

Conjuntos

Los conjuntos en Python, representados por el tipo *set*, son colecciones desordenadas **mutables** de elementos únicos que son **inmutables**. Se definen utilizando llaves {...} y permiten realizar operaciones matemáticas como unión, intersección y diferencia, además de eliminar elementos duplicados.

Creación de conjuntos

```
conjunto = {1,2,3,4,5}
print(f"El conjunto es: {conjunto}")
print(f"El tipo del conjunto es: {type(conjunto)}")
```

```
El conjunto es: {1, 2, 3, 4, 5}
El tipo del conjunto es: <class 'set'>
```

```
conjunto = {78, 4.56, "Hola", False}
```

El siguiente código muestra cómo se crea un conjunto vacío en Python y verifica su tipo:

1. Crear un Conjunto Vacío:

- Se define un conjunto vacío utilizando la función `set()`. A diferencia de las listas y las tuplas, los conjuntos en Python no permiten elementos duplicados y no tienen un orden específico:

```
conjunto = set()
```

```
conjunto = set()
print(conjunto)
print(type(conjunto))
```

```
set()
<class 'set'>
```

Creación de conjuntos inmutables

El siguiente código muestra cómo se crea un conjunto inmutable en Python, conocido como `frozenset`, y verifica su tipo:

- **Crear un `frozenset`:**

- Se define un `frozenset` utilizando la función `frozenset()`, que toma un iterable (como un conjunto) y devuelve un conjunto inmutable. A diferencia de los conjuntos (`set`), los `frozenset` no permiten modificar sus elementos después de su creación:

```
conjunto = frozenset({1, 2, 3, 4, 5})
```

```
conjunto = frozenset({1,2,3,4,5})
print(conjunto)
print(type(conjunto))
```

```
frozenset({1, 2, 3, 4, 5})
<class 'frozenset'>
```

Conversión entre conjuntos y cadenas de caracteres

Este tipo de conversión se realiza igual que con listas y tuplas, aunque como los conjuntos son desordenados, el orden en el que se hace la conversión no tendrá la estructura esperada.

```
# Cadena a un conjunto
cadena = "Hola mundo"
conjunto = set(cadena)
print(conjunto)
```

```
{'m', 'u', 'o', 'd', 'H', 'a', 'n', 'l', ' '}
```

```
# Conjunto a cadena
conjunto = {'H', 'o', 'l', 'a', ' ', 'm', 'u', 'n', 'd', 'o'}
cadena = "".join(conjunto)
print(cadena)
```

```
muodHanl
```

```
frutas = {"Manzana", "Fresa", "Plátano", "Manzana", "Fresa", "Sandía"}
cadena = ",".join(frutas)
print(cadena)
```

Sandía,Manzana,Fresa,Plátano

Añadir elementos a un conjunto

El siguiente código muestra cómo se agrega un nuevo elemento a un conjunto en Python:

- **Agregar un Nuevo Elemento:**

- Se utiliza el método `add()` para agregar un nuevo elemento, "Sandía", al conjunto `frutas`. El método `add()` solo añade el elemento si no está ya presente en el conjunto:

```
frutas.add("Sandía")
```

```
frutas = {"Manzana", "Uva", "Pera"}
frutas.add("Sandía")
print(frutas)
```

{'Pera', 'Manzana', 'Uva', 'Sandía'}

```
frutas.add("Fresa")
print(frutas)
```

{'Manzana', 'Pera', 'Fresa', 'Sandía', 'Uva'}

Crear un conjunto a partir de un conjunto vacío

```
numeros = set()
for i in range(10):
    numeros.add(i)
print(numeros)
```

{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Unión de conjuntos $A \cup B$

Elementos que están en o en :

- En lugar del símbolo + (usado en listas), se usa | (Se puede colocar con Alt+124)
- También se puede utilizar la función *union*

El siguiente código demuestra cómo se realiza una unión de dos conjuntos en Python:

1. Definir los Conjuntos:

- Se definen dos conjuntos, `frutas1` y `frutas2`, con los siguientes elementos:

