COLECCIONES EN PYTHON (3)

set (Conjuntos)

Las colecciones en Python son estructuras de datos que permiten agrupar y organizar elementos de forma eficiente. En otros entornos se conocen como arreglos (Arrays, vectores, Matrices…) Python ofrece 4 colecciones integradas: Listas, Diccionarios (dictionary) y Conjuntos (set).

Objetivos

1. Aprender a utilizar los sets y sobre su manipulación en Python
2. Diferenciar los sets de los diccionarios, listas y tuplas
3. Utilizar operaciones de iterables con un set.
4. Utilizar métodos propios de los sets

set

Un set, también llamado conjunto, es una colección de elementos desordenados que no admite duplicados. Se trata, por tanto, de una estructura de datos equivalente a los conjuntos en matemáticas. Además, como sucede con los elementos de las listas o las claves de los diccionarios, los elementos de un set no necesariamente han de ser del mismo tipo.

Los **set**, se utilizan para almacenar varios elementos en una sola variable.

El **set** es uno de los 4 tipos de datos incorporados en Python que se utilizan para almacenar colecciones de datos, los otros 3 son Lista, Tupla y Diccionario, todos con diferentes cualidades y uso.

Un conjunto es una colección que no está ordenada, no se puede cambiar* y no está indexada

Los set se escriben con llaves: **{ ….. }**

```
IIR2 = {"apple", "banana", "cherry"}
print(IIR2)
print(type(IIR2))
```

Elementos de un set

Los elementos establecidos no están ordenados, no se pueden cambiar y no permiten valores duplicados.

Desordenado

Desordenado significa que los elementos de un conjunto no tienen un orden definido. Los elementos de conjunto pueden aparecer en un orden diferente cada vez que se usan y *no se puede hacer referencia a ellos mediante índice o clave.*

Invariable

Los elementos del conjunto no se pueden cambiar, lo que significa que no podemos cambiar los elementos después de que se haya creado el conjunto.

Una vez creado un conjunto, no se pueden cambiar sus elementos, pero se pueden eliminar elementos y añadir nuevos.

No se permiten duplicados

Los conjuntos no pueden tener dos elementos con el mismo valor.

```
thisset = {"apple", "banana", "cherry", "apple"}
print(thisset)
```

True y 1 se considera el mismo valor:

```
thisset = {"apple", "banana", "cherry", True, 1, 2}
print(thisset)
```

Nota: Los valores False y 0 se consideran el mismo valor en los conjuntos y se tratan como duplicados

```
thisset = {"apple", "banana", "cherry", False, True, 0}
print(thisset)
```

Longitud

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

Diferentes tipos de datos

```
et1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
```

Tipos de datos varios en un mismo set

```
set1 = {"abc", 34, True, 40, "male"}
```

Tipo de dato

```
myset = {"apple", "banana", "cherry"}
print(type(myset))
```

Constructor set

```
thisset = set(("apple", "banana", "cherry")) # note doble parentesis
print(thisset)
```

Acceso a los ítems como iterable

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

Vrificar si un item esta en el set “banana” o si no esta

```
thisset = {"apple", "banana", "cherry"}
print("banana" in thisset)
```

```
thisset = {"apple", "banana", "cherry"}
print("banana" not in thisset)
```

Una vez creado un **set**, no se pueden cambiar sus elementos, pero se pueden añadir nuevos.

Adicionar ítems

```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)
```

Adicioanr ítems de un set a otro

```
thisset = {"apple", "banana", "cherry"}
tropical = {"pineapple", "mango", "papaya"}
thisset.update(tropical)
print(thisset)
```

El objeto en el método update() no tiene que ser un **set**, puede ser cualquier objeto iterable (tuplas, listas, diccionarios, etc.).

Adicionando una lista al set

```
thisset = {"apple", "banana", "cherry"}
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)
```

Eliminando ítems del set
Si el item no existe remove(9 genera error

```
thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)
```

Eliminar sin error con discard

```
thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

Eliminar item aleatorio
También puede usar el método pop() para eliminar un elemento, pero este método eliminará un elemento aleatorio, por lo que no puede estar seguro de qué elemento se elimina.

El valor devuelto del método pop() es el elemento eliminado.

Los **set** no están ordenados, por lo que cuando se usa el método pop(), no se sabe qué elemento se elimina.

```
thisset = {"apple", "banana", "cherry"}
x = IIRR.pop()
print(x)
print(IIRR)
```

Vaciar el **set – sigue existiendo**

```
IIRR = {"apple", "banana", "cherry"}
IIRR.clear()
print(IIRR)
```

Eliminar completamente el **set**
La palabra clave **del** eliminará el conjunto por completo: al no existir genera error

```
IIRR = {"apple", "banana", "cherry"}
del IIRR
print(IIRR)
```

Combinando Set

Hay varias formas de unir dos o más conjuntos en Python.

Los métodos union() y update() unen todos los elementos de ambos conjuntos.

El método intersection() mantiene SOLO los duplicados.

El método difference() mantiene los elementos del primer conjunto que no están en los otros conjuntos.

El método difference() mantiene todos los elementos EXCEPTO los duplicados.

Union

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
```

```
Con el operador: |  obtiene el mismo resultado

set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = set1 | set2
print(set3)
```

Union multiple

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1.union(set2, set3, set4)
print(myset)
```

Otra opción:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set3 = {"John", "Elena"}
set4 = {"apple", "bananas", "cherry"}
myset = set1 | set2 | set3 |set4
print(myset)
```

Unir set y tupla

```
x = {"a", "b", "c"}
y = (1, 2, 3)
z = x.union(y)
print(z)
```

El método **update**() inserta los elementos de set2 en set1:

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1)
```

Ejercicios

1. Verifique si esta presente “apple”

```
fruits = {"apple", "banana", "cherry"}
if "apple"      fruits:
  print("Yes, apple is a fruit!")
```

2. Crear una lista con países y un set con ciudades
   Unirlos en un set

3. Describa que hace el siguiente codigo

```
animales_terrestres = set(["Tortuga", "Lobo", "Perro", "Cangrejo"])
print("Terrestres: ", animales_terrestres)
animales_acuaticos = set(["Tiburón", "Pulpo", "Tortuga", "Cangrejo"])
print("Acuáticos: ", animales_acuaticos)

todos_los_animales = animales_terrestres.union(animales_acuaticos)
print("Unión: ", todos_los_animales)
```

4. Recuerde otras colecciones y vaya tomando las diferencias
   (Listas, Tuplas, sets)

Consulte estos otros métodos para manipular set

Method
add()
clear()
copy()
difference()
difference_update()
discard()
intersection()
intersection_update()
isdisjoint()
issubset()
issuperset()
pop()
remove()
symmetric_difference()
symmetric_difference_update()
union()
update()

Fecha Creación	Enero 27 2024
Responsable	Plinio Neira Vargas
Revisado por	Sonia Escobar
Fecha Revisión	Enero 28 2024