THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:

"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.
"list mystery" problem

- **traversal**: An examination of each element of a list.

- What element values are stored in the following list?

```python
a = [1, 7, 5, 6, 4, 14, 11]
for i in range(0, len(a) - 1):
    if a[i] > a[i + 1]:
        a[i + 1] = a[i + 1] * 2
```

<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>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>12</td>
<td>8</td>
<td>14</td>
<td>22</td>
</tr>
</tbody>
</table>
A multi-counter problem

• Problem: Write a function `most_frequent_digit` that returns the digit value that occurs most frequently in a number.

  • Example: The number 669260267 contains:
    one 0, two 2s, four 6es, 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 multi-counter problem

• We could declare 10 counter variables ...
  counter0, counter1, counter2, counter3, counter4,
  counter5, counter6, counter7, counter8, counter9

• But a better solution is to use a list of size 10.
  • The element at index $i$ will store the counter for digit value $i$.
  • 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>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>0</td>
</tr>
</tbody>
</table>

• How do we build such an list? And how does it help?
Creating a list of tallies

```python
# assume n = 669260267
counts = [0] * 10
while n > 0:
    # pluck off a digit and add to proper counter
    digit = n % 10
    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>0</td>
</tr>
</tbody>
</table>
Tally solution

# Returns the digit value that occurs most frequently in n.
# Breaks ties by choosing the smaller value.
def most_frequent_digit(n):
    counts = [0] * 10
    while n > 0:
        digit = n % 10  # pluck off a digit and tally it
        counts[digit] += 1
        n = n // 10

    # find the most frequently occurring digit
    best_index = 0
    for i in range(1, len(counts)):
        if counts[i] > counts[best_index]:
            best_index = i
    return best_index
Section attendance question

• Read a file of section attendance (*see next slide*):
  yynyyynayaynynyanyaynnanyayyanyyna
  ayyanynyayanaayyananyaanayayayyayynyna
  yynaynyynayynynynnyayyyanayyayynannnnyyayayny

• 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

<table>
<thead>
<tr>
<th>student</th>
<th>123451234512345123451234512345123451234512345</th>
</tr>
</thead>
<tbody>
<tr>
<td>week</td>
<td>1 2 3 4 5 6 7 8 9</td>
</tr>
<tr>
<td>section 1</td>
<td>yynyynayayynyyayyanyyyaynayyyanyanayyyanyyna</td>
</tr>
<tr>
<td>section 2</td>
<td>ayyanyyyyayanaayyanayyanayyanayyanayyanayyanynya</td>
</tr>
<tr>
<td>section 3</td>
<td>yyayaynynyayyayynynnnyyayyanayynannnyyayyayany</td>
</tr>
</tbody>
</table>
def main():
    file = open("sections.txt")
    lines = file.readlines()
    section = 1
    for line in lines:
        points = [0] * 5
        for i in range(len(line)):
            student = i % 5
            earned = 0
            if line[i] == 'y':  # c == 'y' or 'n' or 'a'
                earned = 3
            elif line[i] == 'n':
                earned = 1
            points[student] = min(20, points[student] + earned)
        grades = [0] * 5
        for i in range(len(points)):
            grades[i] = 100.0 * points[i] / 20
        print("Section", section)
        print("Student points:", points)
        print("Student grades:", grades)
        print()
    section += 1
Data transformations

• In many problems we transform data between forms.
  • Example: digits → count of each digit → most frequent digit
  • Often each transformation is computed/stored as an list.
  • For structure, a transformation is often put in its own function.

• Sometimes we map between data and list indexes.
  • by position (store the $i^{th}$ value we read at index $i$)
  • tally (if input value is $i$, store it at array index $i$)
  • explicit mapping (count 'J' at index 0, count 'X' at index 1)

• Exercise: Modify our Sections program to use functions that use lists as parameters and returns.
This program reads a file representing which students attended which discussion sections and produces output of the students' section attendance and scores.

```python
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", section)
    print("Student scores:", points)
    print("Student grades:", grades)
    print()
```

...
# Computes the points earned for each student for a particular section.
def count_points(line):
    points = [0] * 5
    for i in range(len(line)):
        student = i % 5
        earned = 0
        if line[i] == 'y':  # c == 'y' or c == 'n'
            earned = 3
        elif line[i] == 'n':
            earned = 2
        points[student] = min(20, points[student] + earned)
    return points

# Computes the percentage for each student for a particular section.
def compute_grades(points):
    grades = [0] * 5
    for i in range(len(points)):
        grades[i] = 100.0 * points[i] / 20
    return grades