[17 DICIEMBRE, 2016](https://sebamawa.wordpress.com/2016/12/17/arrays-en-python-2/)

**“Arrays” en Python**

Arrays en Python implementados con listas.

En Python no disponemos de arrays al estilo de otros lenguajes. Por ejemplo, en C++ podemos declarar y dimensionar un array en tiempo de compilación con la declaración:

|  |  |
| --- | --- |
| 1**|** | int array[10]; |

para crear un arreglo de diez enteros. O como en el propio C++ o Java lo podemos hacer en tiempo de ejecución con la sentencia:

|  |  |
| --- | --- |
| 1**|** | int array[] = **new** int[10]; |

El dimensionar un array en tiempo de compilación (para lenguajes compilados) o en tiempo de ejecución (para lenguajes compilados o interpretados) dependerá de las necesidades de nuestro programa.

Obviando las inicializaciones por defecto de los lenguajes dadas en sus especificaciones (por ejemplo Java inicializará todos los elementos del array declarado anteriormente en cero), si quisiéramos realizar manualmente esta inicialización, podríamos hacerlo mediante el siguiente bucle:

|  |  |
| --- | --- |
| 1  2 | for (int i=0; i<10; i++)     array[i] = 0 |

Ahora si quisiéramos emular la creación e inicialización del array anterior con una lista en Python, haríamos:

|  |  |
| --- | --- |
| 1  2  3  4 | list = []    for i in range(10):       list[i] = 0 |

Rompe los ojos que en este caso el intérprete de Python lanzará una excepción: ‘IndexError: list assignment out of range’. Es claro que al ser las listas estructuras dinámicas (estructuras cuyo tamaño crece en tiempo de ejecución según las necesidades de los programas), y además solamente haber creado una lista vacía, en el bucle estamos queriendo realizar una asignación en una posición inexistente en la lista, de aquí el clásico error de acceso a memoria que todos nos hemos encontrado alguna vez como programadores.

En Python lo más parecido a un array como los vistos al inicio, sería una lista que se pueda dimensionar e inicializar en el momento de su creación. Para ello este lenguaje nos dispone de las llamadas ‘listas por comprensión’ (List Comprehensions en inglés).

Una comprensión (comprehension) en términos de Python, es una expresión que especifica una secuencia de valores, se puede pensar como una forma compacta de un loop (bucle). Con los ejemplos aclararemos estas ideas.

Por tanto, usaremos comprensiones para inicializar una lista con un tamaño prefijado (dimensión) en el mismo momento de su creación.

La forma más sencilla de crear una lista por compresión en Python es:

|  |  |
| --- | --- |
| 1**|** | list = [expresion for variable in range(N)] |

Por ejemplo, para crear una lista por compresión equivalente al array de 10 enteros ya visto, haríamos:

|  |  |
| --- | --- |
| 1**|** | list = [0 for i in range(10)] |

De esta manera, se tiene una lista de 10 elementos enteros inicializados en 0.

También se podría usar el índice i del for en la expresión, por ejemplo si quisiéramos inicializar la lista en 0, 1, 2, …, 9:

|  |  |
| --- | --- |
| 1**|** | list = [i for i in range(10)] |

o usar una expresión más complicada en la inicialización:

|  |  |
| --- | --- |
| 1**|** | list = [i\*i/2 for i in range(10)] |

observando que en este caso la lista se inicializará con 10 números de punto flotante (floats) por el uso del operador de división en la expresión, y que por tanto el intérprete de Python los asume de ese tipo.

Además, ahora tiene sentido acceder a cualquier posición de la lista en el rango 0..9 para i, con la declaración list[i], y realizar las operaciones que se desee con dichos elementos, como si se tratase de un array de los lenguajes “clásicos”, pero con el plus de que al ser la estructura una lista disponemos de todos sus métodos internos, por ejemplo, el método sort(), que ordena la lista.

**Referencia bibliográfica.**

URL: <https://sebamawa.wordpress.com/2016/12/17/arrays-en-python-2/>