weekday [int {0, 1, . . . , 6}, default 0] A specific integer for the day of the week.

- 0 is Monday
- 1 is Tuesday
- 2 is Wednesday
- 3 is Thursday
- 4 is Friday
- 5 is Saturday
- 6 is Sunday.

startingMonth [int {1, 2, . . . 12}, default 1] The month in which the fiscal year ends.

variation [str, default "nearest"] Method of employing 4-4-5 calendar.

There are two options:

- "nearest" means year end is **weekday** closest to last day of month in year.
- "last" means year end is final **weekday** of the final month in fiscal year.

Attributes

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

pandas.tseries.offsets.FY5253.base

property FY5253.**base**

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

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.FY5253.apply_index

FY5253.**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.FY5253.rollback

FY5253.**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.FY5253.rollforward

FY5253.**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	
copy	
get_rule_code_suffix	
get_year_end	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

FY5253.freqstr
FY5253.kwds
FY5253.name
FY5253.nanos
FY5253.normalize
FY5253.rule_code

pandas.tseries.offsets.FY5253.freqstr

FY5253.**freqstr**

pandas.tseries.offsets.FY5253.kwds

property FY5253.**kwds**

pandas.tseries.offsets.FY5253.name

property FY5253.**name**

pandas.tseries.offsets.FY5253.nanos

property FY5253.**nanos**

pandas.tseries.offsets.FY5253.normalize

FY5253.**normalize = False**

pandas.tseries.offsets.FY5253.rule_code

property FY5253.**rule_code**

Methods

FY5253.apply(self, other)	
FY5253.copy(self)	
FY5253.get_rule_code_suffix(self)	
FY5253.get_year_end(self, dt)	
FY5253.isAnchored(self)	
FY5253.onOffset(self, dt)	
FY5253.is_anchored(self)	
FY5253.is_on_offset(self, dt)	
FY5253.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.FY5253.apply

FY5253.**apply**(*self*, *other*)

pandas.tseries.offsets.FY5253.copy

FY5253.**copy**(*self*)

pandas.tseries.offsets.FY5253.get_rule_code_suffix

FY5253.**get_rule_code_suffix**(*self*)

pandas.tseries.offsets.FY5253.get_year_end

FY5253.**get_year_end**(*self*, *dt*)

pandas.tseries.offsets.FY5253.isAnchored

FY5253.**isAnchored**(*self*)

pandas.tseries.offsets.FY5253.onOffset

FY5253.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.FY5253.is_anchored

FY5253.**is_anchored**(*self*)

pandas.tseries.offsets.FY5253.is_on_offset

FY5253.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.FY5253.__call__

FY5253.**__call__**(*self*, *other*)
Call self as a function.

3.8.30 FY5253Quarter

FY5253Quarter([n, normalize, weekday, ...])	DateOffset increments between business quarter dates for 52-53 week fiscal year (also known as a 4-4-5 calendar).

pandas.tseries.offsets.FY5253Quarter

class pandas.tseries.offsets.**FY5253Quarter**(*n=1*, *normalize=False*, *weekday=0*, *startingMonth=1*, *qtr_with_extra_week=1*, *variation='nearest'*)

DateOffset increments between business quarter dates for 52-53 week fiscal year (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

startingMonth = 1 corresponds to dates like 1/31/2007, 4/30/2007, ... startingMonth = 2 corresponds to dates like 2/28/2007, 5/31/2007, ... startingMonth = 3 corresponds to dates like 3/30/2007, 6/29/2007, ...

Parameters

n [int]

weekday [int {0, 1, ..., 6}, default 0] A specific integer for the day of the week.

- 0 is Monday
- 1 is Tuesday
- 2 is Wednesday
- 3 is Thursday
- 4 is Friday
- 5 is Saturday
- 6 is Sunday.

startingMonth [int {1, 2, ..., 12}, default 1] The month in which fiscal years end.

qtr_with_extra_week [int {1, 2, 3, 4}, default 1] The quarter number that has the leap or 14 week when needed.

variation [str, default "nearest"] Method of employing 4-4-5 calendar.

There are two options:

- "nearest" means year end is **weekday** closest to last day of month in year.
- "last" means year end is final **weekday** of the final month in fiscal year.

