

¿Qué es un conjunto?

Un conjunto en Python es una colección no ordenada de elementos únicos.

Usos de conjuntos

Eliminación de elementos duplicados: Si tienes una lista con elementos repetidos y quieres quitar los duplicados, puedes crear un conjunto a partir de esa lista y luego volver a convertirla en una lista.

Verificación de pertenencia: Puedes utilizar un conjunto para verificar si un elemento específico pertenece a una colección o no. Esto es más rápido que verificar manualmente si un elemento está en una lista o en otro tipo de colección.

Cálculo de intersecciones y diferencias: Si tienes dos o más conjuntos y quieres encontrar la intersección (elementos en común) o la diferencia (elementos exclusivos) entre ellos, puedes utilizar los métodos de intersección y diferencia de los conjuntos.

Análisis de datos: Los conjuntos son útiles en el análisis de datos para realizar operaciones como contar la frecuencia de elementos en una colección o encontrar valores únicos.

Creación de conjuntos

1) Para crear un conjunto especificamos sus elementos entre llaves

2) Puede contener distintos tipos de datos, excepto objetos inmutables

3) Python diferencia diccionarios de objetos por uso de ":" para separar keys y values. Sin embargo la no reconoce la creación de conjuntos con la notación de "{}" directamente

4) Un set puede ser convertido a una lista y viceversa.

Ejemplos con código

```
#1 Creación de conjuntos
conjunto = {1, 2, 3, 4}

#2 Contiene diversos tipos
conjunto = s = {2.16, False, True, " string ", (1, 2)}

#Excepto objetos mutables
conjunto = {[a, b, c]} -> TypeError: unhashable type: 'list'

#3 Python diferencia conjuntos de colecciones
conjunto = {} -> colección
conjunto = set() -> conjunto

#4 Se pueden convertir conjuntos a listas y viceversa
list({1, 2, 3, 4, 5}) -> output: [1,2,3,4,5]
set([1, 2, 2, 3, 4, 4, 5, 6]) -> output: {1,2,3,4,5,6} (sin repetidos)
```



By Fide (fidelp27)
cheatography.com/fidelp27/

Published 12th February, 2023.
Last updated 12th February, 2023.
Page 1 of 3.

Sponsored by [Readable.com](https://readable.com)
Measure your website readability!
<https://readable.com>

Métodos para agregar elementos

```
#add: agrega un elemento al conjunto.
conjunto = set()
conjunto.add(1) -> output: {1}
#update: agrega varios elementos al conjunto.
conjunto = set([1, 2, 3])
conjunto.update([3, 4, 5]) -> output: {1, 2, 3, 4, 5}
```

Métodos para eliminar elementos

```
#remove: elimina un elemento específico del conjunto.
conjunto = set([1, 2, 3, 4, 5])
conjunto.remove(3) -> {1, 2, 4, 5}
#discard: elimina un elemento específico del conjunto, pero no produce un error si el elemento no está en el conjunto.
conjunto = set([1, 2, 3, 4, 5])
conjunto.discard(6) # no produce un error -> output: {1, 2, 3, 4, 5}
#pop: elimina y devuelve un elemento arbitrario del conjunto.
conjunto = set([1, 2, 3, 4, 5])
print(conjunto.pop()) # imprime un elemento arbitrario, por ejemplo: 1 -> output: {2, 3, 4, 5}
#clear: elimina todos los elementos del conjunto.
conjunto = set([1, 2, 3, 4, 5])
conjunto.clear() -> output: set()
```

Métodos de verificación

```
#issubset: verifica si un conjunto es un subconjunto de otro conjunto. Verifica que todos los elementos de "a" están presentes en el conjunto "b"
conjunto_a = set([1, 2, 3])
conjunto_b = set([1, 2, 3, 4, 5])
print(conjunto_a.issubset(conjunto_b)) -> output: True
#issuperset: verifica si un conjunto es un superconjunto de otro conjunto. Si todos los elementos del conjunto "b" están presentes en el conjunto "a"
conjunto_a = set([1, 2, 3, 4, 5])
conjunto_b = set([1, 2, 3])
print(conjunto_a.issuperset(conjunto_b)) -> output: True
#isdisjoint: verifica si dos conjuntos no tienen elementos en común.
conjunto_a = set([1, 2, 3])
conjunto_b = set([4, 5, 6])
print(conjunto_a.isdisjoint(conjunto_b)) -> output: True
```



Métodos de operación

```
#union: devuelve un conjunto que contiene los elementos de ambos conjuntos.
conjuntoA = {1, 2, 3, 4}
conjuntoB = {3, 4, 5, 6}
conjuntoC = conjuntoA.union(conjuntoB) ó conjuntoC = conjuntoA | conjuntoB
-> output: {1, 2, 3, 4, 5}

#inter seccion: devuelve un conjunto que contiene los elementos en común entre dos conjuntos.
conjuntoA = {1, 2, 3, 4}
conjuntoB = {3, 4, 5, 6}
conjuntoC = conjuntoA.intersection(conjuntoB) ó conjuntoC = conjuntoA & conjuntoB
-> output: {3,4}

#diferencia: devuelve un conjunto que contiene los elementos exclusivos de un conjunto en comparación con otro. Elementos únicos en A que no contiene B
conjuntoA = {1, 2, 3, 4}
conjuntoB = {3, 4, 5, 6}
conjuntoC = conjuntoA.difference(conjuntoB) ó conjuntoC = conjuntoA - conjuntoB
-> output: {1,2}

#symmetric_difference: devuelve un conjunto que contiene los elementos que no están en común entre dos conjuntos. Elementos únicos en A al comparar con B y viceversa
conjuntoA = {1, 2, 3, 4}
conjuntoB = {3, 4, 5, 6}
conjuntoC = conjuntoA.symmetric_difference(conjuntoB) ó conjuntoC = conjuntoA ^ conjuntoB
-> output: {1,2,5,6}

#update: es una operación que permite modificar un conjunto existente agregándole elementos de otro conjunto. Modifica directamente el conjunto A a diferencia de la unión que crea un nuevo conjunto
conjuntoA = {1, 2, 3, 4}
conjuntoB = {3, 4, 5, 6}
conjuntoA.update(conjuntoB) ó conjuntoA |= conjuntoB
-> output: {1,2,3,4,5,6}
```

