Data Science Masters :Assignment 8

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 new column 'Y'.

In [49]:

```
# Solution:
import pandas as pd
df = pd.DataFrame({'X':[7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})  # Given Input Series...
pos=1
outList = []
for num in df['X']:
    if(num == 0):
        outList.append(num)
        pos=1
    else:
        outList.append(pos)
        pos +=1
df['Y'] = outList
df
```

Out[49]:

	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 [38]:

```
# Solution...
import pandas as pd
import numpy as np
np.random.seed(0)   # Setting seed value to generate the same random series....
businessDays = pd.bdate_range('2015-01-01', '2015-12-31')  # Business days from 01/2015 to
randomNumbers = np.random.randint(10,1000,len(businessDays)) # Generating Random number fro
s = pd.Series(randomNumbers,businessDays)
s
```

Out[38]:

```
2015-01-01    694
2015-01-02    569
2015-01-05    639
2015-01-06    202
2015-01-07    845
2015-01-08    773
2015-01-09    717
2015-01-12    369
2015-01-13     19
2015-01-14    733
2015-01-15    287
2015-01-16    764
2015-01-19    814
2015-01-20    609
2015-01-21     80
2015-01-22    482
2015-01-23    610
2015-01-26    406
2015-01-27    324
2015-01-28    715
2015-01-29    496
2015-01-30    561
2015-02-02     97
2015-02-03    184
2015-02-04    610
2015-02-05    859
2015-02-06    687
2015-02-09    547
2015-02-10    855
2015-02-11     82
              ...
2015-11-20     35
2015-11-23    474
2015-11-24    966
2015-11-25    899
2015-11-26    896
2015-11-27    127
2015-11-30    455
2015-12-01    605
2015-12-02    683
2015-12-03    882
2015-12-04    938
2015-12-07    238
2015-12-08    517
2015-12-09    773
2015-12-10    131
2015-12-11    336
2015-12-14    479
```

```
2015-12-15    297
2015-12-16    535
2015-12-17    849
2015-12-18    706
2015-12-21    162
2015-12-22    601
2015-12-23     51
2015-12-24    284
2015-12-25    562
2015-12-28    448
2015-12-29    217
2015-12-30    789
2015-12-31    176
Freq: B, Length: 261, dtype: int32
```

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

In [39]:

```
# Solution...
print("Sum of the values in Series (s) for every Wednesday =",s[s.index.weekday == 2].sum()
```

```
Sum of the values in Series (s) for every Wednesday = 26249
```

4) Average For each calendar month

In [40]:

```
# Solution...
print("Avg for each month in Series (s)->")
s.groupby([lambda x: x.month]).mean().round(2)
```

```
Avg for each month in Series (s)->
```

Out[40]:

```
1     532.18
2     597.40
3     538.32
4     458.45
5     516.95
6     539.82
7     593.83
8     479.33
9     361.82
10    496.45
11    562.62
12    489.52
dtype: float64
```

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

In [53]:

```
# Solution...
def custom_groupby(index):  # function to split the given Series (s) into 3 groups G1,G2,G3
    monthPosition = index.month
    if (monthPosition < 5):
        return "G1"
    elif (monthPosition > 4 and monthPosition < 9):
        return "G2"
    elif (monthPosition > 8):
        return "G3"
grouped = s.groupby(custom_groupby)
maxValueGrp = grouped.max()              # Maximum value in each group
dateOfOccurance = grouped.idxmax().map(lambda t: t.strftime('%Y-%m-%d')) # Converting times
print(list(zip(dateOfOccurance,maxValueGrp)))  # Printing Max value with the date for each
```

```
[('2015-02-27', 994), ('2015-07-16', 983), ('2015-11-02', 973)]
```