Attributes

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

pandas.tseries.offsets.FY5253Quarter.base

property FY5253Quarter.**base**

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

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.FY5253Quarter.apply_index

FY5253Quarter.**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.FY5253Quarter.rollback

FY5253Quarter.**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.FY5253Quarter.rollforward

FY5253Quarter.**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	
copy	
get_weeks	
isAnchored	
is_anchored	
is_on_offset	
onOffset	
year_has_extra_week	

Properties

FY5253Quarter.freqstr
FY5253Quarter.kwds
FY5253Quarter.name
FY5253Quarter.nanos
FY5253Quarter.normalize
FY5253Quarter.rule_code

pandas.tseries.offsets.FY5253Quarter.freqstr

FY5253Quarter.**freqstr**

pandas.tseries.offsets.FY5253Quarter.kwds

property FY5253Quarter.**kwds**

pandas.tseries.offsets.FY5253Quarter.name

property FY5253Quarter.**name**

pandas.tseries.offsets.FY5253Quarter.nanos

property FY5253Quarter.**nanos**

pandas.tseries.offsets.FY5253Quarter.normalize

FY5253Quarter.**normalize = False**

pandas.tseries.offsets.FY5253Quarter.rule_code

property FY5253Quarter.**rule_code**

Methods

FY5253Quarter.apply(self, other)	
FY5253Quarter.copy(self)	
FY5253Quarter.get_weeks(self, dt)	
FY5253Quarter.isAnchored(self)	
FY5253Quarter.onOffset(self, dt)	
FY5253Quarter.is_anchored(self)	
FY5253Quarter.is_on_offset(self, dt)	
FY5253Quarter.year_has_extra_week(self, dt)	
FY5253Quarter.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.FY5253Quarter.apply

FY5253Quarter.**apply**(*self*, *other*)

pandas.tseries.offsets.FY5253Quarter.copy

FY5253Quarter.**copy**(*self*)

pandas.tseries.offsets.FY5253Quarter.get_weeks

FY5253Quarter.**get_weeks**(*self*, *dt*)

pandas.tseries.offsets.FY5253Quarter.isAnchored

FY5253Quarter.**isAnchored**(*self*)

pandas.tseries.offsets.FY5253Quarter.onOffset

FY5253Quarter.**onOffset**(*self*, *dt*)

pandas.tseries.offsets.FY5253Quarter.is_anchored

FY5253Quarter.**is_anchored**(*self*)

pandas.tseries.offsets.FY5253Quarter.is_on_offset

FY5253Quarter.**is_on_offset**(*self*, *dt*)

pandas.tseries.offsets.FY5253Quarter.year_has_extra_week

FY5253Quarter.**year_has_extra_week**(*self*, *dt*)

pandas.tseries.offsets.FY5253Quarter.__call__

FY5253Quarter.**__call__**(*self*, *other*)
Call self as a function.

3.8.31 Easter

Easter([n, normalize])	DateOffset for the Easter holiday using logic defined in dateutil.

pandas.tseries.offsets.Easter

class pandas.tseries.offsets.**Easter**(*n=1*, *normalize=False*)

DateOffset for the Easter holiday using logic defined in dateutil.

Right now uses the revised method which is valid in years 1583-4099.

Attributes

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

pandas.tseries.offsets.Easter.base

property Easter.**base**

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

freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.Easter.apply_index

Easter.**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.Easter.rollback

Easter.**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.Easter.rollforward

Easter.**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	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

Easter.freqstr
Easter.kwds
Easter.name
Easter.nanos
Easter.normalize
Easter.rule_code

pandas.tseries.offsets.Easter.freqstr

Easter.**freqstr**

pandas.tseries.offsets.Easter.kwds

property Easter.**kwds**

pandas.tseries.offsets.Easter.name

property Easter.**name**

pandas.tseries.offsets.Easter.nanos

property Easter.**nanos**

pandas.tseries.offsets.Easter.normalize

Easter.**normalize = False**

pandas.tseries.offsets.Easter.rule_code

property Easter.**rule_code**

Methods

Easter.apply(self, other)	
Easter.copy(self)	
Easter.isAnchored(self)	
Easter.onOffset(self, dt)	
Easter.is_anchored(self)	
Easter.is_on_offset(self, dt)	
Easter.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.Easter.apply

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

pandas.tseries.offsets.Easter.copy

Easter.**copy**(*self*)

pandas.tseries.offsets.Easter.isAnchored

Easter.**isAnchored**(*self*)

pandas.tseries.offsets.Easter.onOffset

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

pandas.tseries.offsets.Easter.is_anchored

Easter.**is_anchored**(*self*)

pandas.tseries.offsets.Easter.is_on_offset

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

pandas.tseries.offsets.Easter.__call__

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

3.8.32 Tick

Tick([n, normalize])

Attributes

pandas.tseries.offsets.Tick

class pandas.tseries.offsets.**Tick**(*n=1*, *normalize=False*)

Attributes

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

pandas.tseries.offsets.Tick.base

property Tick.**base**

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

delta	
freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply(self, other)	
apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.Tick.apply

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

pandas.tseries.offsets.Tick.apply_index

Tick.**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.Tick.rollback

Tick.**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.Tick.rollforward

Tick.**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__	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

