**HW 2: Tic-Tac-Toe**

**Overview**

In this assignment you will implement a tic-tac-toe game for two players. The program will prompt each player for their move and then check to see if anyone has won after each move. If the board fills up and nobody has won, the program will declare the game a tie. You will start with a partially implemented game that has some functions implemented and documented as well as some functions that need to be implemented, documented, or both.

This assignment will give you experience working with functions and documentation, especially Python docstrings using the NumPy docstring format. You will also walk through the process of creating the program from scratch in your Discussion Section, so you will gain some experience with using functions to break a complex problem down into smaller problems that you can solve one at a time.

**Instructions**

For all assignment sections,

* Make sure that your code adheres to the [507 Assignment Guide](https://paper.dropbox.com/doc/507-Assignment-Guide--AshR98EaGu3X1IlUreueQi4fAQ-RwRuP1S6RzY21Z04PhJDr).
* See the Grading section for the specific point breakdown that will be used to calculate your grade.

**Main Assignment (100 points)**

1. Copy the file [hw2-ttt.py](https://drive.google.com/open?id=1hpCpEqyU3dzDrysZSkWpFnnhFyUVOIBM) to your computer and open it in your code editor.
2. Find all of the # TODO comments and implement the missing pieces (documentation and/or function implementations).
3. Play-test your game as you develop to make sure that it works properly, and that it handles invalid inputs gracefully (i.e., the program does not crash).
4. Your program should behave as shown in the [HW2 Sample Output](https://docs.google.com/document/d/1Mp91sBcYk8gKunms1tilrA6pLFHjUR3nfkYKIG_2xrI/edit?usp=sharing).
   1. Note that the sample code does not display the winner (Nobody, X, or O) when the game ends as shown in the sample output. You should add this where appropriate.

**Extra Credit #1 (2 points)**

1. Create a new file called **hw2-ec1-connect4.py.**
2. Implement a Connect 4 game that behaves as shown in the [EC1 Sample Output](https://docs.google.com/document/d/1xUrRtsrBY0B8jh75Xu4Oic0nqRi-KFYlENuDKy8ttyg/edit).

**Extra Credit #2 (2 points)**

1. Create a new file called **hw2-ec1-arcade.py.**
2. Implement a program that lets the user select *either* Tic-Tac-Toe *or* Connect 4, and then play the selected game.
3. When the game ends, the user can either select another game to play or quit the program.
4. Your program should behave as shown in the [EC2 Sample Output](https://docs.google.com/document/d/1PFJOfSCC2HjxAWladVq32I2BArxdGvedH6rwLjSBHME/edit).

**Grading**

**Main Assignment**

Refer to sample output for specific examples of how to meet requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| Req | Description | Category | Point Value |
| 1 | Program displays 3x3 game board with numbers in cells and prompts user for X’s move | Behavior | 0 |
| 2 | Program rejects invalid inputs (non-numerics and numbers < 1 or > 9) gracefully and prompts user for valid input as shown in sample output | Behavior | 5 |
| 3 | Program rejects moves into occupied spaces and prompts user for a different move. | Behavior | 5 |
| 4 | When player makes a valid move, board is displayed with their player name in the correct cell. Their name is displayed in that cell for the remainder of the game. | Behavior | 5 |
| 5 | Program correctly determines vertical wins | Behavior | 5 |
| 6 | Program correctly determines diagonal wins | Behavior | 5 |
| 7 | Game ends in a tie if board is full and no winner has been found | Behavior | 5 |
| 8 | Program displays the correct winner (Nobody, X, or O) when the game is over | Behavior | 5 |
| 9 | next\_player() is implemented to match functionality specified in docstring | Code | 5 |
| 10 | make\_move() is implemented to match functionality specified in docstring | Code | 10 |
| 11 | docstring is correctly added for check\_win\_horizontal() to match function implementation | Code | 10 |
| 12 | docstring is correctly added to check\_win\_vertical() | Code | 10 |
| 13 | check\_win\_vertical() is correctly implemented to match docstring | Code | 5 |
| 14 | docstring is correctly added to check\_win\_diagonal() | Code | 10 |
| 15 | check\_win\_diagonal() is correctly implemented to match docstring | Code | 5 |
| 16 | Code style is good (variable and function names, if added, are clear; code layout, indentation, whitespace, etc. complies with [507 Assignment Guide](https://paper.dropbox.com/doc/507-Assignment-Guide--AshR98EaGu3X1IlUreueQi4fAQ-RwRuP1S6RzY21Z04PhJDr)) | Code | 10 |
|  | **Total** |  | **100** |

**Extra Credit #1**

Refer to sample output for specific examples of how to meet requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| Req | Description | Category | Point Value |
| 1 | Program displays board, updates board, and validates input correctly | Behavior | 1 |
| 2 | Program correctly determines winners in horizontal and vertical directions, as well as ties when board is full and no winner fond | Behavior | 0.5 |
| 3 | Program correctly determines winners in diagonal direction, as well as ties when board is full and no winner fond | Behavior | 0.5 |
|  | **Total** |  | **2** |

**Extra Credit #2**

Refer to sample output for specific examples of how to meet requirements. For EC2 credit, the TTT game must earn at least 90 points and run without crashing. The Connect Four game must at least meet requirement 1 and run without crashing. You can get EC2 credit without getting full credit for EC1.

|  |  |  |  |
| --- | --- | --- | --- |
| Req | Description | Category | Point Value |
| 1 | Program displays game menu, validates user input, and launches selected game or quits if user enters “no” when asked if they want to play a game | Behavior | 1 |
| 2 | Selected game is played to the end, then user is asked again if they’d like to play a game. If they say yes, selected game is played. If they say no, program exits. Program continues to run without crashing until the user decides to stop playing. | Behavior | 1 |
|  | **Total** |  | **2** |

**How to submit**

Solutions to each component of the assignment must be **submitted to Canvas** as separate files with the following names. Failure to follow the naming convention will result in no credit for the offending component.

Main Assignment: **hw2-ttt.py**

Extra Credit #1: **hw2-ec1-connect4.py**

Extra Credit #2: **hw2-ec1-arcade.py**

Assignments submitted after the deadline will be assessed a late penalty as stipulated in the syllabus.