In [56]:
```
from numpy import array
from matplotlib import pyplot
from numpy.linalg import inv
from numpy.linalg import qr
```

Here we are setting the data array

This is originally the A vector

however, we will break this A vector into 2 vectors

One vector is going to become the new A vector

the other is going to be the b vector

All in representation of Ax = b

we will use this A and b to find the x vector

In [57]:
```
data = array([[0.05,0.12],
             [0.18, 0.22],
             [0.31, 0.35],
             [0.42, 0.38],
             [0.5, 0.49],])
```

In [58]:
```
print(data)
```

```
[[0.05 0.12]
 [0.18 0.22]
 [0.31 0.35]
 [0.42 0.38]
 [0.5  0.49]]
```

Now we are seperating the vectors into 2

In [59]:
```
#X = data[:,0]
#y = data[:,1]

A, b = data[:,0] , data[:,1]
print(X)
print(y)
```

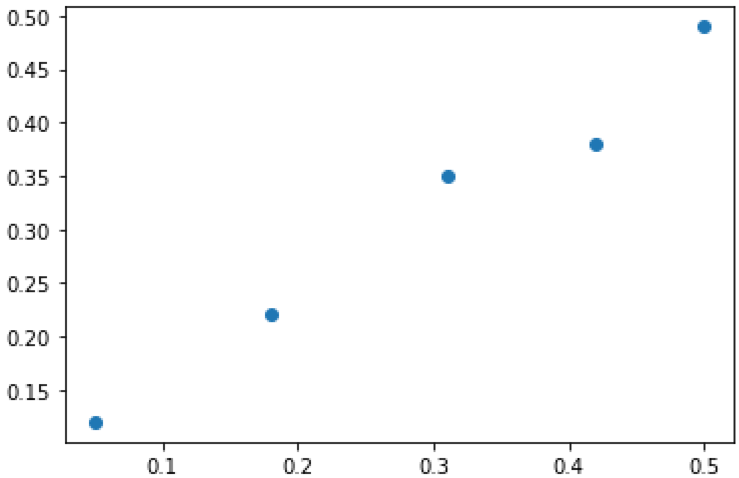
```
[[0.05]
 [0.18]
 [0.31]
 [0.42]
 [0.5 ]]
[0.12 0.22 0.35 0.38 0.49]
```

We are now resphaing the A vector to be 1 dim

because it was originally 2 dim

We are also plotting the points in relation to A and the b vector

In [60]:
```
A = A.reshape((len(A), 1))
#plot dataset
pyplot.scatter(A,b)
pyplot.show()
#bOriginal = inv(data).dot(y)
#pyplot.scatter(bOriginal)
#pyplot.show()
```



We are now using the x vector we found from Ax = b

and plotting it to see its relationship

as you will see, though the A,x, and b vectors are in relation to each other

it does not give us a good prediction as to the general area of

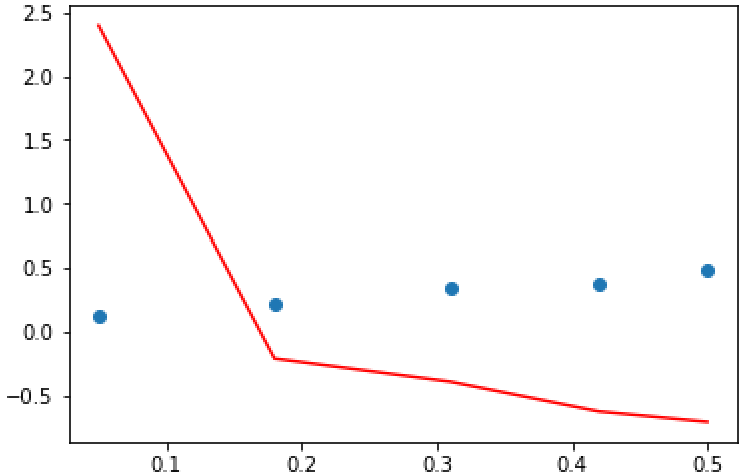
a next potential dat point could be

to find a good relationship, we need to find a common trend between both the A and b

