In [1]:
```
using JLD
using PyPlot
using Distances
import LinearAlgebra: norm
```

In [3]:
```
include("plot_arrow.jl")
```

Out[3]:
```
plot_arrow
```

# 3a (I)

In [4]:
```
w = [2 -2; -2 4];
x = [4; 6];

y = w*x
```

Out[4]:
```
2-element Array{Int64,1}:
 -4
 16
```

# b

In [5]:
```
w = [2 -2 3; -2 4 1];
x = [4; 6; 2];

y = w*x
```

Out[5]:
```
2-element Array{Int64,1}:
  2
 18
```

# c

In [6]:
```
w = [cos(pi/3) -sin(pi/3); sin(pi/3) cos(pi/3)];
x = [1;1];

y = w*x
```

Out[6]:
```
2-element Array{Float64,1}:
 -0.3660254037844385
  1.3660254037844388
```

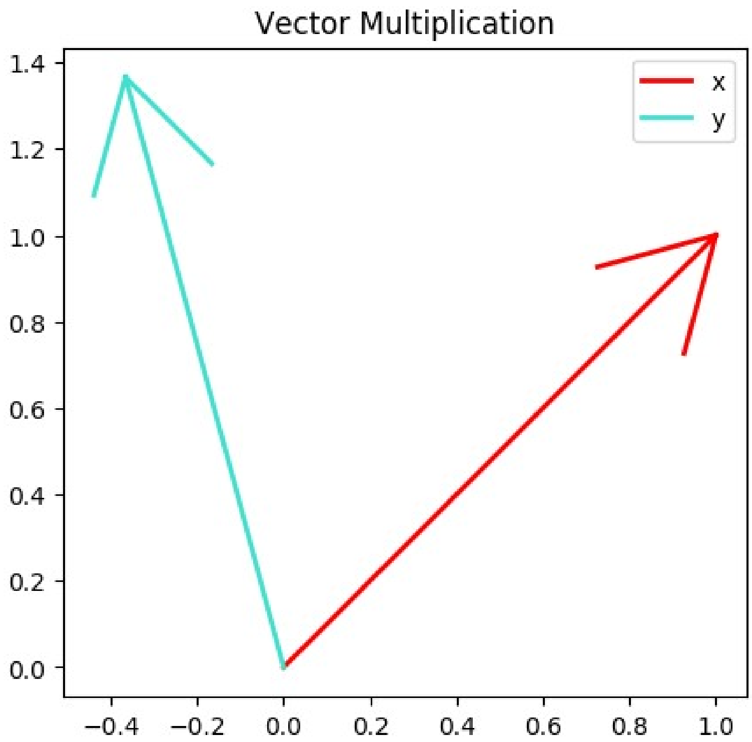
# 3 (II)

In [7]:
```
lgd = ["x", "y"]

figure(1)
clf()
    plot_arrow([0,0], x, linewidth=2, color="red")
    plot_arrow([0,0], y, linewidth=2, color="turquoise")
    axis("scaled")
    title("Vector Multiplication")
    legend(lgd)
```

```
start is [0, 0]
stop  is [1, 1]

start is [0, 0]
stop  is
```




```
[-0.366025, 1.36603]
```

Out[7]: PyObject <matplotlib.legend.Legend object at 0x7fa13a981e10>

In [8]:

```
# Finding the Euclidian lengths of vectors x & y

dist_x = norm(x);
dist_y = norm(y);

println("Euclidian distances of x = ", dist_x, " & of y =  ", dist_y);
println()

##=====
#    Find angle between vectors x & y

#    x_t * y = \x\\y\cos(theta) & x_t = x in this case

#    cos(theta) = (x * y / (dist_x * dist_y))
##=====

theta = acos(dot(x, y)/(dist_x*dist_y));

println("theta = ", theta, " = ", "pi/3 (radians)")
```

```
Euclidian distances of x = 1.4142135623730951 & of y =  1.4142135623730951

theta = 1.0471975511965976 = pi/3 (radians)
```

The resulting vector y is the same length as x, the only difference is that its direction has been rotated pi/3 radians in the anti-clockwise direction.

This is clear when we look at the arrow plot of both vectors.

It makes sense that w here is called a rotation matrix.