Syntactic Sugar for classes with a grid-like instance variable

Note: You can get by totally fine without understanding any of the topics in this article. So if you find this information confusing, you can pretty safely ignore it (for now, at least) and just do things in a way that you do understand.

We'll be writing a number of board games over the next couple of weeks. Many of these board games are played on a two-dimensional grid. In our code, we represent these grids with 2D arrays.

These grids will usually be stored as an instance variable. For example, here's a tic tac toe grid:

```
@grid = [
  [:X, nil, :O],
  [:X, :O, nil],
  [nil, nil, nil]
]
```

Basic 2D array access

We can easily get the value of the cell at the top-right corner of the game board like so:

```
board.grid[0][2] # => :O
```

And we can change the value at a position like so:

```
board.grid[2][0] = :X
```

This works plenty fine. Oftentimes, though, we'll be dealing with a single position variable: a mini array of coordinates. For example, `pos = [0, 0]` would represent the top left corner. By defining two special Ruby methods, `[]` and `[]=`, we can make things a lot cleaner.

Special methods: `[]` and `[]=`

A `Board` class:

```
# board.rb
class Board
  attr_reader :grid

  def initialize
    @grid = Array.new(3) { Array.new(3) }
  end
  def [](pos)
    row, col = pos
    @grid[row][col]
  end

  def []=(pos, mark)
    row, col = pos
    @grid[row][col] = mark
  end

  ...
end
```

After we set up these methods, we can use them in a couple of ways. First off, they can be used just like normal methods; for example, you could say

```
# pos = [0, 3]
board.[](pos, :X)
```

That's ugly, though, and kinda defeats the whole point.

Ruby gives us some lovely syntactic sugar to work with; the bracket methods we wrote work just like the ones on arrays, letting us put parameters inside them. So, instead of writing the above, we can say:

```
# pos = [0, 3]
board[pos] = :X
```

Similarly, you can access a position on the grid in the same manner:

```
# pos = [0, 0]
board[pos] #=> :X
```

Isn't that nice? So much prettier than writing everything out the old way.