We need to do this by finding a best-fit line

In [61]:
```
data2 = array([[2.4],
                  [-0.212],
                  [-0.394],
                 [-0.628],
                 [-0.71],])
bOriginal = data2[:,0]
print(bOriginal)
pyplot.scatter(A,b)
pyplot.plot(A, bOriginal, color='red')
pyplot.show()
```

```
[ 2.4   -0.212 -0.394 -0.628 -0.71 ]
```



These are the dim and cols of A Transpose

The rows are 1 and the cols are 5

so A^T is 1x5

In [62]:
```
ATrans = A.T
print(ATrans)
rows = len(ATrans)
cols = len(ATrans[0])
print(rows)
print(cols)
```

```
[[0.05 0.18 0.31 0.42 0.5 ]]
1
5
```

A best-fit line can be found using the formula mentioned in the research paper

We need a linear representation of this

A best-fit line goes in-between the data pointsin hopes to minimize the distance between each points

We can do this by finding a projection

the goal of a projection is exactly this

This is represented as xhat

In [68]:
```
xhat = inv(ATrans.dot(A)).dot(ATrans).dot(b)
print(xhat)
```

```
[1.00233226]
```

With the above, we calculated the projection of b onto A

As you can see, This is only 1x1 matrix

This is perfect

we need a consistent value (slope) to represent our best-fit line

A^T is a 1x5 matrix

A is a 5x1

Demonstrate how the formula goes about making the matrix a 1x1 matrix

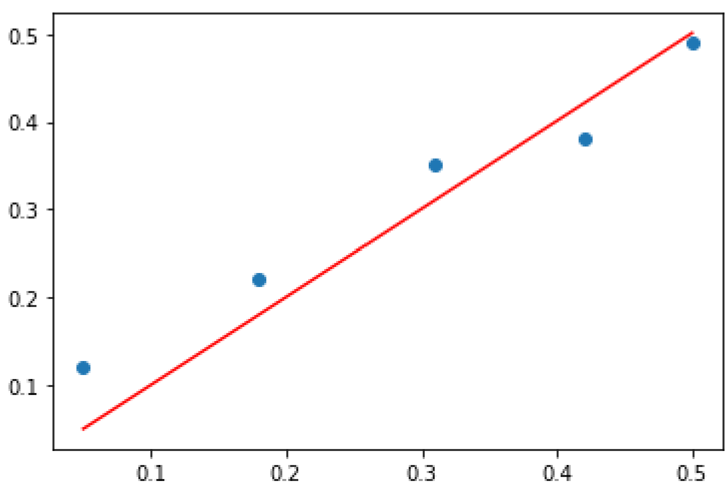
Now we have our slope

We also have our A matrix

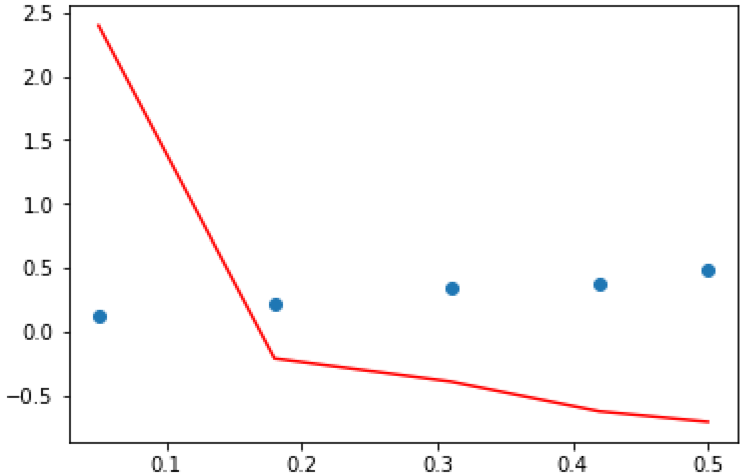
We now use this to find the projecton of b onto A

