

## Lab 8 Feedback

- Mostly minor issues
  - Not closing file objects
  - Off by one errors
  - Not testing invalid cases
  - Rereading case values, unnecessarily
  - Missing name in comments
  - Column formatting issues
  - Slightly inefficient solutions
- Prob 3: Not writing to file using `file` object's `write` method

Mar 25, 2008

Sprenkle - CS111

1

## Deal or No Deal: `printCasesLeft`

```
def printCasesLeft(cases):
    print "Cases Left to Choose from:"
    # number of columns printed
    colCount = 0
    for casePos in xrange(len(cases)):
        # only print if the case hasn't been chosen yet
        if cases[casePos] != CHOSEN:
            print "%3d" % casePos,
            colCount += 1
            # print a newline after every fourth printed case
            if colCount % 4 == 0:
                print
    # add an extra line at the end
    print
```

Mar 25, 2008

Sprenkle - CS111

2

## Deal or No Deal: `printCasesLeft`

Not quite as good alternative

```
def printCasesLeft(cases):
    print "Cases Left to Choose from:"
    linecount = 0
    casePos = 0
    for case in cases:
        # only print if the case hasn't been chosen yet
        if case != CHOSEN:
            print "%3d" % casePos,
            linecount += 1
            # print a newline after every fourth printed case
            if linecount % 4 == 0:
                print
            casePos += 1
    # add an extra line at the end
    print
```

3

## Deal or No Deal: `printBoard`

```
def printBoard(amounts):
    border = ""*30
    print border
    print "The Board: "

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

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

Mar 25, 2008

Sprenkle - CS111

4

## Deal or No Deal: `isValidChoice`

```
def isValidChoice(choice, cases):
    # choice is out of range
    if choice < 0 or choice > len(cases)-1:
        return False

    # case is already chosen
    if cases[choice] == CHOSEN:
        return False

    # handled invalid cases, so this one is valid
    return True
```

Mar 25, 2008

Sprenkle - CS111

5

## Deal or No Deal: `getUserChoice`

```
def getUserChoice(choice, prompt):
    user_choice = input(prompt)

    # keep requesting choice if not valid
    while not isValidChoice(cases, user_choice):
        print "Not a valid choice."
        user_choice = input(prompt)

    # know that user_choice is valid if out of loop
    return user_choice
```

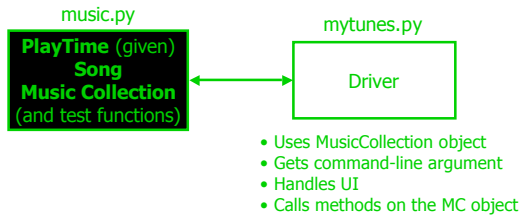
Mar 25, 2008

Sprenkle - CS111

6

## Lab 10 Design

- 2 files: music.py and mytunes.py



Mar 25, 2008

Sprenkle - CS111

7

## Problem: Album Music Files

- Given an album file that has the format
  - > <Artist name>
  - > <Album name>
  - > <number of songs>
  - > <Song name 1>
  - > <Song length 1>
  - > ...
  - > <Song name n>
  - > <Song length n>
- Write algorithm to create Song objects to represent each song

Length has the format  
min:seconds

Mar 25, 2008

Sprenkle - CS111

8

## Example Album File

- Given an album file that has the format

- > <Artist name>
- > <Album name>
- > <number of songs>
- > <Song name 1>
- > <Song length 1>
- > ...
- > <Song name n>
- > <Song length n>

Length has the format  
min:seconds

```

The Killers
Hot Fuss
11
All These Things That I've Done
05:01
Andy, You're a Star
03:14
Believe Me Natalie
05:06
Change Your Mind
03:10
Everything Will Be Alright
05:45
Jenny Was a Friend of Mine
04:06
... [5 more songs...]
  
```

Mar 25, 2008

Sprenkle - CS111

9

## Problem: Library Music Files

- Given a library file that has the format
  - > <number of songs>
  - > <Song artist 1>
  - > <Song album 1>
  - > <Song name 1>
  - > <Song length 1>
  - > ...
  - > <Song artist n>
  - > <Song album n>
  - > <Song name n>
  - > <Song length n>
- Create a MusicLibrary object

Mar 25, 2008

Sprenkle - CS111

10

## Example Library File

- Given a library file that has the format

- > <number of songs>
- > <Song artist 1>
- > <Song album 1>
- > <Song name 1>
- > <Song length 1>
- > ...
- > <Song artist n>
- > <Song album n>
- > <Song name n>
- > <Song length n>

```

2
Better Than Ezra
Before the Robots
Burned
03:41
Better Than Ezra
Before the Robots
Daylight
03:54
  
```

Mar 25, 2008

Sprenkle - CS111

11

## UI Specification

- Checks if user entered a command-line argument
  - > Default library: libraries/mytunes.library
- Read library from file
- Repeatedly gets selected options from the user, until quits
- Repeatedly prompts for new selection if invalid option
- Executes the appropriate code for the selection
- Stops when user quits
- Stores the library into the file

Demonstrate program  
Write pseudocode

Mar 25, 2008

Sprenkle - CS111

12

## UI Pseudocode

```
Use default library if only one command-line argument
Read library from file
while True:
    display menu options
    prompt for selection
    while invalid option
        print error message
        prompt for selection
    break if selected quit
    otherwise, do selected option
Store library to designated file
```

Mar 25, 2008

Sprenkle - CS111

13

## Implementation Plan

- Review PlayTime class
  - How will you create a PlayTime object?
  - How will you use it?
- Implement Song class
  - Test (write test functions, e.g., testSong())
- Implement MusicCollection class
  - Example runs in lab write up
  - Note: in general, methods for classes will not prompt for input (Use input parameters)
  - Test
- Implement driver program

Mar 25, 2008

Sprenkle - CS111

14

## Plan for Implementing a Class

- Write the constructor and string representation/print methods first
- Write function to test them
- While more methods to implement ...
  - Write method
  - Test
- See [counter.py](#) and [card.py](#) for example test functions
- See [PlayTime](#) class for another example class implementation (and testing)

Mar 25, 2008

Sprenkle - CS111

15

## Programming Style

- Write functions and use constants as appropriate
- Comments for methods

Mar 25, 2008

Sprenkle - CS111

16

## Demonstrating MyTunes

- Demonstrate “typical” usage of your program
  - User should try out each available option
- You won’t demo command-line arguments
  - I will test that when I execute your program

Mar 25, 2008

Sprenkle - CS111

17