Classes

Estimated time for completion: 45 minutes

Overview:

In this lab exercise, we will continue with our tic-tac-toe game. Adding functions was a step in the right direction, but modern software is built with classes, not just functions. Our game is ripe for an object-oriented transformation.

These instructions assume you have **Python 3** installed. You can use any OS you choose (OS X, Windows, or Linux).

Goals:

- Create classes
- Define member variables
- Define methods
- Experiment with inheritance

Part 1 – Convert function based tic-tac-toe to classes

Open the **game.py** file (or project if you are using PyCharm) in **Classes/Before/classes_before**. This code is identical to the solution code from final code for the functions lab. If you didn't do or finish the functions lab, study this code until you understand what it is doing (hint: start with **main()**).

1. Quick review of defining classes

Before you get started, here is a quick review of defining classes.

- Classes are defined using the class keyword.
- Methods are defined just as functions, but they usually take a *self* parameter.
- Properties are defined just like methods, but with @property decorators.
- Instances of the classes are created by simply *calling* them.

```python
class Vehicle:

    def __init__(self): # constructor

        self.variable = 7


    def method(self, arg): # a method with an argument

        print("Thanks for the argument:")

        print(arg)


    @property

    def some_prop(self): # a read only property

        return "Some prop value"


# Create a vehicle:

car = Vehicle()

car.method("hello car")

val = car.some_prop
```

2. Convert tic-tac-toe to classes

Now your task will be to define 3 classes:

- Program
- Game
- Board

The functionality of the game should be spread across three classes. Here is one such partitioning.

Class	Method	Is Property
Program	__init__	No
	run	No
Game	__init__	No
	current_player_name	Yes
	winner	Yes
	find_winner	No
	cell_already_played	No
	get_cell	No
	play_cell	No
	has_winner	Yes
	switch_players	No
Board	__init__	No
	build_empty_board	No
	print_board	No
	__str__	No

In addition to this basic structure, each of these classes will need a few fields (e.g. board should have a cells field).

Using this guidance, covert the function-based game to a class-based game. It should run identically.