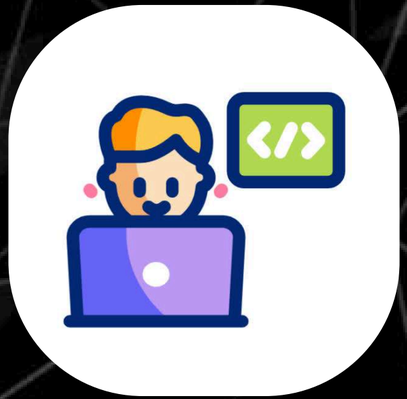
CURSO DE PYTHON DESDE CERO

¿QUÉ ES UN SET EN PYTHON?

Un set (conjunto) es una colección NO ordenada y sin elementos duplicados.



CARACTERÍSTICAS:

No tiene orden → no puedes acceder por índices (set[0] da error).

No permite duplicados → si añades un elemento repetido, se ignora.

Es mutable → puedes agregar y eliminar elementos.

Muy útil para: eliminar duplicados, comparar colecciones y operaciones matemáticas.

EJEMPLO:

```python
numeros = {1, 2, 3, 3}
print(numeros)
# {1, 2, 3}  ← elimina duplicados
```

¿CÓMO DEFINIR UN SET?

✔ 1. Con llaves {}

```
1   frutas = {"manzana", "pera", "uva"}
```

🟦 ¿CÓMO DEFINIR UN SET?

✔ 2. Set vacío (IMPORTANTE)

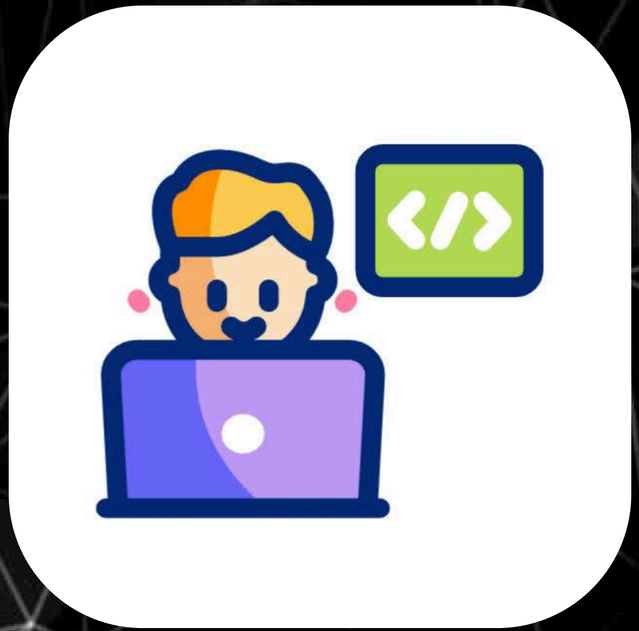
```
# ❌ Esto NO es un set:
vacio = {} # Esto crea un diccionario

# ✔ Debe hacerse así:
vacio = set()
```

🟦 ¿CÓMO DEFINIR UN SET?

✓ 3. Usando set() para convertir otras colecciones

```
1   lista = [1, 2, 2, 3]
2   set_variable = set(lista)   # {1, 2, 3}
```

MÉTODOS PRINCIPALES DE LOS SETS



MÉTODOS DE LOS SETS

add() → Agregar un elemento.

```
1   frutas = {"manzana", "pera"}
2   frutas.add("kiwi")
```

MÉTODOS DE LOS SETS

remove() → Eliminar un elemento, Si no existe, da error.

```
1   frutas = {"manzana", "pera", "uva"}
2   frutas.remove("pera")
```

MÉTODOS DE LOS SETS

discard() → Eliminar un elemento sin error.

```
1   frutas = {"manzana", "pera", "uva"}
2   frutas.discard("naranja")
3   # No da error si no existe
```

MÉTODOS DE LOS SETS

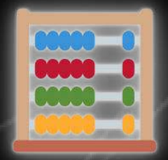
pop() → Elimina un elemento "aleatorio".

```python
frutas = {"manzana", "pera", "uva"}
frutas.pop()
# En Python, los sets (conjuntos) no tienen índices
```

MÉTODOS DE LOS SETS

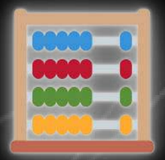
clear() → Vaciar el set.

```
1   frutas = {"manzana", "pera", "uva"}
2   frutas.clear()
```



MÉTODOS PARA OPERACIONES DE CONJUNTOS

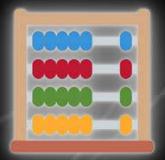
Muy útiles para matemáticas y filtrado.



