1 How-to-count-distance-to-the-previous-zero For each value, count the difference back to the previous zero (or the start of the Series, whichever is closer) create a new column 'Y' Consider a DataFrame df where there is an integer column 'X'

In [9]:
```
#Import the Packages
import pandas as pd
import numpy as np

#Create a Data frame with the given input.
df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})

#Indices where the Zero in the array.
Indice_zero = np.r_[-1, (df['X'] == 0).nonzero()[0]]

# numpy array with elements
idx = np.arange(len(df))

# To find the indices into a sorted array. so that order will remaine before the
df['Y'] = idx - Indice_zero[np.searchsorted(Indice_zero - 1, idx) - 1]
df
```

Out[9]:

	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 [10]:
```
#Import the Packages
import pandas as pd

datetime_index = pd.date_range(start='2015-01-01', end='2015-12-31', freq='B')
x = pd.Series(np.random.rand(len(datetime_index)), index = datetime_index)
print(x)
```

```
2015-11-30    0.352738
2015-12-01    0.226182
2015-12-02    0.706000
2015-12-03    0.609752
2015-12-04    0.810085
2015-12-07    0.059580
2015-12-08    0.567866
2015-12-09    0.976595
2015-12-10    0.254486
2015-12-11    0.073495
2015-12-14    0.082542
2015-12-15    0.282766
2015-12-16    0.096696
2015-12-17    0.788688
2015-12-18    0.553764
2015-12-21    0.012416
2015-12-22    0.269311
2015-12-23    0.999005
2015-12-24    0.594739
```

1. Find the sum of the values in s for every Wednesday.

In [11]:
```
print(x[x.index.weekday == 2].sum())
```

```
28.862119202167147
```

1. Average For each calendar month

In [12]:
```
print(x.resample('M').mean())
```

```
2015-01-31    0.565538
2015-02-28    0.487172
2015-03-31    0.504879
2015-04-30    0.559887
2015-05-31    0.522871
2015-06-30    0.440166
2015-07-31    0.583680
2015-08-31    0.504864
2015-09-30    0.395847
2015-10-31    0.548270
2015-11-30    0.439723
2015-12-31    0.516117
Freq: M, dtype: float64
```

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

In [13]:
```
print(x.groupby(pd.Grouper(freq='4M')).max())
print(x.groupby(pd.Grouper(freq='4M', closed='left')).max())
```

```
2015-01-31    0.997669
2015-05-31    0.992141
2015-09-30    0.979704
2016-01-31    0.999005
dtype: float64
2015-04-30    0.997669
2015-08-31    0.992141
2015-12-31    0.999005
2016-04-30    0.693773
Freq: 4M, dtype: float64
```

In [ ]: