

# **Python for Data Analysis**

# Time Series

# Time Series Data

- *Timestamps*, specific instants in time
- Fixed *periods*, such as the month January 2007 or the full year 2010
- *Intervals* of time, indicated by a start and end timestamp. Periods can be thought of as special cases of intervals
- Experiment or elapsed time; each timestamp is a measure of time relative to a particular start time. For example, the diameter of a cookie baking each second since being placed in the oven

# Basic Settings

```
1 import numpy as np
2 import pandas as pd
3 np.random.seed(12345)
4 import matplotlib.pyplot as plt
5 plt.rc('figure', figsize=(10, 6))
6 PREVIOUS_MAX_ROWS = pd.options.display.max_rows
7 pd.options.display.max_rows = 20
8 np.set_printoptions(precision=4, suppress=True)
```

# Date and Time Data Types and Tools

```
1 from datetime import datetime
2 now = datetime.now()
3 now
4 now.year, now.month, now.day
```

(2019, 3, 1)

# Date and Time Data Types and Tools

```
1 delta = datetime(2011, 1, 7) - datetime(2008, 6, 24, 8, 15)
2 delta
3 delta.days
4 delta.seconds
```

56700

# Date and Time Data Types and Tools

```
1 from datetime import timedelta
2 start = datetime(2011, 1, 7)
3 start + timedelta(12)
4 start - 2 * timedelta(12)
```

```
datetime.datetime(2010, 12, 14, 0, 0)
```

# Date and Time Data Types and Tools

Types in datetime module

Type	Description
<code>date</code>	Store calendar date (year, month, day) using the Gregorian calendar.
<code>time</code>	Store time of day as hours, minutes, seconds, and microseconds
<code>datetime</code>	Stores both date and time
<code>timedelta</code>	Represents the difference between two datetime values (as days, seconds, and microseconds)

---



# Converting Between String and Datetime

```
1 stamp = datetime(2011, 1, 3)
2 str(stamp)
3 stamp.strftime('%Y-%m-%d')
```

'2011-01-03'

# Converting Between String and Datetime

Datetime format specification (ISO C89 compatible)

Type	Description
%Y	4-digit year
%y	2-digit year
%m	2-digit month [01, 12]
%d	2-digit day [01, 31]
%H	Hour (24-hour clock) [00, 23]
%I	Hour (12-hour clock) [01, 12]
%M	2-digit minute [00, 59]
%S	Second [00, 61] (seconds 60, 61 account for leap seconds)
%w	Weekday as integer [0 (Sunday), 6]

# Converting Between String and Datetime

Datetime format specification (ISO C89 compatible)

Type	Description
%U	Week number of the year [00, 53]. Sunday is considered the first day of the week, and days before the first Sunday of the year are "week 0".
%W	Week number of the year [00, 53]. Monday is considered the first day of the week, and days before the first Monday of the year are "week 0".
%z	UTC time zone offset as +HHMM or -HHMM, empty if time zone naive
%F	Shortcut for %Y-%m-%d, for example 2012-4-18
%D	Shortcut for %m/%d/%y, for example 04/18/12

# Converting Between String and Datetime

```
1 value = '2011-01-03'
2 datetime.strptime(value, '%Y-%m-%d')
3 datestrs = ['2011/7/6', '2011/8/6']
4 [datetime.strptime(x, '%Y/%m/%d') for x in datestrs]
```

```
[datetime.datetime(2011, 7, 6, 0, 0), datetime.datetime(2011, 8, 6,
```

# Converting Between String and Datetime

```
1 from dateutil.parser import parse
2 parse('2011-01-03')
```

```
datetime.datetime(2011, 1, 3, 0, 0)
```

```
1 parse('Jan 31, 1997 10:45 PM')
```

```
datetime.datetime(1997, 1, 31, 22, 45)
```

# Converting Between String and Datetime

```
1 datestrs = ['2011-07-06 12:00:00', '2011-08-06 00:00:00']  
2 pd.to_datetime(datestrs)
```

```
DatetimeIndex(['2011-07-06 12:00:00', '2011-08-06 00:00:00'], dtype='datetime64[ns]',  
freq=None)
```

```
1 idx = pd.to_datetime(datestrs + [None])  
2 idx
```

```
DatetimeIndex(['2011-07-06', '2011-08-06', 'NaT'], dtype='datetime64[ns]', freq=None)
```

# Converting Between String and Datetime

```
1 idx[2]
```

NaT

```
1 pd.isnull(idx)
```

array([False, False, True])

# Converting Between String and Datetime

Locale-specific date formatting

Type	Description
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%c	Full date and time, for example 'Tue 01 May 2012 04:20:57 PM'
%p	Locale equivalent of AM or PM
%x	Locale-appropriate formatted date; e.g. in US May 1, 2012 yields '05/01/2012'
%X	Locale-appropriate time, e.g. '04:24:12 PM'



# Time Series Basics

```
1 from datetime import datetime
2 dates = [datetime(2011, 1, 2), datetime(2011, 1, 5),
3          datetime(2011, 1, 7), datetime(2011, 1, 8),
4          datetime(2011, 1, 10), datetime(2011, 1, 12)]
5 ts = pd.Series(np.random.randn(6), index=dates)
6 ts
```

```
2011-01-02    -1.078419
2011-01-05     2.213262
2011-01-07     0.447194
2011-01-08    -0.099447
2011-01-10    -0.573992
2011-01-12     1.727834
dtype: float64
```

