Pandas – time series / date functionality

Pandas contains extensive capabilities and features for working with time series data for all domains (uses the NumPy datetime64 and timedelta64 dtypes)

Pandas captures 4 general time related concepts:

Date times: A specific date and time with timezone support. Similar to datetime.datetime from the standard library.

Time deltas: An absolute time duration. Similar to datetime.timedelta from the standard library.

Time spans: A span of time defined by a point in time and its associated frequency.

Date offsets: A relative time duration that respects calendar arithmetic. Similar to dateutil.relativedelta.relativedelta from the dateutil package.

Pandas – time series / date functionality

Pandas contains extensive capabilities and features for working with time series data for all domains (uses the NumPy datetime64 and timedelta64 dtypes)

Concept	Scalar Class	Array Class	pandas Data Type	Primary Creation Method
Date times	Timestamp	DatetimeIndex	datetime64[ns] or datetime64[ns, tz]	to_datetime or date_range
Time deltas	Timedelta	TimedeltaIndex	timedelta64[ns]	to_timedelta or timedelta_range
Time spans	Period	PeriodIndex	period[freq]	Period or period_range
Date offsets	DateOffset	None	None	DateOffset

Pandas – time series / date functionality

In [1]:
```
import pandas as pd
import numpy as np
import datetime
```

Parsing time from various sources and formats

In [2]:
```
pd.to_datetime(["25/12/2021", np.datetime64("2021-12-25"), datetime.datetime(2021,12,25)])
```

Out[2]:
```
DatetimeIndex(['2021-12-25', '2021-12-25', '2021-12-25'], dtype='datetime64[ns]', freq=None)
```

Generate sequences of fixed-frequency dates and time spans

In [3]:
```
dti = pd.date_range("2021-12-25", periods = 3, freq = "H")
dti
```

Out[3]:
```
DatetimeIndex(['2021-12-25 00:00:00', '2021-12-25 01:00:00',
               '2021-12-25 02:00:00'],
              dtype='datetime64[ns]', freq='H')
```

Defining and converting date times with timezone information

In [4]:
```
dti = dti.tz_localize("UTC")
dti
```

Out[4]:
```
DatetimeIndex(['2021-12-25 00:00:00+00:00', '2021-12-25 01:00:00+00:00',
               '2021-12-25 02:00:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='H')
```

In [5]:
```
dti = dti.tz_convert("US/Pacific")
dti
```

Out[5]:
```
DatetimeIndex(['2021-12-24 16:00:00-08:00', '2021-12-24 17:00:00-08:00',
               '2021-12-24 18:00:00-08:00'],
              dtype='datetime64[ns, US/Pacific]', freq='H')
```

Pandas – time series / date functionality

Resampling or converting a time series to a particular frequency

```
In [6]: idx = pd.date_range("2021-12-25", periods=5, freq="H")
        ts = pd.Series(range(len(idx)), index=idx)
        ts

Out[6]: 2021-12-25 00:00:00    0
        2021-12-25 01:00:00    1
        2021-12-25 02:00:00    2
        2021-12-25 03:00:00    3
        2021-12-25 04:00:00    4
        Freq: H, dtype: int64

In [7]: ts.resample("2H").mean()

Out[7]: 2021-12-25 00:00:00    0.5
        2021-12-25 02:00:00    2.5
        2021-12-25 04:00:00    4.0
        Freq: 2H, dtype: float64
```

Date and time arithmetic with absolute or relative increments

```
In [8]: friday = pd.Timestamp("2021-12-24")
        friday.day_name()

Out[8]: 'Friday'

In [9]: saturday = friday + pd.Timedelta("1 day")
        saturday.day_name()

Out[9]: 'Saturday'

In [10]: monday = friday + pd.offsets.BDay()
         monday.day_name()

Out[10]: 'Monday'
```

Pandas – time series / date functionality

Timestamps vs time spans

Timestamped data is the most basic type of time series data that associates values with points in time

A format argument can be provided

```
In [13]: pd.to_datetime("25-12-2021 00:00", format="%d-%m-%Y %H:%M")
Out[13]: Timestamp('2021-12-25 00:00:00')
```

The span represented by Period can be specified explicitly, or inferred from datetime string format

```
In [14]: pd.Period("2021-12")
Out[14]: Period('2021-12', 'M')

In [15]: pd.Period("2021-12", freq="D")
Out[15]: Period('2021-12-01', 'D')
```

