Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.QuarterEnd.rollback

QuarterEnd.**rollback**(*self*, *dt*)

Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.QuarterEnd.rollforward

QuarterEnd.**rollforward**(*self*, *dt*)

Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

QuarterEnd.freqstr
QuarterEnd.kwds
QuarterEnd.name
QuarterEnd.nanos
QuarterEnd.normalize
QuarterEnd.rule_code

pandas.tseries.offsets.QuarterEnd.freqstr

QuarterEnd.**freqstr**

pandas.tseries.offsets.QuarterEnd.kwds

property QuarterEnd.**kwds**

pandas.tseries.offsets.QuarterEnd.name

property QuarterEnd.**name**

pandas.tseries.offsets.QuarterEnd.nanos

property QuarterEnd.**nanos**

pandas.tseries.offsets.QuarterEnd.normalize

QuarterEnd.**normalize = False**

pandas.tseries.offsets.QuarterEnd.rule_code

property QuarterEnd.**rule_code**

Methods

QuarterEnd.apply(self, other)	
QuarterEnd.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
QuarterEnd.copy(self)	
QuarterEnd.isAnchored(self)	
QuarterEnd.onOffset(self, dt)	
QuarterEnd.is_anchored(self)	
QuarterEnd.is_on_offset(self, dt)	
QuarterEnd.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.QuarterEnd.apply

QuarterEnd.**apply**(*self*, *other*)

pandas.tseries.offsets.QuarterEnd.apply_index

QuarterEnd.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.QuarterEnd.copy

QuarterEnd.**copy**(*self*)

pandas.tseries.offsets.QuarterEnd.isAnchored

QuarterEnd.**isAnchored**(*self*)

pandas.tseries.offsets.QuarterEnd.onOffset

QuarterEnd.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.QuarterEnd.is_anchored

QuarterEnd.**is_anchored**(*self*)

pandas.tseries.offsets.QuarterEnd.is_on_offset

QuarterEnd.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.QuarterEnd.__call__

QuarterEnd.**__call__**(*self*, *other*)

Call self as a function.

3.8.23 QuarterBegin

QuarterBegin([n, normalize, startingMonth])	

Attributes

pandas.tseries.offsets.QuarterBegin

class pandas.tseries.offsets.**QuarterBegin**(*n=1*, *normalize=False*, *startingMonth=None*)

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.QuarterBegin.base

property QuarterBegin.**base**

Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.QuarterBegin.rollback

QuarterBegin.**rollback**(*self*, *dt*)

Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.QuarterBegin.rollforward

QuarterBegin.**rollforward**(*self*, *dt*)

Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

QuarterBegin.freqstr
QuarterBegin.kwds
QuarterBegin.name
QuarterBegin.nanos
QuarterBegin.normalize
QuarterBegin.rule_code

pandas.tseries.offsets.QuarterBegin.freqstr

QuarterBegin.**freqstr**

pandas.tseries.offsets.QuarterBegin.kwds

property QuarterBegin.**kwds**

pandas.tseries.offsets.QuarterBegin.name

property QuarterBegin.**name**

pandas.tseries.offsets.QuarterBegin.nanos

property QuarterBegin.**nanos**

pandas.tseries.offsets.QuarterBegin.normalize

QuarterBegin.**normalize = False**

pandas.tseries.offsets.QuarterBegin.rule_code

property QuarterBegin.**rule_code**

Methods

QuarterBegin.apply(self, other)	
QuarterBegin.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
QuarterBegin.copy(self)	
QuarterBegin.isAnchored(self)	
QuarterBegin.onOffset(self, dt)	
QuarterBegin.is_anchored(self)	
QuarterBegin.is_on_offset(self, dt)	
QuarterBegin.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.QuarterBegin.apply

QuarterBegin.**apply**(*self*, *other*)

pandas.tseries.offsets.QuarterBegin.apply_index

QuarterBegin.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.QuarterBegin.copy

QuarterBegin.**copy**(*self*)

pandas.tseries.offsets.QuarterBegin.isAnchored

QuarterBegin.**isAnchored**(*self*)

pandas.tseries.offsets.QuarterBegin.onOffset

QuarterBegin.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.QuarterBegin.is_anchored

QuarterBegin.**is_anchored**(*self*)

pandas.tseries.offsets.QuarterBegin.is_on_offset

QuarterBegin.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.QuarterBegin.__call__

QuarterBegin.**__call__**(*self*, *other*)
Call self as a function.

