

CptS 111 Introduction to Algorithmic Problem Solving **(<http://piazza.com/wsu/fall2016/cpts111/home>)**

Washington State University (<https://wsu.edu>)

Gina Sprint (<http://eecs.wsu.edu/~gsprint/>)

PA9 Objects (125 points)

Due Monday, December 12 at Midnight.

Learner Objectives

At the conclusion of this programming assignment, participants should be able to:

- Define classes
- Declare objects to instantiate user-defined classes
- Implement class methods
- Implement basic object-oriented programming concepts

Prerequisites

Before starting this programming assignment, participants should be able to:

- Implement lists, tuples, dictionaries
- Work with loops, file I/O, strings

Acknowledgments

Content used in this assignment is based upon information in the following sources:

- Shira Broschat's (<http://eecs.wsu.edu/~shira/>) CptS 111 Tic Tac Toe Assignment

Overview and Requirements

For this assignment you can choose *one of the following two programs* to implement:

1. Tic Tac Toe (see Option 1: Tic Tac Toe section below)
2. Game of your choice (see Option 2: Your Game Your Way section below) I want to emphasize that you are only required to implement one of the above options, **not both**.

Universal Program Details

The following program details apply to either program (Tic Tac Toe or game of your choice).

1. Display instructions at the beginning of the program.
2. Validate user input.
3. Provide updates about game progress to the user.
4. For each class you define, **add an `__init__()` method and getter/setter methods**. You should **not directly access any attributes of an object**. Call the appropriate getter or setter methods instead.
5. You are free to decide which functions/methods to define beyond the ones listed to solve the problem. Break the problem into tasks and build your solution accordingly. You will be graded on your approach. Ask the instructor or your TA for help with this!

Option 1: Tic Tac Toe

Write a program (tictactoe.py) for two-player interactive game to play Tic Tac Toe (<https://en.wikipedia.org/wiki/Tic-tac-toe>). The Tic Tac Toe game board consists of a total of $n \times n$ cells, where n is selected by the user of the game. Your program should randomly select who goes first, X or O. Once a player has been chosen, your program should prompt the player for a position (row and column) in which to draw his/her symbol. Players continue to alternate moves until either a winner has been determined or a "scratch" game occurs. A player wins if his/her symbols align with n in a row diagonally, vertically, or horizontally. A "scratch" game occurs if all cells on the board are occupied and no player aligned n of his/her symbols in a row.

Program Details

The underlying game board should be represented with the following structures:

1. `Coordinate` representing the coordinates of a cell on the board. `Coordinate` should contain the following attributes and methods (at a minimum):
 - A. `row`: a row location
 - B. `col`: a column location
 - C. `__str__()`: returns a string representation of a `Coordinate` i.e. (0, 0)
2. `Cell`, representing a cell on the board. `Cell` should contain the following attributes (at a minimum):
 - A. `occupied` whether or not this cell is occupied
 - B. `symbol`: the symbol to display at this cell (X for player one, O for the other player)
 - C. `location`: the location of this cell on the board (a `Coordinate` object).
 - D. `__str__()`: returns a string representation of a `Cell`, i.e. return the symbol
3. `TicTacToeBoard` representing the Tic Tac Toe board. `Board` should contain the following attributes and methods (at a minimum):
 - A. `n`: dimension of board (entered by user)
 - B. `board`: the Tic Tac Toe $n \times n$ grid (a 2-Dimensional list of `Cell` objects)
 - C. `__str__()`: returns a string representation of a `TicTacToeBoard` i.e. display the game board
 - D. `is_valid_move()` accepts the coordinates of a cell. Returns `True` if the coordinates are valid and the cell is unoccupied; otherwise returns `False`.
 - E. `make_move()`: accepts the coordinates of a cell and a symbol. Marks the cell with the symbol.
 - F. `is_winner()`: accepts a player symbol (X or O). Returns `True` if there are n in a row of the player symbol; otherwise returns `False`

The program should prompt the users to determine if they want to play another game. Keep track of each player's number of wins, losses, and total game played in a `GameStats` class. When the user wants to quit playing Tic Tac Toe games, report both of the user's game stats, including:

- Win to loss ratio
- Percentage of games won
- Number of scratch games

Use functions and methods where appropriate!

Example Run

Here is an example run of the program:

Welcome to Tic Tac Toe! There are two players, player 'X' and player 'O'.
Player X is going first.

```
  0 1 2
0 - - -
1 - - -
2 - - -
```

Player X, please enter the coordinates of your placement: 2 2

```
  0 1 2
0 - - -
1 - - -
2 - - X
```

Player O, please enter the coordinates of your placement: 1 1

```
  0 1 2
0 - - -
1 - O -
2 - - X
```

Player X, please enter the coordinates of your placement: 0 0

```
  0 1 2
0 X - -
1 - O -
2 - - X
```

Player O, please enter the coordinates of your placement: 2 1

```
  0 1 2
0 X - -
1 - O -
2 - O X
```

Player X, please enter the coordinates of your placement: 0 2

```
  0 1 2
0 X - X
1 - O -
2 - O X
```

Player O, please enter the coordinates of your placement: 0 1

```
  0 1 2
0 X O X
1 - O -
2 - O X
```

O won!

Would you like to play again? Enter 'y' to play or 'q' to quit: y

Player O is going first.

```
  0 1 2
```

```
0 - - -
1 - - -
2 - - -
```

...[output omitted for brevity]...