Pandas – time series / date functionality

Time/date components

Property	Description
year	The year of the datetime
month	The month of the datetime
day	The days of the datetime
hour	The hour of the datetime
minute	The minutes of the datetime
second	The seconds of the datetime
microsecond	The microseconds of the datetime
nanosecond	The nanoseconds of the datetime
date	Returns datetime.date (does not contain timezone information)
time	Returns datetime.time (does not contain timezone information)
timetz	Returns datetime.time as local time with timezone information
dayofyear	The ordinal day of year
day_of_year	The ordinal day of year
weekofyear	The week ordinal of the year
week	The week ordinal of the year

Property	Description
dayofweek	The number of the day of the week with Monday=0, Sunday=6
day_of_week	The number of the day of the week with Monday=0, Sunday=6
weekday	The number of the day of the week with Monday=0, Sunday=6
quarter	Quarter of the date: Jan-Mar = 1, Apr-Jun = 2, etc.
days_in_month	The number of days in the month of the datetime
is_month_start	Logical indicating if first day of month (defined by frequency)
is_month_end	Logical indicating if last day of month (defined by frequency)
is_quarter_start	Logical indicating if first day of quarter (defined by frequency)
is_quarter_end	Logical indicating if last day of quarter (defined by frequency)
is_year_start	Logical indicating if first day of year (defined by frequency)
is_year_end	Logical indicating if last day of year (defined by frequency)
is_leap_year	Logical indicating if the date belongs to a leap year

Pandas – time series / date - DateOffset

DateOffset is similar to a Timedelta that represents a duration of time but follows specific calendar duration rules

A Timedelta day will always increment datetimes by 24 hours, while a DateOffset day will increment datetimes to the same time the next day whether a day represents 23, 24 or 25 hours due to daylight savings time

```
In [16]: ts = pd.Timestamp("2021-10-30 016:00:00", tz="Europe/Lisbon")

In [17]: ts + pd.Timedelta(days=1)
Out[17]: Timestamp('2021-10-31 15:00:00+0000', tz='Europe/Lisbon')

In [18]: ts + pd.DateOffset(days=1)
Out[18]: Timestamp('2021-10-31 16:00:00+0000', tz='Europe/Lisbon')
```

DateOffset shifts a date time by the corresponding calendar duration specified. The arithmetic operator (+) or the apply method can be used to perform the shift

```
In [22]: friday
Out[22]: Timestamp('2021-12-24 00:00:00')

In [19]: two_business_days = 2 * pd.offsets.BDay()

In [20]: two_business_days.apply(friday)
Out[20]: Timestamp('2021-12-28 00:00:00')

In [21]: friday + two_business_days
Out[21]: Timestamp('2021-12-28 00:00:00')
```

Pandas – time series / date - DateOffset

Date offsets and associated frequency strings

Date Offset	Frequency	Description
DateOffset	None	Generic offset class, defaults to absolute 24 hours
BDay or BusinessDay	'B'	business day (weekday)
CDay or CustomBusinesDay	'C'	custom business day
Week	'W'	one week, optionally anchored on a day of the week
WeekOfMonth	'WOM'	the x-th day of the y-th week of each month
LastWeekOfMonth	'LWOM'	the x-th day of the last week of each month
MonthEnd	'M'	calendar month end
MonthBegin	'MS'	calendar month begin
BMonthEnd or BusinessMonthEnd	'BM'	business month end
BMonthBegin or BusinesMonthBegin	'BMS'	business month begin
CBMonthEnd or CustomBusinessMonthEnd	'CBM'	custom business month end
CBMonthBegin or CustomBusinessMonthBegin	'CBMS'	custom business month begin
SemiMonthEnd	'SM'	15th (or other day_of_month) and calendar month end
SemiMonthBegin	'SMS'	15th (or other day_of_month) and calendar month begin

Date Offset	Frequency	Description
QuarterEnd	'Q'	calendar quarter end
QuarterBegin	'QS'	calendar quarter begin
BQuarterEnd	'BQ	business quarter end
BQuarterBegin	'BQS'	business quarter begin
FY5253Quarter	'REQ'	retail (aka 52-53 week) quarter
YearEnd	'A'	calendar year end
YearBegin	'AS' or 'BYS'	calendar year begin
BYearEnd	'BA'	business year end
BYearBegin	'BAS'	business year begin
FY5253	'RE'	retail (aka 52-53 week) year
Easter	None	Easter holiday
BusinessHour	'BH'	business hour
CustomBusinessHour	'CBH'	custom business hour
Day	'D'	one absolute day
Hour	'H'	one hour
Minute	'T' or 'min'	one minute
Second	'S'	one second
Milli	'L' or 'ms'	one millisecond
Micro	'U' or 'us'	one microsecond
Nano	'N'	one nanosecond