3.8.24 YearOffset

YearOffset([n, normalize, month])	DateOffset that just needs a month.

pandas.tseries.offsets.YearOffset

class pandas.tseries.offsets.**YearOffset**(*n=1*, *normalize=False*, *month=None*)
DateOffset that just needs a month.

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.YearOffset.base

property YearOffset.**base**
Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.YearOffset.rollback

YearOffset.**rollback**(*self*, *dt*)
Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.YearOffset.rollforward

YearOffset.**rollforward**(*self*, *dt*)
Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

YearOffset.freqstr
YearOffset.kwds
YearOffset.name
YearOffset.nanos
YearOffset.normalize
YearOffset.rule_code

pandas.tseries.offsets.YearOffset.freqstr

YearOffset.**freqstr**

pandas.tseries.offsets.YearOffset.kwds

property YearOffset.**kwds**

pandas.tseries.offsets.YearOffset.name

property YearOffset.**name**

pandas.tseries.offsets.YearOffset.nanos

property YearOffset.**nanos**

pandas.tseries.offsets.YearOffset.normalize

YearOffset.**normalize = False**

pandas.tseries.offsets.YearOffset.rule_code

property YearOffset.**rule_code**

Methods

YearOffset.apply(self, other)	
YearOffset.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
YearOffset.copy(self)	
YearOffset.isAnchored(self)	
YearOffset.onOffset(self, dt)	
YearOffset.is_anchored(self)	
YearOffset.is_on_offset(self, dt)	

continues on next page

Table 299 – continued from previous page

YearOffset.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.YearOffset.apply

YearOffset.**apply**(*self*, *other*)

pandas.tseries.offsets.YearOffset.apply_index

YearOffset.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.YearOffset.copy

YearOffset.**copy**(*self*)

pandas.tseries.offsets.YearOffset.isAnchored

YearOffset.**isAnchored**(*self*)

pandas.tseries.offsets.YearOffset.onOffset

YearOffset.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.YearOffset.is_anchored

YearOffset.**is_anchored**(*self*)

pandas.tseries.offsets.YearOffset.is_on_offset

YearOffset.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.YearOffset.__call__

YearOffset.**__call__**(*self*, *other*)
Call self as a function.

3.8.25 BYearEnd

BYearEnd([n, normalize, month])	DateOffset increments between business EOM dates.

pandas.tseries.offsets.BYearEnd

class pandas.tseries.offsets.**BYearEnd**(*n=1*, *normalize=False*, *month=None*)
DateOffset increments between business EOM dates.

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.BYearEnd.base

property BYearEnd.**base**
Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.BYearEnd.rollback

BYearEnd.**rollback**(*self*, *dt*)

Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.BYearEnd.rollforward

BYearEnd.**rollforward**(*self*, *dt*)

Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

BYearEnd.freqstr
BYearEnd.kwds
BYearEnd.name
BYearEnd.nanos
BYearEnd.normalize
BYearEnd.rule_code

pandas.tseries.offsets.BYearEnd.freqstr

BYearEnd.**freqstr**

pandas.tseries.offsets.BYearEnd.kwds

property BYearEnd.**kwds**

pandas.tseries.offsets.BYearEnd.name

property BYearEnd.**name**

pandas.tseries.offsets.BYearEnd.nanos

property BYearEnd.**nanos**

pandas.tseries.offsets.BYearEnd.normalize

BYearEnd.**normalize = False**

pandas.tseries.offsets.BYearEnd.rule_code

property BYearEnd.**rule_code**

Methods

BYearEnd.apply(self, other)	
BYearEnd.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
BYearEnd.copy(self)	
BYearEnd.isAnchored(self)	
BYearEnd.onOffset(self, dt)	
BYearEnd.is_anchored(self)	
BYearEnd.is_on_offset(self, dt)	
BYearEnd.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.BYearEnd.apply

BYearEnd.**apply**(*self*, *other*)

pandas.tseries.offsets.BYearEnd.apply_index

BYearEnd.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.BYearEnd.copy

BYearEnd.**copy**(*self*)

pandas.tseries.offsets.BYearEnd.isAnchored

BYearEnd.**isAnchored**(*self*)

pandas.tseries.offsets.BYearEnd.onOffset

BYearEnd.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.BYearEnd.is_anchored

BYearEnd.**is_anchored**(*self*)

pandas.tseries.offsets.BYearEnd.is_on_offset

BYearEnd.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.BYearEnd.__call__

BYearEnd.**__call__**(*self*, *other*)

Call self as a function.

