ProfesorenVideo.com

Colecciones de datos

¿Qué són?: Las colecciones son conjuntos de datos del mismo tipo, o diferente, que se almacenan dentro de una única variable.

Las cuatro colecciones de datos que maneja el lenguaje Python son:

- Listas (list).
- Tuplas (tuple).
- Conjuntos (set). ←
- Diccionarios (dictionary).

Los elementos individuales dentro de las listas y las tuplas eran accedidos mediante subíndices que iban dentro de un par de corchetes([ ]). Ejemplos:

miLista[subindice]

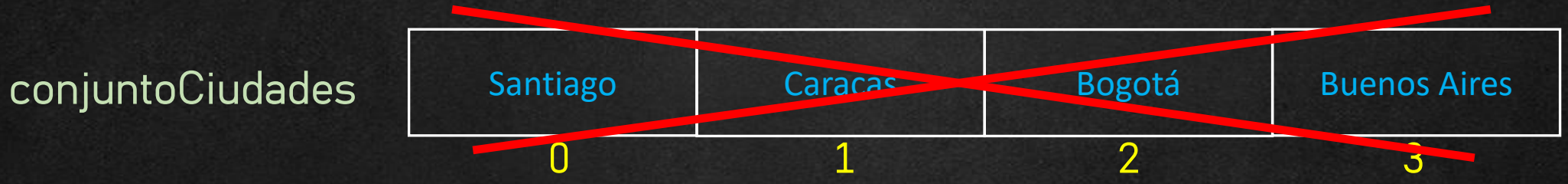
miTupla[subindice]

ProfesorenVideo.com

Conjuntos

Los conjuntos también son usados para para almacenar múltiples elementos en una única variable. Se escriben usando llaves { } .

conjuntoCiudades = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” }



IMPORTANTE: Los elementos individuales de un conjunto NO pueden ser accedidos a través de subíndices. Su ubicación viene dada por un número HASH.

Características de los conjuntos: Son DESORDENADOS, INMUTABLES y NO aceptan DUPLICADOS.

Conjuntos

Los conjuntos también son usados para para almacenar múltiples elementos en una única variable. Se escriben usando llaves { } .

conjuntoCiudades = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” }

Características de los conjuntos: Son INMUTABLES, NO aceptan DUPLICADOS y NO son ORDENADOS.

INMUTABLES (*): Quiere decir que los elementos una vez agregados NO se pueden modificar.

* IMPORTANTE: Los conjuntos a pesar de ser inmutables SÍ permiten remover elementos y agregar nuevos elementos.

ProfesorenVideo.com

Conjuntos

conjuntoCiudades = (“Santiago", “Caracas", “Bogotá“, “Buenos Aires”)

NO son ordenados: Quiere decir que en cada ejecución del programa que los declara, ellos se pueden almacenar en un orden distinto :

```
print (conjuntoCiudades)     # (“Caracas", “Buenos Aires”, “Santiago", “Bogotá“)
```

En otra ejecución del mismo programa:

```
print (conjuntoCiudades)    # (“Bogotá",“Buenos Aires”,“Caracas", “Santiago“)
```

Conjuntos

NO aceptan duplicados: Los elementos de un conjunto deben ser únicos, es decir NO aceptan duplicados :

```
conjuntoCiudades = ("Santiago", "Caracas", "Bogotá", "Buenos Aires")      # Es válido
conjuntoCiudades = ("Santiago", "Caracas", "Caracas", "Buenos Aires")     # Descartado
```

IMPORTANTE: En Python los valores True y 1 representan lo mismo, por tanto si intentamos colocar los dos valores dentro de un conjunto se consideran duplicados.

Exactamente lo mismo ocurre con los valores False y 0.

NOTA: En cualquier situación cuando se intenta agregar un elemento repetido, Python acepta el primero que aparezca y descarta al segundo.

Conjuntos

Ejemplo de elementos duplicados:

Al intentar declarar los siguientes conjuntos que contienen duplicados:

conjuntoCiudades = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” , True, 25, 1 , “Bogotá”}

conjuntoCiudades2 = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” , 0, 27, False }

Python sólo acepta lo siguiente:

conjuntoCiudades = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” , True, 25}

conjuntoCiudades2 = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” , 0, 27 }

ProfesorenVideo.com

Conjuntos

Los conjuntos son NO ordenados: Esto quiere decir que los elementos dentro de un conjunto no disponen de una posición fija. Por ejemplo, si declaramos el siguiente conjunto:

conjuntoCiudades = { “Santiago", “Caracas", “Bogotá“, “Buenos Aires” }

Es muy probable que en la primera ejecución o corrida del programa, al mostrar dicho conjunto con un “print(conjuntoCiudades)” tal vez aparezca en pantalla lo siguiente:

{ “Bogotá", “Buenos Aires", “Caracas“, “Santiago” }

Y en una segunda ejecución del mismo programa aparezca la siguiente salida:

{ “Buenos Aires", “Santiago", “Caracas“, “Bogotá” }

ProfesorenVideo.com

Conjuntos

Porqué los conjuntos son NO ordenados ?:

Esto se debe a que los elementos de un conjunto, al igual que los diccionarios, se almacenan dentro de tablas HASH.

