Problem Statement 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)

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

Out[3]:

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

In [4]:
```
import numpy as np
izero = np.r_[-1, (df['X'] == 0).nonzero()[0]] # indices of zeros -1 is the first
izero
```

Out[4]:
```
array([-1,  2,  7], dtype=int64)
```

In [5]:
```
idx = np.arange(len(df))
idx
```

Out[5]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [6]:
```
df['Y'] = idx - izero[np.searchsorted(izero - 1, idx) - 1] # resetting the values
df
```

Out[6]:

	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 [8]:
```
dtIndex = pd.date_range(start='2015-01-01', end='2015-12-31', freq='B')
dtIndex
```

Out[8]:
```
DatetimeIndex(['2015-01-01', '2015-01-02', '2015-01-05', '2015-01-06',
               '2015-01-07', '2015-01-08', '2015-01-09', '2015-01-12',
               '2015-01-13', '2015-01-14',
               ...
               '2015-12-18', '2015-12-21', '2015-12-22', '2015-12-23',
               '2015-12-24', '2015-12-25', '2015-12-28', '2015-12-29',
               '2015-12-30', '2015-12-31'],
              dtype='datetime64[ns]', length=261, freq='B')
```

In [11]:
```
s = pd.Series(np.random.rand(len(dtIndex)), index=dtIndex)
s
```

Out[11]:
```
2015-01-01    0.408075
2015-01-02    0.339357
2015-01-05    0.701784
2015-01-06    0.551625
2015-01-07    0.717197
2015-01-08    0.943218
2015-01-09    0.986224
2015-01-12    0.367821
2015-01-13    0.531248
2015-01-14    0.329443
2015-01-15    0.717649
2015-01-16    0.616794
2015-01-19    0.961597
2015-01-20    0.506507
2015-01-21    0.018050
2015-01-22    0.415076
2015-01-23    0.643813
2015-01-26    0.479438
2015-01-27    0.104822
2015-01-28    0.479160
2015-01-29    0.244546
2015-01-30    0.705805
2015-02-02    0.652270
2015-02-03    0.766248
2015-02-04    0.867109
2015-02-05    0.530302
2015-02-06    0.569991
2015-02-09    0.597291
2015-02-10    0.274912
2015-02-11    0.491374
                 ...
2015-11-20    0.230635
2015-11-23    0.379858
2015-11-24    0.511312
2015-11-25    0.898540
2015-11-26    0.215027
2015-11-27    0.201587
2015-11-30    0.651728
2015-12-01    0.465443
2015-12-02    0.591186
2015-12-03    0.102234
2015-12-04    0.452400
2015-12-07    0.399892
2015-12-08    0.282958
2015-12-09    0.982159
2015-12-10    0.551244
2015-12-11    0.933416
2015-12-14    0.200326
2015-12-15    0.321567
2015-12-16    0.554946
2015-12-17    0.849588
2015-12-18    0.073914
2015-12-21    0.541381
2015-12-22    0.461799
```

```
2015-12-23    0.246282
2015-12-24    0.878979
2015-12-25    0.014465
2015-12-28    0.771502
2015-12-29    0.524267
2015-12-30    0.087847
2015-12-31    0.897901
Freq: B, Length: 261, dtype: float64
```

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

In [16]:
```
dtIndex.weekday  # check the weekday for wednessday it's 2' start from 0 as Monda
```

Out[16]:
```
Int64Index([3, 4, 0, 1, 2, 3, 4, 0, 1, 2,
            ...
            4, 0, 1, 2, 3, 4, 0, 1, 2, 3],
           dtype='int64', length=261)
```

In [15]:
```
s[dtIndex.weekday == 2].sum()
```

Out[15]:
```
27.210368491125806
```

4) Average For each calendar month

For average need to find the mean

In [17]:
```
s.resample('M', how='mean')
```

```
C:\Users\prashant_gupta1\AppData\Local\Continuum\anaconda3\lib\site-packages\ip
ykernel_launcher.py:1: FutureWarning: how in .resample() is deprecated
the new syntax is .resample(...).mean()
  """Entry point for launching an IPython kernel.
```

Out[17]:
```
2015-01-31    0.534966
2015-02-28    0.601955
2015-03-31    0.434673
2015-04-30    0.465534
2015-05-31    0.436431
2015-06-30    0.484137
2015-07-31    0.542224
2015-08-31    0.484949
2015-09-30    0.543728
2015-10-31    0.518012
2015-11-30    0.535577
2015-12-31    0.486335
Freq: M, dtype: float64
```

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

In [19]:
```
s.groupby(pd.TimeGrouper('4M')).idxmax()
```

```
C:\Users\prashant_gupta1\AppData\Local\Continuum\anaconda3\lib\site-packages\ip
ykernel_launcher.py:1: FutureWarning: pd.TimeGrouper is deprecated and will be
removed; Please use pd.Grouper(freq=...)
  """Entry point for launching an IPython kernel.
```

Out[19]:
```
2015-01-31   2015-01-09
2015-05-31   2015-03-23
2015-09-30   2015-07-10
2016-01-31   2015-12-09
dtype: datetime64[ns]
```