Compsci 101
Modules, How Dictionaries Work

Susan Rodger
April 7, 2022
v is for …

• Viral Video
  • Husky Dog sings with iPAD – 18 million views
  • https://www.youtube.com/watch?v=Mk4bmK-acEM

• Virtual Memory
  • It is and is not there!

• Virtual Reality
  • Augmenting IRL
The Power of Collaboration: Ge Wang, Duke Prof. at Stanford

- **Duke 2000: Music and Computer Science**
  - [http://www.youtube.com/watch?v=ADEHmkL3HBg](http://www.youtube.com/watch?v=ADEHmkL3HBg)

- **About Design in Compsci 308**

  *Our investment into a huge and meticulous design process was a huge factor in making later progress. 35000+ lines of code / design / documentation gave us a project we were all very happy and proud to be a part of.*
Announcements

• APT-7 due TODAY!
• APT-8 out, due Thursday, Apr 14
• Assign 6 Recommender, due Apr 19
  • One grace day, **NO LATE DAYS, must be in Apr 20**

• APT Quiz 2 – 11:30am today thru Sunday, April 10
  • Two Parts, Start on Sakai
  • Rules were sent to you, must be your own work!

• Exam 4 – Tues, April 12, in person
  • See study materials on calendar page on 4/12 date
PFTD

• Collaboration and Creativity
  • The power of working together with code
• Review modules and exceptions
  • Concepts used in Lab 11, leveraging creativity
• How dictionaries are so fast
• Exam review
Why use modules?

• Module – Python file (.py file)
• Can have several modules work together

• Easier to organize code
• Easier to reuse code
• Easier to change code
  • As long as the “what” is the same, the “how” can change
    • Ex: sorted(…), one function many sorting algorithms
In Lab, Modules for Creating

• “MadLibs” → Tag-a-Story
  • User chooses template
  • Computer fills everything in

In lecture I saw a <color> <noun>
For lunch I had a <adjective> <food>
The day ended with seeing a <animal> <verb> in <place>
From <noun> to story

In lecture I saw a <color> <noun>
For lunch I had a <adjective> <food>
The day ended with seeing a <animal> <verb> in <place>

In lecture I saw a magenta house
For lunch I had a luminous hummus
The day ended with seeing a cow sleep in Mombasa
Demo

• Run storyline.py
• Show Haiku’s
• Show Lecture template
• Make modifications
Let's create/modify a story

• Choose a template or make a new one
  • We'll choose lecturetemplate.txt first

• Add a new category/replacement
  • We'll choose number and list some choices

• Run the program and test our modifications
  • Randomized, hard to test, but doable
Main Parts for tag-a-story

- Put everything together, the template and words
  - Storyline.py

- Loading and handling user choosing templates
  - TemplateChooser.py

- Loading and picking the word for a given tag
  - Replacements.py
Main Parts for tag-a-story

• Put everything together, the template and words
  • Storyline.py

• Loading and handling user choosing templates
  • TemplateChooser.py

• Loading and picking the word for a given tag
  • Replacements.py
Creating a story

• Main steps in Storyline.py
  • Get template – use module TemplateChooser
  • Go through template
    • Get words for a tag – use module Replacements
    • Replace tag with word

• Using modules
  • Assume they work
  • Only care *what* they do, not *how* (abstraction!)
Modules in Action:
makeStory() is in Storyline.py

- How can we access TemplateChooser functions?
- import and access as shown

```python
41     def makeStory():
42         
43         let user make a choice of
44         available templates and print
45         the story from the chosen template
46         
47         lines = TemplateChooser.getTemplateLines("templates")
48         st = linesToStory(lines)
49         print(st)
```
Modules in Action: 
makeStory() is in Storyline.py

• How can we access TemplateChooser functions?
  • import and access as shown

```python
def makeStory():
    """
    let user make a choice of
    available templates and print
    the story from the chosen template
    """
    lines = TemplateChooser.getTemplateLines("templates")
    st = linesToStory(lines)
    print(st)
```
Modules in Action:
makeStory() is in Storyline.py

