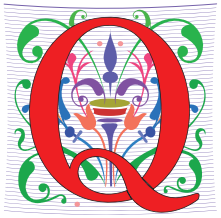
Compsci 101
Dictionaries Practice, Clever GuessWord

```
28  def fastcount(words):
29      d = {}
30      for w in words:
31          if w in d:
32              d[w] += 1
33          else:
34              d[w] = 1
35      return sorted(d.items())
```

Susan Rodger
March 17, 2022

Q is for …



- **QR code**
  - Black and white and read all over
- **Quicksort**
  - Sort of choice before Timsort?
- **QWERTY**
  - When bad ideas persist

Christine Alvarado

- Teaching Professor, UCSD
- PhD Computer Science, MIT
- Her work is in designing CS curriculum that is more accessible and more appealing to all
- LogiSketch – draw and simulate digital circuits

"It's important to choose your own path, and try not to compare yourself to others. You have your own unique circumstance, so what others do or don't do shouldn't really affect your life."

Announcements

- Assignment 4 GuessWord due today!
- APT-5 due Thur, March 24
  - Recommend to do before Exam 3
- Lab 8 Friday, do prelab
- Assignment 5 due March 29
- Exam 2 regrades by March 23
- Exam 2 booster – 8 pts - Take by March 23
- Mid-Semester Survey going out
  - How are we doing, How are UTAs doing, etc
  - 2 extra pts on Exam 3 if 75% of you fill it out!

Exam 3 – in person – Tues, March 22

- Exam is in class on paper – 10:15am
  - Need pen or pencil
- See materials under 3/22 date
  - Exam 3 Reference sheet - part of exam
- Covers
  - Topics: sets, parallel lists, dictionaries, sorting, tuples, (No images)
  - APTs through APT5
  - Labs through Lab 8
  - Assignments through Assignment 4
  - Sakai Quizzes through 3/17

Tuesday
3/22
EXAM 3 Python Reference Sheet for Exam 3 Specific old tests Old Tests Reviewer App

PFTD

- Dictionaries
  - More Practice
  - Fast!
- Family APT
- Clever GuessWord

return [w for w in words if commonCount(w, userword)]

Finish up Jotto

- Last time I made an error:
  - Didn't test updateWordList
- Bad code below. What does this do?

```
def updateWordList(words, numInCommon, userword):
    return [w for w in words if commonCount(w, userword)]
```


Finish up Jotto (2)

- Correct code for updateWordList

```
def updateWordList(words, numInCommon, userword):
    return [w for w in words
            if commonCount(w, userword) == numInCommon]
```

- Now run Jotto. Computer wins a lot!

```
Guess a word with 5 letters: beach
next word is:  femur
num letters in common is:  1
words remaining:  2534
words remaining:  2
next word is:  beach
num letters in common is:  5
words remaining:  1
I won! :D
```

Dictionary Iteration (unordered!)

- Iterate through keys:
  - `for k in d:`
  - `for k in d.keys():`
- Iterate through pairs:
  - `for (k,v) in d.items():`
  - `for k,v in d.items():`

Sorting a list from dictionary - sorted()

d = {'k': 3, 'h': 8, 'a': 12, 'd': 5}

x = sorted(d.keys())
y = sorted(d.values())
z = sorted(d.items())

WordFrequencies Dictionary Example

- Let's see an example that compares using a dictionary vs not using a dictionary

slowcount function Short Code and Long Time

- See module WordFrequencies.py
  - Find # times each word in a list of words occurs
  - We have tuple/pair: word and word-frequency

```
37  def slowcount(words):
38      pairs = [(w,words.count(w)) for w in set(words)]
39      return sorted(pairs)
```

- Think: How many times is **`words.count(w)`** called?
  - Why is **`set(words)`** used in list comprehension?

WordFrequencies with Dictionary

- If start with a million words, then…
- We look at a million words to count # "cats"
  - Then a million words to count # "dogs"
  - Could update with parallel lists, but still slow!
  - Look at each word once: dictionary!
- Key idea: use word as the "key" to find occurrences, update as needed
  - Syntax similar to **`counter[k] += 1`**

Using fastcount

- Update count if we've seen word before
  - Otherwise it's the first time, occurs once

```
28  def fastcount(words):
29      d = {}
30      for w in words:
31          if w in d:
32              d[w] += 1
33          else:
34              d[w] = 1
35      return sorted(d.items())
```


Let's run them and compare them!

