Mini-Lecture 16

Nested Lists
Nested Lists

- Lists can hold any object
- Lists are themselves 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>

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

- An odd symmetry
  - Number of rows of \( d \): \( \text{len}(d) \)
  - Number of cols in row \( r \) of \( d \): \( \text{len}(d[r]) \)

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
- & - & - & - \\
0 & 5 & 4 & 7 & 3 \\
1 & 4 & 8 & 9 & 7 \\
2 & 5 & 1 & 2 & 3 \\
3 & 4 & 1 & 2 & 9 \\
4 & 6 & 7 & 8 & 0 \\
\end{array}
\]
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
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

- 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 nested 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 nested list
  [[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
Slices and Multidimensional Lists

• Create a nested 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 nested list
  [[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
```python
def transpose(table):
    """Returns: copy of table with rows and columns swapped
    Precondition: table is a (non-ragged) 2d List"
    numrows = len(table)  # Need number of rows
    numcols  = len(table[0])  # All rows have same no. cols
    result = []  # Result (new table) accumulator
    for m in range(numcols):
        # Get the column elements at position m
        # Make a new list for this column
        # Add this row to accumulator table
    return result
```

1 2
3 4
5 6

1 3 5
2 4 6
def transpose(table):
    """Returns: copy of table with rows and columns swapped
    Precondition: table is a (non-ragged) 2d List"""
    numrows = len(table)  # Need number of rows
    numcols = len(table[0])  # All rows have same no. cols
    result = []  # Result (new table) accumulator
    for m in range(numcols):
        row = []  # Single row accumulator
        for n in range(numrows):
            row.append(table[n][m])  # Create a new row list
        result.append(row)  # Add result to table
    return result
Functions and 2D Lists

```python
def transpose(table):
    '''Returns: copy of table with rows and columns swapped
    Precondition: table is a (non-ragged) 2d List'''
    numrows = len(table)  # Need number of rows
    numcols = len(table[0])  # All rows have same no. cols
    result = []  # Result (new table) accumulator
    for m in range(numcols):
        row = []  # Single row accumulator
        for n in range(numrows):
            row.append(table[n][m])  # Create a new row list
        result.append(row)  # Add result to table
    return result
```

10/5/18

Nested Lists
JSON: Mixing and Lists and Dictionaries

```
{
  "wind": {
    "speed": 13.0,
    "crosswind": 5.0
  },
  "sky": [
    {
      "cover": "clouds",
      "type": "broken",
      "height": 1200.0
    },
    {
      "type": "overcast",
      "height": 1800.0
    }
  ]
}
```

- **weather.json:**
  - Weather measurements at Ithaca Airport (2017)
  - **Keys:** Times (Each hour)
  - **Values:** Weather readings

- This is a *nested* JSON
  - Values are also dictionaries
  - Containing more dictionaries
  - And also containing lists

See **weather.py**