

Solutions, Week 3, Day 5

1. Create a three dimensional matrix that is at least 5 by 5 by 5 with random contents. What does `myMatrix[:3,1:,:2]` return? What does `myMatrix[:3][1:][::2]` return? Why? Which is preferred?

They return the same thing. The second version of the call, however, creates intermediate arrays, taking elements from the row first, then the column and then the final dimension. As a result, the result is the same, but the call is much slower, as unnecessary arrays are created.

2. Create a square matrix in which every element is initialised to the product of the row and the column index.

```
fromfunction (lambda x, y: x*y, (3, 3))
```

3. Extract the submatrix that is inset by one element in each dimension. That is, if the original array is 6x6, the inner 4x4 array is returned.

```
return myArr[1:-1,1:-1]
```

4. Write a function that accepts an array and returns a new, flattened (one-dimensional) array.

```
def flatten (arr):  
    return arr.reshape ((1,-1))
```

5. The interactions of organisms in an ecology is to be simulated as a 2D matrix, with the entry at a location indicating how many organisms are at that point. Initially the grid will be set up so that there is a maximum of one organism per cell with no organisms immediately adjacent, although they can be “diagonally adjacent” (i.e a checkerboard pattern). Set up a 16 by 16 grid in this fashion.

```
def initcell (x, y):  
    if (x+y)%2 = 0:  
        return 1  
    else  
        return 0
```

```
fromfunction (initcell, (16, 16))
```

6. The experiment has now been moved to three dimensions. Initialise a 16 by 16 by 16 cube, again so that organisms are not immediately adjacent.

```
def initcell (x, y, z):  
    if (x+y+z)%2 = 0:  
        return 1  
    else  
        return 0
```

```
fromfunction (initcell, (16, 16, 16))
```