- Run with Melville and observe time
- Run with Hawthorne and observe time

WOTO-1 Counting Dictionaries
http://bit.ly/101s22-0318-1

APT Family

APT: Family

Problem Statement

You have two lists: `parents` and `children`. The ith element in `parents` is the parent of the ith element in `children`. Count the number of grandchildren (the children of a person's children) for the person in the `person` variable.

Hint: Consider making a helper function that returns a list of a person's children.

Step 1: work an example by hand

```
parents = ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children = ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']
person =  'Junhua'

Returns 3
```


Helper function

```
def childrenOf(parents, children, name):
    <missing code to traverse parallel lists>
    return list of name's children
```


How to traverse parallel lists?

parents: ['Junhua', 'Anshul', 'Junhua', 'Anshul', 'Kerry']
children: ['Anshul', 'Jordan', 'Kerry', 'Paul', 'Kai']
0 1 2 3 4

Assignment 5 - How to play Guess Word Cleverly

- Make it hard for the player to win!
- One way: Try hard words to guess?
  - "jazziest", "joking", "bowwowing"
- Another Way: Keep changing the word, sortof

Clever GuessWord

- Current GuessWord: Pick random secret word
  - User starts guessing
- Can you change secret word?
  - Yes, but must have letters in same place you have told user
    - Change consistent with all guesses
  - Make the user work harder to guess!

Programming A Clever Game

- Instead of guessing a word, you're guessing a *group, category*, or *equivalence class* of words
  Ex: `_ _ _ _ _` and user guesses 'a'
- **`["asked", "adult", "aided", … "axiom"]`**
  - *209 words 'a' as first letter and the only 'a'*
- **`["baked", "cacti", "false", … "walls"]`**
  - *665 words 'a' as second letter and the only 'a'*
- **`["beets", "humor", … "spoof"]`**
  - *2,431 words with no 'a'*
- What should our secret word be? "asked" ,"baked" or "beets“?

Sometimes there will be letters

- The letter “u” has been guessed and is the 2nd letter
  Ex: **`_ u _ _ _`** and user guesses ‘r’
- **`["ruddy", "rummy", "rungs", … "rusty"]`**
  - *5 words start with “ru” and no other “r” or “u”*
- **`["burch", "burly", "burns", … "turns"]`**
  - *17 words only ‘u' as second letter and only ‘r’ third letter*
- **`["bucks", "bucky", … "tufts"]`**
  - *98 words with only “u” second letter and no ‘r'*
- What should our secret word be? "ruddy" ,"burch" or "bucks“?

More Details on Game

- Current secret 8-letter word at random is *catalyst*
  - User guesses 'a', what should computer do?
  - Print `_ a _ a _ _ _ _` and continue?

Creating Groups/Categories

- For each of 7,070 words (8 letters), given word and ‘a’, find its group, represented by a template
- Use dictionary
  - Template is KEY, the VALUE is a list of matching words
- Choose biggest list
- Repeat
- # words smaller over time

Group/Template	Size of Group
_ a _ _ _ _ _ _	587
_ a _ a _ _ _ _	63
_ _ a _ _ _ _ _	498
_ _ _ a _ _ _ _	406
_ _ _ _ _ _ _ _	3,475

Changes to Regular GuessWord

- List of words from which secret word chosen
  - Initially this is all words of specified length
    - User will specify the length of the word to guess
  - After each guess, word list is a new subset
- Keep some functions, modify some, write new ones
- *Changes go in another function* to minimize changes to working program
  - Minimizing changes helps minimize introducing bugs into a working program

Play a game

- `_ _ _ _ _ _ _`
- Secret word is:
  - **`flamer`**
- User guesses:
  - a
- Possible words:
  - 6166

```
_______ : 3441
______a : 80
_____a_ : 233
____a__ : 316
____a_a : 11
___a___ : 549
___a__a : 19
___a_a_ : 10
___aa__ : 1
_a_____ : 962
_a___a_ : 39
_a__a_ : 57
_a_a___ : 40
_a_a_a : 12
_a_aa_ : 3
a______ : 273
a____a : 21
a___a_ : 30
a__a___ : 32
a__a_a : 3
a_a____ : 26
a_a__a : 7
aa_____ : 1
```

