Announcements

• No RQs until after Exam 2
• Assignment 7 due Thursday
• APT Quiz 2 – must do by Wed midnight
• APT 6 due today, APT 7 out
• Exam 2 is Nov 16
• Lab this week!

• Today:
  – More practice with Dictionaries, finish last time

Lab this week! - Odds with poker!
Smarty Hangman

- Version of Hangman that is hard to win.
- Program keeps changing secret word to make it hard to guess!
- User never knows!
- Once a letter is chosen and shown in a location, program picks from words that only have that letter in that location
- Program smart to pick from largest group of words available

Smarty Hangman - Dictionary

- Builds a dictionary of categories
- Start with list of words of correct size
- Repeat
  - User picks a letter
  - Make dictionary of categories based on letter
  - New list of words is largest category
    - Category includes already matched letters
    - List shrinks in size each time

Smarty Hangman Example

- Possible scenario after several rounds

  (secret word: calls) # words possible 176
  You guessed a letter
  You have this many guesses left: 4
  Letters not guessed: bcdfhjklmnpqrstvwxyz
  guessed so far: _ a _ _
  guess a letter or enter + to guess a word: d

  - From list of words with a the second letter.
  From that build a dictionary of list of words
  with no d and with d in different places:
    _ a _ _  147 ← Choose “no d”, most words, 147
    _ add_ 1 ← Only 17 words of this type
    _ a _ d _ 17
    _ a _ d 3
    d add 1
    da_d 1
    da_ _ 6

- Key is string, value is list of strings that fit

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;O _ O _&quot;</td>
<td>[ &quot;OBOE&quot;, &quot;ODOR&quot; ]</td>
</tr>
<tr>
<td>&quot; _ O O _&quot;</td>
<td>[ &quot;NOON&quot;, &quot;ROOM&quot;, &quot;HOOP&quot; ]</td>
</tr>
<tr>
<td>&quot; _ O _O&quot;</td>
<td>[ &quot;SOLO&quot;, &quot;GOTO&quot; ]</td>
</tr>
<tr>
<td>&quot; _ _ _ O&quot;</td>
<td>[ &quot;TRIO&quot; ]</td>
</tr>
<tr>
<td>&quot;O _ _ _&quot;</td>
<td>[ &quot;OATH&quot;, &quot;OXEN&quot; ]</td>
</tr>
<tr>
<td>&quot; _ _ _ _&quot;</td>
<td>[ &quot;PICK&quot;, &quot;FRAT&quot; ]</td>
</tr>
</tbody>
</table>

Everytime guess a letter, build a dictionary based on that letter

- Example: Four letter word, guess o

  Key | Value
  --- | ----
  "O _ O _" | [ "OBOE", "ODOR" ]
  " _ O O _" | [ "NOON", "ROOM", "HOOP" ]
  " _ O _O" | [ "SOLO", "GOTO" ]
  " _ _ _ O" | [ "TRIO" ]
  "O _ _ _" | [ "OATH", "OXEN" ]
  " _ _ _ _" | [ "PICK", "FRAT" ]
Keys can’t be lists

- [“O”,”_”,”O”,”_”] need to convert to a string to be the key representing this list: “O_O_”

Smarty Hangman

- How to start? How to modify assignment 5?
- What helper function(s) would be useful?
- Helper function to create new list of words, new secret word, new template
  - Pass in current list of words, current template (such as [‘_’, ‘a’, ‘_’, ‘_’, ‘_’]) and the letter guessed
  - Pass back a tuple : new secret word, now list of words, new template
    \((x, y, z) = \text{someFunction}(a, b, c)\)

Take another look at the last two problems from last time

Problem 6: Which code is better?

- Problem: Given a string parameter named phrase and string named letter, the function findWords returns a list of all the words from phrase that have letter in them.
- Example:
  - findWords("the circus is coming to town with elephants and clowns", "o") would return ['coming', 'to', 'town', 'clowns']
Different questions about these
bit.ly/101f17-1107-1

```python
def findWords(phrase, letter):
    return [phrase.split()[i] for i in range(len(phrase.split()))
            if letter in phrase.split()[i]]
```

```python
def findWords2(phrase, letter):
    wordlist = phrase.split()
    answer = []
    for i in range(len(wordlist)):
        if letter in wordlist[i]:
            answer.append(wordlist[i])
    return answer
```

How long does phrase.split() take?

“how are you”

```
words = ['how']
words = ['how', 'are']
words = ['how', 'are', 'you']
```

For **N chars** in phrase, takes about **N steps**

How long for this solution?

Have **N chars, M words** in phrase

```python
def findWords(phrase, letter):
    return [phrase.split()[i] for i in range(len(phrase.split()))
            if letter in phrase.split()[i]]
```

- `range(len(phrase.split()))`
- `for i in range(…):
  - If letter in phrase.split()[i]
  - phrase.split()[i]`
How long for this solution?
Have N chars, M words in phrase

```python
def findWords(phrase, letter):
    return [phrase.split()[i] for i in range(len(phrase.split()))
    if letter in phrase.split()[i] ]
```

- `range(len(phrase.split()))`  • N steps
- `for i in range(...):`  • For M words
  - If letter in phrase.split()[i]  • N steps
  - phrase.split()[i]  • N steps

Total: $N + M*(2*N)$
$\sim M*N$

Problem 7 – Which number appears the most times?

- The function `most` has one parameter `nums`, a list of integers. This function returns the number that appears the most in the list.

