Processing lists: builtins

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

Processing lists: The \texttt{map} Function

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

if \texttt{x} is a list of \(n\) items and \(f\) is a function with one parameter:

\[
\text{map}(f, x) \quad \rightarrow \quad \text{calls the function once for each list item}
\]

\[
[f(x[0]), f(x[1]), \ldots, f(x[n-1])] \\
[x[0].m(), x[1].m(), \ldots, x[n-1].m()]
\]

examples:

\[
\text{map(lambda x: x, [1, 2, 3])} \quad \rightarrow \quad [1, 2, 3]
\]

\[
\text{map(lambda x: x, [1, 2, 3], [4, 5, 6])} \quad \rightarrow \quad [(1, 4), (2, 5), (3, 6)]
\]

Processing lists: The \texttt{for} Statement

General form:

\[
\text{for (variable) in (list): (statements)}
\]

\[
\text{executes the body once for each list item}
\]

for \(a\) in \(x\):

\[
\text{print } 3 * a
\]

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

Nested Lists

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

\[
a = [2, 1] \\
b = [3, 1] \\
c = [1, 4, b] \\
x = [1, y, z, 5]
\]

Two Dimensional Lists

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

Each row, col has a value

Store them as lists of lists (row-major order)

\[
d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[4,7,8,0]]
\]

Overview of Two-Dimensional Lists

• Access value at row 3, col 2:

\[
d[3][2] = 8
\]

• Assign value at row 3, col 2:

\[
d[3][2] = 8
\]

• Getting array dimensions:
  * Number of rows of \(d\): \texttt{len(d)}
  * Number of cols in row \(r\) of \(d\): \texttt{len(d[r])}
How Multidimensional Lists are Stored

- \( b = [[9, 6, 4], [5], [7, 7]] \)

- \( b \) holds name of a one-dimensional list
  - Has \( \text{len}(b) \) elements
  - Its elements are (the names of) 1D lists
- \( b[i] \) holds the name of a one-dimensional list (of ints)
  - Has \( \text{len}(b[i]) \) elements

Ragged Lists: Rows w/ Different Length

- \( b = [[17, 13, 19], [28, 95]] \)

- Will see applications of this later

Image Data: 2D Lists of Pixels

Slices and Multidimensional Lists

- Create a 2D List
  >>> \( b = [[9, 6], [4, 5], [7, 7]] \)
- Get a slice
  >>> \( x = b[:2] \)
- Append to a row of \( x \)
  >>> \( x[1].append(10) \)
- \( x \) now has the 2D list \( [[9, 6], [4, 5, 10], [7, 7]] \)

What are the contents of the list (with name) in \( b \)?

- A: \( [[9, 6], [4, 5], [7, 7]] \)
- B: \( [[9, 6], [4, 5, 10]] \)
- C: \( [[9, 6], [4, 5, 10], [7, 7]] \)
- D: \( [[9, 6], [4, 10], [7, 7]] \)
- E: I don’t know