Consider "_ _ _ a _ a" : 11

- Means "_ _ _ a _ a" is key in dictionary
- The value is a list of 11 words
  - have "a' in 4th and 6th position

"_ _ _ a _ a"

['cicada', 'errata', 'guiana', 'guyana', 'ithaca',
'lusaka', 'nevada', 'ottawa', 'sonata', 'tirana', 'urbana']

Play a game

- _ _ _ _ _ _ _
- Secret word is:
  - mounds
- User guesses:
  - o
- Possible words:
  - 3441

```
_______ : 2105
______o : 23
_____o_ : 147
_____oo : 1
____o__ : 148
____o_o : 1
____oo_ : 4
___o___ : 228
___o__o : 2
___o_o_ : 8
___oo__ : 32
__o____ : 528
__o___o : 6
__o__o_ : 41
__o_o__ : 15
__o_o_o : 1
__o_oo_ : 1
__oo___ : 77
__oo_oo : 1
_o_____ : 60
_o___o_ : 3
_o__o__ : 8
_o__oo_ : 1
```


Play a game

- _ _ _ _ _ _ _
- Secret word is:
  - **burkes**
- User guesses:
  - u
- Possible words:
  - 2105

```
_______ : 1441
______u : 2
_____u_ : 36
____u__ : 84
____u_u : 1
___u___ : 107
__u____ : 362
_u_____ : 13
_u__u_ : 11
_u_u__ : 37
u______ : 5
u___u_ : 5
u__u__ : 1
u_u____ : 1
```


Play a game

- _ _ _ _ _ _ _
- Secret word is:
  - **wilted**
- User guesses:
  - i
- Possible words:
  - 1441

```
_______ : 503
______i : 2
_____i_ : 54
____i__ : 158
____i_i : 2
___i___ : 225
___i__i : 1
___i_i_ : 7
___ii__ : 2
__i____ : 355
__i__i_ : 28
__i_i__ : 56
__i_i_i : 2
_i_____ : 28
_i__i__ : 16
_i_i____ : 2
```

Play a game

- _ _ _ _ _ _ _
- Secret word is:
  - **served**
- User guesses:
  - e
- Possible words:
  - 503

```
_____ : 2
____e : 5
___e_ : 13
___e__ : 9
___e_e : 2
___ee_ : 5
__e___ : 42
__e__e : 12
__e_e_ : 23
__ee__ : 36
__ee_e : 9
_e____ : 13
_e___e : 13
_e__e_ : 160
_e__ee : 2
_e_e__ : 59
_e_e_e : 7
_e_ee_ : 3
_ee___ : 6
_ee__e : 5
_ee_e_ : 34
e___e : 1
e__e_ : 5
e__ee : 2
e_e__ : 20
e_ee_ : 2
e_e___ : 9
e_e__e : 1
e_e_e_ : 3
```


Play a game

- _ e _ _ e _
- Secret word is:
  - **tested**
- User guesses:
  - s
- Possible words:
  - 160

```
_e__e_ : 100
_e__es : 16
_e_se_ : 11
_e_ses : 3
_es_e_ : 13
_esse_ : 5
_esses : 1
se__e_ : 7
se__es : 2
se_se_ : 1
se_ses : 1
```


Play a game

- _ e _ _ e _
- Secret word is:
  - **kepler**
- User guesses:
  - r
- Possible words:
  - 100

```
_e__e_ : 45
_e__er : 32
_e_re_ : 1
_er_e_ : 8
_er_er : 6
_erre_ : 1
_errer : 1
re__e_ : 3
re__er : 2
re_re_ : 1
```


Play a game

- _ e _ _ e _
- Secret word is:
  - **wedded**
- User guesses:
  - d
- Possible words:
  - 45

```
_e__e_ : 11
_e__ed : 20
_e_de_ : 2
_e_ded : 4
_ed_e_ : 1
_ed_ed : 2
_edded : 2
de__e_ : 1
de__ed : 2
```

Play a game

- `_e__ed`
- Secret word is:
  - **belted**
- User guesses:
  - l
- Possible words:
  - 20

```
_e__ed : 10
_el_ed : 4
_elled : 5
le__ed : 1
```