3.8.26 BYearBegin

BYearBegin([n, normalize, month])	DateOffset increments between business year begin dates.

pandas.tseries.offsets.BYearBegin

class pandas.tseries.offsets.**BYearBegin**(*n=1*, *normalize=False*, *month=None*)
DateOffset increments between business year begin dates.

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.BYearBegin.base

property BYearBegin.**base**
Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.BYearBegin.rollback

BYearBegin.**rollback**(*self*, *dt*)
Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.BYearBegin.rollforward

BYearBegin.**rollforward**(*self*, *dt*)
Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

BYearBegin.freqstr
BYearBegin.kwds
BYearBegin.name
BYearBegin.nanos
BYearBegin.normalize
BYearBegin.rule_code

pandas.tseries.offsets.BYearBegin.freqstr

BYearBegin.**freqstr**

pandas.tseries.offsets.BYearBegin.kwds

property BYearBegin.**kwds**

pandas.tseries.offsets.BYearBegin.name

property BYearBegin.**name**

pandas.tseries.offsets.BYearBegin.nanos

property BYearBegin.**nanos**

pandas.tseries.offsets.BYearBegin.normalize

BYearBegin.**normalize = False**

pandas.tseries.offsets.BYearBegin.rule_code

property BYearBegin.**rule_code**

Methods

BYearBegin.apply(self, other)	
BYearBegin.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
BYearBegin.copy(self)	
BYearBegin.isAnchored(self)	
BYearBegin.onOffset(self, dt)	
BYearBegin.is_anchored(self)	
BYearBegin.is_on_offset(self, dt)	
BYearBegin.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.BYearBegin.apply

BYearBegin.**apply**(*self*, *other*)

pandas.tseries.offsets.BYearBegin.apply_index

BYearBegin.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.BYearBegin.copy

BYearBegin.**copy**(*self*)

pandas.tseries.offsets.BYearBegin.isAnchored

BYearBegin.**isAnchored**(*self*)

pandas.tseries.offsets.BYearBegin.onOffset

BYearBegin.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.BYearBegin.is_anchored

BYearBegin.**is_anchored**(*self*)

pandas.tseries.offsets.BYearBegin.is_on_offset

BYearBegin.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.BYearBegin.__call__

BYearBegin.**__call__**(*self*, *other*)
Call self as a function.

3.8.27 YearEnd

YearEnd([n, normalize, month])	DateOffset increments between calendar year ends.

pandas.tseries.offsets.YearEnd

class pandas.tseries.offsets.**YearEnd**(*n=1*, *normalize=False*, *month=None*)
DateOffset increments between calendar year ends.

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.YearEnd.base

property YearEnd.**base**
Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.YearEnd.rollback

YearEnd.**rollback**(*self*, *dt*)

Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.YearEnd.rollforward

YearEnd.**rollforward**(*self*, *dt*)

Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

YearEnd.freqstr
YearEnd.kwds
YearEnd.name
YearEnd.nanos
YearEnd.normalize
YearEnd.rule_code

pandas.tseries.offsets.YearEnd.freqstr

YearEnd.**freqstr**

pandas.tseries.offsets.YearEnd.kwds

property YearEnd.**kwds**

pandas.tseries.offsets.YearEnd.name

property YearEnd.**name**

pandas.tseries.offsets.YearEnd.nanos

property YearEnd.**nanos**

pandas.tseries.offsets.YearEnd.normalize

YearEnd.**normalize = False**

pandas.tseries.offsets.YearEnd.rule_code

property YearEnd.**rule_code**

Methods

YearEnd.apply(self, other)	
YearEnd.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
YearEnd.copy(self)	
YearEnd.isAnchored(self)	
YearEnd.onOffset(self, dt)	
YearEnd.is_anchored(self)	
YearEnd.is_on_offset(self, dt)	
YearEnd.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.YearEnd.apply

YearEnd.**apply**(*self*, *other*)

pandas.tseries.offsets.YearEnd.apply_index

YearEnd.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.YearEnd.copy

YearEnd.**copy**(*self*)

pandas.tseries.offsets.YearEnd.isAnchored

YearEnd.**isAnchored**(*self*)

pandas.tseries.offsets.YearEnd.onOffset

YearEnd.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.YearEnd.is_anchored

YearEnd.**is_anchored**(*self*)

pandas.tseries.offsets.YearEnd.is_on_offset

YearEnd.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.YearEnd.__call__

YearEnd.**__call__**(*self*, *other*)

Call self as a function.

