Python

Lab 04: Functions

Exercise 1: A Class to Represent a Car

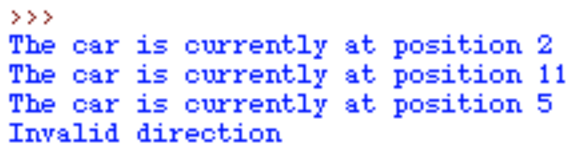
You will make a class to represent a car. Each car has a fixed speed and can drive either forward or backward. A template of the class definition `Car` is provided in the file `car.py`

1.) Implement the constructor (the `__init__` method). The constructor takes in two arguments: a starting position and speed. Both arguments are integers. Set the instantiated car's starting position and speed to these arguments.

2.) Implement the `drive` method which takes in two arguments: `time`, a positive number, and `direction`, which is either the string `'forward'` or `'backward'`. The `drive` method changes the car's position by `time` multiplied by the car's speed. If `direction` is `'forward'`, then increase the position by this amount. If the `direction` if `'backward'`, then decrease the position by this amount. Otherwise if direction is neither `'forward'` nor `'backward'`, print out the error statement `'Invalid direction'`.

3.) Implement the `printPosition` method. This method should be print out `'This car is currently at position <pos>'` where `<pos>` is the car's position. For example, if the car is located at position 2, then calling `printPosition()` should print `This car is currently located at position 2.`

4.) Uncomment the code at the end of the template and run it. Be sure to examine what the code is doing. During the lab check off, we all ask you to explain the code to us. If you implemented the `Car` class correctly, then your code should output:



```
>>>
The car is currently at position 2
The car is currently at position 11
The car is currently at position 5
Invalid direction
```

Exercise 2: Implementing a Robot Class
You will implement a Robot class. A Robot has a name and a strength that determines its ability to fight with other Robots. You will keep track of how many Robots have been created and also have two Robots fight each other. A template of the Robot class definition is provided in robot.py

1.) Notice the class attributes `robot_count` and `robot_list`. Initialize `robot_count` to 0 and `robot_list` to an empty list.

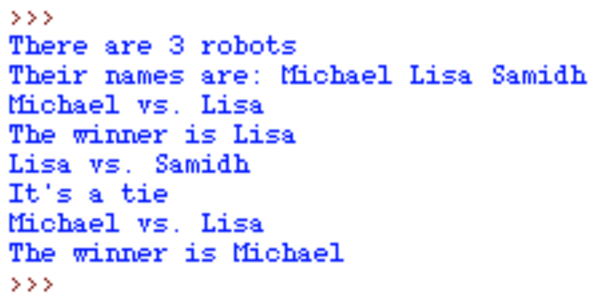
2.) Implement the constructor. The constructor takes in two arguments: `name`, a string, and `strength`, a numeric. Set the instantiated Robot's name and strength to these arguments. Increment `robot_count` by 1, and append this Robot into `robot_list`.

3.) Implement the `exercise` method. A Robot's strength increases by the same amount of `time` that it exercises. In other words, if a Robot exercises for `time` amount of time, then its strength increases by `time`.

4.) Implement the `total_robots` method. Notice that the method is decorated with `@staticmethod` to denote that it is a static method. This method should print out the number of Robots that have been created and the names of all of these Robots.

5.) Implement the `challenge` static method which takes in two inputs: `r1` and `r2` that are both Robot objects. The challenge method makes `r1` and `r2` fight and the winner is the Robot with greater strength. If both Robots have equal strength, then the fight results in a tie.

5.) Uncomment the code at the end of the template. Be sure to examine what the code is doing. During the lab check off, we all ask you to explain the code to us. If you implemented the `Robot` class correctly, then your code should output something like:



```
>>>
There are 3 robots
Their names are: Michael Lisa Samidh
Michael vs. Lisa
The winner is Lisa
Lisa vs. Samidh
It's a tie
Michael vs. Lisa
The winner is Michael
>>>
```