CNA 336 Lab 1: Black Boxes and Turtles

It is rare that someone writes a script without the use of prior code, examples, cookbooks, or boilerplate code. More often than not, a programmer starts with some sample code that can solve a broad class of problems. They then use developer documentation, community support, and other reference material to adjust the sample code towards their specific problem. While helpful, it is not always necessary that a scripter understand the exact inner structure of the libraries or modules in use so long as the desired output is generated from the input. **Black boxes** are systems where it is not necessary to understand its internal structure in order to use it, and a scripter can view the system in terms of inputs and outputs. In today’s lab, we will be exploring how to function with black boxes in Python by skipping ahead a few chapters and working with the Turtle module.

Part 1: Read Chapter 4.1 and 4.3 of How to Think Like a Computer Scientist. You may skip the video if you wish and go straight to the hands‐on exercises.

Part 2: Load up Astronomy Lab and follow the instructions below: <http://interactivepython.org/runestone/static/thinkcspy/Labs/astronomylab.html>

When you load up the Astronomy Lab you will start with over 200 lines of code. It might be confusing but createSSandAnimate(), the last line of code, is executed first. Above it are **function** definitions, which are denoted by def <name>: and line indentation.

1. Click Run, what happens?
2. Refresh the page and go to line 198, numTimePeriods = 10000, and change the number to 200. Click Run.

What changes? What does numTimePeriods control?

1. In question 1 we found that line 204, createSSandAnimate() is the first line executed in the program. Open up Find (Ctrl+F) and enter “createSSandAnimate”. You will have two results. What is the line # of the other result?
2. The other result is a function definition. This is indicated by the word **def** (short for ‘define’) at the start and indented lines after. createSSandAnimate calls three **classes**: SolarSystem, Sun, and Planet.

What line numbers are the SolarSystem, Sun, and Planet classes defined?

1. Within each class you’ll observe that the first indented line starts with def init . This function creates an object, like a Solar System, a Sun, or a Planet. Inside the parenthesis are a list of parameters separated by commas, for example the SolarSystem takes self, width, and height. What are the parameters for Sun and Planet? What are their English names? (hint: below

 init )

1. The Sun initializer takes 5 parameters, but only 1 of these is used outside of the Sun. Which parameter is used in a class outside of the Sun?
2. Change the color of Pluto to Pink. Run the simulation to confirm it changes. What is the line number, and what does the line read now?
3. Double the mass of the sun.

What is the line number, and what does the line read now?

1. Increase the amount the planets move each time period (simulation step speed). What is the line number, and what does the line read now?
2. Add or modify code so that the planets spiral into the sun.

What lines did you add/modify? (Hint: There is more than one way to do this)

EXTRA CREDIT: Modify one of the planets to have a polar orbit of sin (KΘ) It should look like this:



(Image courtesy University of Georgia)