1. add(elem): Añade un elemento al conjunto. Forma de uso: conjunto.add(elemento) Ejemplo básico:

python

```python
frutas = {"manzana", "banana"}
frutas.add("cereza")
print(frutas)  # {'manzana', 'banana', 'cereza'}
```

Ejemplo avanzado:

python

```python
def agregar_tags(post, nuevos_tags):
    post['tags'] = set(post.get('tags', []))
    for tag in nuevos_tags:
        post['tags'].add(tag.lower())
    return post

post = {"titulo": "Python Tips", "tags": ["programación"]}
post = agregar_tags(post, ["Python", "Coding", "TIPS"])
print(post)  # {'titulo': 'Python Tips', 'tags': {'programación', 'python', 'coding', 'tips'}}
```

2. remove(elem): Elimina un elemento del conjunto. Lanza un KeyError si el elemento no existe. Forma de uso: conjunto.remove(elemento) Ejemplo básico:

python

```python
numeros = {1, 2, 3, 4, 5}
numeros.remove(3)
print(numeros)  # {1, 2, 4, 5}
```

Ejemplo avanzado:

python

```python
def eliminar_usuario(usuarios, usuario_id):
    try:
        usuarios.remove(usuario_id)
        return f"Usuario {usuario_id} eliminado correctamente."
```

```python
    except KeyError:
        return f"Usuario {usuario_id} no encontrado."

usuarios_activos = {101, 102, 103, 104}
print(eliminar_usuario(usuarios_activos, 102))  # "Usuario 102 eliminado correctamente."
print(eliminar_usuario(usuarios_activos, 105))  # "Usuario 105 no encontrado."
```

3. discard(elem): Elimina un elemento del conjunto si está presente. Forma de uso: conjunto.discard(elemento) Ejemplo básico:

python

```python
colores = {"rojo", "verde", "azul"}
colores.discard("amarillo")  # No lanza error si el elemento no existe
print(colores)  # {'rojo', 'verde', 'azul'}
```

Ejemplo avanzado:

python

```python
def limpiar_permisos(usuario, permisos_a_quitar):
    for permiso in permisos_a_quitar:
        usuario['permisos'].discard(permiso)
    return usuario

usuario = {"nombre": "Ana", "permisos": {"leer", "escribir", "ejecutar"}}
usuario = limpiar_permisos(usuario, ["ejecutar", "administrar"])
print(usuario)  # {'nombre': 'Ana', 'permisos': {'leer', 'escribir'}}
```

4. pop(): Elimina y devuelve un elemento arbitrario del conjunto. Lanza KeyError si el conjunto está vacío. Forma de uso: elemento = conjunto.pop() Ejemplo básico:

python

```python
numeros = {1, 2, 3}
elemento = numeros.pop()
print(elemento)  # Podría ser 1, 2 o 3
print(numeros)  # El conjunto ahora tiene dos elementos
```

Ejemplo avanzado:

python

```python
def procesar_cola_tareas(cola_tareas):
    tareas_procesadas = set()
    while cola_tareas:
        tarea = cola_tareas.pop()
        print(f"Procesando tarea: {tarea}")
        tareas_procesadas.add(tarea)
    return tareas_procesadas

tareas = {"Tarea1", "Tarea2", "Tarea3"}
procesadas = procesar_cola_tareas(tareas)
print("Tareas procesadas:", procesadas)
```

5. clear(): Elimina todos los elementos del conjunto. Forma de uso: conjunto.clear() Ejemplo básico:

python

```python
numeros = {1, 2, 3, 4, 5}
numeros.clear()
print(numeros)  # set()
```

Ejemplo avanzado:

python

```python
def reiniciar_sistema(estado):
    for subsistema in estado:
        estado[subsistema].clear()
    return "Sistema reiniciado"

estado_sistema = {
    "procesos": {1234, 5678, 9012},
    "conexiones": {"192.168.1.1", "10.0.0.1"},
    "cache": {"datos1", "datos2"}
}
```

```python
print(reiniciar_sistema(estado_sistema))
print(estado_sistema)  # {'procesos': set(), 'conexiones': set(), 'cache': set()}
```