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}
```

2. Unir los Conjuntos:

- Se utiliza el operador de unión | para combinar los elementos de `frutas1` y `frutas2`. La operación de unión de conjuntos devuelve un nuevo conjunto que contiene todos los elementos de ambos conjuntos, sin duplicados:

```
frutas = frutas1 | frutas2
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas1 | frutas2  
print(frutas)
```

```
{'Manzana', 'Sandía', 'Pera', 'Uva', 'Plátano'}
```

- **Otro método de unir los Conjuntos:**

- Se utiliza el método `union()` para combinar los elementos de `frutas1` y `frutas2`. La operación de unión de conjuntos devuelve un nuevo conjunto que contiene todos los elementos únicos de ambos conjuntos:

```
frutas = frutas1.union(frutas2)
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas1.union(frutas2)  
print(frutas)
```

```
{'Manzana', 'Sandía', 'Pera', 'Uva', 'Plátano'}
```

Intersección de conjuntos $A \cap B$

Elementos que están a la vez en y en :

- Se utiliza el símbolo $\&$
- También se puede utilizar la función *intersection*

El siguiente código muestra cómo realizar la intersección de dos conjuntos utilizando el operador $\&$ en Python:

- **Intersección de los Conjuntos:**

- Se utiliza el operador $\&$ para obtener los elementos comunes entre `frutas1` y `frutas2`. La operación de intersección devuelve un nuevo conjunto que contiene solo los elementos que están presentes en ambos conjuntos:

```
frutas = frutas1 & frutas2
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas1 & frutas2  
print(frutas)
```

{'Pera', 'Manzana'}

- **Otro método para obtener la Intersección de los Conjuntos:**

- Se utiliza el método `intersection()` para obtener los elementos comunes entre `frutas1` y `frutas2`. La operación de intersección devuelve un nuevo conjunto que contiene solo los elementos que están presentes en ambos conjuntos:

```
frutas = frutas1.intersection(frutas2)
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas1.intersection(frutas2)  
print(frutas)
```

{'Pera', 'Manzana'}

Diferencia de conjuntos $A - B$

Elementos que están en A y no están en B :

- Se utiliza el símbolo $-$
- También se puede utilizar la función *difference*

El siguiente código muestra cómo realizar la diferencia de dos conjuntos utilizando el operador de resta $-$ en Python:

- **Diferencia de los Conjuntos:**

- Se utiliza el operador de resta $-$ para obtener los elementos que están en `frutas1` pero no en `frutas2`. Esta operación de diferencia devuelve un nuevo conjunto que contiene solo los elementos presentes en `frutas1` que no están en `frutas2`:

```
frutas = frutas1 - frutas2
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}

frutas = frutas1 - frutas2
print(frutas)
```

{'Plátano'}

No es lo mismo obtener la diferencia `frutas1 - frutas2` que la diferencia `frutas2 - frutas1`

```
frutas1 = {"Manzana", "Pera", "Plátano"}
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}

frutas = frutas2 - frutas1
print(frutas)
```

{'Uva', 'Sandía'}

- **Otro método para realizar la Diferencia de los Conjuntos:**

- Se utiliza el método `difference()` de `frutas1` para obtener los elementos que están en `frutas1` pero no en `frutas2`. Este método devuelve un nuevo conjunto que contiene solo los elementos presentes en `frutas1` que no están en `frutas2`:

```
frutas = frutas1.difference(frutas2)
```

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas1.difference(frutas2)  
print(frutas)
```

```
{'Plátano'}
```

No es lo mismo obtener la diferencia `frutas1.difference(frutas2)` que la diferencia `frutas2.difference(frutas1)`

```
frutas1 = {"Manzana", "Pera", "Plátano"}  
frutas2 = {"Uva", "Manzana", "Pera", "Sandía"}  
  
frutas = frutas2.difference(frutas1)  
print(frutas)
```

```
{'Uva', 'Sandía'}
```

