

## Lab Overview

- Review lab 8
- Prep for lab 9

Mar 20, 2012

Sprenkle - CSCI111

1

- Rainbow Dice

Mar 20, 2012

Sprenkle - CSCI111

2

## Lab 8 Feedback

- Define constants for HEADS and TAILS
- Extra step in game module's flipCoin function

```
HEADS=0
TAILS=1
def flipCoin():
    return random.randint(0,1)
```

Equivalent code  
(needs comments)

```
HEADS=0
TAILS=1
def flipCoin():
    flip = random.randint(0,1)
    if flip == HEADS:
        return HEADS
    else:
        return TAILS
```

Mar 20, 2012

Sprenkle - CSCI111

3

## Testing game Module

- How do you know if flipCoin(), rollDie(sides), rollMultipleDice(numSides, numDice) are correct?
- What are good test cases?

Mar 20, 2012

Sprenkle - CSCI111

4

## Common Issues

- Test multiple of 6 the first time
- Test Caesar cipher with a valid file too
  - Found mistakes there
  - Prime example of making a change to script and causing errors
- Comments on game's functions
  - Describe *interface* → tell others how to use
    - What are parameters? Restrictions?
    - What is returned?

Mar 20, 2012

Sprenkle - CSCI111

5

## Difference btw File *Name* and *Object*

- File name is a string
- File object is a file
- Need the file name to create the file object

- Need to remember data types because not explicit in Python
- Use good variable names to help

Mar 20, 2012

Sprenkle - CSCI111

6

## Review: Dictionaries

- How do you create a new dictionary?
- How do you find out if there is a mapping for a key in the dictionary?
- How do you access the value for a key?
- How do you add a mapping?
- How can you iterate through a dictionary?

Mar 20, 2012

Sprenkle - CSCI111

7

## Review: Defining our own classes

- Each object has its own data/attributes/instance variables
  - Card objects have a rank and a suit
- What are defined methods like?
- Special method name for constructor?
- Special name for method that helps with printing?
- Keyword that must be first parameter of every defined method?

Mar 20, 2012

Sprenkle - CSCI111

8

## Card Class (Incomplete)

```
class Card:
    """ A class to represent a standard playing card.
        The ranks are ints: 2-10 for numbered cards, 11=Jack,
        12=Queen, 13=King, 14=Ace.
        The suits are strings: 'clubs', 'spades', 'hearts',
        'diamonds' """
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
        string suit."""
        self.rank = rank
        self.suit = suit
    def getRank(self):
        """Returns the card's rank."""
        return self.rank
    def getSuit(self):
        """Returns the card's suit."""
        return self.suit
```

Doc String

Methods

Identify the instance variables  
• How do we use them in other methods?

card.py

Mar 20, 2012

Sprenkle - CSCI111

9

## Review: Algorithm for Creating Classes

1. Identify need for a class
2. Identify state or attributes of a class/an object in that class
  - Write the constructor (`__init__`) and `__str__` methods
  - Test those methods
3. Identify methods the class should provide
  - How will a user call those methods (parameters, return values)?
    - Develop API
  - Implement methods

Mar 20, 2012

Sprenkle - CSCI111

10

## Lab 9: Dealing with Real Data

- **Problem:** Determine most common first and last names at W&L
  - 4 data files, containing student names
    - Last names, female first names, male first names, all first names
    - 1 name per line
  - Print results in special format for use in Gnuplot
  - What data structure to use?
- Create your own class to help with data
- Create output file used by another application

Mar 20, 2012

Sprenkle - CSCI111

11

## Overview

1. Implement partial solution using a dictionary to map the name to its count
  - handles basic set up of solution, including reading and processing file
2. Implement a class that packages the name (a key) and its count together
  - Data and functionality given
  - Test the class
3. Implement Step 1 with objects of class you created in Step 2
  - Complete solution
4. Graph data generated from Step 3
5. Make web page with graphs

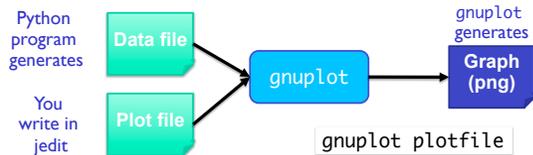
Mar 20, 2012

Sprenkle - CSCI111

12

## Gnuplot

- Portable command-line driven interactive data and function plotting utility for many platforms
- Like a *free* Excel (for the graphing part)

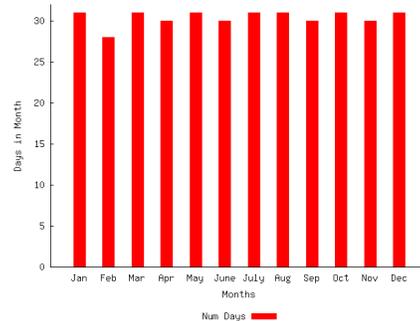


Mar 20, 2012

Sprengle - CSC1111

13

## Generates Graph (PNG)



Mar 20, 2012

Sprengle - CSC1111

14

## Plot File

```

set terminal png large
set output "bars.png" ← Modify to change name of generated output file
set data style boxes
set boxwidth 0.4
set xtics nomirror
set border 11

set xrange [0:13] ← Modify to change x-axis range
set yrange [0:32] ← Modify to change y-axis range

set xlabel "Months" ← Modify to change axes labels
set ylabel "Days in Month"

set xtics ("Jan" 1, "Feb" 2, "Mar" 3, "Apr" 4, "May" 5, "June" 6, \
"July" 7, "Aug" 8, "Sep" 9, "Oct" 10, "Nov" 11, "Dec" 12)
set key below ← Modify to change x-axis labels ("label" x-value)
plot 'bars.dat' using 1:2 fs solid title "Num Days" ← Modify to change input file
  
```

Mar 20, 2012

Sprengle - CSC1111

15

## Data File

```

# number of days in each month of 2010
1 31
2 28
3 31
4 30
5 31
6 30
7 31
8 31
9 30
10 31 ← Y-value
11 30
12 31
  
```

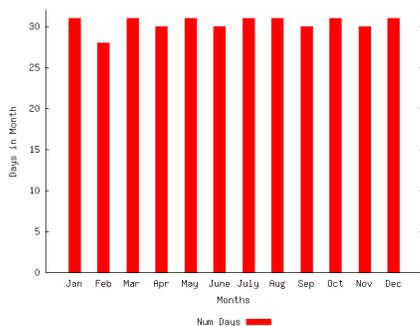
↑ X-coordinate

Mar 20, 2012

Sprengle - CSC1111

16

## Generates Graph (PNG)



Mar 20, 2012

Sprengle - CSC1111

17

## Overview

1. Implement partial solution using a dictionary to map the name to its count
  - handles basic set up of solution, including reading and processing file
2. Implement a class that packages the name (a key) and its count together
  - Data and functionality given
  - Test the class
3. Implement Step 1 with objects of class you created in Step 2
  - Complete solution
4. Graph data generated from Step 3
5. Make web page with graphs

Mar 20, 2012

Sprengle - CSC1111

18