In [1]:

```
"""
1) How-to-count-distance-to-the-previous-zero
For each value, count the difference of the distance from the previous zero (o
of the Series, whichever is closer) and if there are no previous zeros,print t
 Consider a DataFrame df where there is an integer column {'X':[7, 2, 0, 3, 4,
 The values should therefore be [1, 2, 0, 1, 2, 3, 4, 0, 1, 2]. Make this a ne
import pandas as pd
df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
"""

import pandas as pd
import numpy as np


df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
x = (df['X'] != 0).cumsum()
y = x != x.shift()

df['Y'] = y.groupby((y != y.shift()).cumsum()).cumsum()

x = (df['X'] != 0).cumsum()
y = x != x.shift()

df['Y'] = y.groupby((y != y.shift()).cumsum()).cumsum()

print(df)
```

```
   X    Y
0  7  1.0
1  2  2.0
2  0  0.0
3  3  1.0
4  4  2.0
5  2  3.0
6  5  4.0
7  0  0.0
8  3  1.0
9  4  2.0
```

In [2]:
```
"""
Create a DatetimeIndex that contains each business day of 2015 and use it to i
"""

datetimeindex = pd.date_range(start='2015-01-01', end='2015-12-31')
s = pd.Series(np.random.rand(len(datetimeindex)),index=datetimeindex)
print(s)
```

```
2015-01-01    0.451846
2015-01-02    0.944229
2015-01-03    0.360453
2015-01-04    0.820605
2015-01-05    0.444812
2015-01-06    0.741457
2015-01-07    0.612860
2015-01-08    0.559010
2015-01-09    0.015024
2015-01-10    0.286784
2015-01-11    0.909195
2015-01-12    0.880581
2015-01-13    0.466935
2015-01-14    0.862895
2015-01-15    0.451438
2015-01-16    0.110835
2015-01-17    0.670007
2015-01-18    0.200728
2015-01-19    0.074758
2015-01-20    0.632795
2015-01-21    0.390140
2015-01-22    0.490485
2015-01-23    0.285693
2015-01-24    0.971291
2015-01-25    0.726391
2015-01-26    0.721181
2015-01-27    0.327311
2015-01-28    0.939758
2015-01-29    0.500500
2015-01-30    0.563142
                ...
2015-12-02    0.975744
2015-12-03    0.115032
2015-12-04    0.341736
2015-12-05    0.903676
2015-12-06    0.332874
2015-12-07    0.755217
2015-12-08    0.934509
2015-12-09    0.213556
2015-12-10    0.270059
2015-12-11    0.150592
2015-12-12    0.062981
2015-12-13    0.533091
2015-12-14    0.423295
2015-12-15    0.340597
2015-12-16    0.137151
2015-12-17    0.735946
2015-12-18    0.751849
2015-12-19    0.775064
```

```
2015-12-20    0.435587
2015-12-21    0.656655
2015-12-22    0.250608
2015-12-23    0.580641
2015-12-24    0.380445
2015-12-25    0.674857
2015-12-26    0.635313
2015-12-27    0.325243
2015-12-28    0.339970
2015-12-29    0.262100
2015-12-30    0.935564
2015-12-31    0.244298
Freq: D, Length: 365, dtype: float64
```

In [3]:
```
"""
3) Find the sum of the values in s for every Wednesday
"""
s[datetimeindex.weekday_name == 'Wednesday'].sum()
```

Out[3]:
```
25.888337894612206
```

In [4]:
```
"""
4) Average For each calendar month
"""
s.groupby(pd.Grouper(freq='M')).mean()
```

Out[4]:
```
2015-01-31    0.534601
2015-02-28    0.450797
2015-03-31    0.504999
2015-04-30    0.547523
2015-05-31    0.513240
2015-06-30    0.467708
2015-07-31    0.461545
2015-08-31    0.472838
2015-09-30    0.489428
2015-10-31    0.513712
2015-11-30    0.447729
2015-12-31    0.491644
Freq: M, dtype: float64
```

In [5]:
```
"""
5) For each group of four consecutive calendar months in s, find the date on w
"""
s.groupby(pd.Grouper(freq='4M')).max()
```

Out[5]:
```
2015-01-31    0.971291
2015-05-31    0.997163
2015-09-30    0.996816
2016-01-31    0.988635
dtype: float64
```

In [ ]:
```

```