

sets

Los sets son otro tipo de estructuras de datos. Se diferencian de listas, tuplas y diccionarios porque son una colección *mutable* de elementos *inmutables*, *no ordenados* y *sin datos repetidos*.

mutable ✓

ordenado ✗

^{admite}
duplicados ✗

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

métodos

add(item) agrega un elemento al set

```
mi_set_a.add(5)
print(mi_set_a)
>> {1, 2, "tres", 5}
```

clear() remueve todos los elementos de un set

```
mi_set_a.clear()
print(mi_set_a)
>> set()
```

copy() retorna una copia del set

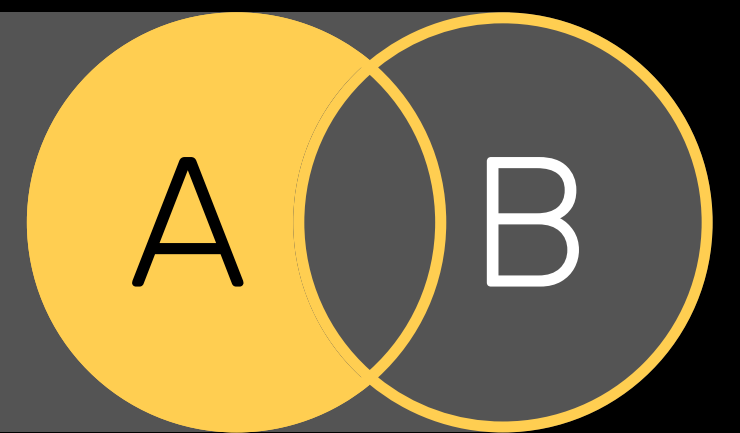
```
mi_set_c = mi_set_a.copy()
print(mi_set_c)
>> {1, 2, "tres"}
```

sets

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

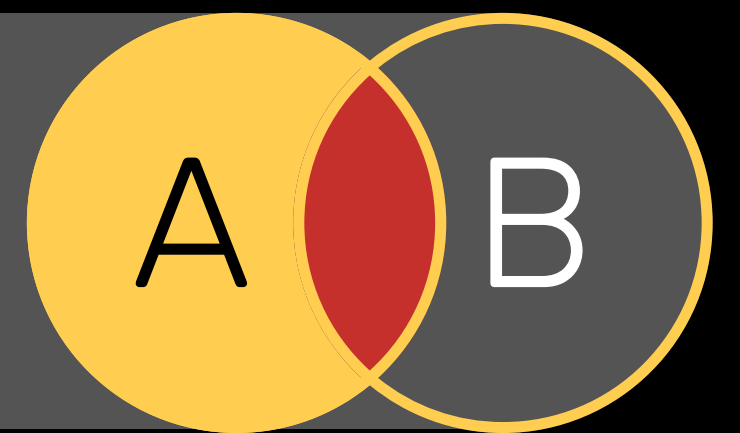
difference(set) retorna el set formado por todos los elementos que únicamente existen en el set A

```
mi_set_c = mi_set_a.difference(mi_set_b)
print(mi_set_c)
>> {1, 2}
```



difference_update(set) remueve de A todos los elementos comunes a A y B

```
mi_set_a.difference_update(mi_set_b)
print(mi_set_a)
>> {1, 2}
```

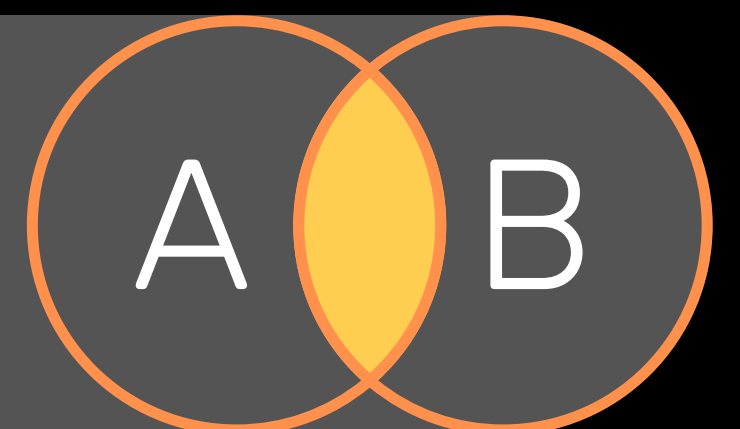


discard(item) remueve un elemento del set

```
mi_set_a.discard("tres")
print(mi_set_a)
>> {1, 2}
```

intersection(set) retorna el set formado por todos los elementos que existen en A y B simultáneamente.

```
mi_set_c = mi_set_a.intersection(mi_set_b)
print(mi_set_c)
>> {'tres'}
```

