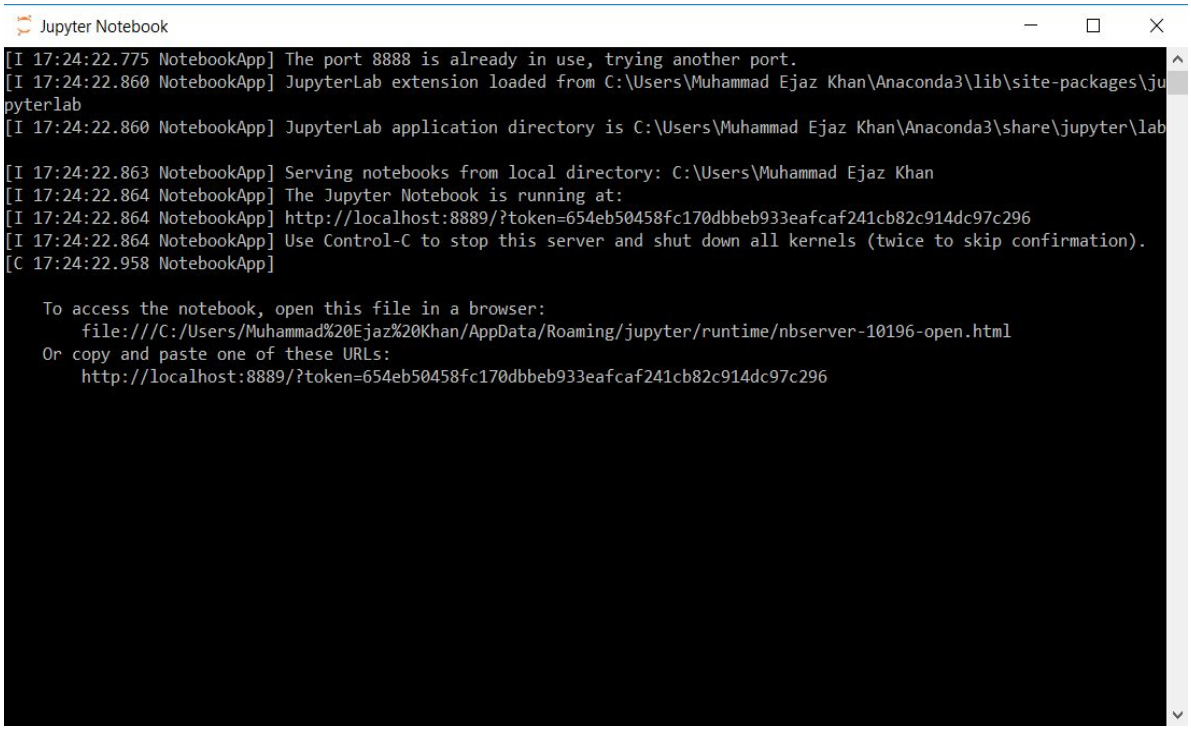
How to set environment

1. First of all download latest Python and install it.
2. In second step download Anaconda and install it.
3. When you install Anaconda, Jupyter Notebook needs to be install, for that open Anaconda Prompt and write the following code.

