Task 1:

Our physics simulator is a lot of fun, but unfortunately (or fortunately, depending on your opinion of physics), it's not obeying many of the laws of physics. In the real world, different materials have different *elasticity* – basically, the amount that something will deform upon collision. For instance, a bowling ball is made out of a very hard material that doesn't deform much, and thus it might bounce off a surface retaining most of its original energy. A tennis ball is somewhat elastic, and although it still bounces nicely, the deformation of the ball causes it to lose more energy, and as it continues to bounce, it'll not bounce as far each time. By contrast, if we made a ball out of jello, it would "splat" when it hits a surface, and not really bounce at all.

Create a `BouncyBall` class, which extends from the `Ball` class. Add a new field to the class, initialised via a parameter in the constructor, that controls "how bouncy" the ball is. Then, override the `move` method, and use the bounciness of the ball to impact what happens when a ball runs into the edge of the world.

Task 2:

Add a `Goat` class to the physics simulator (and certainly, include a suitable picture of a goat to use for it, just like we had pictures of the apple and banana. You can use this one, if you'd like)

Next, override the `move` method in the `Goat` class, but with a twist. Goats are well known for eating anything in sight – fruits, certainly, but also poison ivy, bark off of trees, tennis balls, and whatever else can't outrun them. When the Goat moves, if it finds *any other object* (other than another `Goat`!) within a distance of 10 (along either the x-axis or y-axis; let's not deal with a radius of 10) it will eat it. This should *remove* that object from the `shapes` list. You may find the `isinstance` function useful for making sure that our `Goat` can detect other `Goat`s, and thus doesn't try to eat them.