In [ ]:
```
#How-to-count-distance-to-the-previous-zero
For each value, count the difference back to the previous zero (or the start o
f the Series,
whichever is closer)
create a new column 'Y'
Consider a DataFrame df where there is an integer column 'X'
import pandas as pd
df = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
```

In [9]:
```
import pandas as pd

s = pd.Series([7, 2, 0, 3, 4, 2, 5, 0, 3, 4])

(s.groupby(s.eq(0).cumsum().mask(s.eq(0))).cumcount() + 1).mask(s.eq(0), 0).to
list()
```

Out[9]:
```
[1, 2, 0, 1, 2, 3, 4, 0, 1, 2]
```

In [ ]:
```
#Create a DatetimeIndex that contains each business day of 2015 and use it to
 index a
Series of random numbers.
```

In [11]:
```
import numpy as np
dti = pd.date_range(start='2015-01-01', end='2015-12-31', freq='B')
s = pd.Series(np.random.rand(len(dti)), index=dti)
```

In [14]:
```
dti
```

Out[14]:
```
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 [ ]:
```
# Find the sum of the values in s for every Wednesday.
```

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

Out[15]:
```
28.310617362240155
```

In [ ]:
```
#Average For each calendar month
```

In [16]:
```
s.resample('M', how='mean')
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWar
ning: how in .resample() is deprecated
the new syntax is .resample(...).mean()
  """Entry point for launching an IPython kernel.
```

Out[16]:
```
2015-01-31    0.534815
2015-02-28    0.535890
2015-03-31    0.501440
2015-04-30    0.460362
2015-05-31    0.523951
2015-06-30    0.560641
2015-07-31    0.449447
2015-08-31    0.420447
2015-09-30    0.469834
2015-10-31    0.537100
2015-11-30    0.502354
2015-12-31    0.622382
Freq: M, dtype: float64
```

In [ ]:
```
#For each group of four consecutive calendar months in s, find the date on whi
ch the
highest value occurred.
```

In [17]:
```
s.groupby(pd.TimeGrouper('4M')).idxmax()
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWar
ning: pd.TimeGrouper is deprecated and will be removed; Please use pd.Grouper
(freq=...)
  """Entry point for launching an IPython kernel.
```

Out[17]:
```
2015-01-31   2015-01-07
2015-05-31   2015-05-14
2015-09-30   2015-07-07
2016-01-31   2015-10-09
dtype: datetime64[ns]
```