WORKSHEET D: DATA STRUCTURES

Version 2.0
BSc Computing for Games
COMP110

Dr Ed Powley

Introduction

In this worksheet, you will complete a Python implementation of the pen-and-paper game Noughts and Crosses (also known as OXO or Tic-Tac-Toe).

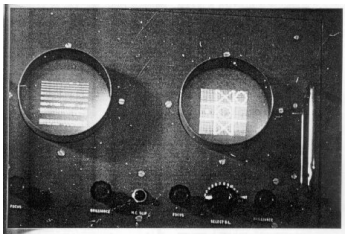
Noughts and Crosses is a two-player game played on a 3×3 grid. Players take turns to place their mark in an empty square of their choosing; usually player 1 marks `O` and player 2 marks `X`. The winner is the first player to get three marks in a row horizontally, vertically or diagonally.

To complete this worksheet:

(a) **Fork** the skeleton project and **open** `oxo.py` in your favourite Python IDE.
(b) **Choose** an appropriate data structure to represent the state of the board.
(c) **Implement** the following methods of the `OxoBoard` class:
  (i) `__init__()`, which should initialise the data structure and any other fields that are required.
  (ii) `get_square(x, y)`, which should return the current contents of the square at coordinates x, y. For this and other functions, x and y have values of 0, 1 or 2: $0, 0$ is the top left corner, $1, 0$ is the top middle, and so on. Cell contents are integers: 0 for an empty square, 1 for a player 1 mark, and 2 for a player 2 mark.
  (iii) `set_square(x, y, mark)`, which should check if the square at coordinates x, y is empty. If it is empty, fill it with the value of `mark` and return `True`; if the square is not empty, leave it alone and return `False`.
  (iv) `is_board_full()`, which should return a boolean indicating whether all spaces on the board are occupied.
  (v) `get_winner()`, which should check if either player has made three in a row. If they have, return the player number (1 or 2). If neither player has made three in a row, return 0. If the board state is such that both players have made three in a row (which cannot occur in a normal game), behaviour is undefined (i.e. your function does not need to handle this case).

It is anticipated that `get_winner()` will be the most challenging of these, so please plan your time accordingly.

Submission instructions



OXO on the EDSAC computer, one of the earliest examples of a computer game.

Begin by **forking** the GitHub repository at the following URL:

`https://github.com/Falmouth-Games-Academy/comp110-worksheet-D`

Edit `oxo.py`, implementing the required functions. **Please do not move or rename the file, or add any other files to your submission.** When you have finished, open a **pull request**.

Attend the scheduled worksheet feedback session in **week 11**, ensuring that you have uploaded all material to GitHub and opened a pull request before

this time.

Marking criteria

Remember that **it is better to submit incomplete work than to submit nothing at all**. Any attempt, even unfinished, will receive a passing grade.

Your work will be marked according to the following criteria:

- **Functional coherence**. Is your implementation correct? Your code will be run through TravisCI to verify that it gives the correct results for a large sample of input values.
- **Sophistication**. Have you made use of appropriate code structures and data structures? Note the emphasis is on **appropriate**; extra credit will **not** be given for unnecessarily complex solutions.
- **Maintainability: readability**. Is your code well commented? Are your identifier names appropriate and descriptive? Have you adhered to appropriate coding standards (e.g. PEP-8)?
- **Maintainability: expandability**. Suppose that we wanted to implement an $n \times n$ variant of Noughts and Crosses. How easily could your code be adapted to this change in requirements?