Nested Lists

- Lists can hold any objects
- Lists are objects
- Therefore lists can hold other lists!

\[ x = [1, [2, 1], [1, 4, [3, 1]], 6] \]

Two Dimensional Lists

<table>
<thead>
<tr>
<th>Table of Data</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  1  2  3</td>
<td></td>
</tr>
<tr>
<td>0  5  4  7  3</td>
<td></td>
</tr>
<tr>
<td>1  4  8  9  7</td>
<td></td>
</tr>
<tr>
<td>2  5  1  2  3</td>
<td></td>
</tr>
<tr>
<td>3  4  1  2  9</td>
<td></td>
</tr>
<tr>
<td>4  6  7  8  0</td>
<td></td>
</tr>
</tbody>
</table>

Store them as lists of lists (row-major order)
\[ d = [[5, 4, 7, 3], [4, 8, 9, 7], [5, 1, 2, 3], [4, 1, 2, 9], [6, 7, 8, 0]] \]

Overview of Two-Dimensional Lists

- Access value at row 3, col 2:
  \[ d[3][2] \]
- Assign value at row 3, col 2:
  \[ d[3][2] = 8 \]
- Getting array dimensions:
  - Number of rows of \( d \): \( \text{len}(d) \)
  - Number of cols in row \( r \) of \( d \): \( \text{len}(d[r]) \)

How Multidimensional Lists are Stored

- \( b = [[9, 6, 4], [8, 7, 7]] \)
- \( b \) holds name of a one-dimensional list
  - Has \( \text{len}(b) \) elements
  - Its elements are (the IDs of) one-dimensional lists
- \( b[i] \) holds the name of a one-dimensional list (of ints)
  - Has \( \text{len}(b[i]) \) elements

Image Data: 2D Lists of Pixels

Ragged Lists: Rows w/ Different Length

- \( b = [[17, 13, 19], [28, 85]] \)
- Will see applications of this later
Slices and Multidimensional Lists

- Only "top-level" list is copied.
- Contents of the list are not altered
- \( b = [[9, 6], [4, 5], [7, 7]] \)

```
\[\begin{array}{c}
\text{id1} \\
9 \\
19384760 \\
\text{id2} \\
6 \\
\text{id3} \\
4 \\
\text{id4} \\
7 \\
\end{array}\]
```

\[\begin{array}{c}
\text{id1} \\
9 \\
\text{id2} \\
4 \\
\text{id3} \\
7 \\
\text{id4} \\
7 \\
\end{array}\]

\[x = b[:2]\]

```
x = b[:2]
```

Clicker: Slices and Lists of Lists

- Create a 2D list
  >>> \( b = [[9, 6], [4, 5], [7, 7]] \)
- Get a slice
  >>> \( x = b[:2] \)
- Append a new row to \( x \)
  >>> \( x.append([2,1]) \)
- Append to a row of \( x \)
  >>> \( x[1].append(10) \)
- What are the contents of the list (with ID stored in) \( x \)?
- What are the contents of the list (with ID stored in) \( b \)?

Processing lists: builtins

- \( \text{sum(x)} \) adds up all the elements in the list \( x \)
  * they had better be numbers!
- \( \text{min(x)} \) or \( \text{max(x)} \) find the minimum resp. maximum value in the list \( x \)
  * they use the same ordering as \( \text{sort()} \)
- \( \text{range(n)} \) produces \([0, 1, 2, \ldots, n-1]\)
  * optional arguments to start somewhere other than zero
- \( \text{list(x)} \) converts \( x \) (a string for example) to a list
  * e.g. \( \text{list('mimsy')} \) produces \(['m', 'i', 'm', 's', 'y']\)

Processing lists: The \text{map} Function

General form: \( \text{map(function), (list)} \)

- if \( x \) is a list of \( n \) items and \( f \) is a function with one parameter:
  \[\text{map(f, x)} \]
  calls the function once for each list item
  \[\{f(x[0]), f(x[1]), \ldots, f(x[n-1])\}\]
- if \( x \) is a list of \( n \) items and \( m \) is a method with no parameters:
  \[\text{map(m, x)} \]
  \[\{x[0].m(), x[1].m(), \ldots, x[n-1].m()\}\]

Examples:

- \( \text{map(lambda, ['a', 'b', 'c', 'd', 'e'])} \) produces \([1, 2, 3, 4, 5]\)
- \( \text{map(str.strip, ['a', 'b', 'c', 'd', 'e'])} \) produces \(['a', 'b', 'c', 'd', 'e']\)

Processing lists: The \text{for} Statement

General form:

\[\text{for } \langle \text{variable} \rangle \text{ in } \langle \text{list} \rangle : \langle \text{statements} \rangle\]

- \( \text{for } a \text{ in } x : \)
  - \( \text{print 3 * a} \)
  - \( \text{print 3 * x[0]} \)
  - \( \text{print 3 * x[1]} \)
  - \( \text{print 3 * x[0]} \)
  - \( \ldots \)
  - \( \text{print 3 * x[n-1]} \)

When the body is executed, the value of \( a \) is the current list item