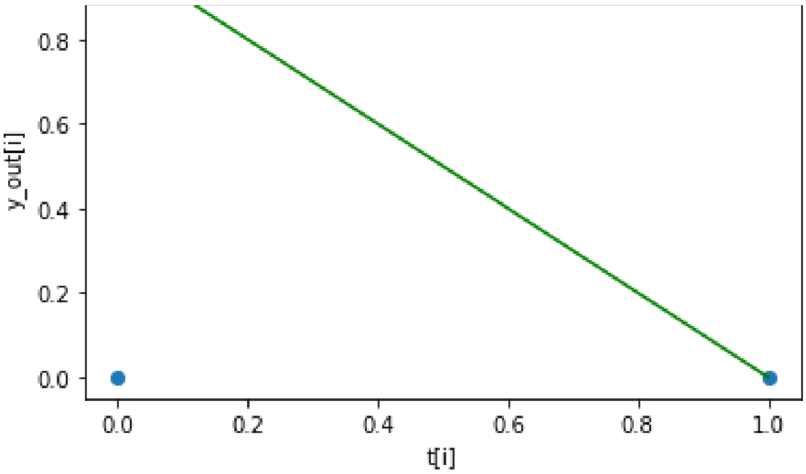
implement perceptron using OR

In [7]:
```
import matplotlib.pyplot as plt
x0=[1,1,1,1]
x1=[0,0,1,1]
x2=[0,1,0,1]
t=[0,1,1,1]
w0=0
w1=0
w2=0
y_out=[0,0,0,0]
for j in range (4):
    for i in range(len(x1)):
        y_in=w0+(x1[i]*w1)+(x2[i]*w2)
        if(y_in>=0):
            y_out[i]=1
        else:
            y_out[i]=0
        if (y_out[i]!=t[i]):
            eta=0.2
            cw0=(eta*(t[i]-y_out[i])*x0[i])
            cw1=(eta*(t[i]-y_out[i])*x1[i])
            cw2=(eta*(t[i]-y_out[i])*x2[i])
            w0=w0+cw0
            w1=w1+cw1
            w2=w2+cw2
        i=i+1
        j=j+1
for i in range(4):
    print(t[i],y_out[i])
%matplotlib inline
m=-int(w1/w2)
c=-int(w0/w2)
x=x1[:]
y=x2[:]
x_line=np.linspace(0,1)

plt.xlabel('t[i]')
plt.ylabel('y_out[i]')
plt.plot(x1,x2,"o")
plt.plot(x_line,m*x_line+c, '-g')
```

```
0 0
1 1
1 1
1 1
```

Out[7]: [<matplotlib.lines.Line2D at 0x11e6dad68>]



there is a convergence of the target and the predicted output. Based on the OR function output for various sets of inputs, we solved for weights based on those conditions and we got a line that perfectly separates positive inputs from those of negative.

