LISTAS

Básicamente, una *lista* es una colección ordenada de objetos, similar al *array dinámico* empleado en otros lenguajes de programación. Puede contener distintos tipos de objetos, es mutable y Python nos ofrece una serie de funciones y métodos integrados para realizar diferentes tipos de operaciones.

Para definir una lista se utilizan corchetes (*[]*) entre los cuales pueden aparecer diferentes valores separados por comas. Esto significa que ambas declaraciones son válidas:

```
>>> lista = []
>>> li = [2, 'a', 4]
```

Al igual que las tuplas, las listas son también *iterables*, así pues, podemos recorrer sus elementos empleando un bucle:

```
>>> for ele in li:
...     print(ele)
...
2
'a'
4
```

A diferencia de las tuplas, los elementos de las listas pueden ser reemplazados accediendo directamente a través del índice que ocupan en la lista. De este modo, para cambiar el segundo elemento de nuestra lista *li*, bastaría como ejecutar la siguiente sentencia:

```
>>> li[1] = 'b'
```

Obviamente, los valores de las listas pueden ser accedidos utilizando el valor del índice que ocupan en la misma:

```
>>> li[2]
4
```

Podemos comprobar si un determinado valor existe en una lista a través del operado *in*, que devuelve *True* en caso afirmativo y *False* en caso contrario:

```
>>> 'a' in li
True
```

Inserciones y borrados

Para añadir un nuevo elemento a una lista contamos con el método *append()*. Como parámetro hemos de pasar el valor que deseamos añadir y este será insertado automáticamente al final de la lista. Volviendo a nuestra lista ejemplo de tres elementos, uno nuevo quedaría insertado a través de la siguiente sentencia:

```
>>> li.append('nuevo')
```

Nótese que, para añadir un nuevo elemento, no es posible utilizar un índice superior al número de elementos que contenga la lista. La siguiente sentencia lanza un error:

```
>>> li[4] = 23
Traceback (most recent call last):
   File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

Sin embargo, el método *insert()* sirve para añadir un nuevo elemento especificando el índice. Si pasamos como índice un valor superior al número de elementos de la lista, el valor en cuestión será insertado al final de la misma, sin tener en cuenta el índice pasado como argumento. De este modo, las siguientes sentencias producirán el mismo resultado, siendo *'c'* el nuevo elemento que será insertado en la lista *li*:

```
>>> li.insert(3, 'c')
>>> li.insert(12, 'c')
```

Por el contrario, podemos insertar un elemento en una posición determinada cuyo índice sea menor al número de valores de la lista. Por ejemplo, para insertar un nuevo elemento en la primera posición de nuestra lista *li*, bastaría con ejecutar la siguiente sentencia:

```
>>> li.insert(0, 'd')
>>> li
>>> ['d', 2, 'a', 4]
```

Si lo que necesitamos es borrar un elemento de una lista, podemos hacerlo gracias a la función *del()*, que recibe como argumento la lista junto al índice que referencia al elemento que deseamos eliminar. La siguiente sentencia ejemplo borra el valor *2* de nuestra lista *li*:

```
>>> del(li[1])
```

Como consecuencia de la sentencia anterior, la lista queda reducida en un elemento. Para comprobarlo contamos con la función *len()*, que nos devuelve el número de elementos de la lista:

```
>>> len(li)
2
```

Obsérvese que la anterior función también puede recibir como argumento una tupla o un string. En general, *len()* funciona sobre tipos de objetos iterables.

También es posible borrar un elemento de una lista a través de su valor. Para ello contamos con el método *remove()*:

```
>>> li.remove('d')
```

Si un elemento aparece repetido en la lista, el método *remove()* solo borrará la primera ocurrencia que encuentre en la misma.

Otro método para eliminar elementos es *pop()*. A diferencia de *remove()*, *pop()* devuelve el elemento borrado y recibe como argumento el índice del elemento que será eliminado. Si no se pasa ningún valor como índice, será el último elemento de la lista el eliminado. Este método puede ser útil cuando necesitamos ambas operaciones (borrar y obtener el valor) en una única sentencia.