# Time Series Basics

```
1 ts.index
```

```
DatetimeIndex(['2011-01-02', '2011-01-05', '2011-01-07', '2011-01-08',  
               '2011-01-10', '2011-01-12'],  
              dtype='datetime64[ns]', freq=None)
```

```
1 ts + ts[::-2]
```

```
2011-01-02    -2.156838  
2011-01-05         NaN  
2011-01-07     0.894388  
2011-01-08         NaN  
2011-01-10    -1.147984  
2011-01-12         NaN  
dtype: float64
```

# Time Series Basics

```
1 ts.index.dtype
```

```
dtype('<M8[ns]')
```

```
1 stamp = ts.index[0]
```

```
2 stamp
```

```
Timestamp('2011-01-02 00:00:00')
```

# Indexing, Selection, Subsetting

```
1 stamp = ts.index[2]  
2 ts[stamp]
```

0.4471938539950634

```
1 ts['2011/1/10']  
2 ts['20110110']
```

-0.5739919980605656

# Indexing, Selection, Subsetting

```
1 longer_ts = pd.Series(np.random.randn(1000),  
2                        index=pd.date_range('2000/1/1', periods=1000))  
3 longer_ts  
4 longer_ts['2001']
```

2001-01-01 -0.151545

2001-01-02 0.401587

2001-01-03 -2.223506

2001-01-04 -0.574654

2001-01-05 0.786210

...

2001-12-29 0.405850

2001-12-30 0.680251

2001-12-31 1.357221

Freq: D, Length: 365, dtype: float64

# Indexing, Selection, Subsetting

```
1 longer_ts['2001-05']
```

```
2001-05-01    0.734799
```

```
2001-05-02   -0.163056
```

```
2001-05-03    0.196350
```

```
2001-05-04    0.727743
```

```
2001-05-05    0.161154
```

```
...
```

```
2001-05-28   -0.548465
```

```
2001-05-29    0.909617
```

```
2001-05-30    1.475824
```

```
2001-05-31    0.584234
```

```
Freq: D, dtype: float64
```

# Indexing, Selection, Subsetting

```
1 ts[datetime(2011, 1, 7):]
```

```
2011-01-07    -0.519439
```

```
2011-01-08    -0.555730
```

```
2011-01-10     1.965781
```

```
2011-01-12     1.393406
```

```
dtype: float64
```

# Indexing, Selection, Subsetting

1	ts
---	----

```
2011-01-02    -1.078419
2011-01-05     2.213262
2011-01-07     0.447194
2011-01-08    -0.099447
2011-01-10    -0.573992
2011-01-12     1.727834
dtype: float64
```

1	ts['2011/1/6':'2011/1/11']
---	----------------------------

```
2011-01-07     0.447194
2011-01-08    -0.099447
2011-01-10    -0.573992
dtype: float64
```



# Indexing, Selection, Subsetting

```
1 ts.truncate(after='2011/1/9')
```

```
2011-01-02    -1.078419
```

```
2011-01-05     2.213262
```

```
2011-01-07     0.447194
```

```
2011-01-08    -0.099447
```

```
dtype: float64
```

# Indexing, Selection, Subsetting

```
1 dates = pd.date_range('2000/1/1', periods=100, freq='W-WED')
2 long_df = pd.DataFrame(np.random.randn(100, 4),
3                          index=dates,
4                          columns=['Colorado', 'Texas',
5                                  'New York', 'Ohio'])
6 long_df.loc['2001-5']
```

	Colorado	Texas	New York	Ohio
<b>2001-05-02</b>	-0.661257	1.134968	0.060154	-0.630441
<b>2001-05-09</b>	-2.303203	1.135953	0.039481	1.492634
<b>2001-05-16</b>	0.219628	-2.433629	-0.071871	-1.438643
<b>2001-05-23</b>	-0.963054	0.689989	-1.173536	0.536754
<b>2001-05-30</b>	0.745970	-0.169002	-0.585304	-0.793187

# Time Series with Duplicate Indices

```
1 dates = pd.DatetimeIndex(['2000/1/1', '2000/1/2', '2000/1/2',  
2                             '2000/1/2', '2000/1/3'])  
3 dup_ts = pd.Series(np.arange(5), index=dates)  
4 dup_ts
```

```
2000-01-01    0  
2000-01-02    1  
2000-01-02    2  
2000-01-02    3  
2000-01-03    4  
dtype: int32
```

# Time Series with Duplicate Indices

```
1 dup_ts.index.is_unique
```

False

```
1 dup_ts['2000/1/3'] # not duplicated
```

4

```
1 dup_ts['2000/1/2'] # duplicated
```

```
2000-01-02    1
2000-01-02    2
2000-01-02    3
dtype: int32
```

# Time Series with Duplicate Indices

```
1 grouped = dup_ts.groupby(level=0)
2 grouped.mean()
```

```
2000-01-01    0
2000-01-02    2
2000-01-03    4
dtype: int32
```

```
1 grouped.count()
```

```
2000-01-01    1
2000-01-02    3
2000-01-03    1
dtype: int64
```

# Date Ranges, Frequencies, and Shifting

```
1 ts
```

```
2011-01-02    -1.078419
2011-01-05     2.213262
2011-01-07     0.447194
2011-01-08    -0.099447
2011-01-10    -0.573992
2011-01-12     1.727834
dtype: float64
```

```
1 resampler = ts.resample('D')
2 resampler
```

```
DatetimeIndexResampler [freq=<Day>, axis=0, closed=left, label=left, convention=start, base=0]
```

# Generating Date Ranges

```
1 index = pd.date_range('2012-04-01', '2012-06-01')
2 index
```

```
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               ...,
               '2012-05-19', '2012-05-20', '2012-05-21', '2012-05-22',
               '2012-05-23', '2012-05-24', '2012-05-25', '2012-05-26',
               '2012-05-27', '2012-05-28', '2012-05-29', '2012-05-30',
               '2012-05-31', '2012-06-01'],
              dtype='datetime64[ns]', freq='D')
```