6. copy(): Devuelve una copia superficial del conjunto. Forma de uso: nuevo_conjunto = conjunto.copy() Ejemplo básico:

python

```python
original = {1, 2, 3}
copia = original.copy()
copia.add(4)
print(original)  # {1, 2, 3}
print(copia)     # {1, 2, 3, 4}
```

Ejemplo avanzado:

python

```python
def crear_grupos(estudiantes, num_grupos):
    grupos = [set() for _ in range(num_grupos)]
    estudiantes_copia = estudiantes.copy()
    for i, estudiante in enumerate(estudiantes_copia):
        grupos[i % num_grupos].add(estudiante)
    return grupos

estudiantes = {"Ana", "Bob", "Carlos", "Diana", "Eva"}
grupos = crear_grupos(estudiantes, 2)
print(grupos)  # [{'Ana', 'Carlos', 'Eva'}, {'Bob', 'Diana'}]
print(estudiantes)  # {'Ana', 'Bob', 'Carlos', 'Diana', 'Eva'}
```

7. union(other_set): Devuelve un nuevo conjunto con elementos de ambos conjuntos. Forma de uso: nuevo_conjunto = conjunto1.union(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
```

```python
union = set1.union(set2)
print(union)  # {1, 2, 3, 4, 5}
```

Ejemplo avanzado:

python

```python
def combinar_preferencias(usuarios):
    todas_preferencias = set()
    for usuario in usuarios:
        todas_preferencias = todas_preferencias.union(usuario['preferencias'])
    return todas_preferencias

usuarios = [
    {"nombre": "Ana", "preferencias": {"jazz", "café", "libros"}},
    {"nombre": "Bob", "preferencias": {"rock", "té", "películas"}},
    {"nombre": "Carlos", "preferencias": {"café", "deportes", "viajes"}}
]
print(combinar_preferencias(usuarios))
# {'jazz', 'café', 'libros', 'rock', 'té', 'películas', 'deportes', 'viajes'}
```

8. intersection(other_set): Devuelve un nuevo conjunto con elementos comunes a ambos conjuntos. Forma de uso: nuevo_conjunto = conjunto1.intersection(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
interseccion = set1.intersection(set2)
print(interseccion)  # {3, 4}
```

Ejemplo avanzado:

python

```python
def encontrar_intereses_comunes(usuarios):
    if not usuarios:
        return set()
```

```
    intereses_comunes = usuarios[0]['intereses']
    for usuario in usuarios[1:]:
        intereses_comunes = intereses_comunes.intersection(usuario['intereses'])
    return intereses_comunes

usuarios = [
    {"nombre": "Ana", "intereses": {"python", "data science", "música"}},
    {"nombre": "Bob", "intereses": {"python", "videojuegos", "música"}},
    {"nombre": "Carlos", "intereses": {"java", "música", "python"}}
]
print(encontrar_intereses_comunes(usuarios))  # {'python', 'música'}
```

9. difference(other_set): Devuelve un nuevo conjunto con elementos en el conjunto original pero no en el otro. Forma de uso: nuevo_conjunto = conjunto1.difference(conjunto2) Ejemplo básico:

python

```
set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7}
diff = set1.difference(set2)
print(diff)  # {1, 2, 3}
```

Ejemplo avanzado:

python

```
def encontrar_habilidades_unicas(equipo1, equipo2):
    habilidades_equipo1 = set(equipo1['habilidades'])
    habilidades_equipo2 = set(equipo2['habilidades'])
    return {
        equipo1['nombre']: habilidades_equipo1.difference(habilidades_equipo2),
        equipo2['nombre']: habilidades_equipo2.difference(habilidades_equipo1)
    }

equipo_a = {"nombre": "Equipo A", "habilidades": ["python", "javascript", "react"]}
equipo_b = {"nombre": "Equipo B", "habilidades": ["python", "java", "docker"]}
```

```python
print(encontrar_habilidades_unicas(equipo_a, equipo_b))
# {'Equipo A': {'javascript', 'react'}, 'Equipo B': {'java', 'docker'}}
```

10. symmetric_difference(other_set): Devuelve un nuevo conjunto con elementos en cualquiera de los conjuntos, pero no en ambos. Forma de uso: nuevo_conjunto = conjunto1.symmetric_difference(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
sym_diff = set1.symmetric_difference(set2)
print(sym_diff)  # {1, 2, 5, 6}
```

Ejemplo avanzado:

python

```python
def comparar_vocabularios(texto1, texto2):
    palabras1 = set(texto1.lower().split())
    palabras2 = set(texto2.lower().split())
    return palabras1.symmetric_difference(palabras2)

texto_a = "El rápido zorro marrón salta sobre el perro perezoso"
texto_b = "El perro perezoso duerme todo el día"
print(comparar_vocabularios(texto_a, texto_b))
# {'rápido', 'zorro', 'marrón', 'salta', 'sobre', 'duerme', 'todo', 'día'}
```

11. issubset(other_set): Verifica si el conjunto es un subconjunto de otro. Forma de uso: conjunto1.issubset(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2}
set2 = {1, 2, 3, 4, 5}
print(set1.issubset(set2))  # True
```

Ejemplo avanzado:

python

```python
def verificar_requisitos(habilidades_requeridas, candidatos):
    return [candidato for candidato in candidatos
            if set(habilidades_requeridas).issubset(candidato['habilidades'])]

requisitos = {'python', 'sql'}
candidatos = [
    {'nombre': 'Ana', 'habilidades': {'python', 'sql', 'java'}},
    {'nombre': 'Bob', 'habilidades': {'python', 'javascript'}},
    {'nombre': 'Carlos', 'habilidades': {'python', 'sql', 'docker'}}
]
print(verificar_requisitos(requisitos, candidatos))
# [{'nombre': 'Ana', 'habilidades': {'python', 'sql', 'java'}},
#  {'nombre': 'Carlos', 'habilidades': {'python', 'sql', 'docker'}}]
```

12. issuperset(other_set): Verifica si el conjunto es un superconjunto de otro. Forma de uso: conjunto1.issuperset(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2, 3, 4, 5}
set2 = {2, 3}
print(set1.issuperset(set2))  # True
```

Ejemplo avanzado:

python

```python
def encontrar_categorias_principales(categorias, subcategorias):
    return [cat for cat, items in categorias.items()
            if set(items).issuperset(subcategorias)]

categorias = {
    'frutas': {'manzana', 'banana', 'cereza', 'dátil'},
    'verduras': {'zanahoria', 'brócoli', 'espinaca'},
    'lacteos': {'leche', 'queso', 'yogur'}
}
```

```python
subcategorias = {'manzana', 'banana'}
print(encontrar_categorias_principales(categorias, subcategorias))  # ['frutas']
```

13. update(other_set): Actualiza el conjunto, añadiendo elementos de otro conjunto. Forma de uso: conjunto1.update(conjunto2) Ejemplo básico:

python

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
set1.update(set2)
print(set1)  # {1, 2, 3, 4, 5}
```

Ejemplo avanzado:

python

```python
def actualizar_inventario(inventario, nuevos_items):
    for categoria, items in nuevos_items.items():
        if categoria in inventario:
            inventario[categoria].update(items)
        else:
            inventario[categoria] = set(items)
    return inventario

inventario = {'electrónica': {'laptop', 'smartphone'}, 'libros': {'novela', 'biografía'}}
nuevos_items = {'electrónica': {'tablet', 'smartwatch'}, 'ropa': {'camiseta', 'pantalón'}}
print(actualizar_inventario(inventario, nuevos_items))
# {'electrónica': {'laptop', 'smartphone', 'tablet', 'smartwatch'},
#  'libros': {'novela', 'biografía'},
#  'ropa': {'camiseta', 'pantalón'}}
```