1) How-to-count-distance-to-the-previous-zero

For each value, count the difference of the distance from the previous zero (or the start of the Series, whichever is closer) and if there are no previous zeros,print the position

Consider a DataFrame df where there is an integer column {'X':[7, 2, 0, 3, 4, 2, 5, 0, 3, 4]}

The values should therefore be [1, 2, 0, 1, 2, 3, 4, 0, 1, 2]. Make this a new column 'Y'.

```
import pandas as pd
df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
```

In [1]:
```
import pandas as pd
def distanceToPreviousZero(df):
    x=0
    df['Y']=pd.Series([0]*df['X'].count())
    for c,i in enumerate(df['X']):
        x=x+1
        if i == 0:
            x=0
        df['Y'][c]=x

df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
distanceToPreviousZero(df)
df
```

Out[1]:

	X	Y
0	7	1
1	2	2
2	0	0
3	3	1
4	4	2
5	2	3
6	5	4
7	0	0
8	3	1
9	4	2

2) Create a DatetimeIndex that contains each business day of 2015 and use it to index a Series of random numbers.

In [2]:
```
import datetime
import pandas
import random

year=2015
start = datetime.datetime(year,1,1)
end = datetime.datetime(year, 12, 31)
index=pandas.bdate_range(start,end)
random.seed(10)
data=[random.uniform(0,1) for i in range(1,len(index)+1)]
df=pandas.DataFrame(data=data,index=index,columns=["RandomNumbers"])
df
```

Out[2]:

	RandomNumbers
2015-01-01	0.571403
2015-01-02	0.428889
2015-01-05	0.578091
2015-01-06	0.206098
2015-01-07	0.813321
2015-01-08	0.823589
2015-01-09	0.653473
2015-01-12	0.160230
2015-01-13	0.520669
2015-01-14	0.327773
2015-01-15	0.249997
2015-01-16	0.952817
2015-01-19	0.996557
2015-01-20	0.044556
2015-01-21	0.860161
2015-01-22	0.603191
2015-01-23	0.381606
2015-01-26	0.283618
2015-01-27	0.674965
2015-01-28	0.456831
2015-01-29	0.685861
2015-01-30	0.661846
2015-02-02	0.132978
2015-02-03	0.767838
2015-02-04	0.982413
2015-02-05	0.969388
2015-02-06	0.613327
2015-02-09	0.044261
2015-02-10	0.004055
2015-02-11	0.133973
...	...

	RandomNumbers
2015-11-20	0.859736
2015-11-23	0.397028
2015-11-24	0.601159
2015-11-25	0.171430
2015-11-26	0.157736
2015-11-27	0.606055
2015-11-30	0.864483
2015-12-01	0.999838
2015-12-02	0.031139
2015-12-03	0.920982
2015-12-04	0.511118
2015-12-07	0.349356
2015-12-08	0.397415
2015-12-09	0.577889
2015-12-10	0.347697
2015-12-11	0.145509
2015-12-14	0.865563
2015-12-15	0.706489
2015-12-16	0.609855
2015-12-17	0.722559
2015-12-18	0.986026
2015-12-21	0.175093
2015-12-22	0.824169
2015-12-23	0.822304
2015-12-24	0.343348
2015-12-25	0.558777
2015-12-28	0.458395
2015-12-29	0.194336
2015-12-30	0.432898
2015-12-31	0.152559

261 rows × 1 columns

3) Find the sum of the values in s for every Wednesday

Note:- I was confused if the total sum of Wednesday is required or if the problem statement is asking for sum of values upto every Wednesday, so I have given two answers respectively.

In [3]:
```
sum=df.asfreq('W-WED',method='pad').RandomNumbers.sum()
sum
```

Out[3]:
```
25.217784010816857
```

In [4]:
```
df.resample("W-WED").sum()
```

Out[4]:

	RandomNumbers
2015-01-07	2.597802
2015-01-14	2.485733
2015-01-21	3.104088
2015-01-28	2.400211
2015-02-04	3.230937
2015-02-11	1.765003
2015-02-18	2.822568
2015-02-25	2.478621
2015-03-04	1.327116
2015-03-11	3.014171
2015-03-18	2.431143
2015-03-25	2.451345
2015-04-01	1.729979
2015-04-08	2.209936
2015-04-15	2.044935
2015-04-22	1.891635
2015-04-29	2.757743
2015-05-06	2.620876
2015-05-13	2.357328
2015-05-20	2.604384
2015-05-27	1.039313
2015-06-03	2.133322
2015-06-10	2.569566
2015-06-17	3.578192
2015-06-24	2.339702
2015-07-01	3.666839
2015-07-08	2.947362
2015-07-15	3.909121
2015-07-22	2.517961
2015-07-29	0.927974
2015-08-05	3.789931
2015-08-12	2.431424
2015-08-19	2.438088
2015-08-26	2.892220
2015-09-02	1.615243
2015-09-09	2.620432
2015-09-16	2.255937
2015-09-23	2.813171

