CS1110

Lecture 10: More with Sequences

Announcements

Assignment 2
Hand it in today by leaving it on the table in front.

Reading
…for next week:
Sections 5.8-5.10
### Processing lists: builtins

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

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

if $x$ is a list of $n$ items and $f$ is a function with one parameter:

$$\texttt{map}(f, x) \rightarrow [f(x[0]), f(x[1]), \ldots, f(x[n–1])]$$

if $x$ is a list of $n$ items and $m$ is a method with no parameters:

$$\texttt{map}(m, x) \rightarrow [x[0].m(), x[1].m(), \ldots, x[n–1].m()]$$

calls the function once for each list item

eamples:

$\texttt{map}(\text{len}, ['a', 'bc', 'defg'])$ produces $[1, 2, 4]$

$\texttt{map}(\text{str.strip}, ['a', ' bc', ' defg'])$ produces $["a", "bc", "defg"]$
### Processing lists: The `for` Statement

**General form:**

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

- `for a in x:`
  - `print 3 * a`

When the body is executed, the value of `a` is the current list item.

Prints `3 * x[0]`
Prints `3 * x[1]`
Prints `3 * x[2]`
... (for each item in the list)
Prints `3 * x[n-1]`
Nested Lists

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

\[
\begin{align*}
  a &= [2, 1] \\
  b &= [3, 1] \\
  c &= [1, 4, b] \\
  x &= [1, a, c, 5]
\end{align*}
\]
# Two Dimensional Lists

## Table of Data

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

Each row, col has a value

## Images

Each row, col has an RGB value

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

\[
d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[6,7,8,0]]
\]
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

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

- \( b \) 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
Image Data: 2D Lists of Pixels

b[0][0] is a white pixel

id1

id2

id3

list

id3

id21

id22

list

 RGB

red

255

green

255

blue

255
Ragged Lists: Rows w/ Different Length

- $b = [[17, 13, 19], [28, 95]]$

- Will see applications of this later
Slices and Multidimensional Lists

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

\[
x = b[:2]
\]
Slices and Multidimensional Lists

- Create a 2D List
  
  ```python
  >>> b = [[9,6],[4,5],[7,7]]
  ```

- Get a slice
  
  ```python
  >>> x = b[:2]
  ```

- Append to a row of x
  
  ```python
  >>> x[1].append(10)
  ```

- x now has the 2D list
  
  ```python
  [[9, 6], [4, 5, 10]]
  ```

- 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