Tick.delta
Tick.freqstr
Tick.kwds
Tick.name
Tick.nanos
Tick.normalize
Tick.rule_code

pandas.tseries.offsets.Tick.delta

property Tick.**delta**

pandas.tseries.offsets.Tick.freqstr

Tick.**freqstr**

pandas.tseries.offsets.Tick.kwds

property Tick.**kwds**

pandas.tseries.offsets.Tick.name

property Tick.**name**

pandas.tseries.offsets.Tick.nanos

property Tick.**nanos**

pandas.tseries.offsets.Tick.normalize

Tick.**normalize = False**

pandas.tseries.offsets.Tick.rule_code

property Tick.**rule_code**

Methods

Tick.copy(self)	
Tick.isAnchored(self)	
Tick.onOffset(self, dt)	
Tick.is_anchored(self)	
Tick.is_on_offset(self, dt)	
Tick.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.Tick.copy

Tick.**copy**(*self*)

pandas.tseries.offsets.Tick.isAnchored

Tick.**isAnchored**(*self*)

pandas.tseries.offsets.Tick.onOffset

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

pandas.tseries.offsets.Tick.is_anchored

Tick.**is_anchored**(*self*)

pandas.tseries.offsets.Tick.is_on_offset

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

pandas.tseries.offsets.Tick.__call__

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

3.8.33 Day

Day([n, normalize])	

Attributes

pandas.tseries.offsets.Day

class pandas.tseries.offsets.**Day**(*n=1*, *normalize=False*)

Attributes

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

pandas.tseries.offsets.Day.base

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

delta	
freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply(self, other)	
apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.Day.apply

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

pandas.tseries.offsets.Day.apply_index

Day.**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.Day.rollback

Day.**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.Day.rollforward

Day.**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__	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

Day.delta
Day.freqstr
Day.kwds
Day.name
Day.nanos
Day.normalize
Day.rule_code

pandas.tseries.offsets.Day.delta

property Day.**delta**

pandas.tseries.offsets.Day.freqstr

Day.**freqstr**

pandas.tseries.offsets.Day.kwds

property Day.**kwds**

pandas.tseries.offsets.Day.name

property Day.**name**

pandas.tseries.offsets.Day.nanos

property Day.**nanos**

pandas.tseries.offsets.Day.normalize

Day.**normalize = False**

pandas.tseries.offsets.Day.rule_code

property Day.**rule_code**

Methods

Day.copy(self)	
Day.isAnchored(self)	
Day.onOffset(self, dt)	
Day.is_anchored(self)	
Day.is_on_offset(self, dt)	
Day.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.Day.copy

Day.**copy**(*self*)

pandas.tseries.offsets.Day.isAnchored

Day.**isAnchored**(*self*)

pandas.tseries.offsets.Day.onOffset

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

pandas.tseries.offsets.Day.is_anchored

Day.**is_anchored**(*self*)

pandas.tseries.offsets.Day.is_on_offset

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

pandas.tseries.offsets.Day.__call__

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

3.8.34 Hour

Hour([n, normalize])	

Attributes

pandas.tseries.offsets.Hour

class pandas.tseries.offsets.**Hour**(*n=1*, *normalize=False*)

Attributes

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

pandas.tseries.offsets.Hour.base

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

delta	
freqstr	
kwds	
name	
nanos	
rule_code	

Methods

apply(self, other)	
apply_index(self, other)	Vectorized apply of DateOffset to DatetimeIndex, raises NotImplentedError for offsets without a vectorized implementation.
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.Hour.apply

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

pandas.tseries.offsets.Hour.apply_index

Hour.**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.Hour.rollback

Hour.**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.Hour.rollforward

Hour.**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__	
copy	
isAnchored	
is_anchored	
is_on_offset	
onOffset	

Properties

Hour.delta
Hour.freqstr
Hour.kwds
Hour.name
Hour.nanos
Hour.normalize
Hour.rule_code

pandas.tseries.offsets.Hour.delta

property Hour.**delta**

pandas.tseries.offsets.Hour.freqstr

Hour.**freqstr**

pandas.tseries.offsets.Hour.kwds

property Hour.**kwds**

pandas.tseries.offsets.Hour.name

property Hour.**name**

pandas.tseries.offsets.Hour.nanos

property Hour.**nanos**

pandas.tseries.offsets.Hour.normalize

Hour.**normalize = False**

pandas.tseries.offsets.Hour.rule_code

property Hour.**rule_code**

Methods

Hour.copy(self)	
Hour.isAnchored(self)	
Hour.onOffset(self, dt)	
Hour.is_anchored(self)	
Hour.is_on_offset(self, dt)	
Hour.__call__(self, other)	Call self as a function.

pandas.tseries.offsets.Hour.copy

Hour.**copy**(*self*)

pandas.tseries.offsets.Hour.isAnchored

Hour.**isAnchored**(*self*)

pandas.tseries.offsets.Hour.onOffset

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

pandas.tseries.offsets.Hour.is_anchored

Hour.**is_anchored**(*self*)

pandas.tseries.offsets.Hour.is_on_offset

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

pandas.tseries.offsets.Hour.__call__

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

3.8.35 Minute

Minute([n, normalize])	

Attributes

pandas.tseries.offsets.Minute

class pandas.tseries.offsets.**Minute**(*n=1*, *normalize=False*)

Attributes

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