Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: _____

# Python Review

1. In this exercise, you will create two classes called `Pawn` and `Rook`. Both classes take in two arguments: a `name`, and a `position`. For example, a Pawn object and Rook object might be instantiated with code such as:

```
p = Pawn("P1", (5, 5))
r = Rook("R1", (3, 2))
```

[5] (a) Create the `Pawn` class.

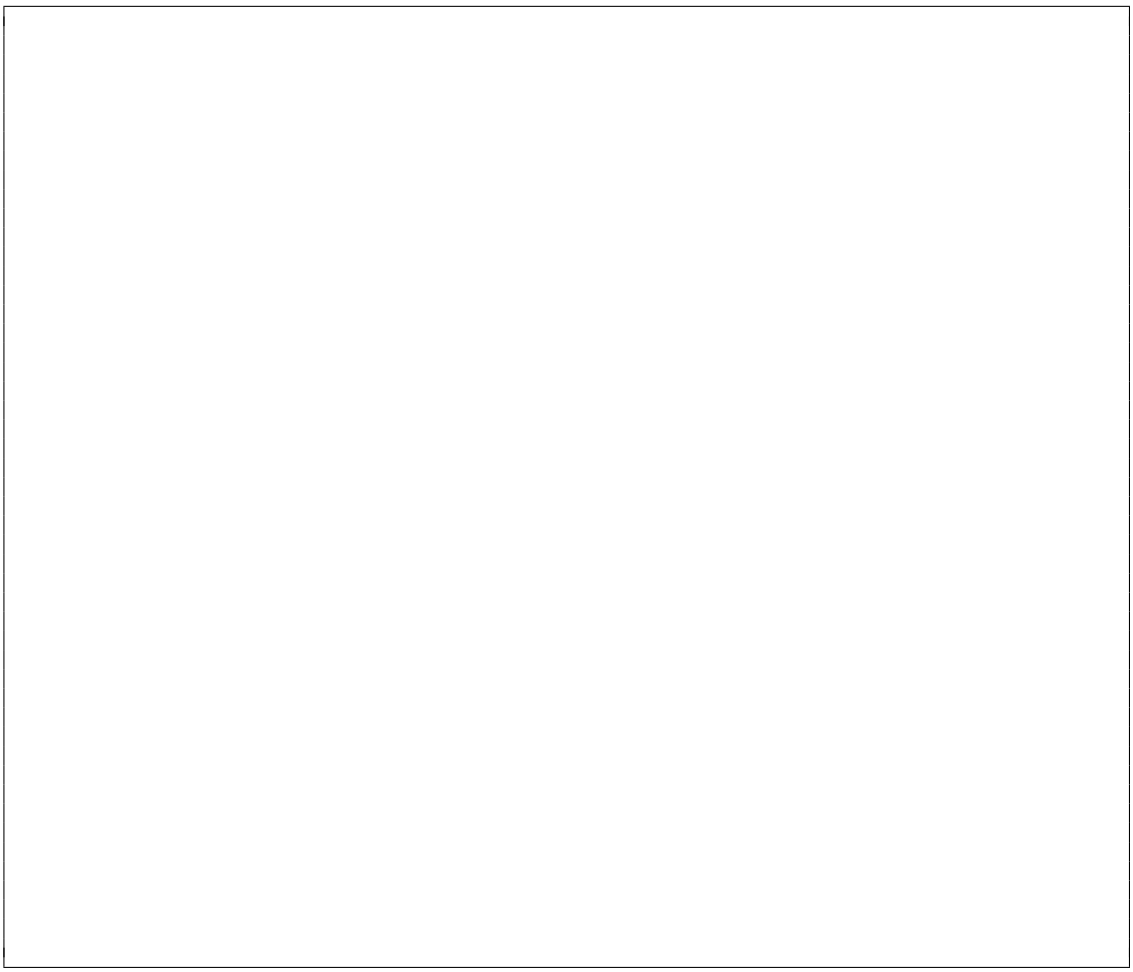
[5] (b) Create the `Rook` class.

2. You will now add a `find(self, targets)` method to the `Rook` class, where `targets` is a list of Pawns or Rooks. This method will return an $(x, y)$ tuple that indicates *how* the Rook should get to the closest (using Manhattan distance) Pawn. For instance, if the closest Pawn is at $(2, 5)$ and a Rook at $(1, 0)$, the function should return $(1, 4)$.

5 (a) Given that you have Pawns at (0, 5), (2, 2), (3, 5), and (-4, -2), compute the Manhattan distance to each Pawn for a Rook that is located at (1, 2).

5 (b) Calculate how far the Rook must travel using $(x, y)$ in order to get to each of the Pawns.

15 (c) Write the method `find(self, targets)`.



0 3. Verify your program using the following test code for check-off:

```
pawns_list = [Pawn("P1", (0,5)), Pawn("P2", (2,2)),
         Pawn("P3", (3,5)), Pawn("P4", (-4,-2))]


rook = Rook("R", (1,2))
print rook.find(pawns)
```

# Game Loop

4. These questions relate to Pygame and the game loop. Use a resolution of 640x480.

10 (a) Write a minimal game loop, using Listing 3.1 as a starting point.

5 (b) Load the `Character Boy` sprite using `pygame.image.load`[1] and store it in the variable `boy`. This code should appear before the game loop, since the `Surface` has to be loaded only once.

5 (c) Fill the background so that it is black.

5 (d) Blit the `boy` to the screen at position `(50, 50)`.

[1] `http://www.pygame.org/docs/ref/image.html#pygame.image.load`

10 (e) Modify the code so that the avatar moves one pixel to the right at each iteration of the game loop. If the unit goes past the screen, reset it so that it wraps around.

5 (f) What happens if you forget to fill the screen? Why does this happen?

5. In the previous questions, you automatically moved the avatar one pixel the right at every iteration of the game loop. In this question, you will control movement through keyboard controls. For all cases, ensure that the avatar "wraps around" the screen. You can use Listing 3.3 as a reference.

   Use the file `game.py` to add your programming logic.

10 (a) Add logic to the game loop so that pressing the right arrow moves the character to the right (`KEYDOWN`). The avatar should keep moving right until the key is released (`KEYUP`).

10 (b) Do the same for the left arrow, moving the avatar one pixel to the left.

15 (c) Add a second avatar, `Character Girl`, to the screen at location $(300, 300)$. Use `A` and `D` to control this character.

You now have a mechanism to test your AI! You can use your arrow keys to move your avatar and ultimately see how an AI opponent will respond.

| Question | Points | Score |
|---|---|---|
| 1 | 10 | |
| 2 | 25 | |
| 3 | 0 | |
| 4 | 40 | |
| 5 | 35 | |
| Total: | 110 | |