

# U3.4

# Conjuntos



# Indice

# Conjuntos en Python

- El tipo set en Python representa conjuntos.
- Un conjunto es una colección desordenada de elementos únicos.

# Qué es el tipo set en Python

- Colección sin orden y elementos únicos.
- Usos principales: verificar pertenencia y eliminar duplicados.

# Creación de conjuntos

- Usar llaves {} o el constructor set().
- Elimina elementos repetidos.

python [Copy code](#)

```
c = {1, 3, 2, 9, 3, 1} # {1, 2, 3, 9}
a = set('Hola Pythonista') # {'a', 'H', 'h', 'y', 'n', 's', 'P', 't', ' '}
unicos = set([3, 5, 6, 1, 5]) # {1, 3, 5, 6}
```

# Conjuntos vs Frozensets

- **set:** Mutable, **seResumen:** Diccionarios en Python Los diccionarios en Python son estructuras de datos que asocian claves con valores. Las claves deben ser inmutables y hashables. Las principales operaciones incluyen almacenar y recuperar valores por clave. Note: Los diccionarios son esenciales y eficientes en Python.

**Resumen: Creación y Manipulación** Crear un diccionario: `{}` o `dict()`. Acceder a elementos: `d[clave]` o `d.get(clave, valor_por_defecto)`. Recorrer un diccionario con `for` y `keys()`, `values()`, `items()`. Añadir, modificar y eliminar elementos en un diccionario. Note: Diccionarios son mutables y versátiles.

**Resumen: Conclusiones** Los diccionarios son ideales para mapear claves a valores en Python. Permite un acceso rápido a

# Acceso a elementos

- No se puede acceder por índice.
- Son objetos que no tienen orden.
- Utilizar un bucle for.

python [Copy code](#)

```
mi_conjunto = {1, 3, 2, 9, 3, 1}
for e in mi_conjunto:
    print(e)
# Output: 1, 2, 3, 9
```



# Añadir elementos a un conjunto

- Método `add()` para un elemento.
- Método `update()` para varios elementos.

python [Copy code](#)

```
mi_conjunto = {1, 3, 2, 9, 3, 1}
mi_conjunto.add(7)    # {1, 2, 3, 7, 9}
mi_conjunto.update([5, 3, 4, 6])  # {1, 2, 3, 4, 5, 6, 7, 9}
```

# Eliminar elementos de un conjunto

- Métodos `discard()`, `remove()`, `pop()` y `clear()`.


python [Copy code](#)

```
mi_conjunto = {1, 3, 2, 9, 3, 1, 6, 4, 5}
mi_conjunto.remove(1) # {2, 3, 4, 5, 6, 9}
mi_conjunto.discard(4) # {2, 3, 5, 6, 9}
mi_conjunto.remove(7) # KeyError: 7
mi_conjunto.discard(7) # No hace nada
mi_conjunto.pop() # 2
mi_conjunto.clear() # set()
```

# Número de elementos en un conjunto

- Usa `len()` para obtener la cantidad de elementos.

python


 Copy code

```
mi_conjunto = set([1, 2, 5, 3, 1, 5])  
len(mi_conjunto) # 4
```

# Verificar pertenencia en un conjunto

- Utiliza el operador `in`.

python

 Copy code


```
mi_conjunto = set([1, 2, 5, 3, 1, 5])
print(1 in mi_conjunto) # True
print(6 in mi_conjunto) # False
print(2 not in mi_conjunto) # False
```

# Operaciones sobre conjuntos

# Unión de conjuntos

- $A \cup B$ : contiene todos los elementos de  $A$  y de  $B$ .

python

 Copy code

```
a = {1, 2, 3, 4}
b = {2, 4, 6, 8}
a | b # {1, 2, 3, 4, 6, 8}
```

# Intersección de conjuntos

- $A \cap B$ : contiene elementos comunes de A y B.

python [Copy code](#)

```
a = {1, 2, 3, 4}
b = {2, 4, 6, 8}
a & b # {2, 4}
```

# Diferencia de conjuntos

- $A - B$ : contiene elementos de  $A$  que no están en  $B$ .

python [Copy code](#)

```
a = {1, 2, 3, 4}
b = {2, 4, 6, 8}
a - b # {1, 3}
```



# Diferencia simétrica de conjuntos

- Contiene elementos que no son comunes en A y B.

python

 Copy code

```
a = {1, 2, 3, 4}
b = {2, 4, 6, 8}
a ^ b # {1, 3, 6, 8}
```

# Inclusión de conjuntos

- $A \subseteq B$ :  $A$  es subconjunto de  $B$ .


python [Copy code](#)

```
a = {1, 2}
b = {1, 2, 3, 4}
a <= b # True
```

# Conjuntos disjuntos

- A y B no tienen elementos en común.


python

 Copy code

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

# Igualdad de conjuntos

- Dos conjuntos son iguales si todos los elementos de uno están en el otro.

```
python  Copy code  
a = {1, 2}  
b = {1, 2}  
a == b # True
```

# Métodos de la clase set

# Métodos de la clase set en Python

- `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 con otro conjunto o iterable.
- `discard(e)` : Elimina el elemento si existe.

# Más métodos de la clase set

- `intersection(iterable)`: Devuelve la intersección con otro conjunto o iterable.
- `isdisjoint(iterable)`: Devuelve True si dos conjuntos son disjuntos.
- `issubset(iterable)`: Devuelve True si el conjunto es subconjunto.
- `issuperset(iterable)`: Devuelve True si el conjunto es superconjunto.

# Métodos de actualización de conjuntos

- `difference_update(iterable)`: Actualiza el conjunto con la diferencia.
- `intersection_update(iterable)`: Actualiza el conjunto con la intersección.
- `symmetric_difference_update(iterable)`: Actualiza con la diferencia simétrica.
- `update(iterable)`: Actualiza el conjunto con la unión.



# Resumen: Conjuntos en Python

# Conjuntos en Python

- Un conjunto es una colección desordenada de elementos únicos.
- Útil para verificar pertenencia y eliminar duplicados.

# Crear un conjunto

- `{elemento1, elemento2, ...}` o `set(iterable)`.
- Elementos repetidos son eliminados.

python [Copy code](#)

```
c = {1, 3, 2, 9, 3, 1} # {1, 2, 3, 9}
a = set('Hola Pythonista') # {'a', 'H', 'h', 'y', 'n', 's', 'P', 't', ' '}
```

# Operaciones con conjuntos

- Operadores como  $|$ ,  $\&$ ,  $-$ ,  $\wedge$  para operaciones de conjuntos.
- Unión, intersección, diferencia, diferencia simétrica.
- Inclusión, conjuntos disjuntos, igualdad.

# Operaciones con conjuntos

python

 Copy code

```
a = {1, 2, 3, 4}
b = {2, 4, 6, 8}
a | b    # Unión
a & b    # Intersección
a - b    # Diferencia
a ^ b    # Diferencia simétrica
```

# Métodos importantes

- `add()`, `remove()`, `clear()`.
- `update()`, `discard()`, `pop()`.
- `len()`, `in`.

¡Gracias por su atención!