Conjuntos y diccionarios

Computación – Curso Servetto

Conjuntos

Python incluye un tipo de datos para conjuntos. Un conjunto es una colección desordenada de elementos sin duplicados. Los usos básicos incluyen la prueba de pertenencia y la eliminación de elementos duplicadas. Los conjuntos también soportan operaciones matemáticas como la unión, la intersección, la diferencia y la diferencia simétrica.

Para crear conjuntos se pueden utilizar llaves o la función set().

Nota: para crear un conjunto vacío hay que utilizar set(), no {}; esto último crea un diccionario vacío (estructura que se ve en la próxima sección).

Ejemplos

```
>>> frutas = {'manzana', 'naranja', 'manzana ', 'pera', 'naranja', 'banana'}
>>> frutas                  # muestra eliminación de duplicados
{'pera', 'banana', 'manzana', 'naranja'}
>>># verificación rápida de pertenencia
>>> 'naranja' in frutas
True
>>> 'ciruela' in frutas
False
>>> frutas |= {'ciruela'}     # incorporación de nuevo elemento con operador de unión
>>> frutas
{'banana', 'naranja', 'manzana', 'pera', 'ciruela'}
>>> frutas -= {'pera'}        # eliminación de elemento con operador de diferencia
>>> frutas
{'banana', 'naranja', 'manzana', 'ciruela'}
>>> # Demostración de operaciones de conjuntos
>>> A={'Juan', 'Esteban', 'Laura', 'Federico', 'Lorena'}
>>> B={'Agustín', 'Alejo', 'Esteban', 'Ana', 'Laura'}
>>> A-B                     # diferencia
{'Federico', 'Lorena', 'Juan'}
>>> B-A
{'Alejo', 'Ana', 'Agustín'}
>>> A|B                     # unión
{'Federico', 'Agustín', 'Ana', 'Lorena', 'Alejo', 'Esteban', 'Laura', 'Juan'}
>>> A&B                     # intersección
{'Esteban', 'Laura'}
>>> A^B                     # diferencia simétrica
{'Ana', 'Lorena', 'Alejo', 'Federico', 'Agustín', 'Juan'}
>>> (A-B)|(B-A)             # expresión equivalente de diferencia simétrica
{'Ana', 'Lorena', 'Alejo', 'Federico', 'Agustín', 'Juan'}
>>> (A|B)-(A&B)             # expresión equivalente de diferencia simétrica
{'Agustín', 'Ana', 'Lorena', 'Alejo', 'Federico', 'Juan'}
```

Diccionarios

A diferencia de las listas y tuplas, que están indexadas por un rango de números, los diccionarios están indexados por claves, que pueden ser de cualquier tipo inmutable; las cadenas y los números siempre pueden ser claves.

Es mejor pensar en un diccionario como un conjunto de pares *clave: valor*, con el requisito de que las claves sean únicas (dentro de un diccionario). Un par de llaves crea un diccionario vacío: {}. Colocando una lista de pares "*clave: valor*" separados por comas dentro de las llaves se añaden los pares al diccionario; esta es también la forma en que los diccionarios se escriben en pantalla.

Las principales operaciones sobre un diccionario representado por una variable d, son almacenar un valor con alguna clave con *d[clave]=valor* y extraer el valor dada la clave con *d[clave]*. También es posible borrar un par *clave: valor* con *del d[clave]*. Si se almacena un valor utilizando una clave que ya está en uso, el valor antiguo asociado a esa clave se olvida. Si se intenta extraer un valor utilizando una clave inexistente se produce un error.

Si se ejecuta *list(d)* con un diccionario, se obtiene una lista de todas las claves utilizadas en el diccionario, en orden de inserción (si se quiere ordenar, basta con utilizar *sorted(d)*). Para comprobar si una sola clave está en el diccionario se utiliza la palabra clave *in*.

Ejemplos

```
>>> palabras={}
>>> palabras
{}
>>> palabras['FIUBA']=1
>>> palabras
{'FIUBA': 1}
>>> palabras['UBA']=5
>>> palabras
{'FIUBA': 1, 'UBA': 5}
>>> 'UBA' in palabras
True
>>> palabras['FIUBA']+=1
>>> palabras
{'FIUBA': 2, 'UBA': 5}
>>> for clave, valor in palabras.items():
        print(clave, valor)


FIUBA 2
UBA 5
```