# Generating Date Ranges

```
1 pd.date_range(start='2012-04-01', periods=20)
```

```
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',  
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',  
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',  
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',  
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20'],  
              dtype='datetime64[ns]', freq='D')
```



# Generating Date Ranges

```
1 pd.date_range(end='2012-06-01', periods=20)
```

```
DatetimeIndex(['2012-05-13', '2012-05-14', '2012-05-15', '2012-05-16',  
               '2012-05-17', '2012-05-18', '2012-05-19', '2012-05-20',  
               '2012-05-21', '2012-05-22', '2012-05-23', '2012-05-24',  
               '2012-05-25', '2012-05-26', '2012-05-27', '2012-05-28',  
               '2012-05-29', '2012-05-30', '2012-05-31', '2012-06-01'],  
              dtype='datetime64[ns]', freq='D')
```

# Generating Date Ranges

```
1 pd.date_range('2000-01-01', '2000-12-01', freq='BM')
```

```
DatetimeIndex(['2000-01-31', '2000-02-29', '2000-03-31', '2000-04-28',  
               '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31',  
               '2000-09-29', '2000-10-31', '2000-11-30'],  
              dtype='datetime64[ns]', freq='BM')
```

# Generating Date Ranges

Base Time Series Frequencies

Alias	Offset Type	Description
D	Day	Calendar daily
B	BusinessDay	Business daily
H	Hour	Hourly
T or min	Minute	Minutely
S	Second	Secondly
L or ms	Milli	Millisecond (1/1000th of 1 second)
U	Micro	Microsecond (1/1000000th of 1 second)
M	MonthEnd	Last calendar day of month
BM	BusinessMonthEnd	Last business day (weekday) of month
MS	MonthBegin	First calendar day of month

# Generating Date Ranges

Base Time Series Frequencies

Alias	Offset Type	Description
BMS	BusinessMonthBegin	First weekday of month
W-MON, W-TUE, ...	Week	Weekly on given day of week: MON, TUE, WED, THU, FRI, SAT, or SUN.
WOM-1MON, WOM-2MON, ...	WeekOfMonth	Generate weekly dates in the first, second, third, or fourth week of the month. For example, WOM-3FRI for the 3rd Friday of each month.
Q-JAN, Q-FEB, ...	QuarterEnd	Quarterly dates anchored on last calendar day of each month, for year ending in indicated month: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC.
BQ-JAN, BQ-FEB, ...	BusinessQuarterEnd	Quarterly dates anchored on last weekday day of each month, for year ending in indicated month
QS-JAN, QS-FEB, ...	QuarterBegin	Quarterly dates anchored on first calendar day of each month, for year ending in indicated month

# Generating Date Ranges

Base Time Series Frequencies

Alias	Offset Type	Description
BQS-JAN, BQS-FEB, ...	BusinessQuarterBegin	Quarterly dates anchored on first weekday day of each month, for year ending in indicated month
A-JAN, A-FEB, ...	YearEnd	Annual dates anchored on last calendar day of given month: JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC.
BA-JAN, BA-FEB, ...	BusinessYearEnd	Annual dates anchored on last weekday of given month
AS-JAN, AS-FEB, ...	YearBegin	Annual dates anchored on first day of given month
BAS-JAN, BAS-FEB, ...	BusinessYearBegin	Annual dates anchored on first weekday of given month

# Generating Date Ranges

```
1 pd.date_range('2012-05-02 12:56:31', periods=5)
```

```
DatetimeIndex(['2012-05-02 12:56:31', '2012-05-03 12:56:31',  
               '2012-05-04 12:56:31', '2012-05-05 12:56:31',  
               '2012-05-06 12:56:31'],  
              dtype='datetime64[ns]', freq='D')
```

```
1 pd.date_range('2012-05-02 12:56:31', periods=5, normalize=True)
```

```
DatetimeIndex(['2012-05-02', '2012-05-03', '2012-05-04', '2012-05-05',  
               '2012-05-06'],  
              dtype='datetime64[ns]', freq='D')
```

# Frequencies and Date Offsets

```
1 from pandas.tseries.offsets import Hour, Minute
2 hour = Hour()
3 hour
```

<Hour>

```
1 four_hours = Hour(4)
2 four_hours
```

<4 \* Hours>

# Frequencies and Date Offsets

```
1 pd.date_range('2000-01-01', '2000-01-03 23:59', freq='4h')
```

```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 04:00:00',  
               '2000-01-01 08:00:00', '2000-01-01 12:00:00',  
               '2000-01-01 16:00:00', '2000-01-01 20:00:00',  
               '2000-01-02 00:00:00', '2000-01-02 04:00:00',  
               '2000-01-02 08:00:00', '2000-01-02 12:00:00',  
               '2000-01-02 16:00:00', '2000-01-02 20:00:00',  
               '2000-01-03 00:00:00', '2000-01-03 04:00:00',  
               '2000-01-03 08:00:00', '2000-01-03 12:00:00',  
               '2000-01-03 16:00:00', '2000-01-03 20:00:00'],  
              dtype='datetime64[ns]', freq='4H')
```



# Frequencies and Date Offsets

```
1 Hour(2) + Minute(30)
```

<150 \* Minutes>

```
1 pd.date_range('2000-01-01', periods=10, freq='1h30min')
```

```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 01:30:00',  
               '2000-01-01 03:00:00', '2000-01-01 04:30:00',  
               '2000-01-01 06:00:00', '2000-01-01 07:30:00',  
               '2000-01-01 09:00:00', '2000-01-01 10:30:00',  
               '2000-01-01 12:00:00', '2000-01-01 13:30:00'],  
              dtype='datetime64[ns]', freq='90T')
```

