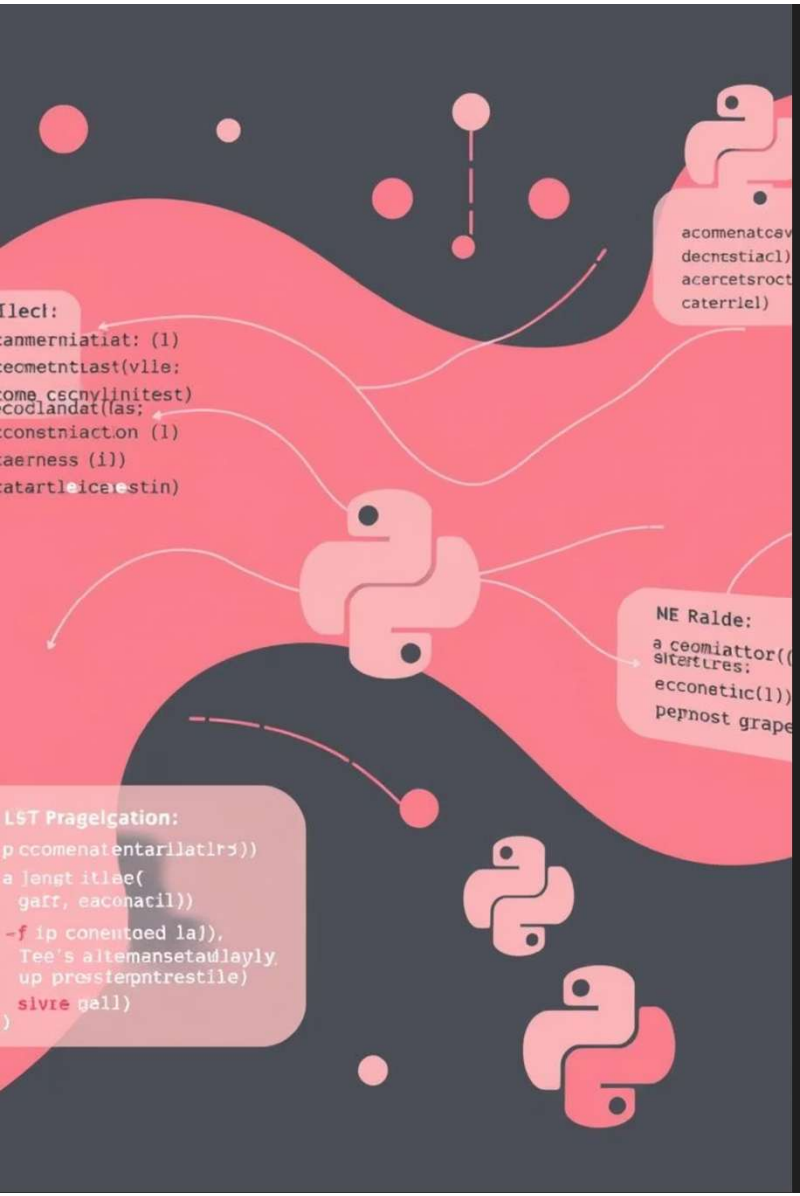


Introducción a las Estructuras de Datos

Python ofrece una variedad de estructuras fundamentales para almacenar y organizar colecciones de datos.

- Listas
- Tuplas
- Conjuntos
- Diccionarios

por conectiva oficial

Listas

1

Colección Ordenada y Mutable

Las listas son colecciones de elementos que mantienen un orden y pueden modificarse. Se definen utilizando corchetes `[]`.

2

Elementos Flexibles

Pueden contener elementos repetidos y de distintos tipos de datos (e.g., números, texto, booleanos).

3

Acceso por Índice

Cada elemento tiene un índice numérico que comienza en 0, lo que permite acceder a ellos fácilmente: `lista[indice]`.

4

Ejemplo Práctico

mi_lista = [1, “Hola", 5,123, True]

Funciones Principales de Listas

Modificación de Elementos

- **append(x)**: añade un elemento al final.
- **extend(iterable)**: añade todos los elementos de un iterable.
- **insert(i, x)**: inserta un elemento **x** en la posición **i**.

Eliminación de Elementos

- **remove(x)**: elimina la primera aparición del valor
- **pop([i])**: elimina y devuelve el elemento en la posición **i**. Si **i** no se especifica, elimina el último.

Ordenación y Reversión

- **sort()**: ordena los elementos de la lista en su lugar.
- **reverse()**: invierte el orden de los elementos de la lista en su lugar.

Tuplas

Ordenadas e Inmutables

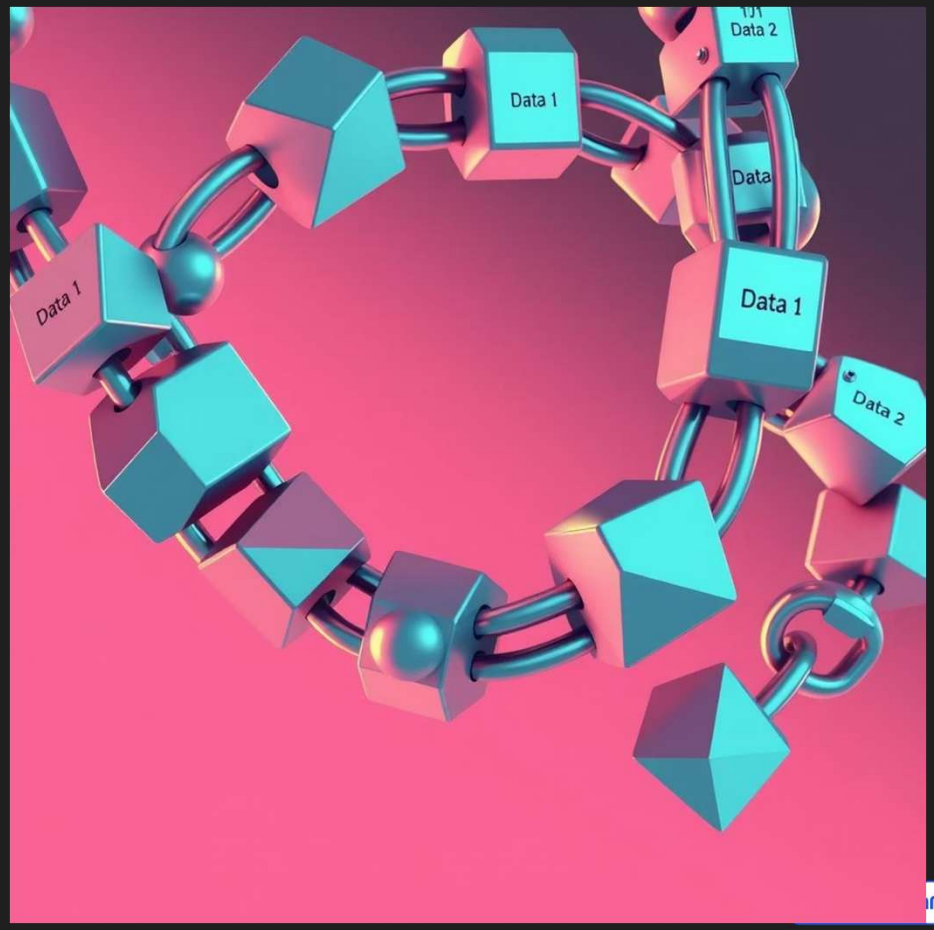
Las tuplas son colecciones de elementos que, al igual que las listas, mantienen un orden. Sin embargo, a diferencia de las listas, las tuplas son inmutables: una vez creadas, no se pueden modificar, añadir ni eliminar elementos. Se definen usando paréntesis `()`.

Datos Constantes

Son ideales para almacenar datos que no deben cambiar durante la ejecución del programa, como las coordenadas geográficas de un lugar o los días de la semana.

Ejemplo

```
dias_semana = ("lunes", "martes", "miércoles")
```



Funciones Principales de Tuplas

Acceso por Índice

Al igual que las listas, los elementos de una tupla se pueden acceder utilizando su índice numérico, que empieza en 0. Por ejemplo: `mi_tupla[0]`.

Conteo de Elementos

count(x): este método devuelve el número de veces que un elemento específico **x** aparece en la tupla.

Búsqueda de Índice

index(x): devuelve el índice de la primera aparición de un elemento **x** en la tupla. Si el elemento no se encuentra, genera un error.

Inmutabilidad

Es importante recordar que las tuplas no tienen métodos para modificar, añadir o eliminar elementos una vez creadas, debido a su naturaleza inmutable.

Conjuntos (Sets)

Colección No Ordenada de Elementos Únicos

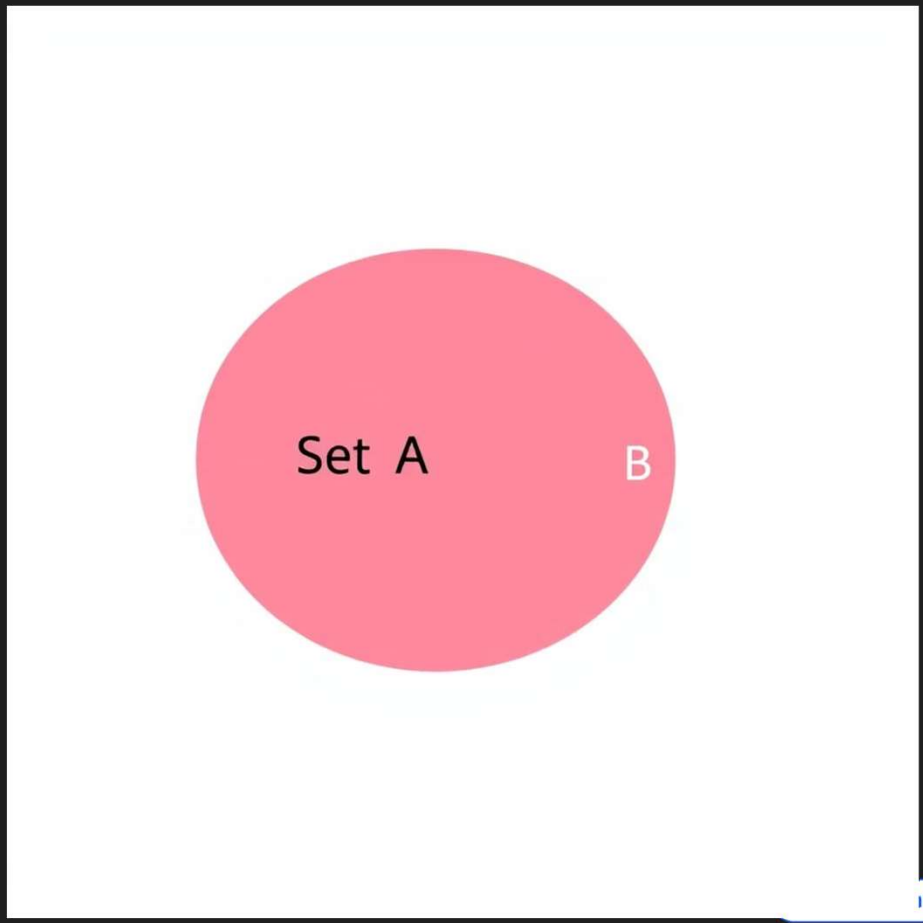
Los conjuntos son colecciones de elementos que no mantienen un orden específico y garantizan que todos sus elementos son únicos (no permiten duplicados). Se definen utilizando llaves `{}`.

Operaciones Matemáticas

Son extremadamente útiles para realizar operaciones matemáticas clásicas entre conjuntos, como la unión, la intersección y la diferencia.

Ejemplo

`frutas = {"manzana", "pera", "naranja", "manzana"}` (Solo contendrá "manzana", "pera", "naranja")



Funciones Principales de Conjuntos

Añadir Elementos

- **add(x)**: Inserta el elemento **x** en el conjunto. Si el elemento ya existe, el conjunto no se modifica.

Eliminar Elementos

- **remove(x)**: Elimina el elemento **x**. Si **x** no existe, genera un `KeyError`.
- **discard(x)**: Elimina **x**. No produce error si el elemento no se encuentra.

Operaciones con Conjuntos

- **union(otro_conjunto)**: Devuelve un nuevo conjunto con todos los elementos de ambos.
- **intersection(otro_conjunto)**: Devuelve un nuevo conjunto con los elementos comunes.
- **difference(otro_conjunto)**: Devuelve un nuevo conjunto con los elementos en el primer conjunto pero no en el segundo.

Diccionarios

Pares Clave:Valor

Son colecciones mutables de elementos que se organizan en pares de **clave:valor**. Las claves deben ser únicas, mientras que los valores pueden repetirse y ser de cualquier tipo de dato. Se definen usando llaves `{}`.

Acceso Rápido

Permiten un acceso extremadamente rápido a los valores a través de sus claves, lo que los hace ideales para almacenar datos estructurados y para búsquedas eficientes.

Ejemplo Representativo

persona = {"nombre": "Ana", "edad": 30, "ciudad": “medellin"}

Funciones Principales de Diccionarios

1

Visualización de Contenido

- **keys()**: devuelve una vista de todas las claves del diccionario.
- **values()**: devuelve una vista de todos los valores.
- **items()**: devuelve una vista de todos los pares (clave, valor) como tuplas.

2

Acceso Seguro

- **get(clave, default)**: Obtiene el valor asociado a la clave. Si la clave no existe, devuelve **default** (o `None` si no se especifica), evitando errores.

3

Manipulación de Pares

- **pop(clave)**: Elimina el par clave-valor asociado a la clave y devuelve el valor. Si la clave no existe, genera un error.
- **update(otro_diccionario)**: Añade o actualiza los pares clave-valor de otro diccionario en el actual.

Resumen y Conclusiones Clave

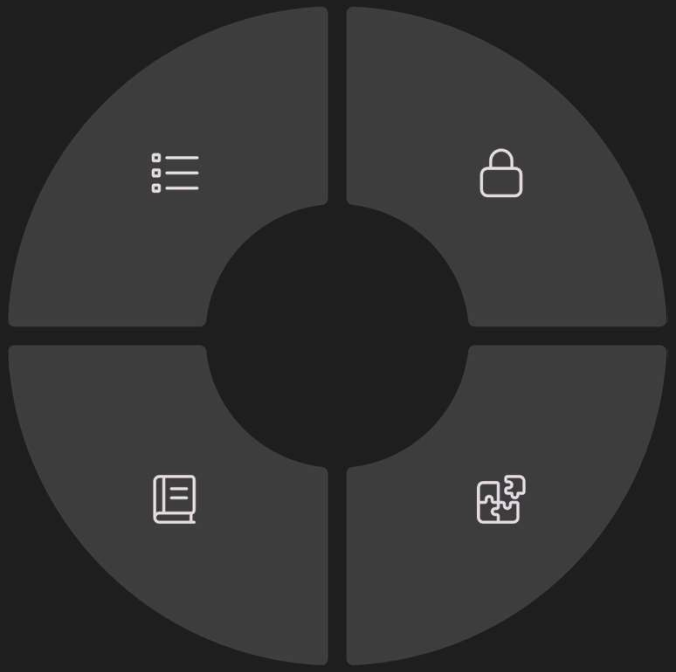
Elegir la estructura de datos adecuada es fundamental para optimizar el rendimiento y la legibilidad de tu código. Cada una de ellas tiene un propósito específico:

Listas

Ordenadas, mutables, permiten duplicados. Para colecciones de ítems que cambian con frecuencia.

Diccionarios

Pares clave-valor, acceso rápido por clave. Para datos estructurados y búsquedas eficientes.



Tuplas

Ordenadas, inmutables. Para datos constantes y estructuras fijas.

Conjuntos

No ordenados, sin duplicados. Ideales para operaciones de pertenencia y matemáticas entre colecciones.