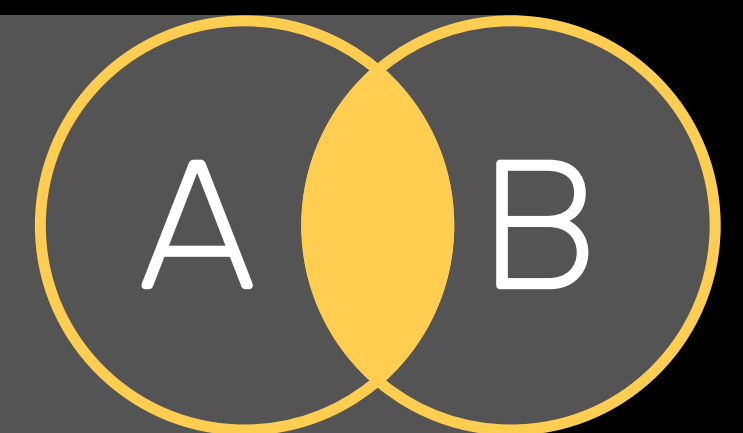


sets

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

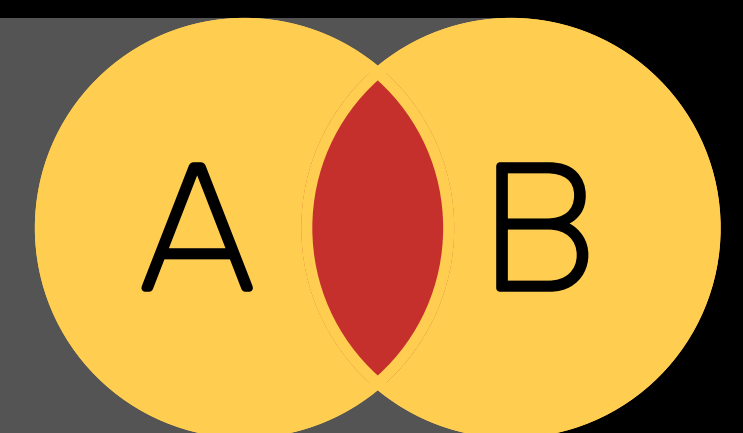
intersection_update(set) mantiene únicamente los elementos comunes a A y B

```
mi_set_b.intersection_update(mi_set_a)
print(mi_set_b)
>> {"tres"}
```



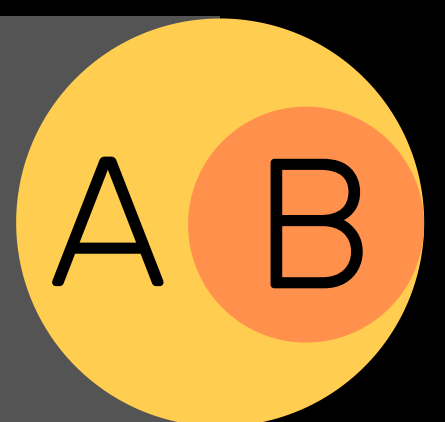
isdisjoint(set) devuelve True si A y B no tienen elementos en común

```
conjunto_disjunto = mi_set_a.isdisjoint(mi_set_b)
print(conjunto_disjunto)
>> False
```



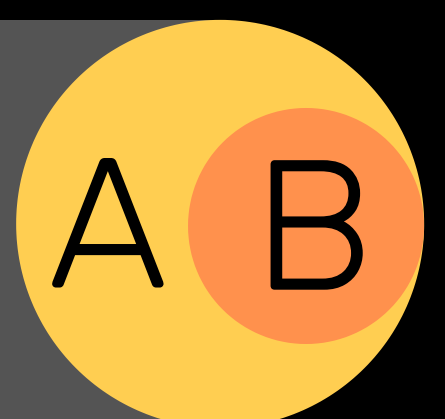
issubset(set) devuelve True si todos los elementos de B están presentes en A

```
es_subset = mi_set_b.issubset(mi_set_a)
print(es_subset)
>> False
```



issuperset(set) devuelve True si A contiene todos los elementos de B

```
es_superset = mi_set_a.issuperset(mi_set_b)
print(es_superset)
>> False
```



sets

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

pop() elimina y retorna un elemento al azar del set

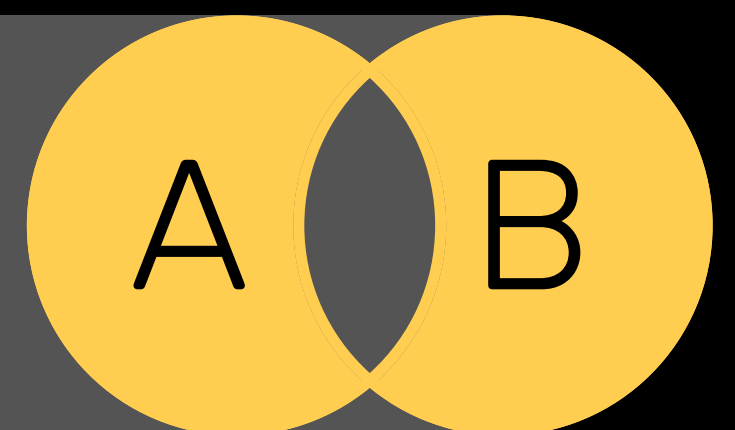
```
aleatorio = mi_set_a.pop()  
print(aleatorio)  
>> {2}
```

remove(item) elimina un item del set, y arroja error si no existe

```
mi_set_a.remove("tres")  
print(mi_set_a)  
>> {1, 2}
```

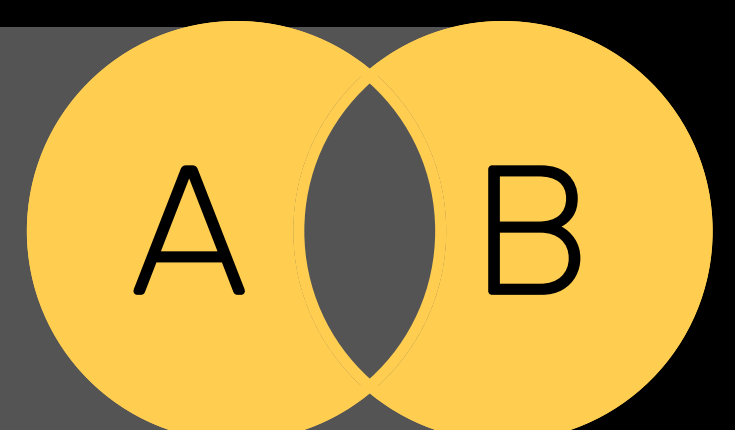
symmetric_difference(set) retorna todos los elementos de A y B, excepto aquellos que son comunes a los dos

```
mi_set_c = mi_set_b.symmetric_difference(mi_set_a)  
print(mi_set_c)  
>> {1, 2, 3}
```



symmetric_difference_update(set) elimina los elementos comunes a A y B, agregando los que no están presentes en ambos a la vez

```
mi_set_b.symmetric_difference_update(mi_set_a)  
print(mi_set_b)  
>> {1, 2, 3}
```

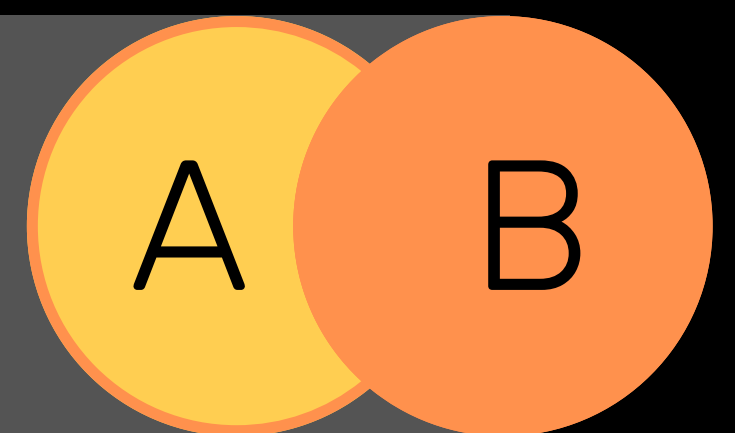


sets

```
mi_set_a = {1, 2, "tres"}    mi_set_b = {3, "tres"}
```

union(set) retorna un set resultado de combinar A y B (los datos duplicados se eliminan)

```
mi_set_c = mi_set_a.union(mi_set_b)
print(mi_set_c)
>> {1, 2, 3, 'tres'}
```



update(set) inserta en A los elementos de B

```
mi_set_a.update(mi_set_b)
print(mi_set_a)
>> {1, 2, 3, 'tres'}
```

