CS1110

Lecture 10: More with Sequences

Announcements

Last chance for a makeup! Prelim 1 conflicts must be registered in CMS by noon Friday (tomorrow!). See email from Maria Witlox.

Assignment 2
…is out! It is a (shortish) paper-and-pencil exercise, due at the next class (Tuesday March 4)

Reading
…for next week:
Sections 5.8-5.10

Slides by D. Gries, L. Lee, S. Marschner, W. White
Nested Lists

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

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

Images

Each row, col has a value

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

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

• \( b = \begin{array}{c}\{9, 6, 4\}, \{5, 7, 7\}\end{array} \)

• \( b \) 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

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
- \( b = \begin{bmatrix} [9, 6], [4, 5], [7, 7] \end{bmatrix} \)

\[
x = b[:, 2]
\]
Clicker: Slices and Lists of Lists

- Create a 2D list
  ```python
  >>> b = [[9,6],[4,5],[7,7]]
  ```
- Get a slice
  ```python
  >>> x = b[:2]
  ```
- Append a new row to `x`
  ```python
  >>> x.append([2,1])
  ```
- Append to a row of `x`
  ```python
  >>> x[1].append(10)
  ```
- What are the contents of the list (with ID stored in) `x`?
  
  A: `[[4,5],[7,7],10,[2,1]]`
  
  B: `[[9,6],[4,5],[2,1],10]`
  
  C: `[[9,6],[4,5,10],[2,1]]`
  
  D: `[[9,6],[4,5,10],[7,7],10]`
  
  E: I don’t know

- What are the contents of the list (with ID stored in) `b`?
  
  A: `[[9,6],[4,5],[7,7]]`
  
  B: `[[9,6],[4,5],[2,1],[7,7]]`
  
  C: `[[9,6],[4,5,10],[2,1],[7,7]]`
  
  D: `[[9,6],[4,5,10],[7,7],[2,1]]`
  
  E: I don’t know
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 map Function**

General form: `map(⟨function⟩, ⟨list⟩)`

- **Example 1**: If `x` is a list of `n` items and `f` is a function with one parameter:
  ```
  map(f, x) → [f(x[0]), f(x[1]), ..., f(x[n-1])]
  ```

- **Example 2**: If `x` is a list of `n` items and `m` is a method with no parameters:
  ```
  map(m, x) → [x[0].m(), x[1].m(), ..., x[n-1].m()]
  ```

Examples:

- `map(len, ['a', 'bc', 'defg'])` produces `[1, 2, 4]`
- `map(str.strip, ['a', ' bc', ' defg '])` produces `['a', 'bc', 'defg']`
Processing lists: The \texttt{for} Statement

General form:

\begin{verbatim}
for \langle variable \rangle in \langle list \rangle:
\langle statements \rangle
\end{verbatim}

for a in x:
\begin{verbatim}
print 3 * a
\end{verbatim}

\begin{verbatim}
print 3 * x[0]
p
print 3 * x[1]
p
print 3 * x[2]
p
...
p
print 3 * x[n-1]
\end{verbatim}

executes the body once for each list item

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