   ```
   Conda install prompt
   ```

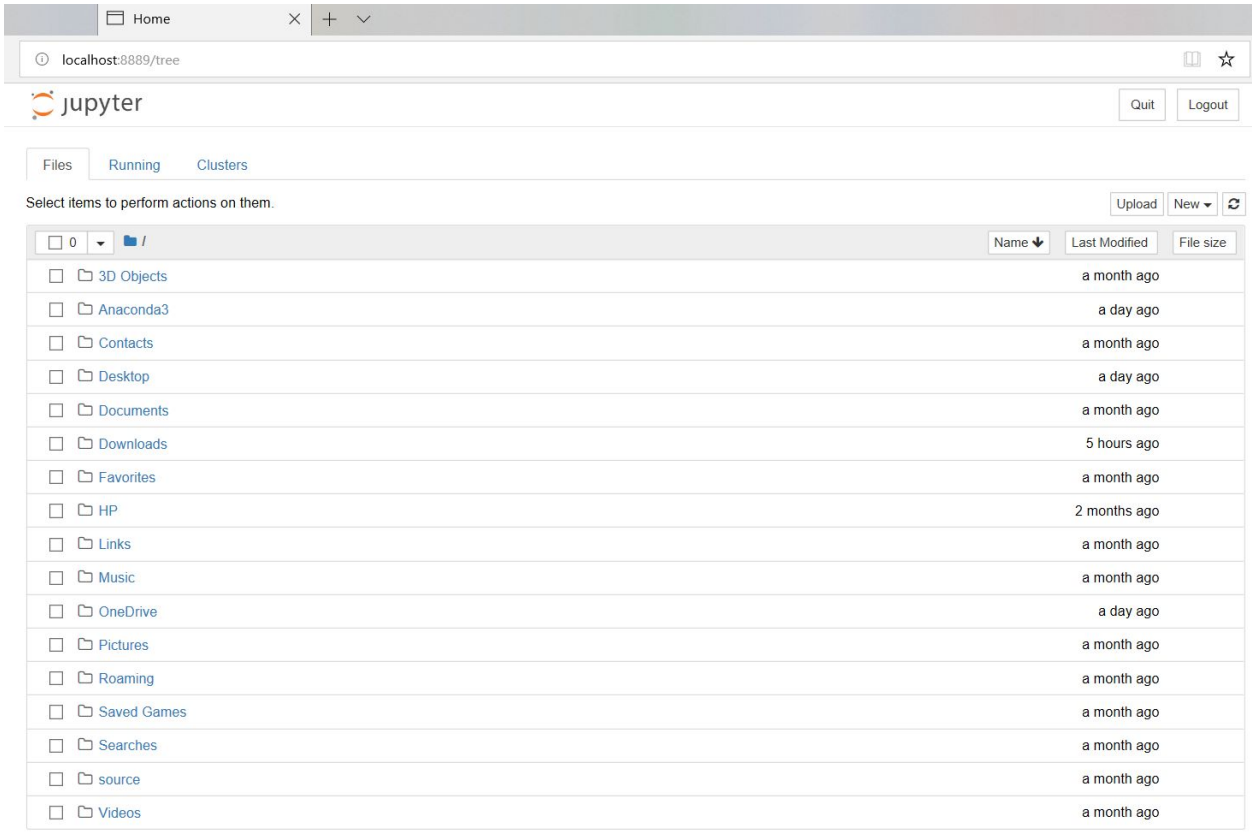
   It will download all needed packages and also Jupyter Notebook.
4. In window search Jupyter Notebook and try to open, you will see that automatically it will run some code in prompt and will open a page in Web browser (Localhost).



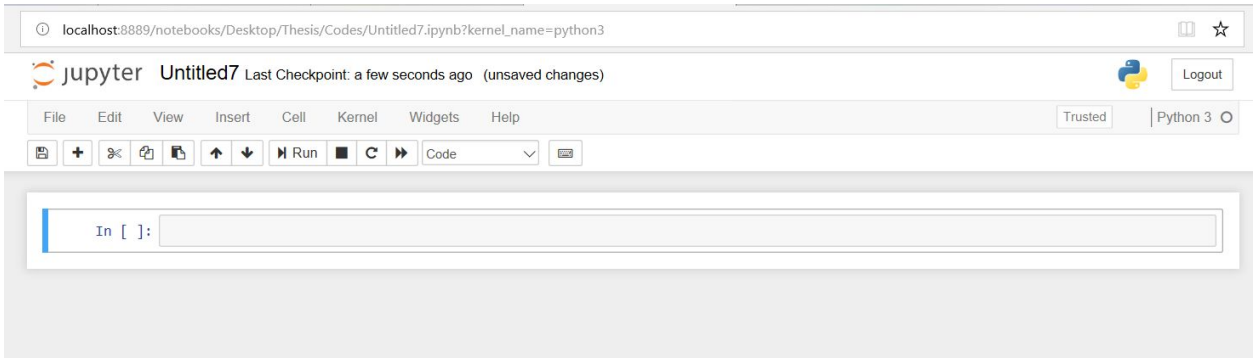
You can also reacher through a link give in prompt i-e

http://localhost:8889/?token=654eb50458fc170dbbeb933eafcaf241cb82c914dc97c296.

You will be directed to the following page



Here you have to select the location of code, where you saved it. After this a new tab will open automatically by selection python code, here you have to write Python code step by step. Below is the window looks like



How to Run the code

Following are the input and the output of the code, you have to follow each step to get the result.

