In [83]:
```
# 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 n
ew column 'Y'.

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

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

#Converting the series value to boolean by using 'eq() to have a 0 and 1 assig
ned to each value for additions
#Using 'cumsum' to get a cumulative sum of values
#Masking the original zeros using 'mask'
#Taking the cumulative count of the values in list using 'cumcount' and adding
 1 to each element
#Masking the original zeros again using 'eq()', replacing it with 0

(S.groupby(S.eq(0).cumsum().mask(S.eq(0))).cumcount() + 1).mask(S.eq(0),0)

#Naming the equation for converting it into a dataframe, naming it to keep it
 simple instead of writing complete eq
Result=(S.groupby(S.eq(0).cumsum().mask(S.eq(0))).cumcount() + 1).mask(S.eq(0
),0)

#creating a new column 'Y'
pd.DataFrame((Result),columns=['Y'])
```

Out[83]:

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

In [84]:
```
# 2) Create a DatetimeIndex that contains each business day of 2015 and use it
 to index a Series of random numbers.

import numpy as np
import pandas as pd

dates = pd.date_range('2015-01-01', '2015-12-31')
s = pd.Series(np.random.rand(dates.shape[0]), index = dates)
print('-----2015 Calendar-----\n',s)
```

```
-----2015 Calendar-----
 2015-01-01    0.901347
2015-01-02    0.005279
2015-01-03    0.377093
2015-01-04    0.474964
2015-01-05    0.317027
2015-01-06    0.041047
2015-01-07    0.127023
2015-01-08    0.147281
2015-01-09    0.169016
2015-01-10    0.535531
2015-01-11    0.146264
2015-01-12    0.431393
2015-01-13    0.217383
2015-01-14    0.174168
2015-01-15    0.277505
2015-01-16    0.574736
2015-01-17    0.362893
2015-01-18    0.764535
2015-01-19    0.829257
2015-01-20    0.342969
2015-01-21    0.116879
2015-01-22    0.035998
2015-01-23    0.327633
2015-01-24    0.851477
2015-01-25    0.085298
2015-01-26    0.276594
2015-01-27    0.756303
2015-01-28    0.020647
2015-01-29    0.083199
2015-01-30    0.066111
                 ...
2015-12-02    0.553937
2015-12-03    0.055285
2015-12-04    0.892508
2015-12-05    0.016420
2015-12-06    0.299073
2015-12-07    0.969998
2015-12-08    0.366197
2015-12-09    0.937445
2015-12-10    0.366431
2015-12-11    0.027302
2015-12-12    0.574812
2015-12-13    0.789879
2015-12-14    0.017903
2015-12-15    0.318742
2015-12-16    0.149610
2015-12-17    0.324010
2015-12-18    0.585829
2015-12-19    0.618614
2015-12-20    0.714349
2015-12-21    0.966170
2015-12-22    0.938273
2015-12-23    0.012786
2015-12-24    0.363698
2015-12-25    0.690529
2015-12-26    0.538996
```

```
2015-12-27    0.950599
2015-12-28    0.328747
2015-12-29    0.899684
2015-12-30    0.793967
2015-12-31    0.065116
Freq: D, Length: 365, dtype: float64
```

In [85]:
```
# 3) Find the sum of the values in s for every Wednesday

#Identifying wednesday using the day of the week with Monday=0 and sunday as
 6, so Wednesday would be 2
#using round to roundoff the decimals to 2 places
print('The sum of all Wednesdays is - ', round(s[s.index.dayofweek ==2].sum(),
2))
```

```
The sum of all Wednesdays is -  24.53
```

In [86]:
```
# 4) Average For each calendar month

# Define list of months
months = ['Jan-15', 'Feb-15', 'Mar-15', 'Apr-15', 'May-15', 'Jun-15', 'Jul-15'
, 'Aug-15',
              'Sep-15', 'Oct-15','Nov-15', 'Dec-15']

# Calculating monthly average by using groupby month on s and identifying th a
verage using mean
monthly_average = s.groupby(s.index.month).mean()

# indexing the mean with the month_list
monthly_average.index = months
print('Please refer to the below table for monthly averages:\n', monthly_avera
ge)
```

```
Please refer to the below table for monthly averages:
 Jan-15    0.332495
Feb-15    0.456637
Mar-15    0.541303
Apr-15    0.532801
May-15    0.525059
Jun-15    0.560563
Jul-15    0.527409
Aug-15    0.467371
Sep-15    0.539121
Oct-15    0.506336
Nov-15    0.499106
Dec-15    0.492909
dtype: float64
```

In [87]:
```
# 5) For each group of four consecutive calendar months in s, find the date on
 which the highest value occurred.

# Creating a list of four consecutive months using for loop with the list 'mon
ths' created above
Four_consecutive_months = [months[i] + "-" + months[i + 3] for i in range(9)]

#Identifying the ids with maximum value for a group of four consecutive months
 for the year.
#range is used as 1,10 since there are 9 such groups
maximum_date_values = [(s[(s.index.month >= i) & (s.index.month <= (i+3))]).id
xmax()
                       for i in range(1,10)]

#Creating a series using the dates with maximum value indexed with 'Four_conse
cutive_months' list.
maximum_date_series = pd.Series(maximum_date_values, index = Four_consecutive_
months)
print('Maximum random values for dates for the year 2015 based on series s -\n
 ', maximum_date_series)
```

```
Maximum random values for dates for the year 2015 based on series s -
  Jan-15-Apr-15   2015-03-09
Feb-15-May-15   2015-05-22
Mar-15-Jun-15   2015-05-22
Apr-15-Jul-15   2015-05-22
May-15-Aug-15   2015-08-21
Jun-15-Sep-15   2015-08-21
Jul-15-Oct-15   2015-08-21
Aug-15-Nov-15   2015-08-21
Sep-15-Dec-15   2015-10-19
dtype: datetime64[ns]
```