Lecture 12: Nested Lists and Dictionaries (Sections 11.1-11.5)

CS 1110
Introduction to Computing Using Python

Announcements

• Be sure to go to section for Labs 11 & 12
• A3: first submission ("part A") due Mar 24; final submission due Mar 28
• Definitive source for due dates is the course webpage, but we try to also put due dates on the Canvas calendar
• A2 grades and solutions available around Wednesday
• Next lecture will be a review session
• Tues 3/30 lecture will be open office hour
• Prelim 1 Study Guide available tonight. Be sure to read it!
• Exam logistics: seat number and Zoom link to be distributed via CMS by end of the week. Online exam takers will be contacted by proctor to do a required short mock exam before actual exam.

Nested Lists

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

```python
b = [3, 1]
c = [1, 4, b]
a = [2, 1]
x = [1, a, c, 5]
```

Two Dimensional Lists

Table of Data

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

E.g., lab number

E.g., student ID

Each row, column of the table stores data (a value). Here, the number of units of product 3 sold by the shop with ID 1 on lab 3

Table of Data

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

Row index

Column index

E.g., product ID

E.g., shop ID

Each row, column of the table stores data (a value). Here, the number of units of product 3 sold by the shop with ID 1

Store them as a list of lists ("row-major order")

```python
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

Example:

```python
d = [[5, 4, 7, 3], [4, 8, 9, 7], [5, 1, 2, 3], [4, 1, 2, 9]]  # Access value at row 3, col 2
>>> d[3][2] = 8  # Assign value at row 3, col 2
>>> d
[[5, 4, 7, 3], [4, 8, 9, 7], [5, 1, 2, 3], [4, 1, 8, 9]]  # Number of rows of d
>>> len(d[2])  # Number of cols in row 2 of d
4
```

How Multidimensional Lists are Stored

```
9 6 4
5 7 7
```

- b holds id of a one-dimensional list
  * Has len(b) elements
- b[i] holds id of a one-dimensional list
  * Has len(b[i]) elements

Ragged Lists: Rows w/ Different Length

```
b = [[17, 19], [28, 95]]
```

A loop to go row to row. Then at each row, set a loop to go column to column. → Nested loops!

Exercise 1

```python
def print_all_rows(my_table):
    """Prints all rows of the table, one row (list) on each line.
    Preconditions: my_table is a table of numbers
    my_table is not empty
    """
```

Exercise 2

```python
def print_all_elements(my_table):
    """Prints all elements of the table, one element on each line.
    Preconditions: my_table is a table of numbers
    my_table is not empty
    """
```
Data Wrangling: Transpose Idea

4 lists: 2 elements in each

How to transpose?
• 1\textsuperscript{st} element of each list gets appended to 1\textsuperscript{st} list
• 2\textsuperscript{nd} element of each list gets appended to 2\textsuperscript{nd} list

Data Wrangling: Transpose Code

def transpose(table):
    
    n_rows = len(table)
    n_cols = len(table[0])  # All rows have same no. cols
    new_table = []  # Result accumulator
    
    for c in range(n_cols):
        row = []  # Single row accumulator
        for r in range(n_rows):
            row.append(table[r][c])  # Build up new row
        new_table.append(row)  # Add new row to new table
    
    return new_table

d = [[1,2],[3,4],[5,6]]
d_v2 = transpose(d)

Dictionaries (Type dict)

Description
• List of key-value pairs
  • Keys are unique
  • Values need not be
• Example: net-ids
  • net-ids are unique (a key)
  • names need not be (values)
  • js1 is John Smith (class '13)
  • js2 is John Smith (class '16)

Python Syntax
• Create with format:
  
  \{key1:value1, key2:value2, ...\}
• Keys must be immutable
  • ints, floats, bools, strings
  • Not lists or custom objects
• Values can be anything
• Example:
  
  d = \{js1:'John Smith',
        js2:'John Smith',
        tm55:'Toni Morrison'\}

Using Dictionaries (Type dict)

• Dictionaries are mutable
  • Can reassign values
  • d[ec1] = 'Ellis'

Using Dictionaries (Type dict)

• Can access elements like a list
• Must use the key, not an index
• Cannot slice ranges
Using Dictionaries (Type `dict`)

- Dictionaries are mutable
  - Can reassign values
    - `d['ec1'] = 'Ellis'`
  - Can add new keys
    - `d['psb26'] = 'Pearl'`

```python
d = {
    'ec1': 'Ezra',
    'ec2': 'Ezra',
    'tm55': 'Toni',
    'psb26': 'Pearl'
}
```

Deleting key deletes both key and value

Be sure to read Textbook 11.1-11.5 for additional examples!

Slices and Multidimensional Lists

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

```python
b = [[9, 6], [4, 5], [7, 7]]
x = b[1]
x[1].append(10)
```

What is now in `x`?
- A: `[[9, 6, 10], [4, 5]]`
- B: `[[9, 6, 10], [4, 5, 10]]`
- C: `[[9, 6], [4, 5, 10], [7, 7]]`
- D: `[[9, 6], [4, 10], [7, 7]]`
- E: I don't know

Slices & Multidimensional Lists (Q1)

- Create a nested 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)
  ```

Slices & Multidimensional Lists (Q2)

- Create a nested 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 nested list `[[9, 6], [4, 5, 10]]`

What is now in `b`?
- A: `[[9, 6, 10], [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