implement perceptron using AND

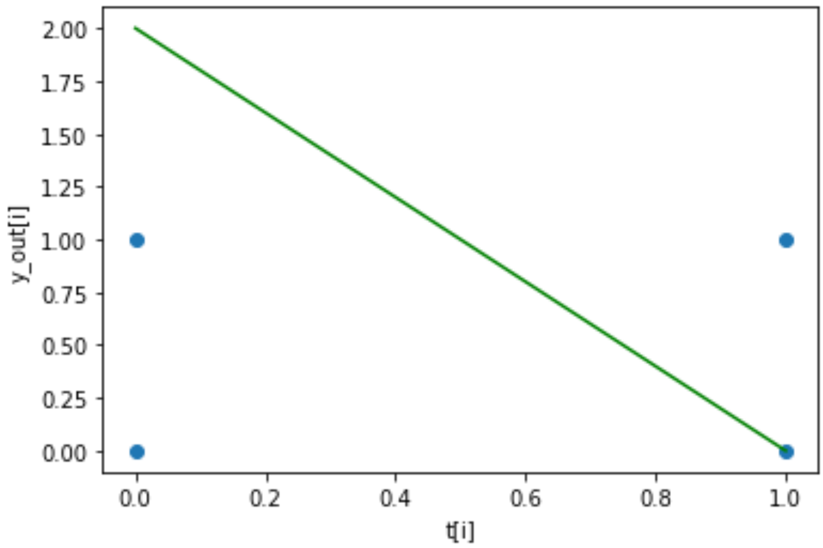
In [7]:
```
import matplotlib.pyplot as plt
x0=[1,1,1,1]
x1=[0,0,1,1]
x2=[0,1,0,1]
t=[0,0,0,1]
w0=0
w1=0
w2=0
y_out=[0,0,0,0]
for j in range (4):
    for i in range(len(x1)):
        y_in=w0+(x1[i]*w1)+(x2[i]*w2)
        if(y_in>=0):
            y_out[i]=1
        else:
            y_out[i]=0
        if (y_out[i]!=t[i]):
            eta=0.2
            cw0=(eta*(t[i]-y_out[i])*x0[i])
            cw1=(eta*(t[i]-y_out[i])*x1[i])
            cw2=(eta*(t[i]-y_out[i])*x2[i])
            w0=w0+cw0
            w1=w1+cw1
            w2=w2+cw2
        i=i+1
        j=j+1
for i in range(4):
    print(t[i],y_out[i])
%matplotlib inline
m=-int(w1/w2)
c=-int(w0/w2)
```

```
x=x1[:]
y=x2[:]
x_line=np.linspace(0,1)

plt.xlabel('t[i]')
plt.ylabel('y_out[i]')
plt.plot(x1,x2,"o")
plt.plot(x_line,m*x_line+c, '-g')
```

```
0 0
0 0
0 0
1 1
```

Out[7]: [<matplotlib.lines.Line2D at 0x11852f908>]



there is a convergence of the target and the predicted output. Based on the AND function output for various sets of inputs, we solved for weights based on those conditions and we got a line that perfectly separates positive inputs from those of negative.

implement perceprton using NAND

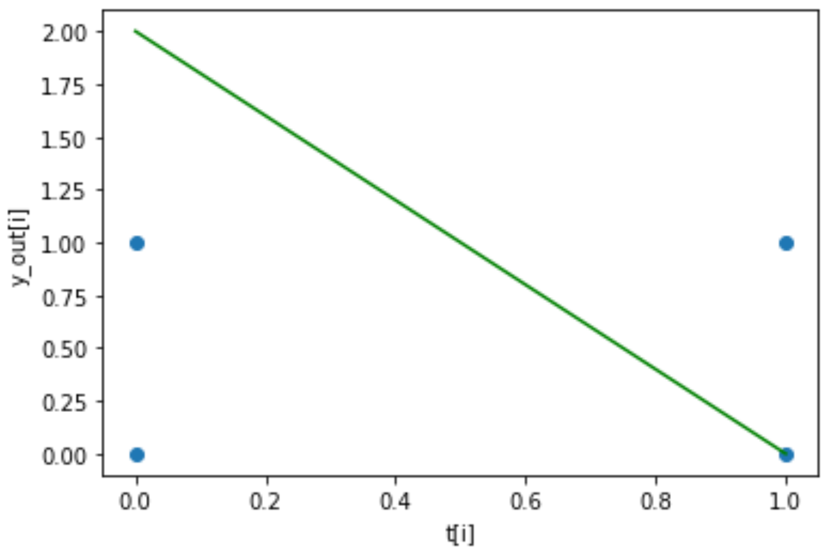
In [3]:
```
import matplotlib.pyplot as plt
x0=[1,1,1,1]
x1=[0,0,1,1]
x2=[0,1,0,1]
t=[1,1,1,0]
w0=0
w1=0
w2=0
y_out=[0,0,0,0]
for j in range (7):
    for i in range(len(x1)):
        y_in=w0+(x1[i]*w1)+(x2[i]*w2)
        if(y_in>=0):
            y_out[i]=1
```

```
                y_out[i]=1
            else:
                y_out[i]=0
            if (y_out[i]!=t[i]):
                eta=0.2
                cw0=(eta*(t[i]-y_out[i])*x0[i])
                cw1=(eta*(t[i]-y_out[i])*x1[i])
                cw2=(eta*(t[i]-y_out[i])*x2[i])
                w0=w0+cw0
                w1=w1+cw1
                w2=w2+cw2
            i=i+1
            j=j+1
for i in range(4):
    print(t[i],y_out[i])
%matplotlib inline
m=-int(w1/w2)
c=-int(w0/w2)
x=x1[:]
y=x2[:]
x_line=np.linspace(0,1)

plt.xlabel('t[i]')
plt.ylabel('y_out[i]')
plt.plot(x1,x2,"o")
plt.plot(x_line,m*x_line+c, '-g')
```

```
1 1
1 1
1 1
0 0
```

