el elemento en posición [^1][^2]
```

```
matriz[^1][^2] = 60
print(matriz)  # [[1, 2, 3], [4, 5, 60], [7, 8, 9]]

# Agregar fila
matriz.append([10, 11, 12])
print(matriz)  # [[1, 2, 3], [4, 5, 60], [7, 8, 9], [10, 11, 12]]
```

8.3 Iterar sobre listas anidadas

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

# Iterar filas y columnas
for fila in matriz:
    for elemento in fila:
        print(elemento, end=" ")
    print()
```

Salida:

```
1 2 3
4 5 6
7 8 9
```

9. Comprensión de listas (list comprehension)

9.1 ¿Qué es list comprehension?

Es una forma **compacta y elegante** de crear listas a partir de otras listas. La sintaxis es:

```
[expresion for elemento in iterable if condicion]
```

9.2 Ejemplo simple: Crear números al cuadrado

Sin list comprehension:

```
numeros = [1, 2, 3, 4, 5]
cuadrados = []

for numero in numeros:
```

```
    cuadrados.append(numero ** 2)

print(cuadrados)  # [1, 4, 9, 16, 25]
```

Con list comprehension:

```
numeros = [1, 2, 3, 4, 5]
cuadrados = [numero ** 2 for numero in numeros]
print(cuadrados)  # [1, 4, 9, 16, 25]
```

9.3 Ejemplos adicionales

```
# Convertir a mayúsculas
palabras = ["hola", "mundo", "python"]
mayusculas = [palabra.upper() for palabra in palabras]
print(mayusculas)  # ['HOLA', 'MUNDO', 'PYTHON']

# Filtrar números pares
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
pares = [numero for numero in numeros if numero % 2 == 0]
print(pares)  # [2, 4, 6, 8, 10]

# Crear tuplas (número, cuadrado)
numeros = [1, 2, 3, 4, 5]
pares_cuadrados = [(n, n**2) for n in numeros if n % 2 == 0]
print(pares_cuadrados)  # [(2, 4), (4, 16)]
```

10. Ejercicio guiado: Gestor de calificaciones

Crearemos un programa que:

1. Almacene calificaciones de estudiantes.
2. Calcule el promedio.
3. Encuentre la máxima y mínima.
4. Muestre las calificaciones ordenadas.
5. Cuente aprobados y reprobados.

10.1 Código completo

```
# Gestor de calificaciones

print("=" * 50)
print("GESTOR DE CALIFICACIONES".center(50))
```

```
print("=" * 50)
print()

calificaciones = []

# Paso 1: Capturar calificaciones
while True:
    try:
        nota = float(input("Ingresa una calificación (0-20) o 'salir': "))

        if nota < 0 or nota > 20:
            print("❌ La calificación debe estar entre 0 y 20")
            continue

        calificaciones.append(nota)
        print(f"✅ Calificación {nota} agregada")

    except ValueError:
        print("✅ Captura de calificaciones terminada")
        break

    print()

# Paso 2: Validar que hay calificaciones
if len(calificaciones) == 0:
    print("❌ No hay calificaciones ingresadas")
else:
    print()
    print("-" * 50)
    print("ANÁLISIS DE CALIFICACIONES")
    print("-" * 50)

    # Mostrar todas las calificaciones
    print(f"Calificaciones ingresadas: {calificaciones}")

    # Calcular estadísticas
    promedio = sum(calificaciones) / len(calificaciones)
    maxima = max(calificaciones)
    minima = min(calificaciones)

    print(f"Promedio: {promedio:.2f}")
    print(f"Máxima: {maxima}")
    print(f"Mínima: {minima}")

    # Contar aprobados y reprobados
    aprobados = sum(1 for nota in calificaciones if nota >= 10)
    reprobados = len(calificaciones) - aprobados

    print(f"Aprobados: {aprobados}")
    print(f"Reprobados: {reprobados}")

    # Mostrar ordenadas
    calificaciones_ordenadas = sorted(calificaciones)
```

```
    print(f"Ordenadas de menor a mayor: {calificaciones_ordenadas}")

    print("-" * 50)
```

10.2 Salida del programa

```
==================================================
         GESTOR DE CALIFICACIONES
==================================================

Ingresa una calificación (0-20) o 'salir': 15
✅ Calificación 15.0 agregada

Ingresa una calificación (0-20) o 'salir': 18
✅ Calificación 18.0 agregada

Ingresa una calificación (0-20) o 'salir': 12
✅ Calificación 12.0 agregada

Ingresa una calificación (0-20) o 'salir': 9
✅ Calificación 9.0 agregada

Ingresa una calificación (0-20) o 'salir': salir
✅ Captura de calificaciones terminada

--------------------------------------------------
ANÁLISIS DE CALIFICACIONES
--------------------------------------------------
Calificaciones ingresadas: [15.0, 18.0, 12.0, 9.0]
Promedio: 13.50
Máxima: 18.0
Mínima: 9.0
Aprobados: 3
Reprobados: 1
Ordenadas de menor a mayor: [9.0, 12.0, 15.0, 18.0]
--------------------------------------------------
```

