利用Python中的pandas(date_range)库生成时间序列(time series)

python万
发布时间：01-09 17:15

在讲pandas时间序列函数之前，我大概介绍下什么是时间序列（time series）。时间序列（time series）简单的说就是各时间点上形成的数值序列，时间序列（time series）分析就是通过观察历史数据预测未来的值。比如股票预测、房价预测分析等。本篇文章主要详细讲解生成时间索引的函数date_range及延伸函数。

pandas.date_range

pandas.date_range()这个函数主要是返回固定频率的时间索引，参数比较多，下面我们依次演示常用的参数用法。

- 根据指定的起止时间，生成时间序列

import pandas as pdpd.date_range(start='2019-1-09', end='2019-1-31')

```
pd.date_range(start='2019-1-09', end='2019-1-31')

DatetimeIndex(['2019-01-09', '2019-01-10', '2019-01-11', '2019-01-12',
               '2019-01-13', '2019-01-14', '2019-01-15', '2019-01-16',
               '2019-01-17', '2019-01-18', '2019-01-19', '2019-01-20',
               '2019-01-21', '2019-01-22', '2019-01-23', '2019-01-24',
               '2019-01-25', '2019-01-26', '2019-01-27', '2019-01-28',
               '2019-01-29', '2019-01-30', '2019-01-31'],
              dtype='datetime64[ns]', freq='D')
```


根据起止时间生成

- 根据起止时间，并指定时间序列数量

pd.date_range(start='2019-1-09', end='2019-1-10',periods=10)

```
pd.date_range(start='2019-1-09', end='2019-1-10',periods=10)

DatetimeIndex(['2019-01-09 00:00:00', '2019-01-09 02:40:00',
               '2019-01-09 05:20:00', '2019-01-09 08:00:00',
               '2019-01-09 10:40:00', '2019-01-09 13:20:00',
               '2019-01-09 16:00:00', '2019-01-09 18:40:00',
               '2019-01-09 21:20:00', '2019-01-10 00:00:00'],
              dtype='datetime64[ns]', freq=None)
```


根据指定数量生成

- 根据开始时间和指定数量生成

pd.date_range(start='2019-1-09',periods=10)

```
pd.date_range(start='2019-1-09',periods=10)

DatetimeIndex(['2019-01-09', '2019-01-10', '2019-01-11', '2019-01-12',
               '2019-01-13', '2019-01-14', '2019-01-15', '2019-01-16',
               '2019-01-17', '2019-01-18'],
              dtype='datetime64[ns]', freq='D')
```


根据开始时间和periods生成

作者最新文章

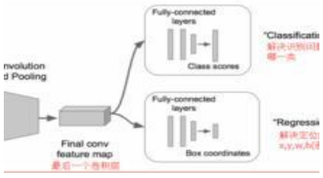
时间序列的平稳性及使用差分法处理非平稳时间序列

python时间序列分析之_用pandas中的rolling函数计算时间窗口数据

Python时间序列分析之_时间重采样（降采样和升采样）

相关文章

CNN——卷积神经网络从R-CNN到Faster R-CNN的理解



Spring中对LookUp注解的处理

如何简单有效地提高代码质量？修改变量名即可！

Python3.7知其然知其所以然-第十章 for循环

你真的理解 Integer 的缓存问题

```
pd.date_range(start='2019-1-09',periods=10,freq='H')
```

```
DatetimeIndex(['2019-01-09 00:00:00', '2019-01-09 01:00:00',
               '2019-01-09 02:00:00', '2019-01-09 03:00:00',
               '2019-01-09 04:00:00', '2019-01-09 05:00:00',
               '2019-01-09 06:00:00', '2019-01-09 07:00:00',
               '2019-01-09 08:00:00', '2019-01-09 09:00:00'],
              dtype='datetime64[ns]', freq='H')
```


