Scoring

You can keep score any way you like. I like to see how long the other ball has been in the exploded state and add that to my total. (Perhaps an easier one would just be to keep track of how many balls you have exploded). So, when a ball collides with something that is exploding (other ball), I would get that time exploded variable from the other ball and set it to be my score. Of course remember, it may be the other ball, colliding with a ball, so we have to test it both ways.

We can also add up all the scores from all the balls to get a total score.

Create this method in `ball.py`:

```python
def setPoints(self, val):
    self.point_val = val
```

When the 2 balls collide, we need to get that otherBall explode time value and set it as our point value.

Back in `BallData.py`, find the evolve method where you are detecting a collision and add the `getTimeExploded` and `setPoints` lines like you see below:

```python
... else:
    if ball.intersects(otherBall) and (ball.is_exploding() or ball.is_shrinking() or
    otherBall.is_exploding() or otherBall.is_shrinking()):
        if not ball.is_exploding() and not ball.is_shrinking():
            ball.exploding = True
            t = otherBall.time_exploded
            ball.set_points(t)
        if not otherBall.is_exploding() and not otherBall.is_shrinking():
            otherBall.exploding = True
            t = ball.time_exploded
            otherBall.set_points(t)
...
```

That will set the points for us. We now need to display it. Open `ball.py` and find the `draw` method. It should look like this:

```python
def draw(self, surface, cell_size):
    if not self.complete:
        (x, y) = self.position
        c = pygame.draw.circle(surface, self.color, (int(x*cell_size),int(y*cell_size)),
        int(self.radius * cell_size))
        if self.point_val != 0:
            textobj = self.font.render(str(self.point_val), 1, self.text_color)
            surface.blit(textobj, c)
```

Now run and see if you can see your points. Note that if it is not generating points when the bomb initially collides, you may need to add it. Look in the evolve method for a line that looks like this:

```python
if ball.intersects(self.bomb) and (self.bomb.is_exploding() or self.bomb.is_shrinking())
and not self.bomb.is_complete():
    ball.exploding = True
```

Immediately after it, and at the same indentation add:

```python
t = self.bomb.time_exploded
ball.set_points(t)
```

Getting the total game points

Essentially for each frame we will loop through all the balls, finding their point values and add them to a total. Then we will display the total on the screen. Find the `evolve` method of the `BallData`. Add the following code somewhere (you could make use of an existing loop if you want)
for ball in self.balls:
    self.total_points += ball.point_val

Navigate to the `newGame` method of the same file and add:

```
self.total_points = 0
```

Find the `draw` method of the same file and add:

```
point_text = "Total Points: " + str(self.total_points)
surface, self.cell_size * self.width/2,
self.cell_size*self.height/2)
```

You will probably want to move that text to another location since that puts it right in the center of the screen

**Other**

- Point multipliers
- Points for specific color circles?