- How can we access TemplateChooser functions?
- import and access as shown

```python
41 def makeStory():
42     """
43     let user make a choice of available templates and print the story from the chosen template """
44     lines = TemplateChooser.getTemplateLines("templates")
45     st = linesToStory(lines)
46     print(st)
```

A function in the file: TemplateChooser.py
Modules in Action:
linesToStory() is in Storyline.py

- We call doWord() – does replacements for words

```python
def linesToStory(lines):
    ""
    lines is a list of strings, each a line from a template file
    Return a string based on substituting for each <tag> in each line
    ""
    story = ""
    for line in lines:
        st = ""
        for word in line.split():
            st += doWord(word) + " "
        story += st.strip() + "\n"
    return story
```
Modules in Action:
linesToStory() is in Storyline.py

- We call doWord() – does replacements for words

```python
def linesToStory(lines):
    """
    lines is a list of strings, each a line from a template file
    Return a string based on substituting for each <tag> in each line
    """

    story = ""
    for line in lines:
        st = ""
        for word in line.split():
            st += doWord(word) + " "
        story += st.strip() + "\n"
    return story
```
Understanding Code/Module

doword is in storyline.py

• What does getReplacement do?
  • How does getReplacement do it?

```python
def doWord(word):
    """
    word is a string
    if word is <tag>, find replacement
    and return it. Else return word
    """
    start = word.find("<")
    if start != -1:
        end = word.find(">")
        tag = word[start+1:end]
        rep = Replacements.getReplacement(tag)
        return rep
    return word
```
Understanding Code/Module

doWord is in Storyline.py

• What does getReplacement do?
  • How does getReplacement do it?

```python
def doWord(word):
    
    word is a string
    if word is <tag>, find replacement
    and return it. Else return word
    
    start = word.find("<")
    if start != -1:
        end = word.find(">")
        tag = word[start+1:end]

        rep = Replacements.getReplacement(tag)
        return rep

    return word
```
Understanding Code/Module

doWord is in Storyline.py

• What does getReplacement do?
  • How does getReplacement do it?

```python
def doWord(word):
    """
    word is a string
    if word is <tag>, find replacement and return it. Else return word
    """

    start = word.find("<")
    if start != -1:
        end = word.find(">")
        tag = word[start+1:end]

        rep = Replacements.getReplacement(tag)
        return rep

    return word
```

A function in the file: Replacements.py
Main Parts for tag-a-story

• Put everything together, the template and words
  • Storyline.py

• Loading and handling user choosing templates
  • TemplateChooser.py

• Loading and picking the word for a given tag
  • Replacements.py
Another module TemplateChooser.py

• Get template
  • TemplateChooser.getTemplateLines(DIR)
  • What:
    • From the templates in the directory DIR (type: str)
    • Return a list of strings, where each element is a line from one of the templates in DIR

• Word for a tag
  • Replacements.getReplacement(TAG)
  • What:
    • Return a random word that matches TAG (type: str)
Where is it called from?

- In module Storyline.py, function makestory

```python
lines = TemplateChooser.getTemplateLines("templates")
```

- Where templates is a folder with three templates:
TemplateChooser.py Steps

- List all templates in the folder
- Get user input that chooses one
- Load that template
- Return as list of strings
TemplateChooser.py Steps

• List all templates in the folder
  • `pathlib` Library

• Get user input that chooses one
  • Handle bad input → try…except

• Load that template
  • Open file, `.readlines()`

• Return as list of strings
These Steps in Code getTemplateLines in TemplateChooser.py

• Read directory of templates, convert to dictionary
  • Let user choose one, open and return it

```python
def getTemplateLines(dirname):
    
    """
    dirname is a string that's the name of a folder
    Prompt user for files in folder, allow user
    to choose, and return the lines read from file
    """

    d = dirToDictionary(dirname)
    lines = chooseOne(d)
    return lines
```
Creating User Menu

dirToDictionary in TemplateChooser.py

• What does this function return? What type?

```python
11  def dirToDictionary(dirname):
12    
18  d = {}
19  index = 0
20  for one in pathlib.Path(dirname).iterdir():
21    d[index] = one
22    # print(type(one))
23    index += 1
24  return d
```
Creating User Menu
dirToDictionary in TemplateChooser.py