It's not important that each input will give output, the output will be combination of several step.

```
In [1]:  import pandas as pd
         import numpy as np
         import csv
         import math
         import matplotlib.pyplot as plt
      #from shapely.geometry.polygon import Polygon
```

```
In [2]:  data_file='t2.6x6.e5.m45.csv'
         data=pd.read_csv(data_file)
```

In the above step, you can select a file that contain the data sets.

```
In [3]:  data
```

Out[3]:

	0000 [x:0.0,y:0.0]	0001 [x:0.0,y:30.0]	0002 [x:0.0,y:60.0]	0003 [x:0.0,y:90.0]	0004 [x:0.0,y:120.0]	0005 [x:0.0,y:150.0]	0006 [x:30.0,y:0.0]	0007 [x:30.0,y:30.0]	0008 [x:30.0,y:60.0]	0009 [x:30.0,y:90.0]	...	[x:120.0,y:(
0	-1.000	28.234	58.997	-1.000	-1.000	-1.000	28.535	43.547	-1.000	-1.000	...	-1
1	30.689	-1.000	31.338	59.463	-1.000	-1.000	41.835	30.552	43.113	-1.000	...	-1
2	58.160	31.336	-1.000	28.517	58.792	-1.000	-1.000	43.573	31.495	40.816	...	-1
3	-1.000	59.824	30.652	-1.000	30.935	59.359	-1.000	-1.000	41.274	30.439	...	-1
4	-1.000	-1.000	58.243	28.470	-1.000	31.209	-1.000	-1.000	-1.000	44.269	...	-1
5	-1.000	-1.000	-1.000	58.833	28.738	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
6	30.301	42.507	-1.000	-1.000	-1.000	-1.000	-1.000	30.087	-1.000	-1.000	...	-1
7	43.486	30.271	41.566	-1.000	-1.000	-1.000	30.488	-1.000	28.788	-1.000	...	-1
8	-1.000	41.340	29.776	40.555	-1.000	-1.000	58.321	31.899	-1.000	28.857	...	-1
9	-1.000	-1.000	43.685	31.086	42.074	-1.000	-1.000	59.755	29.271	-1.000	...	-1
10	-1.000	-1.000	-1.000	43.297	29.083	42.213	-1.000	-1.000	-1.000	30.676	...	-1
11	-1.000	-1.000	-1.000	-1.000	43.828	28.953	-1.000	-1.000	-1.000	-1.000	...	-1
12	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	30.108	42.459	-1.000	-1.000	...	-1
13	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	40.834	29.830	43.243	-1.000	...	-1
14	-1.000	-1.000	59.475	-1.000	-1.000	-1.000	-1.000	43.916	30.365	42.182	...	58
15	-1.000	-1.000	-1.000	58.440	-1.000	-1.000	-1.000	-1.000	44.159	28.974	...	-1
16	-1.000	-1.000	-1.000	-1.000	59.623	-1.000	-1.000	-1.000	-1.000	43.739	...	-1
17	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
18	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
19	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	58.922	-1.000	-1.000	...	42
20	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	59.556	-1.000	...	28
21	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	40
22	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
23	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
24	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	59
25	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	28
26	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
27	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	31
28	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
29	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
30	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
31	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	44
32	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	29
33	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	41
34	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1
35	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	...	-1

36 rows × 36 columns

```
In [4]: mydata=[]
        with open(data_file) as csv_file:
        csv_reader=csv.reader(csv_file,delimiter=',')
        line_count=0
        for row in csv_reader:
        mydata.append(row)

In [5]: myx=[]
        myy=[]
        for each in range(len(mydata[0])):
        x=mydata[0][each][7:10]
        #if (mydata[0][each][14])==':':
        y=mydata[0][each][13:16]

        if (mydata[0][each][13])==':':
        y=mydata[0][each][14:17]

        if (mydata[0][each][13])=='y':
        y=mydata[0][each][15:18]

        #else:
        #y=float(mydata[0][each][14:16])
```

