Las Listas en Python

1. Introducción a las Listas

Las listas en Python son estructuras de datos ordenadas, mutables y que pueden contener elementos de diferentes tipos. Se definen usando corchetes [] y los elementos se separan por comas.

Características principales:

- **Ordenadas**: Mantienen el orden de inserción
- **Mutables**: Se pueden modificar después de su creación
- **Indexables**: Se accede a los elementos por su posición
- **Heterogéneas**: Pueden contener diferentes tipos de datos
- **Dinámicas**: Pueden cambiar de tamaño

2. Creación y Acceso a Listas

Creación de listas:

```
# Lista vacía
lista_vacia = []
lista_vacia2 = list()

# Lista con elementos
numeros = [1, 2, 3, 4, 5]
mixta = [1, "hola", 3.14, True, [1, 2]]

# Lista desde un rango
lista_rango = list(range(10))
```

Acceso a elementos:

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

# Acceso por índice positivo
print(frutas[0])    # "manzana"
print(frutas[2])    # "naranja"

# Acceso por índice negativo
print(frutas[-1])   # "plátano"
print(frutas[-2])   # "naranja"
```

```
# Slicing (rebanado)
print(frutas[1:3])  # ["pera", "naranja"]
print(frutas[:2])   # ["manzana", "pera"]
print(frutas[2:])   # ["naranja", "plátano"]
print(frutas[::2])  # ["manzana", "naranja"]
```

Ejercicios - Creación y Acceso:

Ejercicio 1 (Básico): Crear una lista con los días de la semana y mostrar el primer y último día.

```
dias = ["lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"]
print(f"Primer día: {dias[0]}")
print(f"Último día: {dias[-1]}")
```

Ejercicio 2 (Intermedio): Dada una lista de números, extraer los elementos en posiciones pares y crear una nueva lista con ellos.

```
numeros = [10, 20, 30, 40, 50, 60, 70, 80, 90]
pares = numeros[::2]  # [10, 30, 50, 70, 90]
print(f"Elementos en posiciones pares: {pares}")

# Alternativa con bucle
pares2 = []
for i in range(0, len(numeros), 2):
    pares2.append(numeros[i])
```

Ejercicio 3 (Avanzado): Implementar una función que rote una lista n posiciones a la derecha.

```
def rotar_derecha(lista, n):
    """Rota una lista n posiciones a la derecha"""
    if not lista:
        return lista
    n = n % len(lista)  # Manejar rotaciones mayores al tamaño
    return lista[-n:] + lista[:-n]

# Prueba
numeros = [1, 2, 3, 4, 5]
print(rotar_derecha(numeros, 2))  # [4, 5, 1, 2, 3]
print(rotar_derecha(numeros, 7))  # [4, 5, 1, 2, 3] (7 % 5 = 2)
```

3. Métodos Principales de las Listas

Métodos de modificación:

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

# append() - Añade un elemento al final
lista.append(4)  # [1, 2, 3, 4]

# insert() - Inserta un elemento en una posición
lista.insert(1, 10)  # [1, 10, 2, 3, 4]

# extend() - Añade múltiples elementos
lista.extend([5, 6])  # [1, 10, 2, 3, 4, 5, 6]

# remove() - Elimina la primera ocurrencia de un valor
lista.remove(10)  # [1, 2, 3, 4, 5, 6]

# pop() - Elimina y retorna un elemento por índice
elemento = lista.pop(2)  # elemento = 3, lista = [1, 2, 4, 5, 6]

# clear() - Vacía la lista
lista.clear()  # []
```

Métodos de búsqueda y orden:

```
lista = [3, 1, 4, 1, 5, 9, 2, 6]

# index() - Retorna el índice de la primera ocurrencia
indice = lista.index(1)  # 1

# count() - Cuenta las ocurrencias de un elemento
cantidad = lista.count(1)  # 2

# sort() - Ordena la lista in-place
lista.sort()  # [1, 1, 2, 3, 4, 5, 6, 9]
lista.sort(reverse=True)  # [9, 6, 5, 4, 3, 2, 1, 1]

# reverse() - Invierte la lista
lista.reverse()  # Invierte el orden actual
```

Solo en listas homogeneas

```
# copy() - Crea una copia superficial
copia = lista.copy()
```

4. Operaciones con Listas

Operaciones básicas:

```
# Concatenación
lista1 = [1, 2, 3]
lista2 = [4, 5, 6]
concatenada = lista1 + lista2  # [1, 2, 3, 4, 5, 6]

# Repetición
repetida = [0] * 5  # [0, 0, 0, 0, 0]

# Pertenencia
print(3 in lista1)  # True
print(7 not in lista1)  # True

# Longitud
longitud = len(lista1)  # 3

# Máximo y mínimo
maximo = max([3, 1, 4, 1, 5])  # 5
minimo = min([3, 1, 4, 1, 5])  # 1

# Suma
suma = sum([1, 2, 3, 4, 5])  # 15
```

5. Listas por Comprensión

Las listas por comprensión son una forma concisa y pythónica de crear listas.

Sintaxis básica:

```
# Sintaxis: [expresión for elemento in iterable if condición]

# Básica
```

```
cuadrados = [x**2 for x in range(10)]
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

# Con condición
pares = [x for x in range(20) if x % 2 == 0]
# [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

# Con operación condicional
resultado = [x if x % 2 == 0 else -x for x in range(10)]
# [0, -1, 2, -3, 4, -5, 6, -7, 8, -9]

# Anidadas
matriz = [[i*j for j in range(3)] for i in range(3)]
# [[0, 0, 0], [0, 1, 2], [0, 2, 4]]
```

Resumen

Las listas son una de las estructuras de datos más versátiles y utilizadas en Python. Dominar su uso, desde las operaciones básicas hasta las técnicas avanzadas como las comprensiones de lista y los algoritmos de ordenamiento, es fundamental para cualquier programador Python.

Conceptos clave a recordar:

- Las listas son mutables y ordenadas
- Los métodos principales permiten modificar, buscar y ordenar elementos
- Las comprensiones de lista ofrecen una sintaxis concisa y eficiente
- Las listas pueden contener cualquier tipo de dato, incluyendo otras listas
- El slicing es una herramienta poderosa para trabajar con sublistas

La práctica constante con ejercicios de complejidad creciente es la mejor manera de dominar el uso de las listas en Python.