Out[3]: [<matplotlib.lines.Line2D at 0x123a66080>]



there is a convergence of the target and the predicted output. Based on the NAND function output for various sets of inputs, we solved for weights based on those conditions and we got a line that perfectly separates positive inputs from those of negative.

Implement perceptron learning using XOR

In [ ]:

```
import matplotlib.pyplot as plt
x0=[1,1,1,1]
x1=[0,0,1,1]
x2=[0,1,0,1]
t=[0,1,1,0]
w0=0
w1=0
w2=0
y_out=[0,0,0,0]
for j in range (10):
    for i in range(len(x1)):
        y_in=w0+(x1[i]*w1)+(x2[i]*w2)
        if(y_in>=0):
            y_out[i]=1
        else:
            y_out[i]=0

        if (y_out[i]!=t[i]):
            eta=0.2
            cw0=(eta*(t[i]-y_out[i])*x0[i])
            cw1=(eta*(t[i]-y_out[i])*x1[i])
            cw2=(eta*(t[i]-y_out[i])*x2[i])
            w0=w0+cw0
            w1=w1+cw1
            w2=w2+cw2

        i=i+1
        j=j+1
print(w1,w2,w0)
for i in range(4):
    print(t[i],y_out[i])
%matplotlib inline
```

from the above output of the data one can conclude that there is no convergence of the target output and the predicted output, by this i conclude that preceptron traing using XOR is tough.

Perceptron training on iris dataset

In [9]:

```
import pandas as pd
import numpy as np
```

```
import math
import operator
from sklearn import datasets
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
iris= datasets.load_iris()
X = iris.data
y = iris.target
x1=(X[:,0])[:100]
x2=X[:,2][:100]
t=y[:100]
#np.where(target=0 or -1)
y=[]
for i in range(100):
    y.append(0)
w1=0
w2=0
w0=0
for e in range(3):
    for i in range(100):
        yi=w0+w1*x1[i]+w2*x2[i]
        if(yi<0):
            y[i]=-1
        else:
            y[i]=1

        if(y[i]!=t[i]):
            w1=w1+0.5*(t[i]-y[i])*x1[i]
            w2=w2+0.5*(t[i]-y[i])*x2[i]
            w0=w0+0.5*(t[i]-y[i])
#print(w1, w2,w0)

for i in range(100):
    print(t[i]," ",y[i])


%matplotlib inline
m=-int(w1/w2)
c=-int(w0/w2)
x=x1[:]
y=x2[:]
x_line = np.linspace(0, 10,10)

plt.plot(x,'.')
plt.plot(y,'x')
plt.plot(x_line, m*x_line+c , '-g')

```

```
0   1
0   -1
0   1
0   -1
0   1
```

```
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
0    1
0    -1
1    1
1    1
1    1
1    1
1    1
1    1
1    1
1    1
1    1
1    1
```

```
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
1 1
```

Out[9]: [<matplotlib.lines.Line2D at 0x1056f6828>]