

CompSci 101

Introduction to Computer Science

	ABP	BlueEx	McDon	Loop	Panda	Nasher
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

November 30, 2017

Prof. Rodger

Announcements

- No more RQ!
- Assign 8 due Dec 5, Assign9 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

92 92 92 92 92 92 92 92 92 92 92 92 92 92 92

[illegible][illegible][illegible][illegible][illegible][illegible]

85 85 85 85 85 85 85

84 84 84 84 84 84 84 84 84 84 84 84

83 83 83 83 83 83 83 83 83

82 82 82 82 82 82 82 82

81 81 81 81 81 81 81 81 81

80 80 80 80 80 80

79 79

78 78 78 78

77 77 77 77 77

76 76 76 76

75 75 75 75

74 74

73 73 73

72

71 71 71

69

67 67 67

66

64

63

62

59 59

58

55

25

Assignment 8

From User Rating to Recommendations



Spectre	Martian	Southpaw	Everest	PitchPerfect 2
3	-3	5	-2	-3
2	2	3	2	3
4	4	-2	1	-1

| **What should I choose to see?**

➤ What does this depend on?

| **Who is most like me?**

➤ How do we figure this out

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)
```

not
all
shown
...

- 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

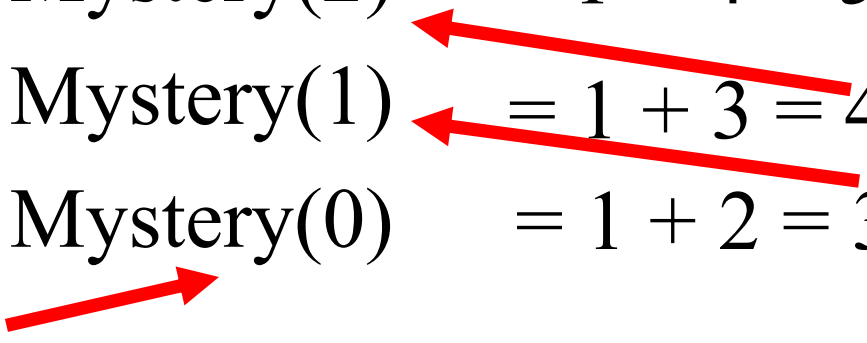
Mystery Recursion

bit.ly/101f17-1130-1

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




Example

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

- $\text{Mystery}(4)$ is $1 + \text{Mystery}(2) = 1 + 4 = 5$
 - $\text{Mystery}(2)$ is $1 + \text{Mystery}(1) = 1 + 3 = 4$
 - $\text{Mystery}(1)$ is $1 + \text{Mystery}(0) = 1 + 2 = 3$
 - $\text{Mystery}(0)$ 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

```
def visit(dirname):  
    for inner in dirname:  
        if isdir(inner):  Is that a directory?  
            visit(inner)  
        else:  If not a directory, it will be a file  
            print name(inner), size(inner)
```

Something Recursion

bitly/101f17-1130-2

