Lab Overview

- Review labs 7, 8
- Prep for lab 9

Caesar Cipher w/Functions

```
def main():
    text = raw_input("Enter some text: ")
    key = input("Enter an integer key (between -25 and 25): ")
    # make sure it's a valid key
    if key < -KEY_BOUND or key > KEY_BOUND:
        print "Invalid key!"
        sys.exit(1)

    message = encoder(text,key)
    print "The encoded message is", message

# encoder takes in some text and integer key and returns
# encoded message
# PRE: Key must be between -25 and 25 inclusive
def encoder(text,key):
    message=""
    for ch in text:
        if ch == " ":
            encode= " "
            message+=encode
        else:
            message += translateLetter(ch,key)
    return messagekey
```

More efficient: constants *not* defined in function

Note: no "side effects" e.g., no printing

Encoding a File

- Similar to last problem
  - Just change *main*

```
def main():
    filename = raw_input("Enter a filename to encode: ")
    key = input("Enter an integer key (between -25 and 25): ")
    # make sure it's a valid key
    if key < -KEY_BOUND or key > KEY_BOUND:
        print "Invalid key!"
        sys.exit(1)
    fileToEncode = file(filename, "r")
    for line in fileToEncode:
        line = line.strip()
        message = encoder(line,key)
        print message

    fileToEncode.close()
```

Alternative:
```
line = line[:-1]
```

Simply use function you defined

Gymnast Scores (Partial Solution)

```
judgeFile = file(FILENAME, "r")

avgDifficulty = judgeFile.readline()
avgDifficulty = float(avgDifficulty)

min = 10
max = 0
total = 0

for x in xrange(6):  # get next 6 execution scores
    line = judgeFile.readline()
    score = float(line)
    if score < min:
        min = score
    if score > max:
        max = score
    total += score
judgeFile.close()
total -= max + min  # exclude high and low scores from total
…
```

Read in separately, Not in loop → inefficient

Keep track of "current" min/max

Comments: what code *means*

Using Lists

- Lab 8: lots of practice with lists
  - Differentiate between using the *positions* and using the *values*
  - Which way to iterate: by positions or by values

- How can we quickly create a list with a constant "step" between values?
- How do we find out what value is in a list at position *p*?

Printing Board

```
    border = "*"*30
    print
    print border
    print " The Board: "

    for count in xrange(len(amounts)/2):
        if amounts[count] != CHOSEN:
            print "$%10.2f" % amounts[count],
        else:
            print "%11s" % "----",

        second_col = count + len(amounts)/2
        if amounts[second_col] != CHOSEN:
            print "   $%10.2f" % amounts[second_col]
        else:
            print "   %11s" % "----"

    print border
```

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

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?

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?

Card Class (Incomplete)

Doc String

```
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
```

Methods

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

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
3. Identify methods the class should provide
   - How will a user call those methods (parameters, return values)?
     - Develop API
   - Implement methods

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
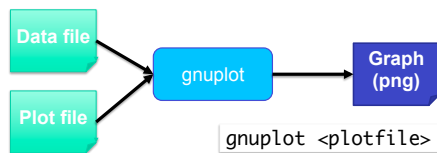- Create output file used by another application

Gnuplot

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

Data file → gnuplot → Graph (png)

Plot file →

`gnuplot <plotfile>`

Plot File

```
set terminal png large

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

set xrange [0:13]    ←  Update to change x-axis range

set yrange [0:32]    ←  Update to change y-axis range

set xlabel "Months"           Update 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         ←  Update to change x-axis labels
                               ("label" x-value)
plot 'bars.dat' using 1:2 fs solid title "Num Days"
          ↖ Update to change input file
```

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
```
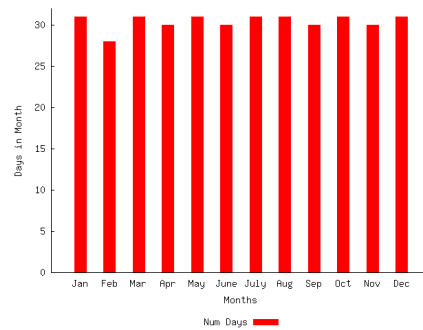
X-coordinate

Generates Graph (PNG)

Lab Outline

- Broken into smaller pieces
- Process data file
  - Determine how many people at W&L have various names
- Generate Gnuplot Data files in Python
  - Has specific format
    ```
    # <name>
    <x coord> <count for name>
    ```
- Create Gnuplot Plot files
  - In jEdit, create a plot file for each data file
- Generate graphs using Gnuplot