11. Ejercicio de evaluación

11.1 Enunciado

Crea un programa que **gestione un inventario de productos** con las siguientes funcionalidades:

1. **Ver productos**: Mostrar todos los productos con nombre, precio y cantidad.
2. **Agregar producto**: Permitir agregar un nuevo producto.
3. **Eliminar producto**: Eliminar un producto por su nombre.

4. **Actualizar cantidad**: Cambiar la cantidad en stock de un producto.
5. **Buscar producto**: Buscar un producto por nombre.
6. **Valor total del inventario**: Calcular el valor total (precio × cantidad de cada producto).
7. **Productos con bajo stock**: Mostrar productos con cantidad < 5.
8. **Salir**: Terminar el programa.

Requisitos:

- Usa una lista de diccionarios para almacenar productos: `[{"nombre": "X", "precio": Y, "cantidad": Z}, ...]`
- Usa operaciones de lista (append, remove, etc.).
- Valida las entradas del usuario.
- Usa bucles y condicionales.

Ejemplo de ejecución:

```
=== GESTOR DE INVENTARIO ===

1. Ver productos
2. Agregar producto
3. Eliminar producto
4. Actualizar cantidad
5. Buscar producto
6. Valor total del inventario
7. Productos con bajo stock
8. Salir

Opción: 2
Nombre del producto: Laptop
Precio: 1000
Cantidad: 5
✅ Producto agregado

Opción: 1
PRODUCTOS EN INVENTARIO
1. Laptop - $1000 - Cantidad: 5

Opción: 6
Valor total del inventario: $5000

Opción: 8
¡Hasta luego!
```

11.2 Pistas para resolver

- Usa `[{"nombre": ..., "precio": ..., "cantidad": ...}, ...]`
- Para buscar: usa `for` con condición `if producto["nombre"] == nombre`

- Para eliminar: usa `productos.remove(producto_encontrado)`
- Para actualizar: accede a `producto["cantidad"] = nueva_cantidad`
- Para valor total: suma `precio × cantidad` de todos los productos

