User Manual - Module vectors32

- Module name vectors32
- class name Vector
- Purpose - Vector algebra, including the scalar (dot) product and vector (cross) product

Instantiation

Let a, b, and c be 3 D vectors with components a.x, a.y, a.z, b.x etc. a and b are created by class instantiation as follows. First import the module and create local copy of the class Vector:

```
from vectors32 import vectors32
Vector = vectors32.Vector
```

Now a vector can be created by instantiating the Vector class as:

```
a = Vector(x, y, z)
```

where x,y and z are the respective components which can be accessed as a.x, a.y, a.z.

Examples

In the following we are providing examples. Preferably, the examples should be entered by hand in the Python Shell of the "Idle" IDE. They can also be entered into a Python Shell that opens up on the terminal after Python has been invoked. At this stage all examples are solved by Python 3.2.

Example

```
>>> v1 = Vector(2.0, 3.0, 4.0)
>>> v2 = Vector(6.6, 5.5, 4.5)
>>> print(v1)
Vector(2.0, 3.0, 4.0)
>>> print(v2)
Vector(6.6, 5.5, 4.5)
```

Notice that print command indicates that it is a "Vector" and shows all three components. A more detailed output could be produced by in output as follows:

```
print('v2 = ', v2)
v2 =  Vector(6.6, 5.5, 4.5)
```

A vector of zero components could be created as:

```
z = Vector()
```

A vector length ("size") is a property easily obtained in the following example.

Example

```
>>> v3 = Vector(5.0, 6.0 ,7.0)
>>> print(v3)
Vector(5.0, 6.0, 7.0)
>>> v3.size
10.488088481701515
```

Vector normalisation is also provided for. (Normalized vector has the same direction as the original vector, but is of a unit length.)

Example

```
>>> v4 = v3.normalize
>>> print(v3)
Vector(5.0, 6.0, 7.0)
>>> print(v4)
Vector(0.476731294623, 0.572077553547, 0.667423812472)
```

Vector addition and subtraction

```
c = a + b
c = a - b
```

Examples

```
>>> w1 = v1 + v2
>>> w2 = w1 - v2
>>> print(w1)
Vector(8.6, 8.5, 8.5)
>>> print(w2)
Vector(2.0, 3.0, 4.0)
```

Multiplication, all forms

Pre-multiply a vector with a scalar (either float or int), then post-multiply a vector with a scalar-

Examples

```
>>> w1 = 2 * v1
>>> print(v1)
Vector(2.0, 3.0, 4.0)
>>> w2 = v2 * 2
>>> print(w2)
Vector(13.2, 11.0, 9.0)
>>> w1 = v1 * 1.5
>>> print(w1)
Vector(3.0, 4.5, 6.0)
```

Scalar Product of two vectors.

Scalar (aka dot) product is the sum of the products of self.x * other.x, self.y * other.y, self.z * other.z. It a scalar. In engineering Structure Analysis Scalar products are useful in calculation of the direction cosines of members or of generalized forces.

Example

```
# Scalar (dot) product.
# The result is scalar.
>>> s = v1 * v2
>>> print(s)
47.7
>>> s = v2 * v1
>>> print(s)
47.7
```

Finally, a vector (aka dot) product is a vector at right angles to the two source vectors. A useful property is a x b == -(b x a). We indicate a vector product operator by "**".

Example

```
# Vector (cross) product.
>>> v1 = Vector(1, 2, 3.0)
>>> v2 = Vector(7., 6, 5)
>>> w1 = v1 ** v2
>>> print(w1)
Vector(-8.0, 16.0, -8.0)
>>> w2 = v2 ** v1
>>> print(w2)
Vector(8.0, -16.0, 8.0)
```

A Practical Example

In the analysis of space frames for each member it necessary to establish its direction in terms of global coordinates. This is conveniently done by vectors. Suppose a member joins node A with coordinates A(0, 0, 0) to a node B(-3, 3, 4). We can immediately determine the direction of this member by its direction cosines as follows:

```
>>> from vectors32 import vectors32
>>> Vector = vectors32.Vector
>>> print('direction cosines of xbar = \n', xbn.x, xbn.y, xbn.z)
direction cosines of xbar =
-0.5144957554275265 0.5144957554275265 0.6859943405700353
```

The direction of the member AB is a necessary information but not adequate information. To analyse a member in space, we also require to define its section properties, its principal axes. To state it simply, if a load is applied in the direction of one of the principal axes, the deflection is in the plane of the load. The direction with respect to the member axes is often defined by specifying an auxiliary point in the plane of one of the principal axes. (There are two mutually orthogonal principal axes in every section. For a symmetrical section, the principal axes coincide with the axes of symmetry.)

All this begins to look quite incomprehensible, so let me define purely in geometrical terms. Given 3 points:

```
A = (0,  0,  0)
B = (-3,  3, 4)
C = (-3,  0, 4)
```

The numbers are global coordinates. We want to define local mutually orthogonal right hand sided coordinate system xbar, ybar, zbar such that xbar coincides with AB and ybar lies in the same plane as lines AB and BC. **xbar** has been calculated already. Let **cbar** be the vector from A to C. The **zbar** is the vector (cross) product of **xbar** and **cbar**, so:

```
>>> cbar = Vector(-3, 0, 4)
>>> zbar = xbar ** cbar
>>> zbn = zbar.normalize
>>> print('direction cosines of zbar = ', zbn.x, zbn.y, zbn.z)
direction cosines of zbar =  0.8 0.0 0.6
```

Thus zbar is defined. Similarly the ybar is obnained as vector product of zbar and xbar:

```
>>> ybar = zbar ** xbar
>>> ybn = ybar.normalize
>>> print('direction cosines of ybar = \n', ybn.x, ybn.y, ybn.z)
direction cosines of ybar =
-0.3086974532565159 -0.8574929257125441 0.41159660434202117
```

In a few lines of code a fairly complex problem of geometry is resolved. Vector analysis has substantially simplified the solution of the problem, even if that may not seem so obvious from the inept description.

Algis Kabaila, 2011-08-12