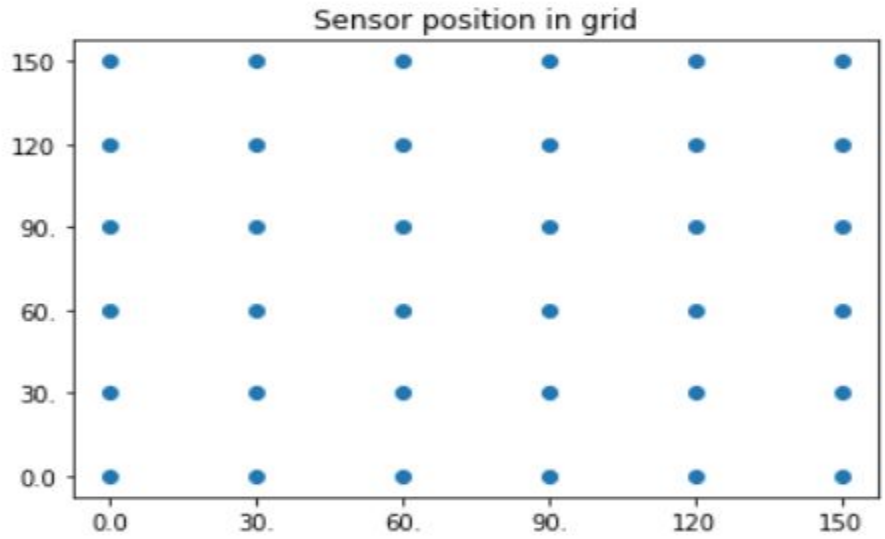
```
        #if y==12.0:
        #y=120.0
        #elif y==15.0:
        #y=150

        myx.append(x)
        myy.append(y)

In [6]: length=len(myx)
        dim=(2,length)
        device_position=np.zeros(dim)

In [7]: device_position[0,:]=myx
        device_position[1,:]=myy

In [8]: device_position
```

Out[8]:
```
array([[  0.,   0.,   0.,   0.,   0.,   0.,  30.,  30.,  30.,  30.,  30.,
         30.,  60.,  60.,  60.,  60.,  60.,  60.,  90.,  90.,  90.,  90.,
         90.,  90., 120., 120., 120., 120., 120., 120., 150., 150., 150.,
        150., 150., 150.],
       [  0.,  30.,  60.,  90., 120., 150.,   0.,  30.,  60.,  90., 120.,
        150.,   0.,  30.,  60.,  90., 120., 150.,   0.,  30.,  60.,  90.,
        120., 150.,   0.,  30.,  60.,  90., 120., 150.,   0.,  30.,  60.,
         90., 120., 150.]])
```

```
In [9]: plt.scatter(myx,myy)
        plt.title('Sensor position in grid')
```

Out[9]:

```
Text(0.5,1,'Sensor position in grid')
```



[] distance measurement

```
In [10]:  D=np.array(mydata)
```

```
In [11]:  D=D[1:,:]
```

```
In [12]:  D
```

Out[12]:

```
array([['-1', '30.381', '-1', ..., '-1', '-1', '-1'],
       ['28.317', '-1', '28.294', ..., '-1', '-1', '-1'],
       ['-1', '31.602', '-1', ..., '-1', '-1', '-1'],
       ...,
       ['-1', '-1', '-1', ..., '-1', '31.936', '-1'],
       ['-1', '-1', '-1', ..., '29.027', '-1', '28.716'],
       ['-1', '-1', '-1', ..., '-1', '28.097', '-1']], dtype='<U21')
```

```
In [13]:  norm_count=len(myx);
          #total number of the localize devices
```

```
In[14]:   def get_d(device_position,n1,n2):
          #calculate the eclidian minimum distance
                                between two point.
          #device1=device_position[:,n1]
          #device2=device_position[:,n2]
          distance=float(D[n2-1,n1-1])
          #if distance==-1:
          #distance=30
  #distance=math.sqrt((y[n2-1]-y[n1-1])**2+(x[n2-1]-x[n1-1])**2)
            return distance

In[15]:   def reboust(a,b,c):
          #set the threshold to be 45
          reboust_thresh=(45*3.142)/180
          #determine the side having the minimum distance
          if ((a<=b) and (a<=c)):
          minvalue=a
          d=b
          e=c
          elif (b<=a and b<=c):
          minvalue=b
          d=a
          e=c
          else:
          minvalue=c
          d=a
          e=b
          #check with the minimum distance
          costh=(d**2+e**2-minvalue**2)/(2*d*e)
          #if it is not reboust then return 0
          #if it is reboust then return 1
          if (minvalue*(1-(costh)**2) < reboust_thresh):
          check=0
          else:
          check=1
          return check
```