- What does this function return? What type?

```python
11   def dirToDictionary(dirname):
12       
18     d = {}
19     index = 0
20     for one in pathlib.Path(dirname).iterdir():
21         d[index] = one
22         # print(type(one))
23         index += 1
24     return d
```

d is:
0 -> haiku.txt
1 -> labtemplate.txt
2 -> lecturetemplate.txt
Folder in Pycharm

Output:

```
0  haiku.txt
1  labtemplate.txt
2  lecturetemplate.txt

------

choose one> 0
the slimy bathtub
reminded them of Africa
chartreuse squeaky brown
```
pathlib Library

• Path:
  “rodger/Pycharm/cps101/lab11/temp/haiku.txt”

• The pathlib library is more recent/Python3
  • Simpler, easier to use than functions from os

• Handles domain specifics!
  • Doesn’t matter if on Windows, Mac, etc.
  • We worry about the what, it handles the how
pathlib Library cont.

• Path:
  “rodger/Pycharm/cps101/lab11/temp/haiku.txt”

• `pathlib.Path(DIR).iterdir()`
  • Returns iterable of Path objects representing each “thing” in the directory DIR

• Path object’s `.parts` – tuple of strings, each element is a piece of a filename’s path
  • (`'rodger'`, `'Pycharm'`, `'cps101'`, `'lab11'`, `'temp'`, `'haiku.txt'`)
• We will return to this, but analyze parts now
  • What's familiar? What's not familiar …

```python
def chooseOne(d):
    while True:
        for key in sorted(d.keys()):
            print("%d\t%s" % (key, d[key].parts[-1]))
        print("-------")
        st = input("choose one> ")
        try:
            val = int(st)
            if 0 <= val and val < len(d):
                return reader(d[val])
        except ValueError:
            print("please enter a number")
```

4/7/22 Compsci 101, Spring 2022 33
Python exceptions

• What should you do if you prompt user for a number and they enter "one"
  • Test to see if it has digits?

• Use exceptions with try: and except:
  • See code in function chooseOne from TemplateChooser.py
Handling Exceptions

- What happens: \( x = \text{int}("123abc") \)

```python
46  st = input("choose one> ")
47  try:
48    val = int(st)
49    if 0 <= val and val < len(d):
50      return reader(d[val])
51  except ValueError:
52    print("please enter a number")
```
WOTO-1 Modules
How do Dictionaries work so fast?

• How are they implemented?
Simple Example

Want a mapping of Soc Sec Num to Names

• Duke's CS Student Union wants to be able to quickly find out info about its members. Also add, delete and update members. Doesn't need members sorted.

  267-89-5431  John Smith
  703-25-6141  Ademola Olayinka
  319-86-2115  Betty Harris
  476-82-5120  Rose Black

• Dictionary d – SSN to names
  • d[‘267-89-5431’] = ‘John Smith’
  • How does it find ‘John Smith’ so fast?
Dictionary $d(\text{SSN}) = (\text{SSN}, \text{name})$

- We actually would map the SSN to the tuple of $(\text{SSN}, \text{name})$.
- That is a lot to display on a slide, so we will just show SSN to name.
- But remember name is really a tuple of $(\text{SSN, name})$.
Simple Example

Let’s look under the hood.
How are dictionaries implemented?

• Dictionaries implemented with a list, in a clever way
• How do we put something into the list fast?
• How do we find it in the list quickly?
  • d[‘267-89-5431’] = ‘John Smith’
• List size is 11 – from 0 to 10
• d[‘267-89-5431’] calculates index location in list of where to put this tuple (SSN,name)
• Use a function to calculate where to store ‘John Smith
  • H(ssn) = (last 2 digits of ssn) mod 11
  • Called a Hash function
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

\[
\begin{align*}
H(267-89-5431) &= 31 \mod 11 = 9 \\
&\quad \text{John Smith} \\
H(703-25-6141) &= 41 \mod 11 = 8 \\
&\quad \text{Ademola Olayinka} \\
H(319-86-2115) &= 15 \mod 11 = 4 \\
&\quad \text{Betty Harris} \\
H(476-82-5120) &= 20 \mod 11 = 9 \\
&\quad \text{Rose Black}
\end{align*}
\]

Collision!
Have a list of size 11 from 0 to 10

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

\[
\begin{align*}
H(267-89-5431) &= 31 \mod 11 = 9 \\
\text{John Smith} \\
H(703-25-6141) &= 41 \mod 11 = 8 \\
\text{Ademola Olayinka} \\
H(319-86-2115) &= 15 \mod 11 = 4 \\
\text{Betty Harris} \\
H(476-82-5120) &= 20 \mod 11 = 9 \\
\text{Rose Black}
\end{align*}
\]

Must resolve collisions
When does this work well?

• When there are few collisions

• You have to deal with collisions

• Use a list large enough to spread out your data
Another way: Use a list of lists

- Insert these into the list
- Insert as (key, value) tuple
  
  (267-89-5431, John Smith)
  
  (in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
  
  John Smith

H(703-25-6141) = 41%11 =  8
  
  Ademola Olayinka

H(319-86-2115 )= 15 %11 =  4
  
  Betty Harris

H(476-82-5120) = 20%11 = 9
  
  Rose Black
Another way: Use a list of lists

- Insert these into the list
- Insert as (key, value) tuple
  (267-89-5431, John Smith)
  (in example, only showing name)

H(267-89-5431) = 31 % 11 = 9
  John Smith
H(703-25-6141) = 41 % 11 = 8
  Ademola Olayinka
H(319-86-2115) = 15 % 11 = 4
  Betty Harris
H(476-82-5120) = 20 % 11 = 9
  Rose Black

Collisions added to list, 2 in list 9
WOTO-2 How Dictionaries Work
Review for Exam 4
Problem 4 Fall 2014 Old Tests

• A programming contest between colleges
• There are problems to solve each has a letter: Problem A through Problem J
• Submit a program for a problem – it is correct or not
• Submit it again if it is not correct.
• Score is total time for problems solved with 20 minute penalty for each wrong submission that was solved eventually!
• Winner is solves most problems – Tie breaker (lowest score
Review for Exam 4
Problem 4 Fall 2014 Old Tests

• Each entry is: 1) school, 2) name of problem, 3) time to solve in minutes, 4) correct or not

• Examples:
['UNC', 'A', '20', 'reject']

['Duke', 'A', '26', 'correct']
Review for Exam 4
Problem 4 Fall 2014 Old Tests

• Each entry is: 1) school, 2) name of problem, 3) time to solve in minutes, 4) correct or not

• Examples:
  ['UNC', 'A', '20', 'reject’]
  UNC submitted problem A in 20 minutes - rejected

  ['Duke', 'A', '26', 'correct’]
  Duke submitted Problem A in 26 minutes - correct
Problem 4 Fall 2014 Old tests
Just look at Duke’s submissions

[...
['Duke', 'A', '26', 'correct'],
['Duke', 'E', '82', 'reject'],
['Duke', 'D', '200', 'correct'],

['Duke', 'E', '210', 'correct'],

Duke score:

Duke score: 26
not correct, no points
plus 200 = 226
plus 210 + 20 (penalty) = 456
Duke has 456 points
Problem 4 Fall 2014 Old tests
Just look at Duke’s submissions

[...]
['Duke', 'A', '26', 'correct'],  Duke score:
['Duke', 'E', '82', 'reject'],  26
['Duke', 'D', '200', 'correct'],  not correct, no points

plus 200 = 226

['Duke', 'E', '210', 'correct'],  plus 210 + 20 (penalty) =

456

Duke has 456 points
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function `listOfSchools(data)`

- returns sorted unique list of schools that submitted a program whether correct or not
- From data should return:

  ```
  ['Duke', 'ECU', 'Elon', 'NCSU', 'UNC', 'USC'].
  ```
Write function `listOfSchools(data)`

- returns sorted unique list of schools that submitted a program whether correct or not
- From data should return:


