Investigación Python

David Sánchez Acevedo

🐍 Introducción a Listas y Cadenas en Python

Una guía breve y clara para principiantes/intermedios sobre cómo trabajar con listas y cadenas en Python, con ejemplos y explicaciones útiles.

📌 Enlace de Listas

🔗 Concatenar Listas

```
python
lista1 = [1, 2, 3]
lista2 = [4, 5]
combinada = lista1 + lista2
print(combinada)  # [1, 2, 3, 4, 5]
```

- `+`: une listas creando una nueva.
- `extend()`: agrega elementos **modificando** la lista original.
-

```
lista1.extend(lista2)
```

🎯 Referencias (aliasing)

```
python
a = [1, 2, 3]
b = a
b.append(4)
```

```
print(a)  # [1, 2, 3, 4]
```

> ⚠️ a y b apuntan a la misma lista. Usa .copy() para evitarlo.

```
python
c = a.copy()
```

🛠️ Agregar, Eliminar y Editar Elementos

➕ Agregar

```
python
frutas = ["manzana", "banana"]
frutas.append("kiwi")
frutas.insert(1, "pera")
```

- `append(x)`: agrega al final
- `insert(i, x)`: inserta en posición

➖ Eliminar

```
python
numeros = [10, 20, 30]
numeros.remove(20)
valor = numeros.pop(1)
del numeros[0]
numeros.clear()
```

- `remove(x)`: elimina el valor
- `pop(i)`: elimina y devuelve por índice
- `del`: elimina por índice o rango

- `clear()`: vacía la lista

📝 Editar

```
python
datos = [5, 10, 15]
datos[1] = 20
```

- También se puede editar usando slicing:

```
datos[0:2] = [100, 200]
```

🧱 Listas Enlazadas (Linked Lists)

Python no incluye listas enlazadas por defecto, pero se pueden implementar con clases.

🔧 Definición básica

```
python
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.sig = None
```

➕ Crear y enlazar nodos

```
python
n1 = Nodo(5)
n2 = Nodo(10)
n3 = Nodo(15)
n1.sig = n2
n2.sig = n3
```

🔁 Recorrer la lista

```
python
actual = n1
while actual:
    print(actual.dato)
    actual = actual.sig
```

🧠 **Ventaja**: inserciones rápidas en cualquier parte.

⚠️ **Desventaja**: acceso por índice es lento.

✂️ Manipulación de Cadenas

🧪 Acceso y Slicing

```
python
texto = "Python"
print(texto[0])     # "P"
print(texto[1:4])   # "yth"
print(texto[::-1])  # "nohtyP"
```

🔨 Métodos comunes

- `.lower()`, `.upper()`, `.capitalize()`
- `.strip()`, `.lstrip()`, `.rstrip()`
- `.replace("a", "b")`
- `.find("sub")`, `"sub" in texto`
- `.split(",")`, `",".join(lista)`

```
python
mensaje = "  Hola Mundo "
print(mensaje.strip().lower())  # "hola mundo"
```

🧮 Funciones Útiles con Listas

🔁 `len()`

Devuelve la cantidad de elementos en una lista:

```
python
mi_lista = [1, 2, 3]
print(len(mi_lista))  # 3
```

🔄 `sorted()` vs `.sort()`

```
nums = [3, 1, 4, 2]
ordenada = sorted(nums)  # Crea nueva lista ordenada
nums.sort()              # Ordena la original
```

> sorted() ➜ NO modifica la original
>
> `.sort()` ➜ SÍ modifica la lista

🔁 Recorrer listas con `for`

```
python
frutas = ["manzana", "pera", "uva"]
for fruta in frutas:
    print(fruta)
```

🔢 `enumerate()`

Te permite obtener el índice y el valor al mismo tiempo:

```
python
for i, fruta in enumerate(frutas):
```

```
    print(i, fruta)
```

📋 Comprensión de listas (List Comprehension)

Forma compacta de crear listas:

```
cuadrados = [x**2 for x in range(5)]
print(cuadrados)  # [0, 1, 4, 9, 16]
```

> condiciones:

```
pares = [x for x in range(10) if x % 2 == 0]
```