- For example, the call `most([3,4,2,2,3,2])` returns 2, as 2 appears more than any other number.
How long for Solution 1?

\[
\text{def } \text{most}(\text{nums}): \\
\text{maxcnt} = 0 \\
\text{maxnum} = -1 \\
\text{cnts} = [0 \text{ for } n \text{ in range}(\text{max(nums)}+1)] \\
\text{for num in nums:} \\
\text{\quad cnts[num] += 1} \\
\text{\quad if cnts[num] > maxcnt:} \\
\text{\quad \quad maxcnt = cnts[num]} \\
\text{\quad \quad maxnum = num} \\
\text{return maxnum}
\]

How long for Solution 2

bit.ly/101f17-1107-2

\[
\text{def } \text{most2}(\text{nums}): \\
\text{maxcnt} = 0 \\
\text{maxnum} = -1 \\
\text{cnts} = [0 \text{ for } n \text{ in range}(\text{max(nums)}+1)] \\
\text{for num in set(nums):} \\
\text{\quad cnt = nums.count(num)} \\
\text{\quad if cnt > maxcnt:} \\
\text{\quad \quad maxcnt = cnt} \\
\text{\quad \quad maxnum = num} \\
\text{return maxnum}
\]

How long for Solution 1?

N numbers

\[
\text{def } \text{most}(\text{nums}): \\
\text{maxcnt} = 0 \\
\text{maxnum} = -1 \\
\text{cnts} = [0 \text{ for } n \text{ in range}(\text{max(nums)}+1)] \\
\text{for num in nums:} \\
\text{\quad cnts[num] += 1} \\
\text{\quad if cnts[num] > maxcnt:} \\
\text{\quad \quad maxcnt = cnts[num]} \\
\text{\quad \quad maxnum = num} \\
\text{return maxnum}
\]

How long for Solution 1?

N numbers

\[
\text{def } \text{most}(\text{nums}): \\
\text{maxcnt} = 0 \\
\text{maxnum} = -1 \\
\text{cnts} = [0 \text{ for } n \text{ in range}(\text{max(nums)}+1)] \\
\text{for num in nums:} \\
\text{\quad cnts[num] += 1} \\
\text{\quad if cnts[num] > maxcnt:} \\
\text{\quad \quad maxcnt = cnts[num]} \\
\text{\quad \quad maxnum = num} \\
\text{return maxnum}
\]

N steps

Loop N times

1 step

3 steps

Total: \(N + 4N\)

\sim N
Example: [5, 3, 4, 3, 3, 5]

- [0, 0, 0, 0, 0, 0] counters from 0 to 5
- 0 1 2 3 4 5 indexes
- After updating the counts we have:
  - [0, 0, 0, 3, 1, 2] 3 3’s, 1 4, 2 5’s
- 0 1 2 3 4 5

---

How long for Solution 2
N numbers (M unique numbers)

```python
def most2(nums):
    maxcnt = 0
    maxnum = -1
    for num in set(nums):
        cnt = nums.count(num)
        if cnt > maxcnt:
            maxcnt = cnt
            maxnum = num
    return maxnum
```

Loop M times
- N steps
- 3 steps
- Total: M * (N+3) ~ M*N
Python shortcut you can ignore

- The zip function, tuples from two lists
- Does something right if lists have different sizes. Look it up

```python
tools = [1, 2, 3, 4, 5]
fruits = ['apple', 'banana', 'cherry', 'date', 'elderberry']
zip_tools_fruits = zip(tools, fruits)
[(1, 'apple'), (2, 'banana'), (3, 'cherry'), (4, 'date'), (5, 'elderberry')]
```

Python functions you CANNOT ignore

- We know how to sort, we call `sorted`
  - Example: sorting tuples
  - Function `sorted` returns a new list, original not changed

```python
xx = [('dog', 3), ('cat', 2), ('fish', 1), ('guava', 2)]
yy = sorted(xx)
[('cat', 2), ('dog', 3), ('fish', 1), ('guava', 2)]
```

- What if sort by numbers instead of words?

Use what you know

- You can re-organize data to sort it as you'd like, list comprehensions are your friend

```python
xx = [('dog', 3), ('cat', 2), ('fish', 1), ('guava', 2)]
...nlist = [(t[1], t[0]) for t in xx]
[(3, 'dog'), (2, 'cat'), (1, 'fish'), (2, 'guava')]
```

APT – SortedFreqs

bit.ly/101f17-1107-3

The returned frequencies represent an alphabetic/lexicographic ordering of the unique words, so the first frequency is how many times the alphabetically first word occurs and the last frequency is the number of times the alphabetically last word occurs

```python
filename: SortedFreqs.py
def freqs(data):
    """
    return list of int values corresponding to frequencies of strings in data, a list of strings
    """

data = ['apple', 'pear', 'cherry', 'apple', 'cherry', 'pear', 'apple', 'banana']
Returns: [3,1,2,2]
```
Ways to count?

bit.ly/101f17-1107-4

- Dictionaries are faster than using lists?
- How fast is list.count(x) for each x?
- For each you are given a list of words…

Shafi Goldwasser

- 2012 Turing Award Winner
- RCS professor of computer science at MIT
  - Twice Godel Prize winner
  - Grace Murray Hopper Award
  - National Academy
  - Co-inventor of zero-knowledge proof protocols

*How do you convince someone that you know [a secret] without revealing the knowledge?*

- Honesty and Privacy

Work on what you like, what feels right, I know of no other way to end up doing creative work

APT Customer Statistics

bit.ly/101f17-1107-5

**Problem Statement**

You will be given a string list names, containing a list of customer names extracted from a database. Your task is to report the customers that occur more than once in this list, and the number of occurrences for each of the repeated customers.

```python
Returns: ['A 4', 'B 2', 'D 3']
```