Two-dimensional lists

- Some data can be organized efficiently in a table (also called a matrix or 2-dimensional list)
- Each cell is denoted with two subscripts, a row and column indicator

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>18</td>
<td>43</td>
<td>49</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>30</td>
<td>32</td>
<td>53</td>
<td>75</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>28</td>
<td>38</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>24</td>
<td>37</td>
<td>58</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>19</td>
<td>40</td>
<td>46</td>
<td>66</td>
</tr>
</tbody>
</table>

\[ B[2][3] = 50 \]
2-d Lists in Python

data = [ [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12] ]

>>> data[0]
[1, 2, 3, 4]
>>> data[1][2]
7
>>> data[2][5]  # index error

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>
Accessing row and column info...

```python
lst = [ [1, 2, 3], [4, 5, 6] ]
print(lst) # prints [[1, 2, 3], [4, 5, 6]]

print(len(lst)) # prints 2
print(len(lst[0])) # prints 3
```
2-d List Example in Python

• Find the sum of all elements in a 2-D list

```python
def matrixSum(table):
    total = 0
    for row in range(0, len(table)):
        for col in range(0, len(table[row])):
            total += table[row][col]
    return total
```

- Number of rows in the table
- Number of columns in the given row of the table
- In a rectangular matrix, this number will be the same for each row so we could use a fixed number for row such as len(table[0])
def matrixSum(table):
    total = 0
    for row in range(0, len(table)):
        for col in range(0, len(table[row])):
            total += table[row][col]
    return total

len(table) = 3
len(table[row]) = 4 for every row
Printing a 2-d list

print(lst)  # not very "pretty", as we saw

def print2d(lst):
    for row in range(len(lst)):
        print(lst[row])

# prints
[ 1, 2, 3 ]
[ 4, 5, 6 ]
2-dimensional lists – beware of aliasing!

• How to make a Tic-Tac-Toe board?

  board = [' ', ' ', ' ']

  board = [' ', ' ', ' '] * 3

  but it just makes a 9-element, 1-d list!

• OK, how about

  board2 = [ [' '] * 3] * 3

  # incorrect due to aliasing (but is 3x3!)

• Correct…

  board = [ ]

  for row in range(3):

    board += [ [' '] * 3]
2-dimensional lists

• Let's play a game of Tic-Tac-Toe (to be continued…)