CONJUNTOS- SET{}

Los conjuntos se utilizan para almacenar varios elementos en una sola variable.

Set es uno de los 4 tipos de datos integrados en Python que se utilizan para almacenar colecciones de datos,

Creación: Los conjuntos se escriben con llaves.

```
Mi_conjunto = {"rosa", "jazmín", "tulipán"}

print(Mi_conjunto)
```

Devolverá:
{"rosa", "jazmín", "tulipán"}

Un conjunto es una colección *desordenada*, *inmutable* y *no indexada*.

Los conjuntos están desordenados, por lo que no puede estar seguro de en qué orden aparecerán los elementos
Los elementos del conjunto pueden aparecer en un orden diferente cada vez que los use, y no se puede hacer referencia a ellos por índice o clave.

Como no tienen índices ni orden… no es posible acceder a sus elementos a través de la posición
Se debe hacer un ciclo *for*

```
Frutas = {"Banana"," Manzana","Mandarina", "Sandia", "Melones", "Kiwi", "Sandia", "Uva"}

for f in Frutas:

  print(f)
```

Contar elementos de un conjunto, len()

Para determinar cuántos elementos tiene un conjunto, use el método ***len( ).***

Los elementos del conjunto pueden ser de cualquier tipo de datos

```
Conj1 = {"lunes", "martes", "miércoles"}
Conj2 = {3, 6, 9, 12, 15,18}
Conj3 = {true, false, true}

print(Conj1)
print(Conj2)
print(Conj3)
```

Devolverá:

{"lunes", "martes", "miércoles"}
{3, 6, 9, 12, 15,18}
{true, false, true}

Cómo saber si un elemento está en un conjunto

Con los conjuntos también se puede usar el operador de pertenencia in para comprobar si un elemento está contenido, o no, en un conjunto:

OPERACIONES DE CONJUNTOS

Unión de conjuntos en Python

La unión de dos conjuntos A y B es el conjunto A ∪ B que contiene todos los elementos de A y de B.
En Python se utiliza el operador **|** para realizar la unión de dos o más conjuntos.

```
a = {1,3,5,7,9}
b = {2,4,6,8}
print(a|b)
```

Devolverá:

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

Intersección de conjuntos

La intersección de dos conjuntos A y B es el conjunto A ∩ B que contiene todos los elementos comunes de A y B.

En Python se utiliza el operador **&** para realizar la intersección de dos o más conjuntos.

```
c = {“Cielo”, “Isa”, “Mari”,” Wendy”, “Eloísa”}
d = {“Gaby”, “Tiago”, “Víctor”, “Cielo”, “Isa”}
print(c&d)
```

Devolverá

{“Cielo”,” Isa”}

Diferencia entre dos conjuntos

La diferencia entre dos conjuntos A y B es el conjunto A \ B que contiene todos los elementos de A que no pertenecen a B.

En Python se utiliza el operador **-** para realizar la diferencia de dos o más conjuntos.

```
e = {"lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo"}
f = {"sábado", "domingo"}

print(e-f)
```

Devolverá:

{"lunes", "martes", "miércoles", "jueves", "viernes”}

Diferencia simétrica de conjuntos

La diferencia simétrica entre dos conjuntos A y B es el conjunto que contiene los elementos de A y B que no son comunes.

En Python se utiliza el operador ^ para realizar la diferencia simétrica de dos o más conjuntos.

```
g = {1, 2, 3, 4, 5}
h = {2, 4, 6, 8}
print(g^h)
```

Devolverá:

{1, 3, 5, 6, 8}

Inclusión de conjuntos

Dado un conjunto A, subcolección del conjunto B o igual a este, sus elementos son un subconjunto de B. Es decir, A es un subconjunto de B y B es un superconjunto de A.

En Python se utiliza el operador **<=** para comprobar si un conjunto A es subconjunto de B y el operador **>=** para comprobar si un conjunto A es superconjunto de B.

```
i = {2,3}
print(g<=h)
print(g>=h)
print(i<=g)
```

Devolverá:

```
False
False
True
```

Conjuntos disjuntos

Dos conjuntos A y B son disjuntos si no tienen elementos en común, es decir, la intersección de A y B es el conjunto vacío.

En Python se utiliza el método **isdisjoint()** de la clase set para comprobar si un conjunto es disjunto de otro

```
a = {1, 2}
b = {3, 4}
print(a.isdisjoint(b))
```

Devolverá: True

Igualdad de conjuntos

Dos conjuntos son iguales si y solo si todos los elementos de un conjunto están contenidos en el otro. Esto quiere decir que cada uno es un subconjunto del otro

```
f = {“verano”, “otoño”,” invierno”, “primavera}
s = {“verano”, “otoño”, “invierno”, “primavera}
print(f==s)
```

Devolverá: True

Métodos de la clase set

MÉTODOS	DESCRIPCIÓN
add(e)	Añade un elemento al conjunto
clear()	Elimina todos los elementos del conjunto.
copy()	Devuelve una copia superficial del conjunto.
difference(iterable)	Devuelve la diferencia del conjunto con el iterable como un conjunto nuevo
difference_update(iterable)	Actualiza el conjunto tras realizar la diferencia con el iterable.
discard(e)	Elimina, si existe, el elemento del conjunto.
intersection(iterable)	Devuelve la intersección del conjunto con el iterable como un conjunto nuevo.
intersection_update(iterable)	Actualiza el conjunto tras realizar la intersección con el iterable.
isdisjoint(iterable)	Devuelve True si dos conjuntos son disjuntos.
issubset(iterable)	Devuelve True si el conjunto es subconjunto del iterable.
issuperset(iterable)	Devuelve True si el conjunto es superconjunto del iterable
pop()	Obtiene y elimina un elemento de forma aleatoria del conjunto.
remove(e)	Elimina el elemento del conjunto. Si no existe lanza un error.
symmetric_difference(iterable)	Devuelve la diferencia simétrica del conjunto con el iterable como un conjunto nuevo.
symmetric_difference_update(iterable)	Actualiza el conjunto tras realizar la diferencia simétrica con el iterable.
union(iterable)	Devuelve la unión del conjunto con el iterable como un conjunto nuevo.
update(iterable)	Actualiza el conjunto tras realizar la unión con el iterable.