Daniel Zingaro
CCSCE 2009 Nifty Ideas Panel
Sample Handout: Lazy Exams

Some of your instructors are lazy. (Present company naturally excluded.) To create your tests and exams, they might use software that randomly selects questions from a prefabricated question bank. These question banks come as part of "instructor resources" for the chosen course textbook. They often include hundreds of questions of various types, such as multiple choice, true/false, short answer, fill-in-the-blank, and so on. In this assignment, you'll write a program that takes a question bank of multiple choice and true/false questions, and randomly selects a user-specified number of each type of question. It will write these questions to a text file, so your lazy instructor can simply print that file and give you your test. It also outputs an answer key (to a different file), to help your lazy instructor mark the test when the time comes.

For an example of a question bank, please consult the "Sample Question Bank" appendix of this assignment.

Question banks consist of zero or more multiple choice questions, and zero or more true/false questions. They may be interleaved in any order; notice that the sample question bank contains multiple choice (MC) and true/false (TF) questions interspersed.

A multiple choice question consists of the following:

- A line containing only `MC`
- One or more lines containing possible answers to the question. The first of these lines is answer A, the second is answer B, and so on; however, these letters are not included in the answer lines. There will be at most ten possible answers for any given multiple choice question. The correct answer is preceded by an asterisk; exactly one such answer has such an asterisk

A true/false question consists of the following:

- A line containing only `TF`
- A line beginning with `T` or `F`, immediately followed by the text of the true/false question itself. The first character indicates whether the correct answer is true or false

Your program must read the question file only once. While reading the file, its data should be stored in Python objects so that you can later use it when randomly choosing questions.

Your program should operate in the following steps.

- Prompt the user for a filename containing question data. You do not have to do error-checking here; you may assume that a valid filename will be given.
- Open the file, and read all of the multiple choice and true/false questions. You may assume that the questions file is well-formed; i.e. that it follows the file format given above.
- Tell the user how many multiple choice questions were found in the questions file, and ask the user how many of them to include in the generated test. If a number less than 0 or greater than the maximum is given, ask the user again until a valid response is received. Your program should be robust here: if the user enters data that is not numeric, your program should not crash!
- Tell the user how many true/false questions were found in the questions file, and ask the user how many of them to include in the generated test. If a number less than 0 or greater than the maximum is given, ask the user again until a valid response is received. If the user enters data that is not numeric, your program should not crash.

- Prompt the user for the name of a file to which the randomly chosen questions will be written (the test file). Your program does not have to check that the filename is valid or that the file does not exist; if the file exists, its contents will be overwritten.
- Prompt the user for the name of a file to which the answer key will be written (the answer file). Your program does not have to check that the filename is valid or that the file does not exist; if the file exists, its contents will be overwritten.
- Randomly generate the user-specified number of multiple choice and true/false questions, saving them in the test file. All multiple choice questions should come first, followed by all true/false questions. For a multiple choice question, answers should be preceded with the letters A, B, C, and so on. Each question should be followed by a line of forty dashes. Each type of question — multiple choice and true/false — should be preceded by a header giving the test-taker instructions for the questions that follow. These instructions should only be included if the generated test has at least one question of the corresponding type. (It makes no sense to give instructions about multiple choice questions if the test doesn't have any!)

Here is a complete example of program execution.

```
File of questions? questions.txt
There are 5 multiple choice questions. Use how many? 2
There are 4 true/false questions. Use how many? 3
Name of test file to write? exam.txt
Name of answer key file to write? exam_answers.txt
```

Assume that the user has chosen to use the sample question bank from the appendix. Please see the "Sample Exam and Answer Key" for a sample output from this program run. Of course, the program randomly selected the questions; if you use exactly the same parameters, your resultant exam will differ each time you run the program.

Save your solution as file `lazy_exam.py`. You should have an `if __name__ == '__main__':` block that acts as the entry point to your program; your code should start running when we click Run in Wing.

This is the first time in the course that I haven't decomposed a larger program into smaller functions. It's your turn to do this! You should use functions to divide your program into meaningful pieces, and to prevent code duplication. Part of your grade will reflect how well you design your program.

```
MC
What is Python's string type called?
string
*str
strng
TF
FPython strings are mutable.
TF
FPython lists are required to be homogeneous collections of elements.
TF
TPython for-loops work on strings, files, and lists.
MC
To receive keyboard input in Python, we can use:
get_input
scanf
*raw_input
do_input
string_input
MC
Python is cooler than:
C
C++
Java
*All of the above
MC
What does Python's range function do?
*Generates lists of integers
Generates tuples of integers
TF
TPython slice syntax can be used with lists.
MC
Which of the following is an integer?
2.5
*3
3.0
```

Appendix: Sample Exam and Answer Key

Exam

```
Multiple Choice
Circle the most appropriate answer for each question below.

1. To receive keyboard input in Python, we can use:
A. get_input
B. scanf
C. raw_input
D. do_input
E. string_input
----------------------------------------
2. Which of the following is an integer?
A. 2.5
B. 3
C. 3.0
----------------------------------------
True/False
Indicate true or false for each question below.

3. Python slice syntax can be used with lists.
----------------------------------------
4. Python strings are mutable.
----------------------------------------
5. Python for-loops work on strings, files, and lists.
----------------------------------------
```

Answer Key

```
1. C
2. B
3. T
4. F
5. T
```