12. Solución del ejercicio de evaluación

```
# Gestor de inventario

print("=" * 50)
print("GESTOR DE INVENTARIO".center(50))
print("=" * 50)
print()

productos = []

while True:
    print("-" * 50)
    print("MENÚ PRINCIPAL")
    print("-" * 50)
    print("1. Ver productos")
    print("2. Agregar producto")
    print("3. Eliminar producto")
    print("4. Actualizar cantidad")
    print("5. Buscar producto")
    print("6. Valor total del inventario")
    print("7. Productos con bajo stock")
    print("8. Salir")
    print("-" * 50)

    try:
        opcion = int(input("Elige una opción (1-8): "))
    except ValueError:
        print("❌ Error: Debes ingresar un número")
        print()
        continue

    # OPCIÓN 1: Ver productos
    if opcion == 1:
        print()
        if len(productos) == 0:
            print("📭 No hay productos en el inventario")
        else:
            print("📋 PRODUCTOS EN INVENTARIO")
            for indice, producto in enumerate(productos, 1):
                nombre = producto["nombre"]
                precio = producto["precio"]
                cantidad = producto["cantidad"]
                print(f"{indice}. {nombre} - ${precio} - Cantidad: {cantidad}")
```

```
        print()

    # OPCIÓN 2: Agregar producto
    elif opcion == 2:
        print()
        nombre = input("Nombre del producto: ")

        # Verificar si ya existe
        existe = any(p["nombre"].lower() == nombre.lower() for p in productos)
        if existe:
            print("❌ El producto ya existe en el inventario")
        else:
            try:
                precio = float(input("Precio: $"))
                cantidad = int(input("Cantidad: "))

                if precio < 0 or cantidad < 0:
                    print("❌ Precio y cantidad deben ser positivos")
                else:
                    productos.append({"nombre": nombre, "precio": precio, "cantidad": cantidad
                    print(f"✅ Producto '{nombre}' agregado")
            except ValueError:
                print("❌ Error: Ingresa valores válidos")
        print()

    # OPCIÓN 3: Eliminar producto
    elif opcion == 3:
        print()
        nombre = input("Nombre del producto a eliminar: ")

        producto_encontrado = None
        for p in productos:
            if p["nombre"].lower() == nombre.lower():
                producto_encontrado = p
                break

        if producto_encontrado:
            productos.remove(producto_encontrado)
            print(f"✅ Producto '{nombre}' eliminado")
        else:
            print(f"❌ Producto '{nombre}' no encontrado")
        print()

    # OPCIÓN 4: Actualizar cantidad
    elif opcion == 4:
        print()
        nombre = input("Nombre del producto: ")

        producto_encontrado = None
        for p in productos:
            if p["nombre"].lower() == nombre.lower():
                producto_encontrado = p
                break
```

```
        if producto_encontrado:
            try:
                nueva_cantidad = int(input("Nueva cantidad: "))
                if nueva_cantidad < 0:
                    print("❌ La cantidad debe ser positiva")
                else:
                    producto_encontrado["cantidad"] = nueva_cantidad
                    print(f"✅ Cantidad actualizada a {nueva_cantidad}")
            except ValueError:
                print("❌ Error: Ingresa un número válido")
        else:
            print(f"❌ Producto '{nombre}' no encontrado")
        print()

    # OPCIÓN 5: Buscar producto
    elif opcion == 5:
        print()
        nombre = input("Nombre del producto a buscar: ")

        producto_encontrado = None
        for p in productos:
            if p["nombre"].lower() == nombre.lower():
                producto_encontrado = p
                break

        if producto_encontrado:
            print(f"✅ Producto encontrado:")
            print(f"   Nombre: {producto_encontrado['nombre']}")
            print(f"   Precio: ${producto_encontrado['precio']}")
            print(f"   Cantidad: {producto_encontrado['cantidad']}")
        else:
            print(f"❌ Producto '{nombre}' no encontrado")
        print()

    # OPCIÓN 6: Valor total del inventario
    elif opcion == 6:
        print()
        if len(productos) == 0:
            print("📭 No hay productos para calcular")
        else:
            valor_total = sum(p["precio"] * p["cantidad"] for p in productos)
            print(f"💰 Valor total del inventario: ${valor_total:.2f}")
        print()

    # OPCIÓN 7: Productos con bajo stock
    elif opcion == 7:
        print()
        bajo_stock = [p for p in productos if p["cantidad"] < 5]

        if len(bajo_stock) == 0:
            print("✅ No hay productos con bajo stock")
        else:
```

```
            print("⚠️ PRODUCTOS CON BAJO STOCK (< 5)")
            for producto in bajo_stock:
                print(f"   {producto['nombre']} - Cantidad: {producto['cantidad']}")
        print()

    # OPCIÓN 8: Salir
    elif opcion == 8:
        print()
        print("✅ ¡Hasta luego! Gracias por usar el gestor")
        break

    # Opción no válida
    else:
        print()
        print("❌ Opción no válida")
        print()
```

13. Ejercicios adicionales para practicar

13.1 Ejercicio: Filtrador de números

Crea un programa que:

- Pida una lista de números.
- Filtre números pares.
- Filtre números mayores a un valor X.
- Muestre el promedio de números impares.

13.2 Ejercicio: Organizador de palabras

Crea un programa que:

- Pida una frase.
- Divida en palabras.
- Ordene alfabéticamente.
- Cuente palabras únicas.
- Encuentre la palabra más larga.

13.3 Ejercicio: Matriz de notas

Crea un programa que:

- Pida una matriz de notas (3 estudiantes, 4 calificaciones cada uno).
- Calcule promedio por estudiante.

- Calcule promedio por calificación.
- Encuentre la nota máxima y mínima general.

13.4 Ejercicio: Duplicados y únicos

Crea un programa que:

- Pida una lista de números.
- Encuentre elementos duplicados.
- Encuentre elementos únicos.
- Cuente la frecuencia de cada número.

13.5 Ejercicio: Carrito de compras

Crea un programa que:

- Tenga una lista de productos con precio.
- Permita al usuario agregar productos al carrito.
- Calcule el total.
- Aplique descuentos (ej: 10% si total > $100).
- Muestre el recibo.