Player O, please enter the coordinates of your placement: 1 2

```
  0 1 2
0 O X X
1 X O O
2 O O X
```

Scratch game, too bad.

Would you like to play again? Enter 'y' to play or 'q' to quit: q

Player X game stats

Win to loss ratio: 0:1

Win percentage: 0.00%

Number of scratch games: 1

Player O game stats

Win to loss ratio: 1:0

Win percentage: 50.00%

Number of scratch games: 1

Bonus (10 pts)

Provide a option for the user to play against the computer. Introduce some basic artificial intelligence so that the computer makes "educated" moves.

Option 2: Your Game Your Way

Write a program (mygame.py) that is a game of your choosing. Your game can be a commonly known game or one you make up. If you choose to implement a game (or version of a game) that you didn't author, cite the game source(s) in your header comment in mygame.py.

Program Details

Your program needs to adhere to the following requirements:

1. A `GameStats` class that tracks the user(s) progress through the game (or multiple iterations of the game, as is the case with Option 1: Tic Tac Toe).
2. At least three additional classes other than `GameStats`. These could be characters and/or weapons in your game, a board and pieces for your game, cards and a deck if it is a card game, etc.
3. A main menu loop. This should display menu options such as:
 - A. Display game rules
 - B. Play game
 - C. Quit
4. A main game loop. The game should exercise iteration in some way, such as taking turns between users.
5. Make use of lists and/or dictionaries.
6. User interaction. The game should allow the user to "play" and make choices.

Feel free to ask me or your TA if your game idea/program meets this requirements. We are happy to hear your ideas, have fun with this assignment!

Note: Keep your game simple and functional. You should strive to turn in a fully functioning, low-complexity game instead of a fancy game with lots of broken features!

Game Ideas

Here are some ideas for games. Most of these games are fairly complex. I encourage you to implement a simplified version of the game, so long as the game meets the requirements specified above.

1. Blackjack (<https://en.wikipedia.org/wiki/Blackjack>)
2. Poker (<https://en.wikipedia.org/wiki/Poker>) or Texas Hold 'em (https://en.wikipedia.org/wiki/Texas_hold_%27em)
3. Dominoes (<https://en.wikipedia.org/wiki/Dominoes>)
4. Battleship ([https://en.wikipedia.org/wiki/Battleship_\(game\)](https://en.wikipedia.org/wiki/Battleship_(game)))
5. Yahtzee (<https://en.wikipedia.org/wiki/Yahtzee>)
6. Connect Four (https://en.wikipedia.org/wiki/Connect_Four)
7. Rock, Paper, Scissors (<https://en.wikipedia.org/wiki/Rock-paper-scissors>)
 - This game is fairly simple, so you would need to implement some basic artificial intelligence, such as in this assignment (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/03_Strings/RockPaperScissors/pr)
 - Also see the variant, Rock, Paper, Scissors, Lizard, Spock (https://en.wikipedia.org/wiki/Rock-paper-scissors#Additional_weapons), that was popularized by The Big Bang Theory. Watch the episode clip here (<https://www.youtube.com/watch?v=x5Q6-wMx-K8>)
8. Minesweeper ([https://en.wikipedia.org/wiki/Minesweeper_\(video_game\)](https://en.wikipedia.org/wiki/Minesweeper_(video_game)))
 - Here (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/08_ClassDesign/Minesweeper/Pr) is a link to a MSU Minesweeper Python assignment to help you get started.
9. Aces Up (https://en.wikipedia.org/wiki/Aces_Up)
 - This is a simplified Solitaire ([https://en.wikipedia.org/wiki/Patience_\(game\)](https://en.wikipedia.org/wiki/Patience_(game))) game.
 - Here (http://www.cse.msu.edu/~cse231/PracticeOfComputingUsingPython/07_ClassUse/Aces/proj09.doc) is a link to a MSU Aces Up Python assignment to help you get started.



Bonus (10 pts)

Present your game to the class during the last day of class (12/7). The presentation should be about 5 minutes and include the following:

1. Name of your game
2. Inspiration for your game
3. Demo
4. The greatest challenge you faced and how you resolved it
5. Directions for future work

Please email me by midnight on 12/6 if you would like to do this.

Note: Your game most likely will not be complete by this day. That is okay! Show us what you have, it will be great!

Submitting Assignments

1. Use the Blackboard tool <https://learn.wsu.edu> (<https://learn.wsu.edu>) to submit your assignment to your TA. You will submit your code to the corresponding programming assignment under the "Content" tab. You must upload your solutions as `<your last name>_pa9.zip` by the due date and time.
2. Your .zip file should contain your .py file and any input/output files your program may write.

Grading Guidelines

This assignment is worth 125 points + 10 points bonus. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- 5 pts for displaying game instructions
- 15 pts for tracking and displaying game information with a `GameStats` object
- 30 pts for defining and using 3 additional classes (`Coordinate`, `Cell`, and `TicTacToeBoard` for Option 1: Tic Tac Toe)
- 10 points for `__init__()` and getter/setter methods
- 10 pts for function and method design beyond the specified minimum requirements
- 15 pts for correct menu loop logic
- 15 pts for correct game loop logic
- 10 pts for validating user input
- 10 pts for utilizing lists or dictionaries (2D list representing the game board grid for Option 1: Tic Tac Toe)
- 5 pts for adherence to proper programming style and comments established for the class

