CMSC 201 Fall 2016

Project 1 – Connect Four

Assignment: Project 1 – Connect Four
Due Date: Wednesday, November 16nd, 2016 by 8:59:59 PM
Value: 80 points

Collaboration: For Project 1, collaboration is not allowed – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Your collaboration statement should state that
`Collaboration was not allowed on this assignment.`

Make sure that you have a complete file header comment at the top of your file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
# Collaboration:
#    COLLABORATION STATEMENT GOES HERE
```

Project 1 is the first assignment where we won't be telling you exactly what to do! You will get the chance to make your own decisions about how you want your program to handle things, what its functions should be called, and how you want to go about designing it.

Project 1 will also be ***substantially longer*** than any of the single homework assignments you've completed so far, so make sure to take the time to plan ahead, and don't do any "cowboy" coding!

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Instructions

For this assignment, you'll need to follow the class coding standards, a set of rules designed to make your code clear and readable. The class coding standards are on the website, linked at the top of the "Assignments" page. You can also access them directly via this URL (http://goo.gl/yEoGfC).

You should be commenting your code, and using constants in your code (not magic numbers or strings).
Any numbers other than 0 or 1 are magic numbers!

Adhere to the coding standards by including function header comments for each of the functions (other than main). Follow the instructions and example provided in the coding standards document when creating your function header comments. Failing to include function header comments will lose you points.

Re-read the coding standards!

You will **lose major points** if you do not following the 201 coding standards.

NOTE: Your filename for this project must be **`proj1.py`**

NOTE: You must use **`main()`** in your file.

Details

For this project, you are going to be building the classic game Connect Four. If you are not familiar with the game, the Wikipedia page has more details (https://en.wikipedia.org/wiki/Connect_Four). In our game, Player 1 is an "x" and Player 2 is an "o". If any player gets four of their pieces in a row (horizontally, vertically, or diagonally), they win. When the board fills up completely and neither player has won, the game ends in a draw.

The game starts by asking the user if they would like to load a game from a file; this is the only time the user has the option to do so.

If they choose not to load a game, the program should prompt the user for the desired size of their board: both row and column should be gotten from the user (in that order). Both row and column must be at least 5; if they enter a smaller number, the program must prompt the user again.

Users then choose a column to put their piece in; the piece will be placed at the "bottom" of the column. **Column numbering must start at 1, not zero!** If the user chooses an invalid column or a column that is already full, the program must prompt the user again.

At any time, instead of choosing their next column, the user can choose to save the game, simply by entering "s". Doing so should immediately save the game to a file (ask the user what they want the filename to be). The format in which the game is saved is up to you, but you should store at least the board and the player whose turn it currently is. After the game is saved, play should continue as normal (ask the same user for their column choice again). See the sample output for an example of this behavior.

After a game ends (in a draw or in a win), the user should be asked if they want to play again. If they choose to play again, they should be asked for the dimensions of the next game's board (row and column).

The user can exit the game by choosing "n" after a game has ended, or by hitting CTRL+C at any point in time.

More Details

Some additional requirements:

1. The game will played by two human players – you do NOT need to worry about a computer player for this assignment.
2. Player 1 always goes first when a new game is started. (The only exception is when loading a game saved by Player 2.)
3. You must show the updated board between each player's turn.
4. You must always tell the user their options (range of numbers, "s" to save, etc.) when asking them for a choice (see sample output).
5. The game must output a message when a draw happens.
6. The game must output a message when one player wins (and must state which player won).
7. If the user chooses to play another game after one ends, the program must reprompt for new row and column sizes, and use them in the new game.

Input Validation

For this project, we will require that you validate input from the user. You can assume that the user will enter the right type of input, but not that they will enter a correct value. In other words, a user will always give an integer when you expect one, but it may be a negative or otherwise invalid value.

You will need to validate the following things:

- When asked to enter the number of rows and columns, the user might enter any whole number. If the number is less than 5, they should be prompted for a new number.
- When asked a yes/no question, the user might enter anything. You should only accept lowercase "y" (for yes) or lowercase "n" (for no). If they enter anything else, they should be prompted again.
- When asked to choose a column to place their piece in, the user might enter any whole number, or a lowercase "s" (to indicate they want to save). They will not enter any other strings. If they choose an invalid integer (a column that does not exist, such as -1 or 99), they should be prompted again.

Sample Output

The sample output for this project is long enough that we have made it available as a separate file. You can find it on the assignment page on the course website, or by clicking here (https://goo.gl/WfKRG9). It contains examples of input validation, winning, drawing, different board sizes, etc.

Your board should be displayed exactly as shown in the sample output. Blank spots should be represented by underscores, player tokens should be represented by "x" (for Player 1) or "o" (for Player 2), and each spot should be separated by a space when it is printed out.

Sample Saved Game Files

We have also provided two sample saved game files for you. They are the same ones that are saved to (and later loaded from) in the sample output. You can download them directly to your proj1 directory using the commands

```
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/save1.txt .
cp /afs/umbc.edu/users/k/k/k38/pub/cs201/save2.txt .
```

Your program does **not** need to follow the layout or format of the provided save files. You can use any method or style you like. As long as your function to load from your saved file in your program works correctly, it can look however you want. We have provided these sample saved game files as an example of one of the many ways to save a Connect Four game.

However you decide to format your save files, it is important to keep in mind what you will need to do to read them in later. Writing the entire board straight to the file is easy: you simply type `write(str(connect4Board))` or something similar. But doing so makes loading very difficult, because you have to handle reading in a board that looks like this:

```
[['_', 'o', 'x', '_', '_', '_', '_',
'_', '_'], ['x', 'x', 'o', 'x', '_',
'o', 'x', '_', 'o'], etc...
```

Advice and Suggestions

Remember what we've covered so far! Do not try to program the entire game up at once, and don't "cowboy code." You need to break this program down into simple pieces that you code and test one at a time.

HINT: You should represent the board as a two-dimensional list.

Here is one possible way you could tackle this problem:

1. Ignore saving, loading, and checking for win or draw. Leave those until you know everything else works.
2. Get the size of the board from the user and create it.
    a. Be able to print out the board.
3. Use a "temporary" **`main()`** to interact and test each function as you create and write them.
4. Figure out how to "switch" turns from one player to another.
5. Be able to validate the user input for choosing a column.
    a. Remember that "s" is a valid option, even if your code doesn't do anything with it just yet.
6. Update the board
    a. Make sure it's the right column and the right player's piece.

Once all of that works, and you've tested it completely, then you can start writing the code to handle draws, wins, and saves and loads. (Work on them separately! Draw is probably the easiest.)
Pay close attention to the sample output provided when testing your own code.

Submitting

Once your **`proj1.py`** file is complete, it is time to turn it in with the **`submit`** command. (You may turn in the file multiple times as you complete each function. To do so, simply submit **`proj1.py`** each time you complete a part of the homework. Each new **`submit`** will overwrite the old file.)

You must be logged into your GL account, and you must be in the same directory as your Project 1 python file. To double-check this, you can type **`ls`**.

```
linux1[3]% ls
proj1.py
linux1[4]%
```

To submit your Project 1 python files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`PROJ1`**. Type in (all on one line) **`submit cs201 PROJ1 proj1.py`** and press enter.

```
linux1[4]% submit cs201 PROJ1 proj1.py
Submitting proj1.py...OK
linux1[5]%
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**