CS102, Python, Practical 6, Python Classes

Please email your solution to Question 2 as an attachment to the course lecturer at the following address:

martin.meere@nuigalway.ie

Please note the following:

(a) **your program should be saved as**

planet_yourfirstname_yourlastname.py;

(b) **as a backup, put your name and student id number in a comment at the head of the program;**

(c) **the deadline for submission is March the 23rd;**

(d) **programs that do not work at all will receive 0%;**

(e) **include sufficient comments for the program to be comprehensible;**

(f) **you may get assistance from the tutors (I encourage you to do so in fact) or the lecturer, but people who simply copy or share work risk losing some or all of their marks for this practical.**

1. *Sphere Class.* Write a `Sphere` class to represent a solid sphere and save it in a module named `spherical.py`. Your class should contain the following methods.

   `__init__(self,radius)` The constructor. It creates a sphere having the given `radius`.

   `get_radius(self)` Accessor method that returns the radius of the sphere.

   `surface_area(self)` Returns the surface area of a sphere.

   `volume(self)` Returns the volume of a sphere.

   Include a docstring for the class and for each of the methods. Using the `help` command in IDLE, print all of the information in your docstrings to the screen.

   Using your `Sphere` class, write a program in another file to calculate the surface area and volume of a sphere. The program should prompt the user to enter the radius of the sphere, and then print its surface area and volume to the screen.

Hint: In your program, you can import your `Sphere` class using the command:

```
from spherical import Sphere
```

2. *Planet Class.* On pages 115 to 120 of your notes, an algorithm for tracking the motion of a planet about the Sun is described. Write a `Planet` class to model the motion of a planet about the Sun, and save it in a module named `planetary_motion.py`. Your class should contain the following methods.

   `__init__(self, x_pos, y_pos, x_vel, y_vel)` The constructor. Here `(x_pos,y_pos)` and `(x_vel,y_vel)` give the position and velocity, respectively, of the planet at a given moment.

   `get_x_pos(self)` Accessor method that returns the first position coordinate of the planet. Similar accessor methods should be included for `y_pos`, `x_vel`, and `y_vel`.

   `update(self,dt)` This method updates the position and velocity of the planet for a time `dt` later; the appropriate algorithm is given on page 120 of your notes. For example, to update `x_vel`, a command of the following kind can be used:

   $$\texttt{x\_vel\_new} = \texttt{self.x\_vel} + \texttt{dt} * \texttt{self.\_\_f1}(\texttt{self.x\_pos}, \texttt{self.y\_pos})$$

   where `__f1` is defined below. Once ALL of the new (updated) quantities have been calculated, we can then update the instance variables using commands of the following kind

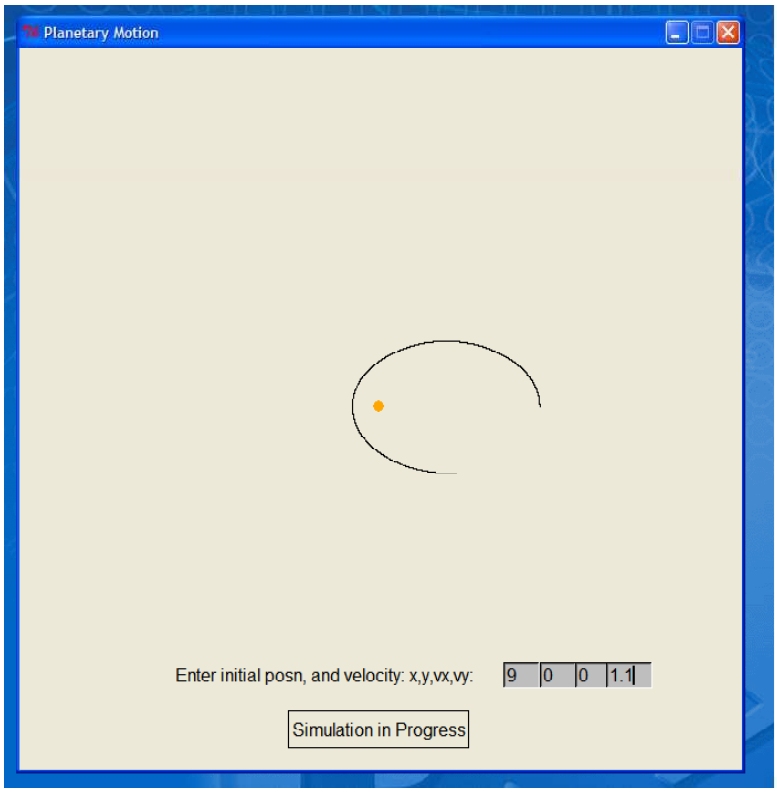
   $$\texttt{self.x\_vel} = \texttt{x\_vel\_new}$$

   `__f1(self,x,y)` Helper function that returns the first component of the force - see page 118 of your notes. A similar function `__f2(self,x,y)` for the second component of the force should also be defined. See page 113 of your notes for an example of a private helper function in the `DieView` class.

   Include a docstring for the class, and one for each of the methods.

3. You have been emailed a program called plot_planetary_motion.py which plots the orbit of a planet about the Sun. The program needs the `Planet` class of Question 2 to run, and so you will need to write this class before trying to run the program. The plotter should provide a useful check on your `Planet` class.

   A snapshot of the GUI created by the program is given below.



Snapshot of the GUI created by the program plot_planetary_motion.py.

Notice that to run the program, values for intial position, `(x,y)`, and the initial velocity, `(vx,vy)`, need to be entered into `Entry` boxes. You might like to begin by trying the values `(x,y,vx,vy)`$=(9,0,0,1.1)$. If you're interested, you might also like to try the values given in the following table.

	x	y	vx	vy
Venus	0.7	0	0	7.4
Earth	1	0	0	6.3
Mars	1.5	0	0	5.1
Jupiter	5.2	0	0	2.75
Saturn	9.5	0	0	2
Halley's Comet*	-0.6	0	0	11.3

* To see Halley's comet properly, you will need to set the value for t in the program to be about 76 (remember, our time unit is an Earth year) and you will need to extend the window to the right considerably.