Python [Lists]

1. Python Lists
2. Acceder a valores
3. Actualizar valores
4. Insertar valores
5. Eliminar valores
6. Iterar valores
7. Ordenar listas
8. Copiar listas
9. Concatenar listas
10. Lists methods

Python [Lists]

Python [Lists]

En Python, existen **cuatro estructuras integradas** para **almacenar múltiples valores/objetos en una sola variable iterativa**: Listas, Tuplas, Sets y Diccionarios.
Las listas **se comportan como un vector** clásico tal y como podemos encontrar en otros lenguajes.

- Las listas se representan con claudátors: **coches = ["Audi", "BMW", "Volkswagen"]**
- Las listas son indexadas.
- El primer elemento de una lista tiene índice 0 y consecutivamente.
- Se dice que las listas son ordenadas puesto que cada elemento tiene un valor de índice distinto, pero sus elementos se almacenarán en el orden de inserción.
- Los valores de las listas pueden ser actualizados.
- Las listas aceptan inserciones, eliminaciones y modificaciones de ítems.
- Las listas aceptan valores repetidos.
- Una lista puede contener elementos de distinto tipaje: **lista = ["abc", 34, True, 40, "male"]**

Acceder a valores

Python [Lists]: **acceder** a valores

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

Acceder	Python code
Un valor mediante **índice** *(error si 'out of bounds')*	**print**(coches**[2]**)
Un valor mediante **índice negativo** *(último elemento)*	**print**(coches**[-1]**)
Múltiples valores usando **rango** *(tres opciones)* **(incluye el primer índice del rango ¡pero no el último!)* ***(si obviamos el fin de rango, muestra hasta el final)*	**print**(coches**[0:2]**) **/** **print**(coches**[:2]**) **/** **print**(coches**[1:]**)
¿Un **valor existe** en lista?	**if** "BMW" **in** coches: **print**("Sí, hay un BMW en la lista")

Actualizar valores

Python [Lists]: **actualizar** valores

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

Actualizar	Python code
Un valor existente mediante el **índice**	coches**[1]** = **"Renault"**
Un rango de elementos Esto cambiará los elementos coches[0] y coches[1] **(Recuerda que el índice final del rango no está incluido)*	coches**[0:2]** = **["Renault", "Mercedes"]**
Cambiar un elemento por dos **Con esto hemos cambiado el valor "Audi" por "Citroën" y "Lamborgini" haciendo crecer la lista con un elemento más. Podrías hacer también la acción inversa: cambiar dos elementos por uno.*	coches**[0:1]** = **["Citroën", "Lamborgini"]**

Insertar valores

Python [Lists]: **insertar** valores

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

Insertar	Python code
Elemento(s) **al final**	coches.**append("Dacia")**
Elemento(s) **en un índice**	coches.**insert(2, "Peugeot")**
Una lista *(u otra estructura iterativa)* **al final**	coches.**extend(["Lamborgini, "Lexus"])**

Eliminar valores

Python [Lists]: **eliminar** valores

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

Eliminar	Python code
Un **valor**	coches.**remove("Audi")**
Un **elemento** por **índice** *(dos opciones)*	coches.**pop(2)** / **del** coches**[2]**
El **último** elemento	coches.**pop()**
Todos los **valores**	coches.**clear()**
Todos los **valores** y **la estructura**	**del coches**

Iterar valores

Python [Lists]: **iterar** valores

Iterar	Python code

while	**while** i < **len**(coches)**:**
	  print(coches**[i])**
	  i = i + 1

for	**for** i **in range(len**(coches))**:** **print(**coches**[i])**
for-each	**for** c **in** coches**:** **print(**c**)**
for-enumerate	**for** i, c **in enumerate(**coches**):** **print(**i, "->", c**)**
List Comprehensive shorthand	**[print(**c**) for** c **in** coches**]**

https://docs.python.org/2/tutorial/datastructures.html#looping-techniques

Ordenar valores

Python [Lists]: **ordenar** valores

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

Ordenar	Python code
Numérica o **alfabéticamente**	coches**.sort()**
Descendientemente *(dos opciones)*	coches**.sort(reverse = true)** / coches**.reverse()**
Alfabéticamente *case insensitive*	coches**.sort(key = str.lower)**

Copiar listas

Python [Lists]: **copiar listas**

Dada una lista: **coches = ["Audi", "BMW", "Volkswagen"]**

- En Python una lista no se puede copiar usando el operador de asignación '='. En su lugar, esto provoca que dos variables apunten a la misma estructura guardada en RAM.
- Para crear una nueva variable que contenga la misma estructura y valores que otra lista dada, tenemos dos opciones:

cochesB = coches**.copy()** ó **cochesC = list(**coches**)**

Concatenar listas

Python [Lists]: **concatenar listas**

Dadas dos listas:

```
coches = ["Audi", "BMW", "Volkswagen"]
motos = ["Vespa", "Honda", "Suzuki"]
```

- Disponemos de tres formas para concatenarlas:

```
vehículos = coches + motos        ''' Crea una nueva lista con los valores de coches y motos '''

coches.extend(motos)              ''' Añade los valores de motos al final de coches '''

for c in motos:                   ''' Añade los valores de motos al final de coches '''
    coches.append(c)
```

[Lists] methods

Python [Lists]: **methods**

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Fuente: https://www.w3schools.com/python/python_lists_methods.asp

“Mediocre es aquel que exige mucho a los demás y poco de si mismo.”

Séneca (4 aC – 65 dC)