La ubicación real de cada elemento de conjunto dentro de la tabla hash viene dada por un número hash. Éste número hash es devuelto por una función hash.

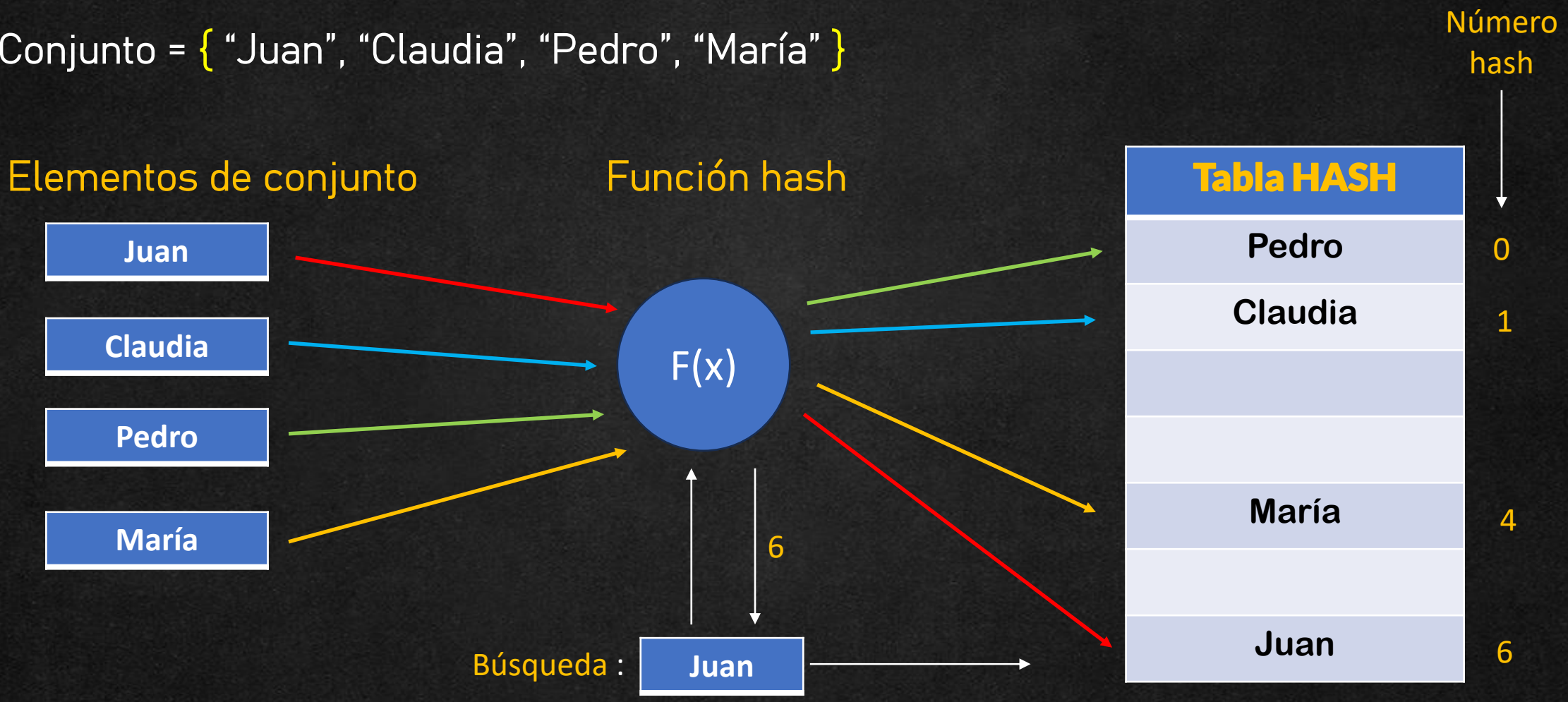
La función hash en cada nueva corrida o ejecución hace uso de una semilla (seed) aleatoria distinta, por lo tanto el número hash asociado a un elemento de conjunto será distinto en cada nueva ejecución.

Veámoslo de manera gráfica para su mejor comprensión ...

ProfesorenVideo.com

Conjuntos

miConjunto = { “Juan”, “Claudia”, “Pedro”, “María” }



ProfesorenVideo.com

Conjuntos

IMPORTANTE !!!

NO se preocupe por la implementación de los algoritmos de tablas HASH, sólo preocúpese por aprender las características de los conjuntos y aprender a trabajar con dichas colecciones de datos. El resto del trabajo lo hace Python por usted.

ProfesorenVideo.com

Características

Lista	Tupla	Conjunto
list	**tuple**	**set**
- Ordenadas	- Ordenadas	- NO son ordenados
- Aceptan duplicados	- Aceptan duplicados	- NO aceptan duplicados
- Son modificables	- NO son modificables	- NO son modificables *
- Se escriben con [ ]	- Se escriben con ( )	- Se escriben con { }
- Usan subíndices: lista[índice]	- Usan subíndices: tupla[índice]	- NO usan subíndices: usa posiciones hash

Muchas gracias por su atención