```
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

```
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 $11 + (22+1) + (33+2) + (44+3) + (55+4) = 175$

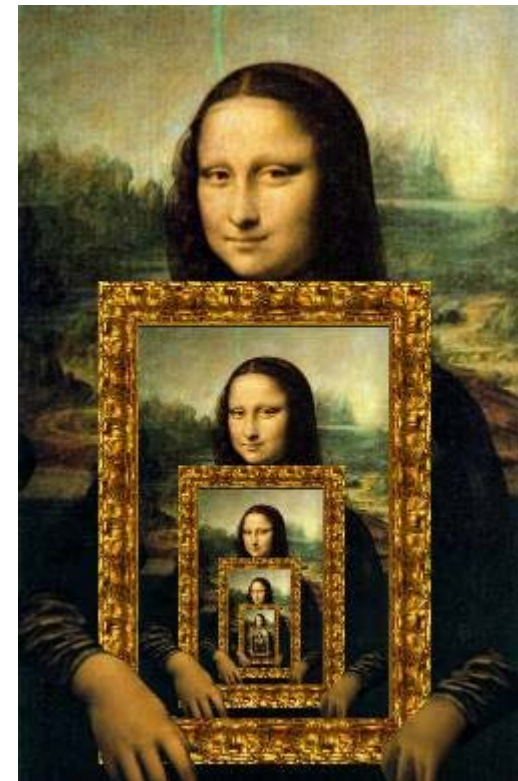
APT Bagels Recursively

bit.ly/101f17-1130-3

- A)
- ```
def bagelCount(orders):
 if len(orders) > 0:
 return orders[0]/12 + orders[0] + bagelCount(orders[1:])
 else:
 return 0
```
- B)
- ```
def bagelCount(orders):  
    if len(orders) > 0:  
        return orders[-1]/12 + orders[-1] + bagelCount(orders[:-1])  
    else:  
        return 0
```
- C)
- ```
def bagelCount(orders):
 return orders[0] + orders[0]/12 + bagelCount(orders[1:])
```
- D)
- ```
def bagelCount(orders):  
    if len(orders)>1:  
        return orders[1] + orders[1]/12 + bagelCount(orders[2:])  
    else:  
        return bagelCount(orders[0])
```

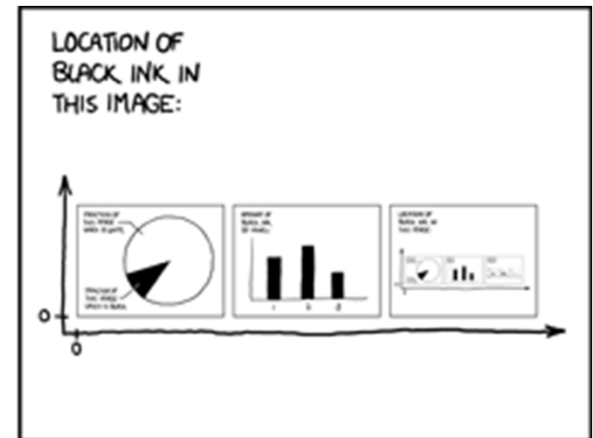
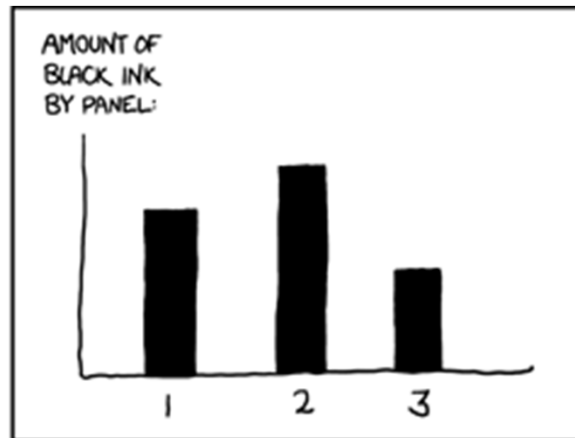
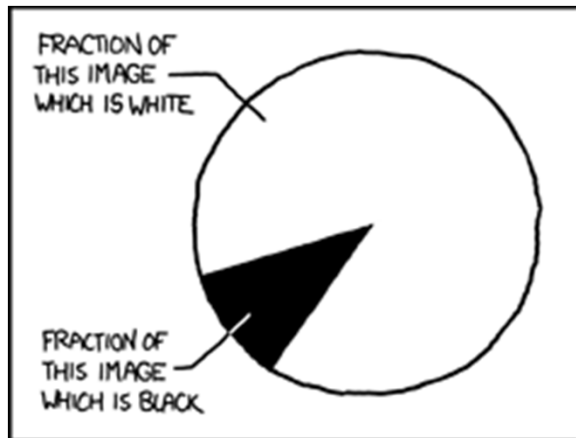
Recursion in Pictures

- <http://xkcd.com/543/>



More: Recursion in Pictures

- <http://xkcd.com/688/>



What is Computer Science?

- ... "it is the study of automating algorithmic processes that scale."
 - https://en.wikipedia.org/wiki/Computer_science
- 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:

A, a, 9 or ...	Any character matches itself
.	Matches any character
\w	Matches alphanumeric and _
\d	Matches digit
\s	Matches whitespace

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

[a-zA-B]	Brackets create character class
(regex)	Tag or group a regex
\1 or \2	Matches previously grouped regex
{1} or {n}	Repeat regex 1 or n times

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}z`
- Words that start with a consonant:
 - `^[^aeiou]` double meaning of `^`

Regex examples tried and explained

- Five letter words ending in p? Starts 'd'?
 - $\text{\textasciitilde}\backslash w\backslash w\backslash w\backslash wp\$$ but not $\text{\textasciitilde}\text{\textasciitilde}\text{\textasciitilde}\text{\textasciitilde}\text{\textasciitilde}p\$$
- Seven letter words, or seven ending with 'z'
 - Difference between $\text{\textasciitilde}\backslash w\{7\}\$$ and $\text{\textasciitilde}\backslash w\{7\}$
- Start and end with the same two letters like sense and metronome, decipher this:
 - $\text{\textasciitilde}(\backslash w\backslash w)\text{\textasciitilde}^*\backslash 1\$$
- Start and end with three letters reversed, like despised and foolproof?

Summary of Regular Expressions

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

Regex Questions

bit.ly/101f17-1130-4

Answer Questions

bit.ly/101f17-1130-5

SortByFreqs APT

Sort items by their frequency, break ties alphabetically

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