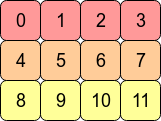
A contiguous array is just an array stored in an unbroken block of memory: to access the next value in the array, we just move to the next memory address.

Consider the 2D array arr = np.arange(12).reshape(3,4). It looks like this:

[](https://i.sstatic.net/BJIVL.png)

In the computer's memory, the values of arr are stored like this:

[enter image description here](https://i.sstatic.net/MXrA6.png)

This means arr is a **C contiguous** array because the *rows* are stored as contiguous blocks of memory. The next memory address holds the next row value on that row. If we want to move down a column, we just need to jump over three blocks (e.g. to jump from 0 to 4 means we skip over 1,2 and 3).

Transposing the array with arr.T means that C contiguity is lost because adjacent row entries are no longer in adjacent memory addresses. However, arr.T is **Fortran contiguous** since the *columns* are in contiguous blocks of memory:

[](https://i.sstatic.net/g6Nb0.png)

Performance-wise, accessing memory addresses which are next to each other is very often faster than accessing addresses which are more "spread out" (fetching a value from RAM could entail a number of neighbouring addresses being fetched and cached for the CPU.) This means that operations over contiguous arrays will often be quicker.

* ‘ALIGNED’ (‘A’) - ensure a data-type aligned array