

Lab 7 Feedback

- Missing function comments
 - Others need to know how to use the function
- Extra step in game module's flipCoin function

```
HEADS=0
TAILS=1

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

Equivalent code

```
HEADS=0
TAILS=1

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

Mar 11, 2008

Sprengle - CS111

1

Lab 7 Feedback

- Reading case values from a file
 - Need to be numbers, not strings
 - Convert to floats before appending to list

Mar 11, 2008

Sprengle - CS111

2

Palindrome Problem

```
# returns True iff phrase is a palindrome (ignoring spaces
# and capitalization)
def isPalindrome(phrase):
    phrase = phrase.replace(" ", "")
    phrase = phrase.lower()
    for pos in xrange(len(phrase)/2):
        if phrase[pos] != phrase[-(pos+1)]:
            return False
    return True
```

```
def main():
    user_phrase = raw_input("Enter your phrase: ")
    if isPalindrome( user_phrase ):
        print user_phrase, "is a palindrome"
    else:
        print user_phrase, "is not a palindrome"
```

Mar 11, 2008

Sprengle - CS111

3

Where is Documentation Coming From?

- Comes from the code itself in "**doc strings**"
 - i.e., "documentation strings"
- Doc strings are simply strings *after* the function header
 - Typically use triple-quoted strings because documentation goes across several lines

```
def printVerse(animal, sound):
    """ prints a verse of Old MacDonald, filling in
    the strings for animal and sound """
```

Mar 11, 2008

Sprengle - CS111

4

Comments

- Comment functions, programs/modules at high level
- Comment inside functions at low-level (implementation details)
- Example (derived from students' submissions):

```
# This function calculates the speeding ticket fine, if speeding.
# PRE: Input params are two positive numbers representing the
# speed limit and the clocked speed
# POST: Return the computed fine
def calculateFine(speed_limit, clock_speed):
    ...
```

Mar 11, 2008

Sprengle - CS111

5

Why is This a Good Comment?

```
# This function calculates the speeding ticket fine, if speeding.
# PRE: Input params are two positive numbers representing the
# speed limit and the clocked speed
# POST: Return the computed fine
def calculateFine(speed_limit, clock_speed):
    ...
```

- Describes the input parameters' types and implies the type of the output
- Says that if you give it negative numbers, you could get a goofy answer back
- Does not include implementation details
- **Tells caller how to use the function**

Mar 11, 2008

Sprengle - CS111

6

Another Example

```
# Encodes a single character.
# PRE: Input parameters are a single, lowercase character
# string (char) and an integer key
# POST: returns the encoded character
def translateLetter(char, key):
```

Mar 11, 2008

Sprenkle - CS111

7

Commenting Exercise

- Write a comment for this function:

```
def encode(toEncode, key):
```

Mar 11, 2008

Sprenkle - CS111

8

Commenting Exercise

- Write a comment for this function:

```
# Encodes a lowercase string using a key, preserving spaces in
# the string
# PRE: a lowercase string to be encoded (toEncode) and the
# integer key
# POST: returns the encoded string
def encode(toEncode, key):
```

Mar 11, 2008

Sprenkle - CS111

9

Commenting Notes

- Well-named parameters make documentation easier
- I'm not strict on the pre/post format.
- Just need to be clear on
 - > what the function does (at high level)
 - > types of parameters
 - > type of the output
- ➔ The caller knows what to pass to the function and if they should assign the output to a variable

Mar 11, 2008

Sprenkle - CS111

10

Descend Sort a List w/ 3 elements

```
def descendSort3Nums(list3):
    if list3[1] > list3[0]:
        # swap 'em
        tmp = list3[0]
        list3[0] = list3[1]
        list3[1] = tmp

    if list3[2] > list3[1]:
        tmp = list3[1]
        list3[1] = list3[2]
        list3[2] = tmp

    if list3[1] > list3[0]:
        tmp = list3[0]
        list3[0] = list3[1]
        list3[1] = tmp
```

```
def main():
    list = [1,2,3]
    descendSort3Nums(list)
    print list
```

Note: Function does not return anything. Simply modifies the list3 parameter.

Mar 11, 2008

Sprenkle - CS111

11

Passing Lists and Dictionaries as Parameters

- Lists and dictionaries are **mutable**
- Can be modified inside of functions
 - > Modifications seen outside of functions
- Examples from yesterday:
 - > addStudent(majorToCount, major)
 - > descendSort3Nums(list3)
- majorToCount and list3 were modified in function
 - > Modifying major, however, would not change major outside of the function because major is a string

Mar 11, 2008

Sprenkle - CS111

12

Lab 8 Overview

- List and Dictionary problems
- Common names at W&L
- *Deal or No Deal*

Mar 11, 2008

Sprenkle - CS111

13

Dealing with Real Data

- You will determine the most common first and last names at W&L
 - 3 data files, containing student names
 - Last names, female first names, male first names
 - 1 name per line
 - Print in special format, use Unix tools to find most common names
- What data structure to use?

Mar 11, 2008

Sprenkle - CS111

14

Implementing *Deal or No Deal*

- Given: partial solution in code
 - Complete `main()` function, some additional functions
- Your job:
 - Read, understand given code
 - Fill in the functions for a complete solution

Mar 11, 2008

Sprenkle - CS111

15

Modeling *Deal or No Deal*

- Cases, numbered 0 to 25
 - Have dollar amounts in them
- How can we represent when a case has been opened?

1000000	1000	5	...	750000
0	1	2	...	25

- Board
 - Which dollar amounts have been chosen, which are still in play

.01	1	5	...	1000000
0	1	2	...	25

Mar 11, 2008

Sprenkle - CS111

16

Modeling *Deal or No Deal*

- Cases, numbered 0 to 25
 - Have dollar amounts in them
- CHOSEN = -1**
means case opened:
Don't display on board,
Don't allow user to select again

1000000	1000	5	...	CHOSEN
0	1	2	...	25

- Board
 - Which dollar amounts have been chosen, which are still in play

.01	CHOSEN	5	...	1000000
0	1	2	...	25

Mar 11, 2008

Sprenkle - CS111

17

Functionality

- Read in values contained in cases from a file
 - Done in Lab 7
- Have user select from remaining cases
 - Make sure choice is valid
- Display remaining cases
 - Print four to a row
- Display remaining amounts on board
 - Left column is smaller amounts

Mar 11, 2008

Sprenkle - CS111

18

How to print remaining cases?

- Cases, numbered 0 to 25

➤ Have dollar amounts in them

1000000	1000	5		CHOSEN
0	1	2	...	25

- Board

➤ Which dollar amounts have been chosen, which are still in play

.01	CHOSEN	1000		-1
0	1	2	...	25