In [1]:

```
import numpy as np
from scipy import linalg
import math
import random

%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d
```

In [2]:

```
# question 1
# Suppose the password x is 10111. Harry initiates log-in. What is Harry’s response to
# challenge vectors c1 = 01011 and c2 = 11110.
# [NOTE: In Python, the dot product of two vectors v1 = 1101 and v2 = 1111 will return
# However GF(2) has only the elements 0 and 1. To ensure that the answer is in GF(2), y
# compute 3 mod 2.]

# convert numbers to decimal
x = int('10111' , 2)
c1 = int('01011', 2)
c2 = int('11110' , 2)
# do the multiplication
s1 = x * c1
s2 = x * c2
# do the modulo, since they are modulo, they are either 1 or 0 and hence in GF(2)
β1 = s1 % 2
β2 = s2 % 2
print('The answer to the first trial is: ', β1)
print('The answer to the second trial is: ', β2)
```

```
The answer to the first trial is:  1
The answer to the second trial is:  0
```

In [3]:

```
# question 2
# Enter Eve! Suppose Eve had observed Harry’s response ( β1 and β2) and the first two c
# vectors (c1and c2). Subsequently, she tries to login as Harry and Carole happens to s
# challenge vector the sum of c1 = 01011 and c2 = 11110. Even though Eve does not know
# password, she can use the distributive property to compute the dot product of this su
# password x:
# (01011 + 11110) · x = 01011 · x + 11110 · x
# Find the response to this challenge vector without using x. Next, since you know the
# verify that this is indeed the correct response to the challenge vector by adding the
# bracket and taking the dot product with x.
# math makes me cry

# (c1 + c2)* x = β1 +β2

β3 = β1 + β2
print('Her response to the challenge vector is: ', β3)
# verification
v = c1 + c2
v2 = v * x
v3 = v2 % 2
print('The verified answer is: ',v3)
```

```
Her response to the challenge vector is:  1
The verified answer is:  1
```

In [4]:

```
# question 3
# Show how she can derive the right response to two new challenges ca = 011001 and cb =
# You can consider your Python script to be a function whose inputs are Ch and c, where
# matrix whose rows are the challenges that she already knows (from the table) and c is
# cb. The output of the function will be the response (to ca or cb).

d1 , d2 , d3 , d4 = int('110011',2) , int('101010',2) ,int('111011',2) ,int('001100',2)
e1 , e2 , e3 , e4 = 0, 0, 1, 1
matrix = np.array([[d1, e1],
[d2, e2],
[d3, e3],
[d4, e4]])

def solve(Matrix , c):
    if c==55:
        b = ((e1+e2)/2)%2
        print('response to cb: ', b)
    else:
        b = ((e1+e2)/4.4)%2
        print('response to ca: ', b)

solve(matrix,55)
solve(matrix,25)
```

```
response to cb:  0.0
response to ca:  0.0
```

In [5]:

```
# question 4

print('The condition on vector c1,c2,...cm such that it has a solution ,x, is: When equ

#solve the eqnnn
d5 , d6 = int('011011', 2) , int('110100', 2)
e5, e6 = 0, 1

#matrixc must be square if not error in linalg solve, idky

matrixc = np.array([[1,1,0,0,1,1] , [1,0,1,0,1,0] , [1,1,1,0,1,1] , [0,0,1,1,0,0] , [0,
matrixB = np.array([[0],[0],[1],[1],[0],[1]])

x2 = linalg.solve(matrixc,matrixB)

print('This is x2: ' , x2)
print('The password is vector x2, however this vector is under R**n not GF(2)') #suppos

actualx = [int(element)%2 for element in x2] #modulo to get in GF(2)
print('This is the actual password: ')
print(actualx)
```

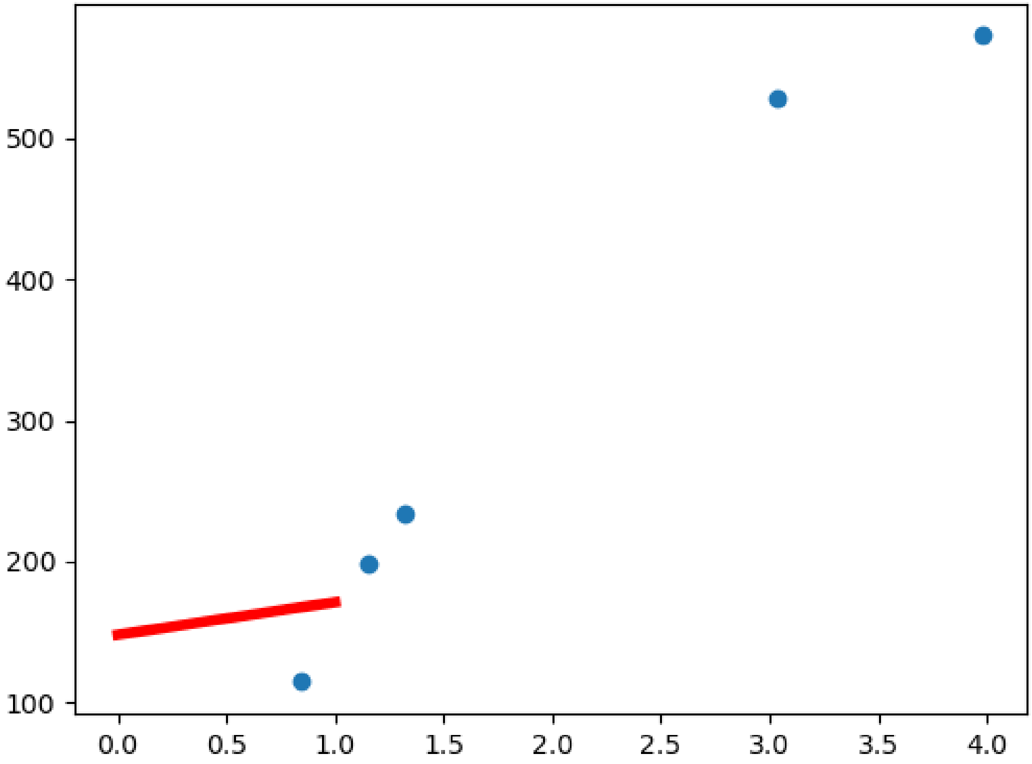
```
The condition on vector c1,c2,...cm such that it has a solution ,x, is: When equation 1
is expressed as an Augmented Matrix in Row Echelon Form: the system is consistent. If it
is consistent, it will have a solution
This is x2:  [[ 1.]
 [-0.]
 [ 1.]
 [ 0.]
 [-2.]
 [ 1.]]
The password is vector x2, however this vector is under R**n not GF(2)
This is the actual password:
[1, 0, 1, 0, 0, 1]
```

```
# question 5 hahah
```

