Hints for Handling N-d Structures

Nested Iteration

One of the big difficulties in *HyperMines* is arbitrary-depth iteration. In *Mines*, you could write the following:

```
for r in range(nrows):
    for c in range(ncols):
        if (some condition on r, c):
            # Do something with (r, c)
```

In 3 dimensions, you could imagine generalizing it to the following:

```
for x in range(width):
    for y in range(height):
        for z in range(depth):
            if (some condition on x, y, z):
                # Do something with (x, y, z)
```

And in 4 dimensions:

```
for x in range(width):
    for y in range(height):
        for z in range(depth):
            for a in range(time):
                if (some condition on x, y, z, a):
                    # Do something with (x, y, z, a)
```

But this won't work for *HyperMines* (do you see why?). Instead, can you write a recursive function that works for an *arbitrary* number of dimensions?

Suggested Functions

Here is a list of useful auxiliary functions that the reference solution uses. If you use any of them, remember to add your own docstrings/doctests!

- A function that, given an N-d array and a tuple/list of coordinates, returns the value at those coordinates in the array.
- A function that, given an N-d array, a tuple/list of coordinates, and a value, replaces the value at those coordinates in the array with the given value.
- A function that, given a list of dimensions and a value, creates a new N-d array with those dimensions, where each value in the array is the given value.
- A function that, given a game, returns the state of that game (`'ongoing'`, `'defeat'`, or `'victory'`).
- A function that returns all the neighbors of a given set of coordinates in a given game.
- A function that returns all possible coordinates in a given board.