CompSci 101
Introduction to Computer Science

February 9, 2016
Prof. Rodger

score = [10, 8, 10, 9]

Announcements

• Reading and RQ8 due next time
• Assignment 3 extended one day – due Wed
  – Assignment 4 out late today
• APT 3 is due on Thursday
• APT Quiz 1 due tonight by midnight

• Today:
  – 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
String Functions – What is output?

```python
name = "VVDarth Vader Darth VaderVVV"
nm = name.strip("V")

phrase = "mississippi"
phrase = phrase.replace("ss","pp")

last = "Darth Vader or Darth Vader"
last = last.replace("a","o").replace("or","es")

b = "the end is near oh dear"
a = b.endswith(‘s’)
```

Making Decisions

```
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: ==, !=
- Look at if examples: miscIf.py
Lists

A list is a collection of objects
scores = [99, 78, 91, 84]

allAboutMe = ["Mo", 25, "934-1234"]

club = ['Mo', 'Jo', 'Po', 'Flo', 'Bo']

- 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 isVowel1(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

def isVowel2(letter):
    answer = False
    if letter == 'a':
        answer = True
    if letter == 'e':
        answer = True
    if letter == 'i':
        answer = True
    if letter == 'o':
        answer = True
    if 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

```
[10, 8, 10, 9]  
```

```python
score = [10, 8, 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:
  
  ```python
  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/101sp16-0209-2

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

<table>
<thead>
<tr>
<th>Problem Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strange abbreviations are often used to write text messages on uncomfortable mobile devices. One particular strategy for encoding text composed of alphabetic characters and spaces is the following:</td>
</tr>
<tr>
<td>• Spaces are maintained, and each word is encoded individually. A word is a consecutive string of alphabetic characters.</td>
</tr>
<tr>
<td>• If the word is composed of vowels, it is written exactly as in the original message.</td>
</tr>
<tr>
<td>• If the word has at least one consonant, write only the consonants that do not have another consonant immediately before them. Do not write any vowels.</td>
</tr>
<tr>
<td>• The letters considered vowels in these rules are 'a', 'e', 'i', 'o' and 'u'. All other letters are considered consonants.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename: Txmsg.py</td>
</tr>
</tbody>
</table>
| def getmessage(original):
    return String that is 'textized' version of string parameter original

Examples

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

1. "text message"
   Returns "tx msg"

2. "ps i love u"
   Returns: "p l v u"

3. "please please me"
   Returns: "p p p m"

4. "back to the user"
   Returns "b t t s"

5. "aiou bcdghjklmpqrstwxyz"
   Returns: "aiou b"
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