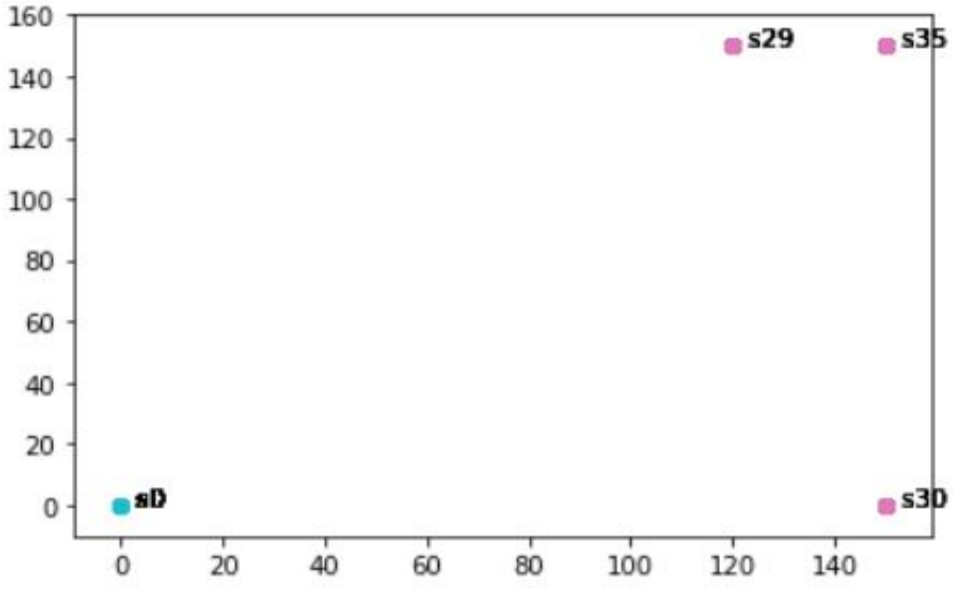
```
In[16]:   length=100
          dim=(length,4)
          Quadsi=np.zeros(dim)
          i=0
          count=1
          for j in range(1,norm_count+1):
          dij=get_d(device_position,i,j)
          #get the distance between node i and j
               #if there is no distance between i and j, then
continue
          if (dij==-1):
          continue
          for k in range(1,norm_count):
          djk=get_d(device_position,j,k)
          #get distance between j and k
          dik=get_d(device_position,i,k)
          #get distance between i and k
          if (k==j or djk==-1 or dik==-1):
          Continue
          if(reboust(dij,dik,djk)==0):
          #check if dij,dik,djk is reboust else continue
          continue
          for l in range(1,norm_count+1):
          dkl=get_d(device_position,k,l);
          if (l==k or l==j or dkl==-1):
          continue
          dlj=get_d(device_position,l,j)
          if (dlj==-1) or (dlj==-1):
               #if there is no distance between l and j, then
continue
          continue
          dil=get_d(device_position,i,l)
          if (dil==-1):
               #if there is no distance between i and l, then
continue
          continue
          #then check if all distance are reboust
          if (reboust(djk,dkl,dlj)==1 & reboust(dij,dil,dlj)==1
          &   reboust(dik,dil,dkl)==1):
```

```
#add reboust nodes to the Quadsi
d=[i,j,k,l]
print(d)
 d=[int(x) for x in d]
 Quadsi[count,:]=d
 count=count+1
```

```
[0, 29, 30, 35]
[0, 29, 35, 30]
[0, 30, 29, 35]
[0, 30, 35, 29]
[0, 35, 29, 30]
[0, 35, 30, 29]
```

In[17]:

```
for each in range(len(Quadsi)):
xq=[int(x) for x in Quadsi[each]]
xx=device_position[0,xq]
yy=device_position[1,xq]
plt.scatter(xx,yy)
for c in range(len(xx)):
s=str(xq[c])
s='  s'+s
plt.text(xx[c],yy[c],s,fontsize=10,fontweight='light')
#plt.hold(True)
#plt.show()
```


160
140
120
100
80
60
40
20
0
0
20
40
60
80
100
120
140
s29
s35
s30