# Conjuntos (Sets) en Python

Los conjuntos en Python son una colección desordenada y sin duplicados de elementos. Son muy útiles para eliminar duplicados, realizar operaciones de pertenencia y operaciones matemáticas como unión, intersección, diferencia y diferencia simétrica. A continuación, se presentan las propiedades y métodos más importantes de los conjuntos en Python, junto con ejemplos de código.

## Propiedades de los Conjuntos

### 1. Elementos únicos

Los conjuntos no permiten elementos duplicados. Si se intenta agregar un elemento que ya existe en el conjunto, este será ignorado.

frutas = {"manzana", "banana", "naranja", "banana"}  
print(frutas) # Salida: {"manzana", "banana", "naranja"}

### 2. Desordenados

Los elementos en un conjunto no tienen un orden específico, a diferencia de las listas y tuplas.

numeros = {1, 3, 5, 2, 4}  
print(numeros) # Salida: {1, 2, 3, 4, 5} (el orden puede variar)

### 3. Mutabilidad

Los conjuntos son mutables, lo que significa que sus elementos pueden ser agregados o eliminados después de su creación.

numeros = {1, 2, 3}  
numeros.add(4) # Agrega el elemento 4 al conjunto  
print(numeros) # Salida: {1, 2, 3, 4}

### 4. Longitud

La longitud de un conjunto (número de elementos) se puede obtener utilizando la función len().

vocales = {"a", "e", "i", "o", "u"}  
print(len(vocales)) # Salida: 5

## Métodos de los Conjuntos

### 1. add()

Agrega un elemento al conjunto. Si el elemento ya existe, no se realiza ninguna acción.

frutas = {"manzana", "banana", "naranja"}  
frutas.add("kiwi")  
print(frutas) # Salida: {"manzana", "banana", "naranja", "kiwi"}

### 2. update()

Agrega múltiples elementos al conjunto desde un iterable (como una lista, tupla, etc.).

numeros = {1, 2, 3}  
numeros.update([4, 5, 6])  
print(numeros) # Salida: {1, 2, 3, 4, 5, 6}

### 3. remove() / discard()

Elimina un elemento específico del conjunto. remove() genera un KeyError si el elemento no existe, mientras que discard() no genera ningún error.

vocales = {"a", "e", "i", "o", "u"}  
vocales.remove("i")  
print(vocales) # Salida: {"a", "e", "o", "u"}  
vocales.discard("x") # No genera ningún error, simplemente no hace nada

### 4. union()

Devuelve un nuevo conjunto que contiene los elementos de ambos conjuntos.

conjunto1 = {1, 2, 3}  
conjunto2 = {3, 4, 5}  
union = conjunto1.union(conjunto2)  
print(union) # Salida: {1, 2, 3, 4, 5}

### 5. intersection()

Devuelve un nuevo conjunto que contiene los elementos comunes a ambos conjuntos.

conjunto1 = {1, 2, 3}  
conjunto2 = {3, 4, 5}  
interseccion = conjunto1.intersection(conjunto2)  
print(interseccion) # Salida: {3}

### 6. difference()

Devuelve un nuevo conjunto que contiene los elementos que están en el conjunto actual pero no en el otro.

conjunto1 = {1, 2, 3}  
conjunto2 = {3, 4, 5}  
diferencia = conjunto1.difference(conjunto2)  
print(diferencia) # Salida: {1, 2}

### 7. symmetric\_difference()

Devuelve un nuevo conjunto que contiene los elementos que están en uno de los conjuntos pero no en ambos.

conjunto1 = {1, 2, 3}  
conjunto2 = {3, 4, 5}  
diferencia\_simetrica = conjunto1.symmetric\_difference(conjunto2)  
print(diferencia\_simetrica) # Salida: {1, 2, 4, 5}

Los conjuntos son una estructura de datos muy útil en Python, especialmente cuando se necesita eliminar duplicados, realizar operaciones de pertenencia o realizar operaciones matemáticas con conjuntos. Son ampliamente utilizados en diversos contextos, como el procesamiento de datos, la optimización de algoritmos y la teoría de conjuntos.