**Note:** sorted

Unique schools (use sets)

Returns list (must convert set back to list)
Write function listOfSchools(data)

def listOfSchools(data):

setSchools = set([])
for item in data:
    setSchools.add(item[0])
alist = list(setSchools)
return sorted(alist)
Write function `listOfSchools(data)`

def listOfSchools(data):
    schools = [ ]
    for item in data:
        schools.append(item[0])
    aset = set(schools)
    return sorted(aset)
data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function problemsAttempted(data)

• Returns list of problems attempted
• **Would return list:**
  • ['A', 'C', 'B', 'E', 'D', 'G', 'F', 'H']
  • Note doesn’t say anything about the order but implies one of each.
Write function `problemsAttempted(data)`

- Returns list of problems attempted
- **Would return list:**
  - `['A', 'C', 'B', 'E', 'D', 'G', 'F', 'H']`
  - Note doesn’t say anything about the order but implies one of each.

- **Need to loop over the lists in data**
  - Collect the names of problems attempted
  - Get the unique ones
Write function `problemsAttempted(data)`

def problemsAttempted(data):
    problems = set([])
    for item in data:
        problems.add(item[1])
    return list(problems)
Write function `problemsAttempted(data)`

```python
def problemsAttempted(data):
    problems = set([])
    for item in data:
        problems.add(item[1])
    return list(problems)
```
Problem 4 Fall 2014 Old tests
data is list of lists of submissions

data = [
  ['UNC', 'A', '20', 'reject'],
  ['Duke', 'A', '26', 'correct'],
  ['UNC', 'A', '33', 'reject'],
  ['ECU', 'A', '34', 'correct'],
  ['Elon', 'A', '34', 'correct'],
  ['USC', 'G', '44', 'reject'],
  ['UNC', 'A', '45', 'correct'],
  ['NCSU', 'B', '60', 'reject'],
  ['USC', 'C', '72', 'reject'],
  ['Duke', 'E', '82', 'reject'],
  ['USC', 'C', '90', 'correct'],
  ['UNC', 'B', '98', 'reject'],
  ['NCSU', 'B', '103', 'correct'],
  ['NCSU', 'A', '115', 'correct'],
  ['USC', 'A', '116', 'correct'],
  ['ECU', 'F', '202', 'reject'],
  ['Duke', 'D', '200', 'correct'],
  ['Duke', 'E', '210', 'correct'],
  ['UNC', 'B', '212', 'reject'],
  ['USC', 'G', '220', 'reject'],
  ['NCSU', 'D', '222', 'correct'],
  ['Elon', 'H', '225', 'correct'],
  ['NCSU', 'H', '230', 'reject']
]
Write function problemsNotAttempted(problems, data)

• problems is a list of all possible problems

• Returns a list of the problems not attempted
Write function
problemsNotAttempted(problems, data)

• problems is a list of all possible problems
  • ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

• Returns a list of the problems not attempted

• We know how to get all the problems attempted!
  • Call problemsAttempted!
• Put both lists in sets and set operation!
Write function

problemsNotAttempted(problems, data)

```python
def problemsNotAttempted(problems, data):
    attempted = problemsAttempted(data)
    setProbs = set(problems)
    setAttempted = set(attempted)
    setNotAttempted = setProbs - setAttempted
    return list(setNotAttempted)
```
Write function
problemsNotAttempted(problems, data)

def problemsNotAttempted(problems, data):
    attempted = problemsAttempted(data)
    setProbs = set(problems)
    setAttempted = set(attempted)
    setNotAttempted = setProbs - setAttempted
    return list(setNotAttempted)
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function
dictProblemstoSchoolsSolved(data)

• Returns a dictionary of letters for problems mapped to list of schools that solved that problem
  • ‘B’ mapped to ['NCSU']
  • ‘A’ mapped to ['Duke', 'ECU', 'Elon', 'UNC', 'NCSU', 'USC']
  • ‘D’ mapped to ['Duke', 'NCSU']
  • Etc
Write function
dictProblemstoSchoolsSolved(data)