指定以小时为频率

比较上面可以看出，date_range中默认以天为频率，如果我们需要其他单位的频率必须用freq这个参数指定，并且可以是基础频率的倍数，如下：

pd.date_range(start='2019-1-09',periods=10,freq='12H')

```
pd.date_range(start='2019-1-09',periods=10,freq='12H')
```

```
DatetimeIndex(['2019-01-09 00:00:00', '2019-01-09 12:00:00',
               '2019-01-10 00:00:00', '2019-01-10 12:00:00',
               '2019-01-11 00:00:00', '2019-01-11 12:00:00',
               '2019-01-12 00:00:00', '2019-01-12 12:00:00',
               '2019-01-13 00:00:00', '2019-01-13 12:00:00'],
              dtype='datetime64[ns]', freq='12H')
```


freq=12H

这里可选的频率有很多，大家在使用的时候查看官方文档即可，这里不在一一举例，附一张官方文档中的图。

Alias	Description
B	business day frequency
C	custom business day frequency
D	calendar day frequency
W	weekly frequency
M	month end frequency
SM	semi-month end frequency (15th and end of month)
BM	business month end frequency
CBM	custom business month end frequency
MS	month start frequency
SMS	semi-month start frequency (1st and 15th)
BMS	business month start frequency
CBMS	custom business month start frequency
Q	quarter end frequency
BQ	business quarter end frequency
QS	quarter start frequency
BQS	business quarter start frequency
A, Y	year end frequency
BA, BY	business year end frequency
AS, YS	year start frequency
BAS, BYS	business year start frequency
BH	business hour frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

频率可选值

pd.date_range(start='2019-01-09', end='2019-01-14', closed=None)pd.date_range(start='2019-01-09', end='2019-01-14', closed='left')pd.date_range(start='2019-01-09', end='2019-01-14', closed='right')

```
pd.date_range(start='2019-01-09', end='2019-01-14', closed=None)
```

```
DatetimeIndex(['2019-01-09', '2019-01-10', '2019-01-11', '2019-01-12',
               '2019-01-13', '2019-01-14'],
              dtype='datetime64[ns]', freq='D')
```

```
pd.date_range(start='2019-01-09', end='2019-01-14', closed='left')
```

```
DatetimeIndex(['2019-01-09', '2019-01-10', '2019-01-11', '2019-01-12',
               '2019-01-13'],
              dtype='datetime64[ns]', freq='D')
```

```
pd.date_range(start='2019-01-09', end='2019-01-14', closed='right')
```

```
DatetimeIndex(['2019-01-10', '2019-01-11', '2019-01-12', '2019-01-13',
               '2019-01-14'],
              dtype='datetime64[ns]', freq='D')
```

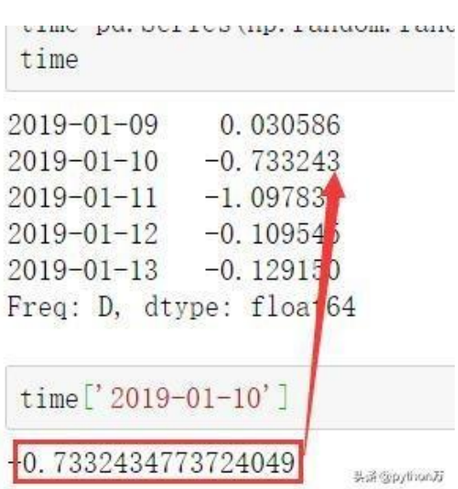

closed空值起止时间

- 时间序列作为索引并根据索引取值

```
index=pd.date_range(start='2019-01-09', end='2019-01-13')
time=pd.Series(np.random.randn(5),index=index)
time
```

```
2019-01-09    0.030586
2019-01-10   -0.733243
2019-01-11   -1.097832
2019-01-12   -0.109545
2019-01-13   -0.129150
Freq: D, dtype: float64
```


生成时间序列



根据时间索引取值

truncate过滤

time.truncate(before='2019-01-12')

可以看到truncate这个函数将before指定日期之前的值全部过滤出去。既然有before，那么就有after，如下：

time.truncate(after='2019-01-12')

```
2019-01-09    0.030586
2019-01-10   -0.733243
2019-01-11   -1.097832
2019-01-12   -0.109545
Freq: D, dtype: float64
```


过滤after之后的数据

pandas.Timestamp

这个类主要是用来生成时间戳的，如下：

pd.Timestamp('2019-01-10')

```
pd.Timestamp('2019-01-10')
Timestamp('2019-01-10 00:00:00')
```


pandas.Timedelta

pd.Timedelta('2day')

pd.Timestamp('2019-01-10')+pd.Timedelta('2day')