# Week of month dates

```
1 rng = pd.date_range('2012-01-01', '2012-09-01', freq='WOM-3FRI')  
2 list(rng)
```

```
[Timestamp('2012-01-20 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-02-17 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-03-16 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-04-20 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-05-18 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-06-15 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-07-20 00:00:00', freq='WOM-3FRI'),  
Timestamp('2012-08-17 00:00:00', freq='WOM-3FRI')]
```

# Shifting (Leading and Lagging) Data

```
1 ts = pd.Series(np.random.randn(4),  
2                 index=pd.date_range('2000/1/1', periods=4, freq='M'))  
3 ts
```

```
2000-01-31    -0.150786  
2000-02-29     1.502825  
2000-03-31    -0.539127  
2000-04-30     1.044848  
Freq: M, dtype: float64
```

# Shifting (Leading and Lagging) Data

```
1 ts.shift(2)
```

```
2000-01-31      NaN
2000-02-29      NaN
2000-03-31    -0.150786
2000-04-30     1.502825
Freq: M, dtype: float64
```

```
1 ts.shift(-2)
```

```
2000-01-31    -0.539127
2000-02-29     1.044848
2000-03-31         NaN
2000-04-30         NaN
Freq: M, dtype: float64
```

computing percent changes in a time series or multiple timeseries as DataFrame columns.

```
ts / ts.shift(1) - 1
```

# Shifting (Leading and Lagging) Data

```
1 ts.shift(2, freq='M')
```

```
2000-03-31    -0.150786
2000-04-30     1.502825
2000-05-31    -0.539127
2000-06-30     1.044848
Freq: M, dtype: float64
```

```
1 ts.shift(3, freq='D')
```

```
2000-02-03    -0.150786
2000-03-03     1.502825
2000-04-03    -0.539127
2000-05-03     1.044848
dtype: float64
```

```
1 ts.shift(1, freq='90T')
```

```
2000-01-31 01:30:00    -0.150786
2000-02-29 01:30:00     1.502825
2000-03-31 01:30:00    -0.539127
2000-04-30 01:30:00     1.044848
Freq: M, dtype: float64
```

# Shifting dates with offsets

```
1 from pandas.tseries.offsets import Day, MonthEnd
2 now = datetime(2011, 11, 17)
3 now + 3 * Day()
```

Timestamp('2011-11-20 00:00:00')

```
1 now + MonthEnd()
```

Timestamp('2011-11-30 00:00:00')

```
1 now + MonthEnd(2)
```

Timestamp('2011-12-31 00:00:00')

# Shifting dates with offsets

```
1 offset = MonthEnd()  
2 offset.rollforward(now)
```

```
Timestamp('2011-11-30 00:00:00')
```

```
1 offset.rollback(now)
```

```
Timestamp('2011-10-31 00:00:00')
```

# Shifting dates with offsets

```
1 ts = pd.Series(np.random.randn(20),  
2                 index=pd.date_range('2000/1/15', periods=20, freq='4d'))  
3 ts  
4 ts.groupby(offset.rollforward).mean()
```

```
2000-01-31    -0.370374  
2000-02-29     0.335717  
2000-03-31    -0.677309  
dtype: float64
```



# Shifting dates with offsets

```
1 ts.resample('M').mean()
```

```
2000-01-31    -0.370374
```

```
2000-02-29     0.335717
```

```
2000-03-31    -0.677309
```

```
Freq: M, dtype: float64
```

# Periods and Period Arithmetic

```
p = pd.Period(2007, freq='A-DEC')  
p
```

```
Period('2007', 'A-DEC')
```

```
p + 5  
p - 2
```

```
Period('2005', 'A-DEC')
```

```
pd.Period('2014', freq='A-DEC') - p
```

```
<7 * YearEnds: month=12>
```

# Periods and Period Arithmetic

```
rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')  
rng
```

```
PeriodIndex(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06'], dtype='period[M]', freq='M')
```

```
pd.Series(np.random.randn(6), index=rng)
```

```
2000-01    2.089154  
2000-02   -0.060220  
2000-03   -0.167933  
2000-04    0.631634  
2000-05   -1.594313  
2000-06   -1.519937  
Freq: M, dtype: float64
```

# Periods and Period Arithmetic

```
values = ['2001Q3', '2002Q2', '2003Q1']  
index = pd.PeriodIndex(values, freq='Q-DEC')  
index
```

```
PeriodIndex(['2001Q3', '2002Q2', '2003Q1'], dtype='period[Q-DEC]', freq='Q-DEC')
```

# Period Frequency Conversion

```
1 p = pd.Period('2007', freq='A-DEC')  
2 p  
3 p.asfreq('M', how='start')
```

Period('2007-01', 'M')

```
1 p.asfreq('M', how='end')
```

Period('2007-12', 'M')

