CompSci 101
Introduction to Computer Science

November 30, 2017

Prof. Rodger
Announcements

• No more RQ!
• Assign 8 due Dec 5, Assign 9 due Dec 8-11
• APT 8 due Dec 7
• Be a UTA – sign up – or Peer Tutor

• Today:
  – Review Recursion
  – Regular Expressions
  – Assignment 8 Recommender
## Exam 2 Scores

<table>
<thead>
<tr>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
</tr>
<tr>
<td>91</td>
</tr>
<tr>
<td>90</td>
</tr>
<tr>
<td>89</td>
</tr>
<tr>
<td>88</td>
</tr>
<tr>
<td>87</td>
</tr>
<tr>
<td>86</td>
</tr>
<tr>
<td>85</td>
</tr>
<tr>
<td>84</td>
</tr>
<tr>
<td>83</td>
</tr>
<tr>
<td>82</td>
</tr>
<tr>
<td>81</td>
</tr>
<tr>
<td>80</td>
</tr>
<tr>
<td>79</td>
</tr>
<tr>
<td>78</td>
</tr>
<tr>
<td>77</td>
</tr>
<tr>
<td>76</td>
</tr>
<tr>
<td>75</td>
</tr>
<tr>
<td>74</td>
</tr>
<tr>
<td>73</td>
</tr>
<tr>
<td>72</td>
</tr>
<tr>
<td>71</td>
</tr>
<tr>
<td>70</td>
</tr>
<tr>
<td>69</td>
</tr>
</tbody>
</table>

**compsci 101 fall 2017**
Assignment 8
From User Rating to Recommendations

What should I choose to see?
  - What does this depend on?

Who is most like me?
  - How do we figure this out

<table>
<thead>
<tr>
<th></th>
<th>Spectre</th>
<th>Martian</th>
<th>Southpaw</th>
<th>Everest</th>
<th>PitchPerfect 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-3</td>
<td>5</td>
<td>-2</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>-2</td>
<td>1</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

compsci 101 fall 2017
ReadAllFood modules: Food Format

• All Reader modules return a tuple of strings: itemlist and dictratings dictionary

Sarah Lee
(DivinityCafe) (3)
(IlForno) (3)
(TheSkillet) (-3)
(LoopPizzaGrill) (3)
(FarmStead) (3)
(Tandoor) (5)
(PandaExpress) (-3)

Melanie
(McDonalds) (1)
(Tandoor) (3)
(DivinityCafe) (5)
(TheCommons) (3)
(TheSkillet) (1)
(IlForno) (3)
(PandaExpress) (3)
J J
(TheSkillet) (1)

• Translated to list and dictionary:

['IlForno', 'TheCommons', 'FarmStead', 'DivinityCafe', 'PandaExpress', 'TheSkillet', 'Tandoor', 'LoopPizzaGrill', 'McDonalds']

{"Sung-Hoon": [-1, 1, -1, 0, 3, -3, -3, 5, 1], "Wei": [0, 3, 1, 1, 0, 0, 5, 3, -1], "Sly one": [1, 3, 0, 5, 0, 3, 3, 3, 0], "Nana Grace": [3, 3, 0, 5, 0, 0, 1, -5, -1], "Melanie": [3, 3, 0, 5, 3, 1, 3, 0, 1], "J J": [0, 0, 1, 0, 1, 1, 3, -1, 1], "Harry": [0, 5, 3, 5, -5, 1, 0, -1, -3], "Sarah Lee": [3, 0, 3, 3, -3, -3, 5, 3, 0]}
Follow 12-step process

• ReadFood first!
  – Read input and save it
  – Get list of restaurants – use that ordering! Set?
  – For each person
    • For each restaurant and its rating
      – Must find location of restaurant in itemlist
      – Then update appropriate counter
  – Print any structure you create to check it
Recursion Review

• Function calls a clone of itself
  – Smaller problem
  – Must be a way out of recursion
Mystery Recursion
bit.ly/101f17-1130-1

def Mystery(num):
    if num > 0:
        return 1 + Mystery(num/2)
    else:
        return 2 + num
Example

```python
def Mystery(num):
    if num > 0:
        return 1 + Mystery(num/2)
    else:
        return 2 + num
```

- Mystery(4) is $1 + Mystery(2) = 1 + 4 = 5$
- Mystery(2) is $1 + Mystery(1) = 1 + 3 = 4$
- Mystery(1) is $1 + Mystery(0) = 1 + 2 = 3$
- Mystery(0) is 2
Review: Recursion to find ALL files in a folder

- A folder can have sub folders and files
- A file cannot have sub files

```python
def visit(dirname):
    for inner in dirname:
        if isdir(inner):
            visit(inner)
        else:
            print name(inner), size(inner)
```

Is that a directory?
If not a directory, it will be a file
def Something(data):
    # data is a list of integers
    if len(data) == 0:
        return 0
    if data[0] % 2 == 0:  # it is even
        return data[0] + Something(data[1:]),
    else:
        return Something(data[1:])
Revisit the APT Bagels
Recursively

```python
filename: Bagels.py

def bagelCount(orders) :
    
    return number of bagels needed to fulfill
    the orders in integer list parameter orders

```

1. orders = [1,3,5,7]

   Returns: 16

   No order is for more than a dozen, return the total of all orders.

