

CompSci 101

Introduction to Computer Science

Nov. 7, 2017

Prof. Rodger

0	1	2	3	4	5	6
"apple"	"fig"	"apple"	"banana"	"cherry"	"fig"	"apple"

compsci 101 fall 2017

1

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

compsci 101 fall 2017

2

Announcements

- No RQs until after Exam 2
 - Assignment 7 due ~~Thursday~~ - **now Monday**
 - 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

compsci 101 fall 2017

3

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

compsci 101 fall 2017

5

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

compsci 101 fall 2017

6

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: bcd fghjklmnpqrstvwxyz
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
_a_d_ 17 ← Only 17 words of this type
_ad_ 3
dadd_ 1
da_d_ 1 ← Only 1 word of this type
da_ 6
```

compsci 101 fall 2017

7

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"]

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

compsci 101 fall 2017

8

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) = 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

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

```
def findWords2(phrased, 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”

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

```
def findWords(phrased, 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

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

How long for this solution?

Have N chars, M words

```
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
```

- `phrase.split()`
- `for each word`
 - `If letter in word`
 - `Answer.append`

How long for this solution?

Have N chars, M words

```
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
```

- `phrase.split()` • N steps
 - `for each word` • For M words
 - `If letter in word` – `len(word)` steps
 - `Answer.append` – 1 step
- } N steps
- Total: $N + N = \sim N$

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?

```
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
```

compsei 101 fall 2017

21

How long for Solution 2?

bit.ly/101f17-1107-2

```
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
```

compsei 101 fall 2017

22

How long for Solution 1?

N numbers

```
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
```

compsei 101 fall 2017

23

How long for Solution 1?

N numbers

```
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
```

N steps
Loop N times
1 step
3 steps
Total: $N + 4N$
 $\sim N$

compsei 101 fall 2017

24

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

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)

```
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
```

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

```
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: $M * (N+3)$
 $\sim M*N$

Python shortcut you can ignore

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

```
words = ['dog', 'cat', 'fish', 'guava']
counts = [3, 2, 1, 5]
cc = zip(word,counts)
```

```
[('dog', 3), ('cat', 2), ('fish', 1), ('guava', 5)]
```

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

```
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

```
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')]
```

```
yy = sorted(nlist)
```

```
[(1, 'fish'), (2, 'cat'), (2, 'guava'), (3, 'dog')]
```

APT – SortedFreqs bit.ly/101f17-1107-3

Specification

```
filename: SortedFreqs.py

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

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

```
data = ["apple", "pear", "cherry", "apple", "cherry", "pear", "apple", "banana"]
```

```
Returns: [3,1,2,2]
```


Ways to count?

bit.ly/101f17-1107-4

0	1	2	3	4	5	6
"apple"	"fig"	"apple"	"banana"	"cherry"	"fig"	"apple"

- 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.

```
names = ["A", "B", "A", "C", "A", "B", "A", "D", "D", "D"]
```

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