# Period Frequency Conversion

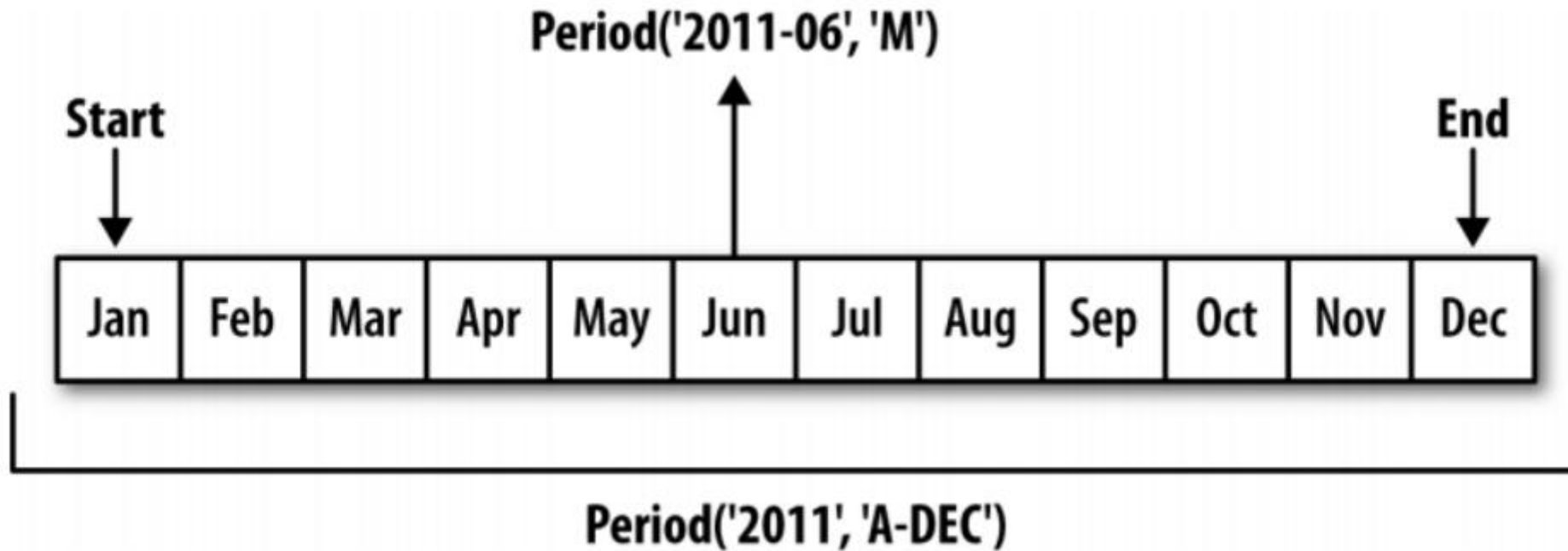
```
1 p = pd.Period('2007', freq='A-JUN')  
2 p  
3 p.asfreq('M', 'start')
```

Period('2006-07', 'M')

```
1 p.asfreq('M', 'end')
```

Period('2007-06', 'M')

# Period Frequency Conversion



# Period Frequency Conversion

```
1 p = pd.Period('2007-8', 'M')  
2 p.asfreq('A-JUN')
```

```
Period('2008', 'A-JUN')
```



# Period Frequency Conversion

```
1 rng = pd.period_range('2006', '2009', freq='A-DEC')
2 ts = pd.Series(np.random.randn(len(rng)), index=rng)
3 ts
```

2006 -0.272657

2007 -1.692615

2008 1.423830

2009 -0.407890

Freq: A-DEC, dtype: float64

```
1 ts.asfreq('M', how='start')
```

2006-01 -0.272657

2007-01 -1.692615

2008-01 1.423830

2009-01 -0.407890

Freq: M, dtype: float64

# Period Frequency Conversion

```
1 ts.asfreq('B', how='end')
```

2006-12-29    -0.272657

2007-12-31    -1.692615

2008-12-31     1.423830

2009-12-31    -0.407890

Freq: B, dtype: float64

# Quarterly Period Frequencies

```
1 p = pd.Period('2012Q4', freq='Q-JAN')  
2 p
```

Period('2012Q4', 'Q-JAN')

Year 2012												
M	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
Q-DEC	2012Q1			2012Q2			2012Q3			2012Q4		
Q-SEP	2012Q2			2012Q3			2012Q4			2013Q1		
Q-FEB	2012Q4		2013Q1			2013Q2			2013Q3			Q4

Different quarterly frequency conventions

# Quarterly Period Frequencies

```
1 p.asfreq('D', 'start')
```

```
Period('2011-11-01', 'D')
```

```
1 p.asfreq('D', 'end')
```

```
Period('2012-01-31', 'D')
```

# Quarterly Period Frequencies

```
1 p4pm = (p.asfreq('B', 'e') - 1).asfreq('T', 's') + 16 * 60
2 p4pm
```

```
Period('2012-01-30 16:00', 'T')
```

```
1 p4pm.to_timestamp()
```

```
Timestamp('2012-01-30 16:00:00')
```

# Quarterly Period Frequencies

```
1 rng = pd.period_range('2011Q3', '2012Q4', freq='Q-JAN')
2 ts = pd.Series(np.arange(len(rng)), index=rng)
3 ts
```

2011Q3 0

2011Q4 1

2012Q1 2

2012Q2 3

2012Q3 4

2012Q4 5

Freq: Q-JAN, dtype: int32

# Quarterly Period Frequencies

```
1 new_rng = (rng.asfreq('B', 'e') - 1).asfreq('T', 's') + 16 * 60
2 ts.index = new_rng.to_timestamp()
3 ts
```

```
2010-10-28 16:00:00    0
2011-01-28 16:00:00    1
2011-04-28 16:00:00    2
2011-07-28 16:00:00    3
2011-10-28 16:00:00    4
2012-01-30 16:00:00    5
dtype: int32
```

# Converting Timestamps to Periods (and Back)

```
1 rng = pd.date_range('2000-01-01', periods=3, freq='M')
2 ts = pd.Series(np.random.randn(3), index=rng)
3 ts
```

```
2000-01-31    0.756332
2000-02-29   -1.288602
2000-03-31    0.867534
Freq: M, dtype: float64
```

```
1 pts = ts.to_period()
2 pts
```

```
2000-01    0.756332
2000-02   -1.288602
2000-03    0.867534
Freq: M, dtype: float64
```



# Converting Timestamps to Periods (and Back)

```
1 rng = pd.date_range('2000/1/29', periods=6, freq='D')
2 ts2 = pd.Series(np.random.randn(6), index=rng)
3 ts2
```

```
2000-01-29    -0.252765
2000-01-30    -0.894590
2000-01-31     0.955842
2000-02-01    -1.653984
2000-02-02    -0.262528
2000-02-03    -0.976094
Freq: D, dtype: float64
```

```
1 ts2.to_period('M')
```

```
2000-01    -0.252765
2000-01    -0.894590
2000-01     0.955842
2000-02    -1.653984
2000-02    -0.262528
2000-02    -0.976094
Freq: M, dtype: float64
```

# Converting Timestamps to Periods (and Back)

```
1 pts = ts2.to_period()  
2 pts
```

```
2000-01-29    -0.252765  
2000-01-30    -0.894590  
2000-01-31     0.955842  
2000-02-01    -1.653984  
2000-02-02    -0.262528  
2000-02-03    -0.976094  
Freq: D, dtype: float64
```

```
1 pts.to_timestamp(how='end')
```

```
2000-01-29 23:59:59.999999999    -0.252765  
2000-01-30 23:59:59.999999999    -0.894590  
2000-01-31 23:59:59.999999999     0.955842  
2000-02-01 23:59:59.999999999    -1.653984  
2000-02-02 23:59:59.999999999    -0.262528  
2000-02-03 23:59:59.999999999    -0.976094  
Freq: D, dtype: float64
```

# Creating a PeriodIndex from Arrays

```
1 data = pd.read_csv('macrodata.csv')  
2 data.head(5)
```

	year	quarter	realgdp	realcons	realinv	realgovt	realdpi	cpi	m1	tbilrate	unemp
0	1959.0	1.0	2710.349	1707.4	286.898	470.045	1886.9	28.98	139.7	2.82	
1	1959.0	2.0	2778.801	1733.7	310.859	481.301	1919.7	29.15	141.7	3.08	
2	1959.0	3.0	2775.488	1751.8	289.226	491.260	1916.4	29.35	140.5	3.82	
3	1959.0	4.0	2785.204	1753.7	299.356	484.052	1931.3	29.37	140.0	4.33	
4	1960.0	1.0	2847.699	1770.5	331.722	462.199	1955.5	29.54	139.6	3.50	

# Creating a PeriodIndex from Arrays

1	data.year
---	-----------

0	1959.0
1	1959.0
2	1959.0
3	1959.0
4	1960.0
5	1960.0

...

200	2009.0
201	2009.0
202	2009.0

Name: year, Length: 203, dtype: float64

1	data.quarter
---	--------------

0	1.0
1	2.0
2	3.0
3	4.0
4	1.0
5	2.0

...

200	1.0
201	2.0
202	3.0

Name: quarter, Length: 203, dtype: float64

# Creating a PeriodIndex from Arrays

```
1 index = pd.PeriodIndex(year=data.year, quarter=data.quarter,  
2                        freq='Q-DEC')  
3 index
```

```
PeriodIndex(['1959Q1', '1959Q2', '1959Q3', '1959Q4', '1960Q1', '1960Q2',  
            '1960Q3', '1960Q4', '1961Q1', '1961Q2',  
            ...,  
            '2007Q2', '2007Q3', '2007Q4', '2008Q1', '2008Q2', '2008Q3',  
            '2008Q4', '2009Q1', '2009Q2', '2009Q3'],  
            dtype='period[Q-DEC]', length=203, freq='Q-DEC')
```

# Creating a PeriodIndex from Arrays

```
1 data.index = index
2 data.infl
```

1959Q1	0.00
1959Q2	2.34
1959Q3	2.74
1959Q4	0.27
1960Q1	2.31
1960Q2	0.14
...	
2008Q4	-8.79
2009Q1	0.94
2009Q2	3.37
2009Q3	3.56

Freq: Q-DEC, Name: infl, Length: 203, dtype: float64

# Resampling and Frequency Conversion

```
1 rng = pd.date_range('2000-01-01', periods=100, freq='D')
2 ts = pd.Series(np.random.randn(len(rng)), index=rng)
3 ts
```

2000-01-01 -1.493407

2000-01-02 1.167858

2000-01-03 0.969001

2000-01-04 -2.536487

2000-01-05 0.362754

...

2000-04-08 -1.040816

2000-04-09 0.426419

Freq: D, Length: 100, dtype: float64

# Resampling and Frequency Conversion

```
1 ts.resample('M').mean()
```

```
2000-01-31    0.027804
2000-02-29    0.194619
2000-03-31    0.166734
2000-04-30    0.239600
Freq: M, dtype: float64
```

```
1 ts.resample('M', kind='period').mean()
```

```
2000-01    0.027804
2000-02    0.194619
2000-03    0.166734
2000-04    0.239600
Freq: M, dtype: float64
```



# Resampling and Frequency Conversion

resample method arguments

Argument	Description
<code>freq</code>	String or DateOffset indicating desired resampled frequency, e.g. 'M', '5min', or <code>Second(15)</code>
<code>how='mean'</code>	Function name or array function producing aggregated value, for example 'mean', 'ohlc', <code>np.max</code> . Defaults to 'mean'. Other common values: 'first', 'last', 'median', 'ohlc', 'max', 'min'.
<code>axis=0</code>	Axis to resample on, default <code>axis=0</code>
<code>fill_method=None</code>	How to interpolate when upsampling, as in 'ffill' or 'bfill'. By default does no interpolation.
<code>closed='right'</code>	In downsampling, which end of each interval is closed (inclusive), 'right' or 'left'. Defaults to 'right'
<code>label='right'</code>	In downsampling, how to label the aggregated result, with the 'right' or 'left' bin edge. For example, the 9:30 to 9:35 5-minute interval could be labeled 9:30 or 9:35. Defaults to 'right' (or 9:35, in this example).

