CURSO DE PYTHON - PROGRAMACIÓN BÁSICA

==========

TEMA: Listas y Estructuras de Datos en Python

1. INTRODUCCIÓN A LAS LISTAS

Las listas en Python son estructuras de datos mutables que pueden contener elementos de diferentes tipos. Se definen usando corchetes [].

Ejemplo básico:

```
mi_lista = [1, 2, 3, 4, 5]
frutas = ["manzana", "banana", "naranja"]
mixta = [1, "texto", 3.14, True]
```

2. OPERACIONES BÁSICAS

- Acceso por índice: lista[0] devuelve el primer elemento
- Slicing: lista[1:3] devuelve elementos desde índice 1 hasta 3 (sin incluir 3)
- Append: lista.append(elemento) añade al final
- Insert: lista.insert(indice, elemento) inserta en posición específica
- Remove: lista.remove(elemento) elimina la primera ocurrencia
- Pop: lista.pop() elimina y devuelve el último elemento

Ejemplos:

```
numeros = [1, 2, 3, 4, 5]
numeros.append(6) # [1, 2, 3, 4, 5, 6]
numeros.insert(0, 0) # [0, 1, 2, 3, 4, 5, 6]
numeros.remove(3) # [0, 1, 2, 4, 5, 6]
```

3. LIST COMPREHENSIONS

Las list comprehensions son una forma elegante y concisa de crear listas.

Sintaxis básica:

```
nueva_lista = [expresion for item in iterable if condicion]
```

Ejemplos:

```
# Cuadrados de números del 1 al 10
cuadrados = [x**2 for x in range(1, 11)]
# Números pares
pares = [x for x in range(20) if x % 2 == 0]
# Transformar strings
mayusculas = [s.upper() for s in ["hola", "mundo"]]
```

4. MÉTODOS IMPORTANTES

- len(lista): devuelve la longitud
- sorted(lista): devuelve una copia ordenada
- lista.sort(): ordena in-place
- lista.reverse(): invierte el orden
- lista.count(elemento): cuenta ocurrencias
- lista.index(elemento): devuelve índice de primera ocurrencia

5. EJERCICIOS PRÁCTICOS

Ejercicio 1: Crear una función que tome una lista y devuelva solo los números positivos.

Ejercicio 2: Implementar una función que elimine duplicados de una lista.

Ejercicio 3: Escribir una función que ordene una lista de tuplas por el segundo elemento.

6. CASOS AVANZADOS

- Listas anidadas: matrices y estructuras complejas
- Iteración con enumerate(): acceso a índice y valor simultáneamente
- zip(): combinar múltiples listas
- map() y filter(): programación funcional con listas

CONCEPTOS CLAVE PARA RECORDAR:

- Las listas son mutables (se pueden modificar)

- Los índices empiezan en 0
- Los índices negativos cuentan desde el final (-1 es el último)
- Las list comprehensions son más eficientes que loops tradicionales
- Siempre verificar si la lista está vacía antes de acceder elementos