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]})

**Code:**

import pandas as pd

import numpy as np

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

izero = np.r\_[-1, (df['X'] == 0).nonzero()[0]] # indices of zeros

idx = np.arange(len(df))

df['Y'] = idx - izero[np.searchsorted(izero - 1, idx) - 1]

print(df['Y'])

**Output:**

0 1

1 2

2 0

3 1

4 2

5 3

6 4

7 0

8 1

9 2

Name: Y, dtype: int64

2) Create a DatetimeIndex that contains each business day of 2015 and use it to index a

Series of random numbers.

**Code:**

import pandas as pd

dti = pd.date\_range(start='2015-01-01', end='2015-12-31', freq='B')

print(dti)

print('\n\n')

#Index the series, s be the index

s = pd.Series(np.random.rand(len(dti)), index=dti)

print(s)

**Output:**

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')

2015-01-01 0.280242

2015-01-02 0.552706

2015-01-05 0.852426

2015-01-06 0.562905

2015-01-07 0.585875

2015-01-08 0.484258

2015-01-09 0.902398

2015-01-12 0.343157

2015-01-13 0.776381

2015-01-14 0.900921

2015-01-15 0.629210

2015-01-16 0.586195

2015-01-19 0.578564

2015-01-20 0.782542

2015-01-21 0.784371

2015-01-22 0.392744

2015-01-23 0.433930

2015-01-26 0.490754

2015-01-27 0.836696

2015-01-28 0.486445

2015-01-29 0.405107

2015-01-30 0.322072

2015-02-02 0.290147

2015-02-03 0.358461

2015-02-04 0.168942

2015-02-05 0.585980

2015-02-06 0.327918

2015-02-09 0.827112

2015-02-10 0.277477

2015-02-11 0.819430

...

2015-11-20 0.725139

2015-11-23 0.310721

2015-11-24 0.867789

2015-11-25 0.015982

2015-11-26 0.643568

2015-11-27 0.906863

2015-11-30 0.536162

2015-12-01 0.949486

2015-12-02 0.245943

2015-12-03 0.020938

2015-12-04 0.176345

2015-12-07 0.799896

2015-12-08 0.819685

2015-12-09 0.951056

2015-12-10 0.043044

2015-12-11 0.929727

2015-12-14 0.599806

2015-12-15 0.782314

2015-12-16 0.809454

2015-12-17 0.509283

2015-12-18 0.204521

2015-12-21 0.821563

2015-12-22 0.879348

2015-12-23 0.762870

2015-12-24 0.461678

2015-12-25 0.624431

2015-12-28 0.463589

2015-12-29 0.353559

2015-12-30 0.291333

2015-12-31 0.178086

Freq: B, Length: 261, dtype: float64

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

**Code:**

s[dti.weekday == 2].sum()

**Output:**

26.814802987731042

4) Average For each calendar month

**Code:**

s.resample('M').mean()

**Output:**

2015-01-31 0.589541

2015-02-28 0.454435

2015-03-31 0.516619

2015-04-30 0.540800

2015-05-31 0.581563

2015-06-30 0.578953

2015-07-31 0.563426

2015-08-31 0.550628

2015-09-30 0.539029

2015-10-31 0.503502

2015-11-30 0.483782

2015-12-31 0.551215

Freq: M, dtype: float64

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

highest value occurred.

**Code:**

s.groupby(pd.Grouper(freq='4M')).idxmax()

**Output:**

2015-01-31 2015-01-09

2015-05-31 2015-04-16

2015-09-30 2015-08-24

2016-01-31 2015-10-23

dtype: datetime64[ns]