CompSci 101
Introduction to Computer Science

<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>

Nov 2, 2017
Prof. Rodger
Announcements

• No Reading/RQ until after Exam 2
• Assignment 5 due, Assignment 6 due Nov 8
• APT 6 due Tuesday
• APT Quiz 2 - Sunday-Wednesday

• Today:
  – Debugging
  – Which code is better?
Assignment 7 – Demo
Smarty, Evil, Frustrating Hangman

• Computer changes secret word every time player guesses to make it "hard" to guess
  – Must be consistent with all previous guesses
  – Idea: the more words there are, harder it is
    • Not always true!

• Example of greedy algorithm
  – Locally optimal decision leads to best solution
  – More words to choose from means more likely to be hung
Canonical Greedy Algorithm

• How do you give change with fewest number of coins?
  – Pay $1.00 for something that costs $0.43
  – Pick the largest coin you need, repeat
Greedy not always optimal

• What if you have no nickels?
  – Give $0.31 in change
  – Algorithms exist for this problem too, not greedy!
Smarty Hangman

• When you guess a letter, you're really guessing a category (secret word "salty")
_ _ _ _ _ and user guesses 'a'

  – "gates", "cakes", "false" are all a the same, in 2cd position
  – "flats", "aorta", "straw", "spoon" are all a in different places

• How can we help ensure player always has many words to distinguish between?
number of misses left: 8
secret so far: _ _ _ _ _ _ _ (word is catalyst)
# possible words: 7070
guess a letter: a
a___a___a 1
...
_a______ 587
__aa____ 1
...
__a_____ 498
_________ 3475
___a____ 406
...
____a___ 396
# keys = 48

number of misses left: 7
letters guessed: a
...
(word is designed)
# possible words: 3475
guess a letter:
Debugging Output and Game Play

• Sometimes we want to see debugging output, and sometimes we don't
  – While using microsoft word, don't want to see the programmer's debugging statements
  – Release code and development code

• You'll approximate release/development using a global variable DEBUG
  – Initialize to False, set to True when debugging
  – Ship with DEBUG = False
Look at howto and categorizing words

• Play a game with a list of possible words
  – Initially this is all words
  – List of possible words changes after each guess

• Given template "_ _ _ _", list of all words, and a letter, choose a secret word
  – Choose all equivalent secret words, not just one
  – Greedy algorithm, choose largest category
Computing the Categories

• Loop over every string in words, each of which is consistent with guess (template)
  – This is important, also letter *cannot* be in guess
  – Put letter in template according to word
  – _ _ _ a _ t might become _ _ _ a n t

• Build a dictionary of templates with that letter to all words that fit in that template.

• How to create key in dictionary?
Everytime guess a letter, build a dictionary based on that letter

- Example: Four letter word, guess o

<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>

- Key is string, value is list of strings that fit
Keys can’t be lists

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

- **software 'bug'**
- **Start small**
  - Easier to cope
  - Simplest input?
- **Judicious 'print'**
  - Debugger too
- **Python tutor**
  - Visualizes data
  - step through
- **Verify the approach being taken, test small, test frequently**
  - How do you 'prove' your code works?
Debugging Problems

• Today the main focus is on debugging.
• There are several problems. Trace by hand to see if you can figure out if they are correct or not, or what to do to correct them.
Debug 1 – Does it work?
bit.ly/101f17-1102-1

• The function `sizes` has a parameter named `words` that is a list of strings. This function returns a list of the sizes of each string. For example, `sizes(['This', 'is', 'a', 'test'])` should return the list `[4, 2, 1, 4]`

```python
def sizes(words):
    nums = []
    for w in words:
        nums = len(w)
    return nums
```
Debug 2 – Does it work?

bit.ly/101f17-1102-2

• The function `buildword` has a parameter `words` that is a list of strings. This function returns a string that is made up of the first character from each word in the list. For example, `buildword(['This', 'is', 'a', 'test'])` returns 'Tiat'

def buildword(words):
    answer = ''
    for w in words:
        answer += w[:1]
    return answer
Debug 3 – Does it work?

- The function *middle* has a parameter *names* that is a list of strings, which each string is in the format "firstname:middlename:lastname". This function returns a list of strings of the middlenames.

For example, the call `middle( "Jo:Mo:Tree", "Mary:Sue:Perez", "Stephen:Lucas:Zhang" )` returns

[ ‘Mo’, ‘Sue’, ‘Lucas’]
Debug 3 – Does it work?
bit.ly/101f17-1102-3

• The function `middle` has a parameter `names` that is a list of strings, which each string is in the format "firstname:middlename:lastname". This function returns a list of strings of the middlenames.

```
def middle(names):
    middlelist = []
    for name in names:
        name.split("":"")
        middlelist.append(name[1])
    return middlelist
```
Debug 4 – Does it work?
bit.ly/101f17-1102-4

• The function `removeOs` has one string parameter named `names`. This function returns a string equal to `names` but with all the lowercase o's removed. For example, `removeOs('Mo Moo Move Over')` returns 'M M Mve Over'

```python
def removeOs(word):
    position = word.find("o")
    while position != -1:
        word = word[:position] +
        word[position+1:]
    return word
```

compsci 101 fall 2017
Problem 5 – Does it work?

bit.ly/101f17-1102-5

• The function uniqueDigits has one int parameter number. This function returns the number of unique digits in number. For example, the call uniqueDigits(456655) should return 3.

def uniqueDigits(number):
    digits = []
    while number > 0:
        digits.append(number % 10)
        number = number / 10
    return len(digits)
Which code is better?

• For the next two problems, we will look at two examples of code that both work in solving the problem, and think about which code is better.
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']
Consider two solutions, which is better? bit.ly/101f17-1102-6

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

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
```
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.
def most(nums):
    maxcnt = 0
    maxnum = -1
    cnts = [0 for n in range(max(nums)+1)]
    for num in nums:
        cnts[num] += 1
        if cnts[num] > maxcnt:
            maxcnt = cnts[num]
            maxnum = num
    return maxnum
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