Tipo de datos conjuntos: set, frozenset

set

Los conjuntos (`set`): Me permiten guardar conjuntos (desornedos) de datos (a los que se puede calcular una función hash), en los que no existen repeticiones. Es un tipo de datos mutable.

Normalmente se usan para comprobar si existe un elemento en el conjunto, eliminar duplicados y cálculos matématicos, como la intersección, unión, diferencia,...

Definición de set. Constructor set

Podemos definir un tipo `set` de distintas formas:

```
>>> set1 = set()
>>> set1
set()
>>> set2=set([1,1,2,2,3,3])
>>> set2
{1, 2, 3}
>>> set3={1,2,3}
>>> set3
{1, 2, 3}
```

Frozenset

El tipo `frozenset` es un tipo inmutable de conjuntos.

Definición de frozenset. Constructor frozenset

```
>>> fs1=frozenset()
>>> fs1
frozenset()
>>> fs2=frozenset([1,1,2,2,3,3])
>>> fs2
frozenset({1, 2, 3})
```

Operaciones básicas con set y frozenset

De las operaciones que estudiamos en el apartado "Tipo de datos secuencia" los conjuntos sólo aceptan las siguientes:

- Recorrido
- Operadores de pertenencia: `in` y `not in` .

Entre las funciones definidas podemos usar: `len` , `max` , `min` , `sorted` .

Los set son mutables, los frozenset son inmutables

```
>>> set1={1,2,3}
>>> set1.add(4)
>>> set1
{1, 2, 3, 4}
>>> set1.remove(2)
>>> set1
{1, 3, 4}
```

El tipo `frozenset` es inmutable por lo tanto no posee los métodos `add` y `remove` .

Métodos de set y frozenset

```
set1.add                          set1.issubset
set1.clear                        set1.issuperset
set1.copy                         set1.pop
set1.difference                   set1.remove
set1.difference_update            set1.symmetric_difference
set1.discard                      set1.symmetric_difference_update
set1.intersection                 set1.union
set1.intersection_update          set1.update
set1.isdisjoint
```

Veamos algunos métodos, partiendo siempre de estos dos conjuntos:

```
>>> set1={1,2,3}
>>> set2={2,3,4}

>>> set1.difference(set2)
{1}
>>> set1.difference_update(set2)
>>> set1
{1}

>>> set1.symmetric_difference(set2)
{1, 4}
>>> set1.symmetric_difference_update(set2)
>>> set1
{1, 4}

>>> set1.intersection(set2)
{2, 3}
>>> set1.intersection_update(set2)
>>> set1
{2, 3}

>>> set1.union(set2)
{1, 2, 3, 4}
>>> set1.update(set2)
>>> set1
{1, 2, 3, 4}
```

Veamos los métodos de añadir y eliminar elementos:

```
>>> set1 = set()
>>> set1.add(1)
>>> set1.add(2)
>>> set1
{1, 2}
>>> set1.discard(3)
>>> set1.remove(3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 3
>>> set1.pop()
1
>>> set1
{2}
```

Y los métodos de comprobación:

```
>>> set1 = {1,2,3}
>>> set2 = {1,2,3,4}
>>> set1.isdisjoint(set2)
False
>>> set1.issubset(set2)
True
>>> set1.issuperset(set2)
False
>>> set2.issuperset(set1)
True
```

Por último los métodos de `frozenset`:

```
fset1.copy                  fset1.isdisjoint            fset1.symmetric_difference
fset1.difference            fset1.issubset              fset1.union
fset1.intersection          fset1.issuperset
```