14. Tabla resumen de métodos y funciones

Método/Función	Descripción	Ejemplo
`append()`	Agregar al final	`lista.append(5)`
`insert()`	Insertar en posición	`lista.insert(0, 5)`
`pop()`	Eliminar por índice	`elemento = lista.pop(0)`
`remove()`	Eliminar por valor	`lista.remove(5)`
`index()`	Encontrar índice	`i = lista.index(5)`
`count()`	Contar ocurrencias	`n = lista.count(5)`
`sort()`	Ordenar la lista	`lista.sort()`
`reverse()`	Invertir la lista	`lista.reverse()`
`clear()`	Limpiar la lista	`lista.clear()`
`copy()`	Copiar la lista	`copia = lista.copy()`

Método/Función	Descripción	Ejemplo
`extend()`	Extender con otra lista	`lista.extend([4, 5])`
`len()`	Longitud	`n = len(lista)`
`sum()`	Suma de elementos	`total = sum(lista)`
`max()`	Máximo	`máximo = max(lista)`
`min()`	Mínimo	`mínimo = min(lista)`
`sorted()`	Ordenada nueva lista	`nueva = sorted(lista)`
`enumerate()`	Índice + elemento	`for i, v in enumerate(lista):`
`zip()`	Múltiples listas	`for a, b in zip(l1, l2):`

15. Errores comunes con listas

Error 1: Confundir append() con extend()

```
lista = [1, 2, 3]

# ❌ Agrega la lista completa como un elemento
lista.append([4, 5, 6])
print(lista)  # [1, 2, 3, [4, 5, 6]]

# ✅ Extiende con los elementos individuales
lista = [1, 2, 3]
lista.extend([4, 5, 6])
print(lista)  # [1, 2, 3, 4, 5, 6]
```

Error 2: Modificar lista mientras se itera

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

# ❌ Evita esto
for numero in numeros:
    if numero == 3:
        numeros.remove(3)

# ✅ Itera sobre copia
for numero in numeros[:]:
    if numero == 3:
        numeros.remove(3)
```

Error 3: Referencia vs copia

```
lista1 = [1, 2, 3]

# ❌ Referencia (comparte contenido)
lista2 = lista1
lista2[^0] = 10
print(lista1)  # [10, 2, 3]

# ✅ Copia
lista2 = lista1.copy()
lista2[^0] = 10
print(lista1)  # [1, 2, 3]
```

Error 4: Índice fuera de rango

```
lista = [1, 2, 3]

# ❌ Error
print(lista[^5])  # IndexError

# ✅ Verificar primero
if len(lista) > 5:
    print(lista[^5])
```

Error 5: Olvidar que range() no incluye el final

```
# ❌ Solo hasta 4
for i in range(5):
    print(i)  # 0, 1, 2, 3, 4

# ✅ Si quieres del 1 al 5
for i in range(1, 6):
    print(i)  # 1, 2, 3, 4, 5
```

16. Preguntas frecuentes del PDF 4

P: ¿Cuál es la diferencia entre append() y insert()?

R: `append()` agrega al final. `insert()` agrega en una posición específica.

P: ¿Puedo tener listas con diferentes tipos?

R: Sí, Python permite listas mixtas: `[1, "hola", 3.14, True]`.

P: ¿Qué es slicing?

R: Es una forma de obtener un rango de elementos: `lista[inicio:fin:paso]`.

P: ¿Cómo hago una copia verdadera de una lista?

R: Usa `.copy()` o `lista[:]` para copias superficiales.

P: ¿Cuándo uso list comprehension?

R: Cuando necesitas crear una nueva lista basada en transformaciones o filtros de otra lista.

P: ¿Qué es una lista anidada?

R: Una lista que contiene otras listas, útil para representar matrices o estructuras complejas.

17. Casos de uso del mundo real

17.1 Procesamiento de datos

```
# Obtener números mayores a 50
datos = [10, 55, 30, 75, 20, 60]
mayores = [n for n in datos if n > 50]
print(mayores)  # [55, 75, 60]
```

17.2 Manipulación de texto

```
# Convertir palabras a mayúsculas
palabras = ["python", "java", "javascript"]
mayusculas = [p.upper() for p in palabras]
print(mayusculas)  # ['PYTHON', 'JAVA', 'JAVASCRIPT']
```

17.3 Análisis de datos

```
# Calcular estadísticas
calificaciones = [85, 90, 78, 92, 88, 76, 95]

promedio = sum(calificaciones) / len(calificaciones)
mejor = max(calificaciones)
peor = min(calificaciones)
```

Conclusión

Felicidades por completar el PDF 4. Ahora entiendes:

- Cómo crear y acceder a elementos en listas.
- Operaciones básicas (append, remove, insert, pop).
- Métodos de búsqueda y ordenamiento.
- Slicing para obtener rangos.
- Iteración con for, enumerate() y zip().
- Listas anidadas y estructuras complejas.
- List comprehension para código conciso.
- Casos de uso prácticos del mundo real.

En el siguiente PDF (Diccionarios y Tuplas), aprenderás nuevas estructuras de datos que complementan las listas: **diccionarios** para almacenar datos asociados clave-valor, y **tuplas** para datos inmutables. Estas estructuras son fundamentales para programas más complejos.