1 Administrative Topics

- We take the quiz

2 Symbol tables and inheritance

When a child class is derived from a parent class, its symbol tables entries are “copied” from the parent class’s symbol table (i.e. the values are pointers to the corresponding entries in the parent’s table). Any method that overrides the parent’s method has a value that points to the new function (instead of an arrow to the parent table). Consider the Shape class from the project. It has methods \_init\_ and draw (in addition to a bunch of mutators that I will ignore for this example). The Square class is derived from the Shape class, and it overrides the \_init\_ method. So we have these two symbol tables as shown in Fig. 1.
Figure 1: Symbol tables for parent class Shape and child class Square
When we make an instance of the Square class (i.e. a Square object), the object’s symbol table contains entries pointing to the Square table’s entries. It also contains data field (e.g. distance). If we were to call the draw method on the object, Python would look in the object’s symbol table for the draw entry, and follow the arrow up to the Square table’s draw entry, and up to the Shape table’s draw entry until it finally finds the function definition. See Fig. 2 for the arrows.

![Diagram of symbol tables for Shape, Square, and a Square object](image)

Figure 2: Symbol tables for parent class Shape, child class Square, and a Square object
3 Drawing Rectangles

Here is a hint: If you want to draw a rectangle with sides of arbitrary length, then use a distance of 1 and lots of F’s. To construct a string with lots of F’s, use the “multiplication” operator, i.e. if width contains an integer, you can say width * ’F’

4 Drawing Complex Shapes

We talked about how to design a complex shape class where the object contains a list of Shapes that can be drawn. I won’t talk about it here, but have uploaded the code.