# #

In [69]:
```
projb = A.dot(xhat)
pyplot.scatter(A,b)
pyplot.plot(A,projb,  color='red')
pyplot.show()
```



In [70]:
```
pyplot.scatter(A,b)
pyplot.plot(A, bOriginal, color='red')
pyplot.show()
```



In [71]:
```
### Lets do this suing QR  Decomposition now
```

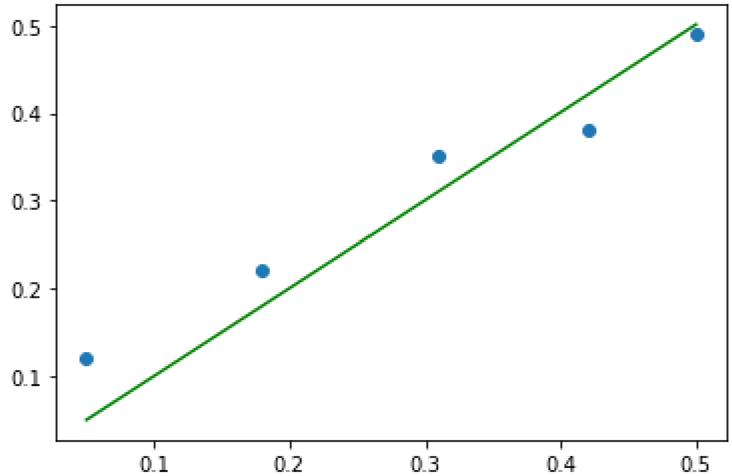
In [72]:
```
Q,R = qr(X)
print("This is Q: ")
print(Q)
rows = len(Q)
cols = len(Q[0])
print(rows)
print(cols)
print("This is R:  ")
print(R)
rows = len(R)
cols = len(R[0])
print(rows)
print(cols)
```

```
This is Q:
[[-0.06697096]
 [-0.24109545]
 [-0.41521995]
 [-0.56255606]
 [-0.6697096 ]]
5
1
This is R:
[[-0.74659226]]
1
1
```

We will ow use QR Decomposition in the same fashion as mentioned in the research paper

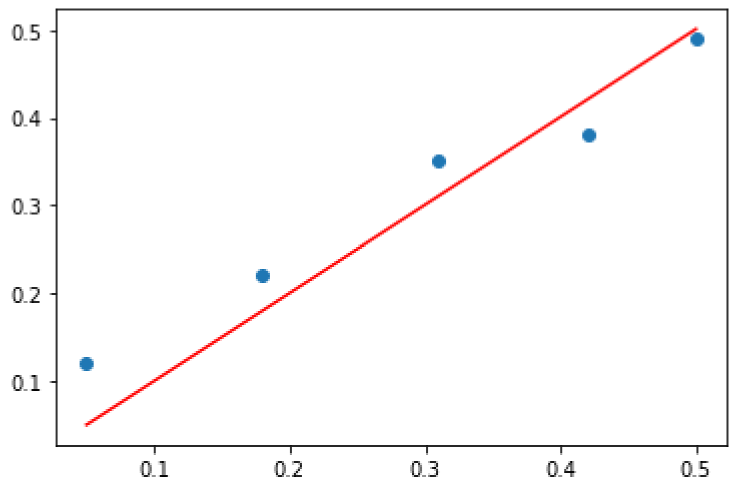
In [73]:
```
xhat = inv(R).dot(Q.T).dot(b)
print(xhat)

xhatQR = A.dot(xhat)

pyplot.scatter(A,b)
pyplot.plot(A, xhatQR, color = 'green')
pyplot.show()
```

```
[1.00233226]
```



In [74]:

```
projb = A.dot(xhat)
pyplot.scatter(A,b)
pyplot.plot(A,projb,  color='red')
pyplot.show()
```



In [ ]:

In [ ]:

In [ ]: