Announcements

• Reading and RQ8 due next time
• Assignment 3 due tonight
  – Assignment 4 out, due Sept 29
• APT 3 is due on Tuesday
• APT Quiz 1 take Sunday-Tuesday midnight
  – Friday – practice APT quiz available

• Today - EOWF:
  – Solving problems with lists, ifs.

Getting help

• Consider a peer tutor – one hour of one on one help a week.
  – Many take advantage of this
  – contact peer tutoring center

• Are you getting too much help?
  – After solving APT
  – Can you solve again with a blank sheet of paper or blank file and no help?

• Are you using 7 step process to solve?

Are you Learning How to Debug?

• Print is your friend!
• Create variables!
• Isolate the problem
  – Comment out sections until you can isolate where the problem is
• Python Tutor – trace
  – Doesn’t work with files but comment out file and create variable with sample input
Assignment 3 - Earthquakes

• Write QuarryBlastQuakes – return the list of earthquakes that are something from a Quarry such as a Quarry Blast Quakes –

• quakes is a list of earthquake strings in correct format
def QuarryBlastQuakes(quakes):

Pattern to build and return new list

initialize newlist
for item in oldlist:
    if item fits criteria
        put item in newlist
return newlist

Assignment 3 - Earthquakes

• Write QuarryBlastQuakes – return the list of earthquakes that are something from a Quarry such as a Quarry Blast Quakes –
    – Description starts with “Quarry”
• quakes is a list of earthquake strings in correct format
def QuarryBlastQuakes(quakes):

def quarryQuakes(quakes):

compsci 101, fall 2016
How do you use quarryQuakes?

- In main:

  ```python
  print "Quarry quakes"
  ```

Making Decisions in Python

```
if condition1:
    Block of code to do if condition is true
elif condition2:
    Block of code to do if condition1 false, condition2 is true
else:
    Block of code to do if other conditions false
```

- Can have many elifs, leave out elif, leave out else
Making Decisions tools

- Boolean values: True, False
- Boolean operators: and, or, not

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X and Y</th>
<th>X or Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

- Relational operators: <, <=, >, >=
- Equality operators: ==, !=

Lists

- A list is a collection of objects
  - scores = [99, 78, 91, 84]
  - allAboutMe = [“Mo”, 25, “934-1234”]
- Lists are mutable – use [num] to change a value
- Lists are indexed starting at 0, or -1 from the end
- Functions: max, min, len, sum
- Slice lists [:]

List Examples

```python
def isVowel(letter):
    answer = False
    if letter == 'a':
        answer = True
    elif letter == 'e':
        answer = True
    elif letter == 'i':
        answer = True
    elif letter == 'o':
        answer = True
    elif letter == 'u':
        answer = True
    return answer

scores = [10, 8, 10, 9]
print scores
scores[2] = 5
print scores
print max(scores), len(scores), print sum(scores)
print scores[1:]
print scores[1], scores[-1]
scores.append(4)
scores += [5]
scores += [5]
print scores
```
List before/after modification

0 1 2 3

10 8 10 9

score = [10, 8, 10, 9]

0 1 2 3

score [2] = 5

10 8 5 10 9

Design pattern of accumulation

*for item in something*

- Summing to tally a count
  
  value += 1

- Building a new string by concatenating
  
  str += ch

- Building a new list by appending
  
  lst.append(element)
  OR
  lst += [element]

Processing List Items

- Process all the items in a list, one item at a time
- Format: for variable in list:
  process variable
- Example:

  ```python
  sum = 0
  nums = [6, 7, 3, 1, 2]
  for value in nums:
      sum = sum + value
  print sum
  ```

Learn list functions

```python
nums = [6, 7, 3, 1, 2]
print sum(nums)
```
Problem: Sum up even numbers in list of numbers

- Could do it similar to two slides back
- OR Build a list of the correct numbers, then sum

How to build list of evens and sum?

bit.ly/101f16-0922-3

def sumUpEven(nums):
    answer = question1
    for item in nums:
        if question2:
            question3
    return question4

Problem: What is length of longest string in list of strings?

From APT 3 - TxMsg
http://www.cs.duke.edu/csed/pythonapt/txmsg.html
Examples

- Do one by hand?
- Explain to partner?
- Identify Pythonic/programming challenges?

**Debugging APTs: Going green**

- TxMsg APT: from ideas to code to green
  - What are the main parts of solving this problem?
  - Transform words in original string
    - Abstract that away at first
    - Finding words in original string
      - How do we do this?

```python
def getMessage(original):
    ret = ""
    for word in original.split():
        ret = ret + " " + transform(word)
    return ret  # initial space?
```

Why use helper function 'transform'?

- Structure of code is easier to reason about
  - Harder to develop this way at the beginning
  - Similar to accumulate loop, build on what we know

- We can debug pieces independently
  - What if transform returns "" for every string?
  - Can we test transform independently of getMessage?

**Python via Problem Solving**

In the loop for TxMsg we saw:

```
    ret = ret + " " + transform(word)
```

- Why does this leave "extra" space at front?
- Eliminate with `ret.strip()`

Alternate: collect transform words in list, use `join` to return

Rather than construct string via accumulation and concatenation, construct list with `append`