3.8.28 YearBegin

YearBegin([n, normalize, month])	DateOffset increments between calendar year begin dates.

pandas.tseries.offsets.YearBegin

class pandas.tseries.offsets.**YearBegin**(*n=1*, *normalize=False*, *month=None*)

DateOffset increments between calendar year begin dates.

Attributes

base	Returns a copy of the calling offset object with n=1 and all other attributes equal.

pandas.tseries.offsets.YearBegin.base

property YearBegin.**base**

Returns a copy of the calling offset object with n=1 and all other attributes equal.

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

rollback(self, dt)	Roll provided date backward to next offset only if not on offset.
rollforward(self, dt)	Roll provided date forward to next offset only if not on offset.

pandas.tseries.offsets.YearBegin.rollback

YearBegin.**rollback**(*self*, *dt*)

Roll provided date backward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

pandas.tseries.offsets.YearBegin.rollforward

YearBegin.**rollforward**(*self*, *dt*)

Roll provided date forward to next offset only if not on offset.

Returns

TimeStamp Rolled timestamp if not on offset, otherwise unchanged timestamp.

__call__	
apply	
apply_index	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

YearBegin.freqstr
YearBegin.kwds
YearBegin.name
YearBegin.nanos
YearBegin.normalize
YearBegin.rule_code

pandas.tseries.offsets.YearBegin.freqstr

YearBegin.**freqstr**

pandas.tseries.offsets.YearBegin.kwds

property YearBegin.**kwds**

pandas.tseries.offsets.YearBegin.name

property YearBegin.**name**

pandas.tseries.offsets.YearBegin.nanos

property YearBegin.**nanos**

pandas.tseries.offsets.YearBegin.normalize

YearBegin.**normalize = False**

pandas.tseries.offsets.YearBegin.rule_code

property YearBegin.**rule_code**

Methods

YearBegin.apply(self, other)	
YearBegin.apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
YearBegin.copy(self)	
YearBegin.isAnchored(self)	
YearBegin.onOffset(self, dt)	
YearBegin.is_anchored(self)	
YearBegin.is_on_offset(self, dt)	
YearBegin.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.YearBegin.apply

YearBegin.**apply**(*self*, *other*)

pandas.tseries.offsets.YearBegin.apply_index

YearBegin.**apply_index**(*self*, *other*)

Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.

Parameters

i [DatetimeIndex]

Returns

y [DatetimeIndex]

pandas.tseries.offsets.YearBegin.copy

YearBegin.**copy**(*self*)

pandas.tseries.offsets.YearBegin.isAnchored

YearBegin.**isAnchored**(*self*)

pandas.tseries.offsets.YearBegin.onOffset

YearBegin.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.YearBegin.is_anchored

YearBegin.**is_anchored**(*self*)

pandas.tseries.offsets.YearBegin.is_on_offset

YearBegin.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.YearBegin.__call__

YearBegin.**__call__**(*self*, *other*)
Call self as a function.

3.8.29 FY5253

FY5253([n, normalize, weekday, ...])	Describes 52-53 week fiscal year.

pandas.tseries.offsets.FY5253

class pandas.tseries.offsets.**FY5253**(*n=1*, *normalize=False*, *weekday=0*, *startingMonth=1*, *variation='nearest'*)
Describes 52-53 week fiscal year. This is also known as a 4-4-5 calendar.

It is used by companies that desire that their fiscal year always end on the same day of the week.

It is a method of managing accounting periods. It is a common calendar structure for some industries, such as retail, manufacturing and parking industry.

For more information see: http://en.wikipedia.org/wiki/4-4-5_calendar

The year may either:
- end on the last X day of the Y month.
- end on the last X day closest to the last day of the Y month.

X is a specific day of the week. Y is a certain month of the year

Parameters

n [int]