OPERACIONES DE CONJUNTOS

union() → Unión de conjuntos:
Combina todos los elementos de ambos conjuntos sin duplicados.

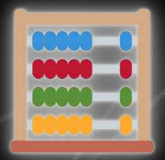
```
1   a = {1, 2}
2   b = {2, 3}
3   print(a.union(b))   # {1, 2, 3}
```



OPERACIONES DE CONJUNTOS

intersection() → Intersección (elementos comunes):
Devuelve solo los elementos que existen en ambos conjuntos.

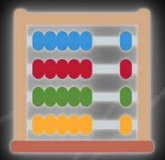
```
1   a = {1, 2, 3}
2   b = {2, 3, 4}
3   print(a.intersection(b))   # {2, 3}
```



OPERACIONES DE CONJUNTOS

difference() → Diferencia:
Devuelve los elementos que están en el primer conjunto pero no en el segundo.

```
1   a = {1, 2, 3}
2   b = {2}
3   print(a.difference(b))   # {1, 3}
```



OPERACIONES DE CONJUNTOS

symmetric_difference() → Elementos NO comunes:
Devuelve los elementos que NO se repiten en ambos conjuntos.

```
1   a = {1, 2}
2   b = {2, 3}
3   print(a.symmetric_difference(b))   # {1, 3}
```

🔥 OPERACIONES DE CONJUNTOS

Muy útiles para matemáticas y filtrado.

🔥 OPERACIONES DE CONJUNTOS

issubset() → Es Subconjunto:
Devuelve True si todos los elementos del conjunto A están dentro del conjunto B.

```
1   a = {1, 2}
2   b = {2, 3}
3   print(a.issubset(b)) # False
```

🔥 OPERACIONES DE CONJUNTOS

issuperset() → Es superconjunto:
Devuelve True si el conjunto A contiene todos los elementos del conjunto B.

```
1   a = {1, 2}
2   b = {2, 3}
3   print(a.issuperset(b)) # False
```

🔥 OPERACIONES DE CONJUNTOS

isdisjoint() → Es disjunto:
Devuelve True si no comparten ningún elemento.

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

🔥 OPERACIONES DE CONJUNTOS

update() → Es actualizar:
Agrega al conjunto A todos los elementos del conjunto B (modifica A).

```python
a = {1, 2}
b = {2, 3}
a.update(b)
print(a) # {1, 2, 3}
```

🔥 OPERACIONES DE CONJUNTOS

intersection_update() → Es actualización e intersección:
Deja en A solo los elementos que están en ambos conjuntos (modifica A).

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

🔥 OPERACIONES DE CONJUNTOS

difference_update() → Es actualización y diferencia:
Elimina de A los elementos que están en B (modifica A).

```python
a = {1, 2}
b = {2, 3}
a.difference_update(b)
print(a) # {1}
```

🔥 OPERACIONES DE CONJUNTOS

symmetric_difference_update() → actualización de diferencia simétrica: Deja en A solo los elementos que no se repiten entre A y B (modifica A).

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

EJEMPLO COMPLETO

```python
numeros = {1, 2, 3}
numeros.add(4)
numeros.add(2)          # ignorado por ser duplicado
print(numeros)          # {1, 2, 3, 4}

numeros.remove(3)
print(numeros)          # {1, 2, 4}

otros = {4, 5, 6}

print(numeros.union(otros))             # {1, 2, 4, 5, 6}
print(numeros.intersection(otros))      # {4}
print(numeros.difference(otros))        # {1, 2}
```

📌 RESUMEN

- Set: Colección no ordenada y sin duplicados
- Definición: {} o set()
- Mutable: ✔ Sí
- Acceso por índice: ❌ No

📌 RESUMEN

- Set: Colección no ordenada y sin duplicados
- Definición: {} o set()
- Métodos clave: add, remove, discard, clear
- Operaciones: union, intersection, difference, etc.

NOS VEMOS EN UN PRÓXIMO VIDEO DE ESTE CURSO, SALUDOS 🚀