Physics 503: Scientific Computing

Homework 04

Topic: Object Oriented Programming

Due: Friday March 2 by midnight.

Assignment

Use object oriented programming techniques to create a 3D Vector object (class) . The object should initialize with a list of three components (x,y,z) which are optional (test if they are supplied, if not give default values of [1,0,0]). It should be called like this: v1=Vector([2.4,3.1,0.5]) or v1=Vector() for default values.

The Vector object should have the following methods: (assume v1 is a Vector object)

1. v1.setCoords([x,y,z])
2. v1.x (as well as y and z) This would return the x component by v1.x, etc.
3. v1.mag() - return the magnitude of the vector
4. v1.dot(v2) - return the scalar product of two vectors
5. v1.__add__(v2) which allows a user to add two vectors using the syntax 'vsum=v1+v2'. Add a complimentary magic method to subtract one vector from another.
   Write a script which imports this class and puts it through a series of tests.

Notes and Resources

Here is a link to a pretty good and concise introduction into object oriented programming in Python. There are MANY others as well (as a google search will uncover).

http://www.freenetpages.co.uk/hp/alan.gauld/tutclass.htm

Some Vector analysis

1. The magnitude of a vector is computed by $|v| = \sqrt{v_x^2 + v_y^2 + v_z^2}$
2. The dot product of two vectors returns a scalar (number) is $\vec{v_1} \bullet \vec{v_2} = v_{1x}v_{2x} + v_{1y}v_{2y} + v_{1z}v_{2z}$
3. $\vec{C} = \vec{A} + \vec{B} \Rightarrow C_x = A_x + B_x, C_y = A_y + B_y, C_z = A_z + B_z$

For Fun (Optional)

Add a method to compute the cross (or vector) product. You could use a __multiply__ magic method so calling v1 * v2 returns the cross product.

If you want to get fancy, you could also have a method that converts a vector from the Cartesian coordinate system to cylindrical and/or spherical (e.g components convert from [x,y,z] to [r,theta,z]).

Another fun thing would be to be able to rotate the vector with a set of Euler angles – this gets a bit complicated in 3D, but is pretty easy in 2D.