Short Description

You will implement a simple 2 player racing game. Each player rolls a 6-sided die and moves forward in a lane based on the value of the die. In order to win the game, a player must move their game piece exactly onto the last position.

Learning Outcomes

- Practice modeling real-life objects into code
- Discover new language features[1]
- Practice using loops, functions, and logic
- Learn how to organize your software using functions

Detailed Explanation and Things To Do

- Design and write code for a program that implements the race as shown in the sample scenario below.
- There are two ***players***: Player 'x' and Player 'o'.
- Both players run in a single ***lane*** that is *8 positions long*. Players take turns to roll the die. When the game starts, Player 'x' takes the first turn.
- Each player starts at the first position in the lane and moves forward based on the value of the rolled die. The symbol '*' is used to denote both players are occupying a position simultaneously. This scenario occurs at the start of the game when both players are occupying the first position simultaneously.
- The first player who reaches the last position wins the game and the game will be over. The player must reach exactly the last position. If a player "overshoots" the last position with a roll of the die, the player stays in their current position. For instance, if the player is in position 7, and rolls a 6, the player stays at 7. This player needs to roll 1 to reach the last position. In the sample scenario below, there are two such cases, and an extra line is printed:
  `"The roll was too high..."`

[1] To create this game, you may need to use a few programming language features that have not been used yet in class. Discovering new language features and how to use them is an integral part of problem-solving in computing science and an essential skill that you should learn. Think about what you need to do, search the web for python3 programming examples, and/or use the Python documentation to help you find the programming constructs that you need. If you get stuck, ask your TA for help/hints about the programming constructs you need to use.

- If one of the players (either 'x' or 'o') reaches the exact spot of the other player, it "kicks" that player back to the start. For example, if player 'x' is in position 5, player 'o' is in position 6, and player 'x' rolls a 1, then that player moves to position 6 and player 'o' is kicked back to the first position. In the sample scenario below, there is one such case, and an extra line is printed:
  `x kicked the rival!`

- Your code must implement the following major logical tasks as user-defined functions with the names given (`in brackets`):

  - Rolling the die (`roll_die`)

  - Displaying the state of the game as a one-line text (`display_state`)

  - Updating the position of a player after the die has been rolled (`update_position`)

  - Checking if a player has won which means if it has reached the last position or not (`check_game_over`)

  - Change the turn after each move by returning the opponent of a given player (`opponent`)

  - A game loop should call the other functions above. It should wait for the user to press enter before each die roll. It should then update players' positions, display the state, switch players' turns until one player wins, and at the end show when the game is over.
    The game loop should be implemented in the main function (`main`).

- Your program should play just one game.

- Make sure you are following code quality standards outlined in the Software Quality Tests.

Sample Scenario

```
Players begin in the starting position
************************************
update: * - - - - - - -
************************************
Player x press enter to roll!
Player x rolled a 4
************************************
update: o - - - x - - -
************************************
```

```
Player o press enter to roll!
Player o rolled a 5
************************************
update: - - - - x o - -
************************************
Player x press enter to roll!
Player x rolled a 4
The roll was too high, player x stays in this position
************************************
update: - - - - x o - -
************************************
Player o press enter to roll!
Player o rolled a 4
The roll was too high, player o stays in this position
************************************
update: - - - - x o - -
************************************
Player x press enter to roll!
Player x rolled a 1
x kicked the rival!
************************************
update: o - - - - x - -
************************************
Player o press enter to roll!
Player o rolled a 4
************************************
update: - - - - o x - -
************************************
Player x press enter to roll!
Player x rolled a 2
************************************
update: - - - - o - - x
************************************
Player x has won!
```

Submission Information

- You are required to submit the solution for this lab exercise by the due date provided. For submission purposes, the file with your code should be named:

  `race.py`