Most common coding Q's I have been getting:

1) For *non-uniform* distributions (for example $= cx^9$) we use the following process to figure out a good number for the numerical constant in the density function (in this case c).

$$Q_{tot} = \int_i^f dq$$

$$Q_{tot} = \int_i^f \lambda\, dx$$

$$Q_{tot} = \int_i^f cx^9\, dx$$

$$Q_{tot} = \frac{c}{10}[x^9]_i^f$$

Rearrange to solve this for the constant!

$$c = Q_{tot} \times (some\ \#'s)$$

At the top of your code, you probably have a constant something like `Q = 1e-12`.
You also have the other constants which appear in your formula for the density's numerical density (i.e c).
Make the code compute your numerical constant using the formula you derive.

2) For non-uniform density with total charge zero (i.e. half the rod is positive and half is negative) the above process needs a slight modification. Do the above process for the positive half of the rod only. This means do the exact same thing but use $\frac{Q_{tot}}{2}$ and change your limits of integration to include only the positive half of the rod.

3) If using an arc instead of a rod use ds instead of dx to derive your formula for the numerical constant.
THINK: for an arc $ds = R\, d\theta$.

$$Q_{tot} = \int_i^f \lambda\, ds = \int_i^f \lambda\, R\, d\theta$$

4) If you wish to verify your rod has the correct charge (to verify you did the above steps correctly) do the following:
   a. *Before* the FOR loop which draws the balls, initialize a constant: `Q_check = 0`.
   b. *Inside* the FOR loop drawing the balls, AFTER you wrote `ball.dq = lambda * dx`, put in a line of code that says `Q_check += ball.dq`.
   c. *After* the FOR loop, put in a print statement for `Q_check`.
   d. Hopefully you discover `Q_check` is approximately equal to the total amount of charge we expect from a paper & pencil calculation.
   e. For non-uniform cases with total charge zero, you could modify the above work with an absolute value function (`Q_check += abs(ball.dq)`).

5) People get confused on **22.24** (non-uni arc) due to the coordinate system. I am not offended if you use the standard coordinates and use angles radians$(-45°)$ to $radians(+45°)$. This essentially does the problem.

   If you really want to rotate the system, modify the ball positions in the for loop as follows:
   ```
   ball.pos = -1 * R * vec( cos(pi/2-theta), sin(pi/2-theta), 0 )
   ```
   You will also need to play around with `theta_min` and `theta_max`.
   I think you might try angles like `radians(135)`, `radians(225)`, etc.
   May need to play around with minus signs as well (possibly adding a minus sign to the density constant).