Two-dimensional arrays

Conceptually, a two-dimensional (2D) array is a rectangular grid (or matrix) of cells. The size of a 2D array is fixed and is indicated by the width and height of the grid. For example, a 2D array of size 16x10 has 16 rows and 10 columns. That is, the 16x10 array contains 160 items, organized into 16 rows of 10 cells each.

Given a function that maps each point in a multidimensional space to a unique point in a one-dimensional (1D) space, it is possible to represent the multidimensional space using the 1D space. Consequently, given a function that maps each cell in a 2D array to a unique index in a 1D array (or Python list), it is possible to store the 2D array in the 1D array. The two functions that most programmers and programming languages use to implement this mapping are row- major ordering and column-major ordering.

In row-major ordering the elements in each row of a 2D array are stored at consecutive indices in the 1D array and all elements in a given row are stored at lower indices than any of the elements in a subsequent row. Specifically, given a 2D array of width W stored using row-major ordering, the index in the 1D array that corresponds to row x, column y is:

```
x * W + y
```

For example, consider the 2x3 array:

```
0 1 2
3 4 5
```

Using row-major ordering, the 2D array would be represented as the Python list:

```
[ 0, 1, 2, 3, 4, 5 ]
```

In column-major ordering the elements in each column of a 2D array are stored at consecutive indices in the 1D array and all elements in a given column are stored at lower indices than any of the elements in a subsequent column. Specifically, given a 2D array of height H stored using column-major ordering, the index in the 1D array that corresponds to row x, column y is:

```
x + y * H
```

Consider the 2x3 array from the previous example. Using column-major ordering, the 2D array would be represented as the Python list:

```
[ 0, 3, 1, 4, 2, 5 ]
```