Borrar elementos de un conjunto

En el siguiente código, se utiliza la función `remove()` para eliminar un elemento específico de un conjunto en Python:

- **Eliminar un Elemento con `remove()`:**

- Se utiliza la función `remove()` para eliminar el elemento “Pera” del conjunto `frutas`. La función `remove()` busca el elemento especificado y lo elimina del conjunto. Si el elemento no se encuentra en el conjunto, se genera una excepción `KeyError`:

```
frutas.remove("Pera")
```

```
# Utilizando la función remove  
frutas = {"Manzana", "Pera", "Uva"}  
frutas.remove("Pera")  
print(frutas)
```

```
{'Manzana', 'Uva'}
```

En el siguiente código, se utiliza la operación de diferencia de conjuntos para eliminar varios elementos de un conjunto en Python:

- **Eliminar Elementos con la Diferencia de Conjuntos:**

- Se utiliza el operador `-=` para actualizar el conjunto `frutas` eliminando los elementos que están en el conjunto `{"Pera", "Uva"}`. El operador `-=` realiza la diferencia de conjuntos y actualiza el conjunto original:

```
frutas -= {"Pera", "Uva"}
```

```
# Utilizando la diferencia de conjuntos
frutas = {"Manzana", "Pera", "Uva"}
frutas -= {"Pera", "Uva"}
print(frutas)
```

```
{'Manzana'}
```

Pertenencia en un conjunto

```
frutas = {"Manzana", "Pera", "Uva"}
"Manzana" in frutas
```

```
True
```

```
frutas = {"Manzana", "Pera", "Uva"}
"Manzana" not in frutas
```

```
False
```

Iterar sobre un conjunto

```
frutas = {"Uva", "Manzana", "Sandía", "Pera"}
for fruta in frutas:
    print(fruta)
```

```
Pera
```

```
Uva
```

```
Manzana
```

```
Sandía
```


Iterar sobre múltiples conjuntos

```
frutas1 = {"Uva", "Manzana", "Sandía", "Pera"}
frutas2 = {"Fresa", "Plátano", "Durazno", "Naranja"}

for f1, f2 in zip(frutas1, frutas2):
    print(f1,f2)
```

Pera Durazno
Uva Fresa
Manzana Naranja
Sandía Plátano

Crear una copia de un conjunto

```
frutas = {"Uva", "Manzana", "Sandía", "Pera"}
frutas_copia = frutas
frutas is frutas_copia
```

True

```
frutas_copia = frutas.copy()
frutas is frutas_copia
```

False

Funciones matemáticas con conjuntos

```
# Suma de todos los elementos de un conjunto
numeros = {4,1,8,7,9}
suma = sum(numeros)
print(f"La suma es: {suma}")
```

La suma es: 29

```
# Valor mínimo de un conjunto
numeros = {4,1,8,7,9}
minimo = min(numeros)
print(f"El valor minimo es: {minimo}")
```

El valor minimo es: 1

```
# Valor máximo de un conjunto
numeros = {4,1,8,7,9}
maximo = max(numeros)
print(f"El valor máximo es: {maximo}")
```

El valor máximo es: 9

```
# Longitud de un conjunto
numeros = {4,1,8,7,9}
longitud = len(numeros)
print(f"La longitud es: {longitud}")
```

La longitud es: 5

Problema con los conjuntos de conjuntos

En Python, no se pueden crear conjuntos de conjuntos (o conjuntos de otros conjuntos) debido a que los conjuntos deben ser “hashables”. Esto significa que los elementos dentro de un conjunto deben ser inmutables y su valor debe ser fijo durante su vida útil para que puedan ser utilizados como claves en un diccionario y como miembros en un conjunto.

Los conjuntos en Python son mutables; es decir, puedes agregar o eliminar elementos después de que el conjunto ha sido creado. Dado que los conjuntos son mutables, no cumplen con el criterio de ser hashables. Esto impide que un conjunto contenga otros conjuntos, ya que, al ser mutable, no garantiza que su valor sea constante y fiable para las operaciones de hash.