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'.

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

Functional Approach

In [24]:

```
import pandas as pd
import numpy as np
```

In [12]:

```
def complex(df):
    res=[]
    flag1=0

    ix=df.index[df['X'] == 0].tolist()

    for i in df.index.values:
        for j in ix:
            if i < j and flag1==0 :
                res.append(i+1)
                break

            elif i == j and flag1==0:
                res.append(0)
                break

            elif i > j:
                flag=j
                flag1=1

        if flag1==1:
            if df.iloc[i][0] == 0:
                res.append(0)
            else:
                res.append(i-flag)

    df['Y'] = res
    return df
```

In [13]:

```
frame1 = pd.DataFrame({'X': [7, 2, 0, 3, 4, 2, 5, 0, 3, 4]})
frame_out1=complex(frame1)
frame_out1
```

Out[13]:

	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

In [14]:

```
frame2 = pd.DataFrame({'X': [1,7, 2, 0, 3, 4, 2, 3, 5, 0, 3, 4]})
frame_out2=complex(frame2)
frame_out2
```

Out[14]:

	X	Y
0	1	1
1	7	2
2	2	3
3	0	0
4	3	1
5	4	2
6	2	3
7	3	4
8	5	5
9	0	0
10	3	1
11	4	2

Non function approach

In [9]:

```
df=pd.DataFrame({'X': [7, 2, 0, 3, 0, 4, 2, 5, 0, 3, 4]})
ix=df.index[df['X'] == 0].tolist()

res=[]
flag1=0

for i in df.index.values:
    for j in ix:
        if i < j and flag1==0 :
            res.append(i+1)
            break

        elif i == j and flag1==0:
            res.append(0)
            break

        elif i > j:
            flag=j
            flag1=1

    if flag1==1:
        if df.iloc[i][0] == 0:
            res.append(0)
        else:
            res.append(i-flag)

df['Y'] = res
```

In [10]:

```
df
```

Out[10]:

	X	Y
0	7	1
1	2	2
2	0	0
3	3	1
4	0	0
5	4	1
6	2	2
7	5	3
8	0	0
9	3	1
10	4	2

2. Create a DatetimeIndex that contains each business day of 2015 and use it to index a series of random numbers.

Find the sum of the values in s for every Wednesday
Average For each calendar month
For each group of four consecutive calendar months in s, find the date on whic
h the highest value occurred

In [15]:

```
myindex=pd.date_range('1/1/2015','31/12/2015')
```

In [16]:

```
import random
mydata=[]
for i in range(1,366):
    mydata.append(random.randint(1,101))
```

In [18]:

```
s=pd.Series(data=mydata,index=myindex)
```

Find the sum of the values in s for every Wednesday

In [23]:

```
s[s.index.to_datetime().weekday_name =='Wednesday'].sum()
```

Out[23]:

```
2698
```

Average For each calendar month

In [20]:

```
s.groupby(s.index.to_datetime().month).mean()
```

Out[20]:

```
1     49.548387
2     48.321429
3     41.354839
4     43.000000
5     39.290323
6     35.766667
7     54.258065
8     54.000000
9     54.100000
10    55.290323
11    47.966667
12    47.419355
dtype: float64
```

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

In [21]:

```
s.groupby(pd.TimeGrouper('4M')).max()
```

Out[21]:

```
2015-01-31    101
2015-05-31    101
2015-09-30     97
2016-01-31    100
Freq: 4M, dtype: int64
```