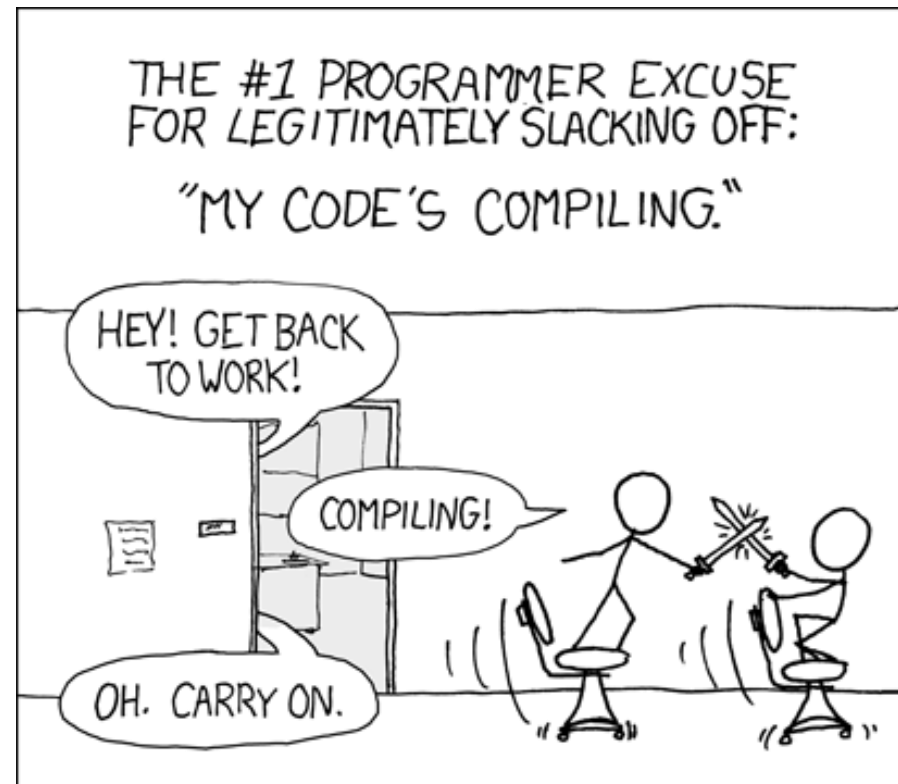


# CSc 110, Autumn 2017

## Lecture 24: Lists for Tallying; Text Processing

Adapted from slides by Marty Stepp and Stuart Reges



# "list mystery" problem

- **traversal:** An examination of each element of an list.
- What element values are stored in the following list?

```
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
```

<i>index</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
<i>value</i>	1	7	10	12	8	14	22

# 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:

<i>index</i>	0	1	2	3	4	5	6	7	8	9
<i>value</i>	1	0	2	0	0	0	4	1	0	0

- How do we build such an list? And how does it help?

# Creating a list of tallies

```
# 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
```

<i>index</i>	0	1	2	3	4	5	6	7	8	9
<i>value</i>	1	0	2	0	0	0	4	1	0	0

# 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*):

```
yynyyynayayynyyyayanyyyaynayyayyanayyyyanyayna  
ayyanyyyyayanaayyanayyyananayayaynyayayynynya  
yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny
```

- 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

<b>student</b>		12345	12345	12345	12345	12345	12345	12345	12345	12345	12345
<b>week</b>		1	2	3	4	5	6	7	8	9	
<b>section</b>	1	yyynyynayayynyyayanyyyaynayyayyanayyyanyayna									
<b>section</b>	2	ayyanyyyyayanaayyanayyyananayayaynyayayynynya									
<b>section</b>	3	yyayaynyyayyanynnyyyayyanayaynannnyyayyayayny									

- Each line represents a section.
- A line consists of 9 weeks' worth of data.
  - Each week has 5 characters because there are 5 students.
- Within each week, each character represents one student.
  - a means the student was absent (+0 points)
  - n means they attended but didn't do the problems (+1 points)
  - y means they attended and did the problems (+3 points)



# Section attendance answer

```
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  $\rightarrow$  count of each digit  $\rightarrow$  most frequent digit
  - Often each transformation is computed/stored as a list.
  - For structure, a transformation is often put in its own function.
- Sometimes we map between data and list indexes.
  - by position (store the  $i^{\text{th}}$  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.

# List param/return answer

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

```
...
```

# List param/return answer

...

**# 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
```