2. orders = [11,22,33,44,55]

   Returns: 175 since 11 + (22+1) + (33+2) + (44+3) + (55+4) = 175
def bagelCount(orders):
    if len(orders) > 0:
        return orders[0]/12 + orders[0] + bagelCount(orders[1:])
    else:
        return 0

A)

def bagelCount(orders):
    if len(orders) > 0:
        return orders[-1]/12 + orders[-1] + bagelCount(orders[:-1])
    else:
        return 0

B)

def bagelCount(orders):
    return orders[0] + orders[0]/12 + bagelCount(orders[1:])

C)

def bagelCount(orders):
    if len(orders) > 1:
        return orders[1] + orders[1]/12 + bagelCount(orders[2:])
    else:
        return bagelCount(orders[0])

D)
Recursion in Pictures

More: Recursion in Pictures

What is Computer Science?

• … "it is the study of automating algorithmic processes that scale."

• If you need to find one email address on a webpage, you don't need computer science
  – If you need to scrape every email address, that number in the 10's to 100's, you could use help
How do you solve a problem like …

• How many words end in "aria"?
  – Start with "aria"? Contain "aria"?
  – Why would you care about this?

• Can you find ola@cs.duke.edu, susan.rodger@duke.edu, and andrew.douglas.hilton@gmail.com when searching through a webpage source?
  – What is the format of a "real" email address?
Examples of regex's at work

• What do aria$ and ^aria and aria share?
  – Answers to previous question

• What about the regex .+@.+  
  – Turns out that . has special meaning in regex,  
    so does +, so do many characters

• We'll use a module RegexDemo.py to check  
  – Uses the re Python library  
  – Details won't be tested, regex knowledge will
Regex expressions

• Regex parts combined in powerful ways
  – Each part of a regex "matches" text, can extract matches using programs and regex library
  – ^ is start of word/line, $ is end

• Expressions that match single characters:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, a, 9 or …</td>
<td>Any character matches itself</td>
</tr>
<tr>
<td>.</td>
<td>Matches any character</td>
</tr>
<tr>
<td>\w</td>
<td>Matches alphanumeric and _</td>
</tr>
<tr>
<td>\d</td>
<td>Matches digit</td>
</tr>
<tr>
<td>\s</td>
<td>Matches whitespace</td>
</tr>
</tbody>
</table>
Regex expressions

• Repeat and combine regex parts
  – * means 0 or more occurrences/repeats
  – + means 1 or more occurrences/repeats
  – ? Means (after * or +) to be non-greedy

• Expressions match more than one character

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a-zA]</td>
<td>Brackets create character class</td>
</tr>
<tr>
<td>(regex)</td>
<td>Tag or group a regex</td>
</tr>
<tr>
<td>\1 or \2</td>
<td>Matches previously grouped regex</td>
</tr>
<tr>
<td>{1} or {n}</td>
<td>Repeat regex 1 or n times</td>
</tr>
</tbody>
</table>
Regex examples tried and explained

• Five letter words ending in p? Starts 'd'?  
  – `^\w\w\w\wp$` but not `....p$`

• Seven letter words, or seven ending with 'z'
  – Difference between `^\w\{7\}$` and `^\w\{7\}`

• Words that start with a consonant:
  – `^[^aeiou]`  \textit{double meaning of} `^`
Regex examples tried and explained

• Five letter words ending in p? Starts 'd'?
  – ^\w\w\w\wp$  but not . . . . p$

• Seven letter words, or seven ending with 'z'
  – Difference between ^\w{7}$ and ^\w{7}$

• Start and end with the same two letters like sense and metronome, decipher this:
  – ^(\w\w).*\1$

• Start and end with three letters reversed, like despised and foolproof?
# Summary of Regular Expressions

<table>
<thead>
<tr>
<th><strong>regex</strong></th>
<th><strong>purpose</strong></th>
<th><strong>regex</strong></th>
<th><strong>purpose</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>any character</td>
<td>*</td>
<td>zero or more of previous regex</td>
</tr>
<tr>
<td>\w</td>
<td>any alphanumeric character (and _)</td>
<td>+</td>
<td>one or more of previous regex</td>
</tr>
<tr>
<td>\s</td>
<td>any whitespace character</td>
<td>*? or +?</td>
<td>non-greedy version of either * or +</td>
</tr>
<tr>
<td>\d</td>
<td>any digit character</td>
<td>()</td>
<td>tag/group a regular expression</td>
</tr>
<tr>
<td>[]</td>
<td>character class, e.g., [A-Z] or [aeiou]</td>
<td>\1, \2, ..</td>
<td>match numbered tagged/grouped regex</td>
</tr>
<tr>
<td>{n}</td>
<td>n occurrences of preceding regex</td>
<td>^</td>
<td>beginning of line/string</td>
</tr>
<tr>
<td>[^...]</td>
<td>not the characters in the class, e.g., [^aeiou]</td>
<td>$</td>
<td>end of line/string</td>
</tr>
</tbody>
</table>

compsci 101 fall 2017
Regex Questions
bit.ly/101f17-1130-4
SortByFreqs APT
Sort items by their frequency, break ties alphabetically

data = ["apple", "pear", "cherry", "apple", "pear", "apple", "banana"]
Returns: ["apple", "pear", "banana", "cherry"]