- Returns a dictionary of letters for problems mapped to list of schools that solved that problem
  - ‘B’ mapped to ['NCSU']
  - ‘A’ mapped to
    - ['Duke', 'ECU', 'Elon', 'UNC', 'NCSU', 'USC']
  - ‘D’ mapped to ['Duke', 'NCSU']
  - Etc
- Each letter - create a list and append schools to it
Write function
dictProblemsToSchoolsSolved(data)

def dictProblemsToSchoolsSolved(data):
    d = {}

Write function
dictProblemsToSchoolsSolved(data)

def dictProblemsToSchoolsSolved(data):
    d = {}
    for item in data:
        if item[3] == 'correct':
            if item[1] in d:  # already in
                d[item[1]].append(item[0])
            else:  # not in yet,
                add d[item[1]] = [item[0]]
    return d
Problem 4 Fall 2014 Old tests
data is list of lists of submissions

```python
data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]```
Write function

dictSchoolsToNumSubmissions(data)

• Returns a dictionary of schools mapped to the number of submissions they had (rejected or correct)
  • ‘Duke’ mapped to 4
  • ‘UNC’ mapped to 5
  • Etc
Write function
dictSchoolsToNumSubmissions(data)

• Returns a dictionary of schools mapped to the number of submissions they had (rejected or correct)
  • ‘Duke’ mapped to 4
  • ‘UNC’ mapped to 5
  • Etc

• Counting dictionary!
Write function

dictSchoolsToNumSubmissions(data)

def dictSchoolsToNumSubmissions(data):
d = {}

Write function
dictSchoolsToNumSubmissions(data)

def dictSchoolsToNumSubmissions(data):
    d = {}
    for item in data:
        if item[0] in d:
            d[item[0]] += 1
        else:
            d[item[0]] = 1
    return d
WOTO-3 Solving problems
Problem 4 Fall 2014 Old tests

data is list of lists of submissions

data = [
    ['UNC', 'A', '20', 'reject'],
    ['Duke', 'A', '26', 'correct'],
    ['UNC', 'A', '33', 'reject'],
    ['ECU', 'A', '34', 'correct'],
    ['Elon', 'A', '34', 'correct'],
    ['USC', 'G', '44', 'reject'],
    ['UNC', 'A', '45', 'correct'],
    ['NCSU', 'B', '60', 'reject'],
    ['USC', 'C', '72', 'reject'],
    ['Duke', 'E', '82', 'reject'],
    ['USC', 'C', '90', 'correct'],
    ['UNC', 'B', '98', 'reject'],
    ['NCSU', 'B', '103', 'correct'],
    ['NCSU', 'A', '115', 'correct'],
    ['USC', 'A', '116', 'correct'],
    ['ECU', 'F', '202', 'reject'],
    ['Duke', 'D', '200', 'correct'],
    ['Duke', 'E', '210', 'correct'],
    ['UNC', 'B', '212', 'reject'],
    ['USC', 'G', '220', 'reject'],
    ['NCSU', 'D', '222', 'correct'],
    ['Elon', 'H', '225', 'correct'],
    ['NCSU', 'H', '230', 'reject']
]
Write function easiestProblem(data)

• Returns a tuple of two items
  • The name of the problem that was solved by the most schools
  • A sorted list of the schools that solved that problem
• If a tie, then pick any one
• Returns:
Write function easiestProblem(data)

- Need to calculate the problem that was solved the most
- Need to find that problem’s list of schools in the dictionary we already built
  - Will need to call that function
- Can do both as you walk through the dictionary!
Write function easiestProblem(data)

def easiestProblem(data):
    d = dictProblemsToSchoolsSolved(data)
    maxProb = ('', [])
    for (key, value) in d.items():
        if len(value) > len(maxProb[1]):
            maxProb = (key, value)
    return maxProb
Write function easiestProblem(data)

def easiestProblem(data):
    d = dictProblemsToSchoolsSolved(data)
    maxProb = ("",[])
    for (key,value) in d.items():
        if len(value) > len(maxProb[1]):
            maxProb = (key, value)
    return maxProb