Play a game

- `_e__ed`
- Secret word is:
  - **vented**
- User guesses:
  - t
- Possible words:
  - 4

```
_e__ed : 4
_e_ted : 1
_etted : 4
te_ted : 1
```


Greedy Algorithms

- "Choosing largest group" -> *greedy algorithm*
  - Make a locally optimal decision that works in the long run
  - Choose largest group to make game last …
- Greed as in "it chooses the best current choice every time, which results in getting the best overall result"
- Canonical example? Change with coins
  - Minimize # coins given for change: 57 cents

Making change for 57 cents

- When choose next coin, always pick biggest
- With half-dollar coins

- With quarters and no half dollars

When greedy doesn't work

- What if no nickels? Making change for 31 cents:

Woto-2 Clever GuessWord
http://bit.ly/101s22-0317-2

Problem Solving

- Given Brodhead University. They have a basketball team.
- Data on players and how they did when playing against another team.

- List of lists named datalist
  - Each list has
    - school opponent name
    - player name
    - Points player scored
    - Whether game was ‘won’ or ‘lost’

Example: lists of 20 lists
datalist =

```
[  ['Duke', 'Bolton', '2', 'lost’],
   ['NCSU', 'Stone', '12', 'won’],
   ['Duke', 'Kreitz', '3', 'lost’],
   ['Duke', 'Pura', '6', 'lost’],
   ['GT', 'Dolgin', '4', 'lost’],
   ['WFU', 'Laveman', '20', 'won’],
   ['ECU', 'Parlin', '15', 'won’],
   ['UNC', 'Stone', '17', 'won’],
   ['UNC', 'Dolgin', '12', 'won’],
   ['UNC', 'Kreitz', '5', 'won’],
   ['Duke', 'Stone', '16', 'lost’],
   ['Duke', 'Laveman', '13', 'lost’],
   ['NCSU', 'Kreitz', '8', 'won'],
   ['NCSU', 'Dolgin', '18', 'won’],
   ['NCSU', 'Parlin', '13', 'won'],
   ['GT', 'Bolton', '7', 'lost’],
   ['GT', 'Stone', '9', 'lost'],
   ['WFU', 'Parlin', '14', 'won'],
   ['ECU', 'Laveman', '16', 'won’],
    ['ECU', 'Pura', '15', 'won’] ]
```

1) Write function dictPlayerToNumGamesPlayedIn

Build a dictionary of players mapped to number of games they have played in.

def dictPlayerToNumGamesPlayedIn(datalist):

With previous example, player ‘Laveman’ would be mapped to 3 games

Woto-3 Players and Games Played in http://bit.ly/101s22-0317-3

Calculate list of players who played in 3 or more games, give (player name, number of games played in), sort by player name

[('Dolgin', 3), ('Kreitz', 3), ('Laveman', 3), ('Parlin', 3), ('Stone', 4)]

```
[  ['Duke', 'Bolton', '2', 'lost’],
   ['NCSU', 'Stone', '12', 'won’],
   ['Duke', 'Kreitz', '3', 'lost’],
   ['Duke', 'Pura', '6', 'lost’],
   ['GT', 'Dolgin', '4', 'lost’],
   ['WFU', 'Laveman', '20', 'won’],
   ['ECU', 'Parlin', '15', 'won’],
   ['UNC', 'Stone', '17', 'won’],
   ['UNC', 'Dolgin', '12', 'won’],
   ['UNC', 'Kreitz', '5', 'won’],
   ['Duke', 'Stone', '16', 'lost’],
   ['Duke', 'Laveman', '13', 'lost’],
   ['NCSU', 'Kreitz', '8', 'won'],
   ['NCSU', 'Dolgin', '18', 'won’],
   ['NCSU', 'Parlin', '13', 'won'],
   ['GT', 'Bolton', '7', 'lost’],
   ['GT', 'Stone', '9', 'lost'],
   ['WFU', 'Parlin', '14', 'won'],
   ['ECU', 'Laveman', '16', 'won’],
    ['ECU', 'Pura', '15', 'won'] ]
```


You should be able to:

- Build a dictionary
- Use a dictionary to help solve a problem