CSc 110, Spring 2017

Lecture 20: Lists for Tallying; Text Processing

Adapted from slides by Marty Stepp and Stuart Reges
Value/Reference Semantics - Review

• Variables of type int, float, boolean, store values directly:
  
  | age | 20 |
  | cats | 3 |

• Values are copied from one variable to another:
  
  cats = age
  
  | age | 20 |
  | cats | 20 |

• Variables of other types (like lists) store references to memory:
  
  grades
  
  | index | 0 | 1 | 2 |
  | value | 89 | 78 | 93 |

• References are copied from one variable to another:
  
  scores = grades
  
  | scores |  |  |  |
Lists for Tallying
Extracting digits

• Given a number, how do we extract the digits one at a time? Ex: 590823

• Hint: use % and //

```python
>>> n = 590823
>>> n % 10
3
>>> n = n // 10
>>> n
59082
>>> n % 10
2
>>> n = n // 10
>>> n
5908
>>> n % 10
8
>>> 
```
A tallying problem

• Problem: Write a function `most_frequent_digit(n)` that returns the digit of a number `n` that occurs most frequently.

  • Example: the number 669260267 contains:
    one 0, two 2s, four 6s, one 7, and one 9.
    `most_frequent_digit(669260267)` returns 6.

  • If there is a tie, return the digit with the lower value.
    `most_frequent_digit(57135203)` returns 3.
A tallying problem

• This is well-suited for a list.

• Note that there are 10 digits. Consider a list of 10 elements.
  • The value at index $i$ holds the number of occurrences of digit $i$
  • Example for 669260267:

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating a list of tallies

```python
# assume n = 669260267
counts = [0] * 10
while (n > 0):
    # pluck off a digit and add to its counter
digit = n % 10
counts[digit] = counts[digit] + 1
n = n // 10
```

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
def most_frequent_digit(n):
    counts = [0] * 10
    while (n > 0):
        digit = n % 10  # pluck off a digit and tally it
        counts[digit] = count[digit] + 1
        n = n // 10

    best_index = 0
    for i in range(1, len(counts)):
        if (counts[i] > counts[best_index]):
            best_index = i
    return best_index
Data transformations

• In many problems we transform data between forms.
  • Example: digits $\rightarrow$ count of each digit $\rightarrow$ most frequent digit
  • A transformation is computed/stored as a list.

• Sometimes we map between data and list indexes.
  • tally  (if digit is $i$, store its count at index $i$)

• The problem structure affects the mapping
Section attendance question

• Read a file of section attendance (*see next slide for structure*): 

• And produce the following output:

Section 1
Student points: [20, 16, 17, 14, 11]
Student grades: [100.0, 80.0, 85.0, 70.0, 55.0]

Section 2
Student points: [16, 19, 14, 14, 8]
Student grades: [80.0, 95.0, 70.0, 70.0, 40.0]

Section 3
Student points: [16, 15, 16, 18, 14]
Student grades: [80.0, 75.0, 80.0, 90.0, 70.0]

• Students earn 3 points for each section attended up to 20.
Section input file

• Each line represents 9 weeks of attendance data for a section.
• Each week has 5 characters because there are 5 students in all sections.
• Within each week, each character represents one student's attendance:
  • a means the student was absent (+0 points)
  • n means they attended but didn't do the problems (+2 points)
  • y means they attended and did the problems (+3 points)
Section input file (fragment)

- Look at 2 weeks of one section (one line of the file):

<table>
<thead>
<tr>
<th>student</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>value</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>a</td>
<td>y</td>
<td>a</td>
</tr>
</tbody>
</table>

- For index i, a student is i % 5.

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

Need a list of length 5 to calculate the cumulative points for each student
Get the points for each student of a section

#Computes points earned for each student in a particular section.
def count_points(line):
    points = [0] * 5
    for i in range(0, len(line)):
        student = i % 5
        earned = 0
        if (line[i] == 'y'):
            earned = 3
        elif (line[i] == 'n'):
            earned = 2
        points[student] = points[student] + earned
    return points

Note: fix the code to cap the points earned
# This program reads a file representing which students attended which discussion sections and produces output of the students' section attendance and scores.

def main():
    file = open("sections.txt")
    lines = file.readlines()
    section = 1
    for line in lines:
        # process one section
        points = count_points(line)
        grades = compute_grades(points)
        results(section, points, grades)
        section += 1

# Produces all output about a particular section.
def results(section, points, grades):
    print("Section " + str(section))
    print("Student scores: " + str(points))
    print("Student grades: " + str(grades))
    print()
def count_points(line):
    points = [0] * 5
    for i in range(0, len(line)):
        student = i % 5
        earned = 0
        if (line[i] == 'y'):  # values are 'y', 'n' or 'a'
            earned = 3
        elif (line[i] == 'n'):
            earned = 2
        points[student] = min(20, points[student] + earned)
    return points

def compute_grades(points):
    grades = [0] * 5
    for i in range(0, len(points)):
        grades[i] = 100.0 * points[i] / 20
    return grades