# Resampling and Frequency Conversion

resample method arguments

Argument	Description
<code>loffset=None</code>	Time adjustment to the bin labels, such as <code>'-1s' / Second(-1)</code> to shift the aggregate labels one second earlier
<code>limit=None</code>	When forward or backward filling, the maximum number of periods to fill
<code>kind=None</code>	Aggregate to periods ( <code>'period'</code> ) or timestamps ( <code>'timestamp'</code> ); defaults to kind of index the time series has
<code>convention=None</code>	When resampling periods, the convention ( <code>'start'</code> or <code>'end'</code> ) for converting the low frequency period to high frequency. Defaults to <code>'end'</code>

# Downsampling

```
1 rng = pd.date_range('2000-01-01', periods=12, freq='T')
2 ts = pd.Series(np.arange(12), index=rng)
3 ts
```

```
2000-01-01 00:00:00    0
2000-01-01 00:01:00    1
2000-01-01 00:02:00    2
2000-01-01 00:03:00    3
...
2000-01-01 00:09:00    9
2000-01-01 00:10:00   10
2000-01-01 00:11:00   11
Freq: T, dtype: int32
```

# Downsampling

- Which side of each interval is *closed*
- How to label each aggregated bin, either with the start of the interval or the end

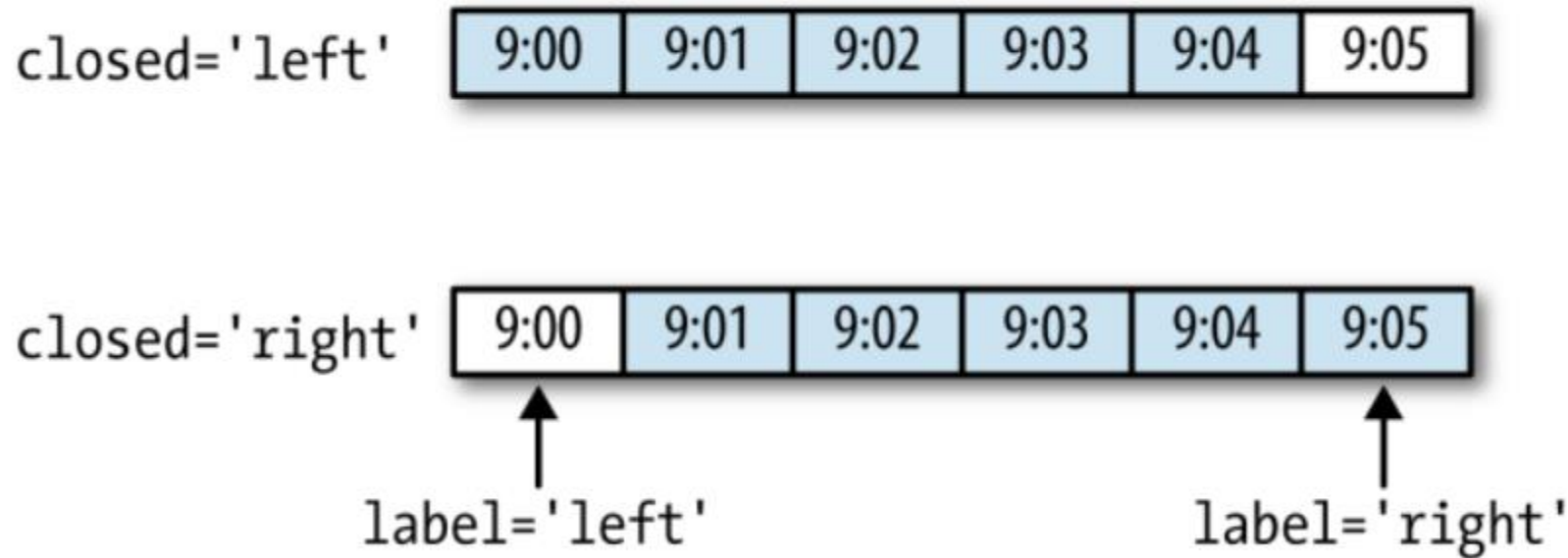
```
1 ts.resample('5min', closed='right')
```

DatetimeIndexResampler [freq=<5 \* Minutes>, axis=0, closed=right, label=left, convention=start, base=0]

```
1 ts.resample('5min', closed='right').sum()
```

```
1999-12-31 23:55:00    0
2000-01-01 00:00:00   15
2000-01-01 00:05:00   40
2000-01-01 00:10:00   11
Freq: 5T, dtype: int32
```

# Downsampling



minute resampling of closed, label conventions

# Downsampling

```
1 ts.resample('5min', closed='right', label='right').sum()
```

```
2000-01-01 00:00:00    0
2000-01-01 00:05:00   15
2000-01-01 00:10:00   40
2000-01-01 00:15:00   11
Freq: 5T, dtype: int32
```

```
1 ts.resample('5min', closed='right',
2               label='right', loffset='-1s').sum()
```

```
1999-12-31 23:59:59    0
2000-01-01 00:04:59   15
2000-01-01 00:09:59   40
2000-01-01 00:14:59   11
Freq: 5T, dtype: int32
```

# Open-High-Low-Close (OHLC) resampling

```
1 ts.resample('5min').ohlc()
```

	open	high	low	close
2000-01-01 00:00:00	0	4	0	4
2000-01-01 00:05:00	5	9	5	9
2000-01-01 00:10:00	10	11	10	11

# Upsampling and Interpolation

```
1 frame = pd.DataFrame(np.random.randn(2, 4),  
2                       index=pd.date_range('2000/1/1', periods=2,  
3                                           freq='W-WED'),  
4                       columns=['Colorado', 'Texas', 'New York', 'Ohio'])  
5 frame
```

	Colorado	Texas	New York	Ohio
2000-01-05	2.268799	0.146326	0.508391	-0.196713
2000-01-12	-3.745356	-1.520113	-0.346839	-0.696918



# Upsampling and Interpolation

```
1 df_daily = frame.resample('D').asfreq()  
2 df_daily
```

	Colorado	Texas	New York	Ohio
2000-01-05	2.268799	0.146326	0.508391	-0.196713
2000-01-06	NaN	NaN	NaN	NaN
2000-01-07	NaN	NaN	NaN	NaN
2000-01-08	NaN	NaN	NaN	NaN
2000-01-09	NaN	NaN	NaN	NaN
2000-01-10	NaN	NaN	NaN	NaN
2000-01-11	NaN	NaN	NaN	NaN
2000-01-12	-3.745356	-1.520113	-0.346839	-0.696918

# Upsampling and Interpolation

```
1 frame.resample('D').ffill()
```

	Colorado	Texas	New York	Ohio
<b>2000-01-05</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-06</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-07</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-08</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-09</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-10</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-11</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-12</b>	-3.745356	-1.520113	-0.346839	-0.696918

# Upsampling and Interpolation

```
1 frame.resample('D').ffill(limit=2)
```

	Colorado	Texas	New York	Ohio
<b>2000-01-05</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-06</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-07</b>	2.268799	0.146326	0.508391	-0.196713
<b>2000-01-08</b>	NaN	NaN	NaN	NaN
<b>2000-01-09</b>	NaN	NaN	NaN	NaN
<b>2000-01-10</b>	NaN	NaN	NaN	NaN
<b>2000-01-11</b>	NaN	NaN	NaN	NaN
<b>2000-01-12</b>	-3.745356	-1.520113	-0.346839	-0.696918

# Upsampling and Interpolation

```
1 frame.resample('W-THU')
```

DatetimeIndexResampler [freq=<Week: weekday=3>, axis=0, closed=right, label=right, convention=start, base=0]

```
1 frame.resample('W-THU').ffill()
```

	Colorado	Texas	New York	Ohio
2000-01-06	2.268799	0.146326	0.508391	-0.196713
2000-01-13	-3.745356	-1.520113	-0.346839	-0.696918

# Resampling with Periods

```
1 frame = pd.DataFrame(np.random.randn(24, 4),
2                       index=pd.period_range('2000-1', '2001-12',
3                                             freq='M'),
4                       columns=['Colorado', 'Texas', 'New York', 'Ohio'])
5 frame[:5]
```

	Colorado	Texas	New York	Ohio
<b>2000-01</b>	0.873921	-1.180212	-0.208885	-0.549671
<b>2000-02</b>	-1.252880	-1.276761	1.881156	1.108227
<b>2000-03</b>	-1.751994	-0.973899	0.908732	-0.509226
<b>2000-04</b>	-1.023400	-0.412273	-1.073039	-0.601411
<b>2000-05</b>	0.222178	0.949363	0.704186	-1.358964

# Resampling with Periods

```
1 annual_frame = frame.resample('A-DEC').mean()  
2 annual_frame
```

	Colorado	Texas	New York	Ohio
2000	0.154403	-0.402994	-0.115005	0.032294
2001	0.326664	0.377224	-0.324770	-0.607382

# Resampling with Periods

```
1 # Q-DEC: Quarterly, year ending in December
2 annual_frame.resample('Q-DEC')
```

PeriodIndexResampler [freq=<QuarterEnd: startingMonth=12>, axis=0, closed=right, label=right, convention=start, base=0]

```
1 annual_frame.resample('Q-DEC').ffill()
```

	Colorado	Texas	New York	Ohio
<b>2000Q1</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2000Q2</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2000Q3</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2000Q4</b>	0.154403	-0.402994	-0.115005	0.032294

<b>2001Q1</b>	0.326664	0.377224	-0.324770	-0.607382
<b>2001Q2</b>	0.326664	0.377224	-0.324770	-0.607382
<b>2001Q3</b>	0.326664	0.377224	-0.324770	-0.607382
<b>2001Q4</b>	0.326664	0.377224	-0.324770	-0.607382

# Resampling with Periods

```
1 annual_frame.resample('Q-DEC', convention='end').ffill()
```

	Colorado	Texas	New York	Ohio
<b>2000Q4</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q1</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q2</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q3</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q4</b>	0.326664	0.377224	-0.324770	-0.607382



# Resampling with Periods

- In downsampling, the target frequency must be a *subperiod* of the source frequency.
- In upsampling, the target frequency must be a *superperiod* of the source frequency.

```
1 annual_frame.resample('Q-MAR').ffill()
```

	Colorado	Texas	New York	Ohio
<b>2000Q4</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q1</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q2</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q3</b>	0.154403	-0.402994	-0.115005	0.032294
<b>2001Q4</b>	0.326664	0.377224	-0.324770	-0.607382

<b>2002Q1</b>	0.326664	0.377224	-0.324770	-0.607382
<b>2002Q2</b>	0.326664	0.377224	-0.324770	-0.607382
<b>2002Q3</b>	0.326664	0.377224	-0.324770	-0.607382