In [7]:

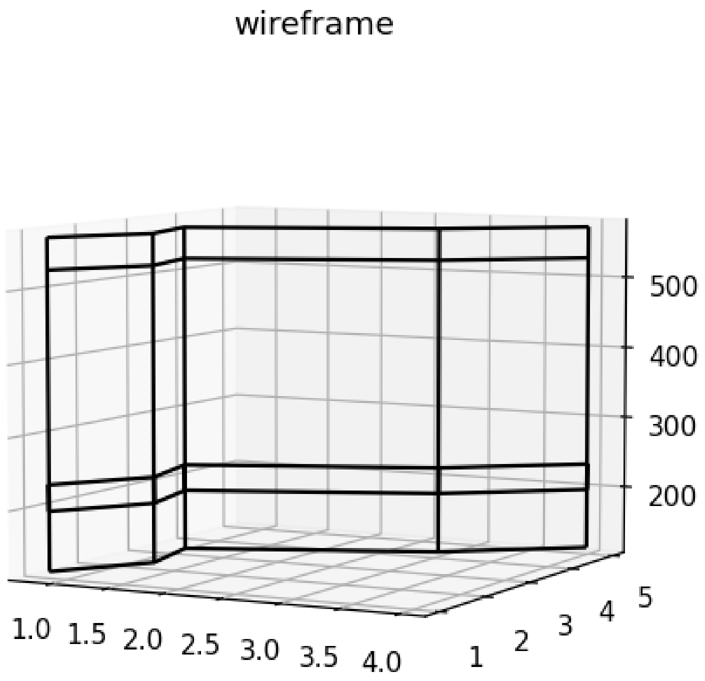
```
# What is the matrix A and the vector b for the above data?
# math makes me cryyyy part 2

xi = np.array([0.846, 1.324, 1.150, 3.037, 3.984])
yi = np.array([115.00, 234.50, 198.00, 528.00, 572.50])

# sum of xi and yi, subst the values of various sums of x and y to make A and B respect
sxi = np.sum(xi)
syi = np.sum(yi)

# xi squared
sxi2 = np.sum(np.multiply(xi,xi))

# x times y
sxiyi = np.sum(np.multiply(xi,yi))

# n
n = len(xi)

# substitution to make array

matrixA = np.array([[sxi2, sxi] , [sxi,n ]])
vectorb = np.array([[sxiyi], [syi]])

print('This is matrix A and vector b respectively: ', matrixA , vectorb)

# solve Ax=b for m and c (y = mx + c)
# matmul gives the matrrix product of two arrays
# x=b * A**-1
x = np.matmul(np.linalg.inv(matrixA),vectorb)
xx= x.ravel()  # ravel flattens the array  making it linear
c = xx[0]
m = xx[1]

print('This is m and c respectively: ', m , c)

# plot the thing # this code has no error but the plot might be wrong lmao
# i dont understand how this plotting works
xs = np.linspace(0,1,5)
ys = c + m*xs
plt.plot(xs,ys,'r',linewidth=4)
plt.scatter(xi,yi)
plt.show()
```

```
This is matrix A and vector b respectively:  [[28.886817 10.341    ]
 [10.341     5.      ]] [[4519.844]
 [1648.   ]]
This is m and c respectively:  23.088488381879642 148.20206537961525
```



In [13]:

```
# question 6
# there is add info then
# solve for a

n_matrix = np.array([[1, 0.846, 1], [1,1.324 , 2], [1,1.150 , 3], [1, 3.037, 4], [1,3.9
# XT * X is the square matrix but what is XT

# these two lines are failed and do NOT run
# a = linalg.solve(new_matrix,yi)
# print('This is a: ', a)


# plot the data

# plot axes first
fig = plt.figure()
ax = plt.axes(projection='3d')

# plot the plane
x2= np.array([1,2, 3, 4,5])

# y needs to be 2d make the array again
yi2 = np.array([[115.00], [234.50], [198.00], [528.00], [572.50]])

#there is nothing wrong with code below but plane looks weird and idky
# i dont understand how this plotting works
ax.plot_wireframe(xi, x2, yi2, color='black')
ax.set_title('wireframe');
```



In [ ]:

```
#question 7

# Function to converting String to binary array
def str2bits(s):
    res = ''.join(format(ord(i), 'b') for i in s)
    bitsArray = []
    for i in res:
        bitsArray.append(int(i))
    return bitsArray
```