	RandomNumbers
2015-09-30	3.956969
2015-10-07	2.486206
2015-10-14	2.282833
2015-10-21	2.176658
2015-10-28	2.679554
2015-11-04	2.501141
2015-11-11	3.518900
2015-11-18	2.071519
2015-11-25	2.594533
2015-12-02	2.659251
2015-12-09	2.756760
2015-12-16	2.675114
2015-12-23	3.530151
2015-12-30	1.987753
2016-01-06	0.152559

4) Average For each calendar month

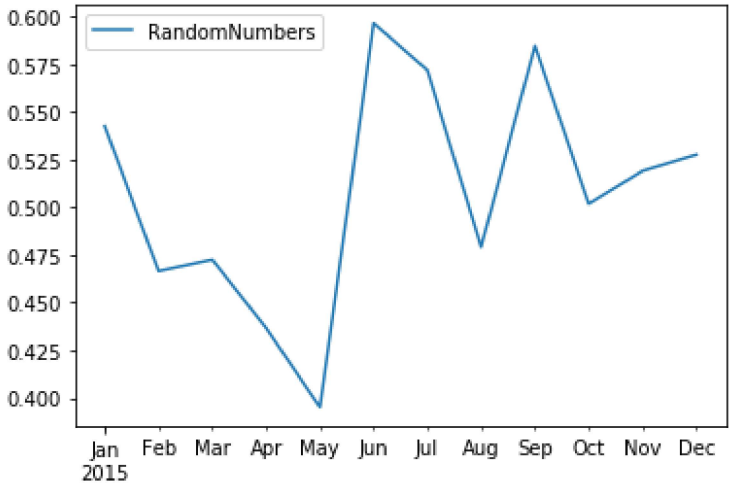
In [5]:
```
df.resample('M').mean()
```

Out[5]:

	RandomNumbers
2015-01-31	0.542525
2015-02-28	0.466507
2015-03-31	0.472481
2015-04-30	0.436730
2015-05-31	0.395145
2015-06-30	0.596535
2015-07-31	0.571826
2015-08-31	0.479210
2015-09-30	0.584672
2015-10-31	0.501792
2015-11-30	0.519057
2015-12-31	0.527535

In [6]:
```
%matplotlib inline
df.resample('M').mean().plot()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1ceb07a5fd0>
```



5) For each group of four consecutive calendar months in s, find the date on which the highest value occurred.

In [7]:
```
df=pandas.DataFrame(data=data,index=index,columns=["RandomNumbers"])
#df.resample('4M',convention='s').max() --> This is a partial code, however it is
df1=pd.DataFrame(columns=["RandomNumbers"])
start_year=df["RandomNumbers"].index[0].year
start_month=df["RandomNumbers"].index[0].month
end_year=df["RandomNumbers"].index[df["RandomNumbers"].count()-1].year
end_month=df["RandomNumbers"].index[df["RandomNumbers"].count()-1].month
curr_year=start_year
curr_month=start_month
prev_curr_year = curr_year
prev_curr_month = curr_month
while curr_year<=end_year and curr_month<=end_month:
    curr_month = curr_month + 4
    if curr_month > 12:
        curr_month = 1
        curr_year = curr_year + 1
    start_range=str(prev_curr_year) + '-' + str(prev_curr_month)
    if curr_month == 1:
        end_range=str(curr_year-1) + '-' + str(12)
    else:
        end_range=str(curr_year) + '-' + str(curr_month-1)
    #print(start_range,":",end_range)
    df2=df[start_range:end_range]
    df2=df2[df2["RandomNumbers"] == df2["RandomNumbers"].max()]
    df1=df1.append(df2)
    prev_curr_year = curr_year
    prev_curr_month = curr_month
df1
```

Out[7]:

	RandomNumbers
2015-01-19	0.996557
2015-07-30	0.999698
2015-12-01	0.999838

In [ ]: