In [75]:
```
#native date types in python
#creating date object using datetime module
from datetime import datetime
date=datetime(year=2022,month=12,day=25)
print(date)
#using dateutil module and parser to parse date in string format and create object
from dateutil import parser
date1=parser.parse("24th of December,2022")
print(date1)
print("Day on which ",date," lands=")
print(date.strftime("%A"))#prinitng day form date object
```

```
2022-12-25 00:00:00
2022-12-24 00:00:00
Day on which  2022-12-25 00:00:00  lands=
Sunday
```

In [76]:
```
#numpy dates are encodeed with particular data type hence much efficient than python datetime
import numpy as np
date=np.array(['2015-07-04'],dtype=np.datetime64)
print(date)
#in this format it is possible to perform all vectorised operations on dates
print("performing vectorised operations on date array=")
print(date+np.arange(10))
print(np.datetime64("2022-12-24 12:00"))
print(np.datetime64("2022-12-24 12:00:59.59","Y"))#year code is "Y"
print(np.datetime64("2022-12-24 12:00:59.59","M"))#Month code is "M"
print(np.datetime64("2022-12-24 12:00:59.5923456","s"))#seconds code is "s"
print(np.datetime64("2022-12-24 12:00:59.5923456","ms"))#milli seconds code is "ms"
```

```
['2015-07-04']
performing vectorised operations on date array=
['2015-07-04' '2015-07-05' '2015-07-06' '2015-07-07' '2015-07-08'
 '2015-07-09' '2015-07-10' '2015-07-11' '2015-07-12' '2015-07-13']
2022-12-24T12:00
2022
2022-12
2022-12-24T12:00:59
2022-12-24T12:00:59.592
```

In [77]:
```
#dates and times in pandas
#to_datetime in pandas
import pandas as pd
date=pd.to_datetime("25th of December,2022")
print(date)
date#timestamp object allows us to use all native python features along with vectorised numpy operations
```

```
2022-12-25 00:00:00
```

Out[77]:
```
Timestamp('2022-12-25 00:00:00')
```

In [78]:
```
#vectorised operations on timestamp object
date + pd.to_timedelta(np.arange(12),"D")
```

Out[78]:
```
DatetimeIndex(['2022-12-25', '2022-12-26', '2022-12-27', '2022-12-28',
               '2022-12-29', '2022-12-30', '2022-12-31', '2023-01-01',
               '2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05'],
              dtype='datetime64[ns]', freq=None)
```

In [79]:
```
#pandas time series-Indexing by time
festivals=["Christmas","New Year","Diwali","Ramzan"]
index=["2022-12-25","2022-1-1","2022-11-12","22-4-21"]
ser=pd.Series(festivals,index=index)
print(ser)
print(ser["2022-12-25":"2022-11-12"])
```

```
2022-12-25    Christmas
2022-1-1       New Year
2022-11-12       Diwali
22-4-21          Ramzan
dtype: object
2022-12-25    Christmas
2022-1-1       New Year
2022-11-12       Diwali
dtype: object
```

In [80]:
```
#timestamp and datetimeIndex object-timestamp is the replacement for datetime in python and is based on numpy.dateitme64
#passing single date into pd.to_datetime() returns timestamp while passing multiple dates returns datetimeIndex
dates=pd.to_datetime(["2022-12-25","24th of December,2022","20221223","2022-Dec-22"])
dates
```

Out[80]:
```
DatetimeIndex(['2022-12-25', '2022-12-24', '2022-12-23', '2022-12-22'], dtype='datetime64[ns]', freq=None)
```

In [81]:
```
#for time periods pandas provides period type object which encodees frequency based on numpy.datetime64,index is TimedetlaIndex
dates=dates.to_period("D")#D indicates daily frequency
dates
```

Out[81]:
```
PeriodIndex(['2022-12-25', '2022-12-24', '2022-12-23', '2022-12-22'], dtype='period[D]')
```

In [82]:
```
#TimedeltaIndex is created when a date is subtrated from another date
print(dates-dates[0])
```

```
Index([<0 * Days>, <-1 * Day>, <-2 * Days>, <-3 * Days>], dtype='object')
```

In [83]:
```
#regular sequences in pandas dates
#date_range,period_range,timedelta_range
print(pd.date_range("2022-12-1","2022-12-31"))
#specifying date_range with number of periods
print(pd.date_range("2022-12-1",periods=7))#"D" represents that period is 1 day
print(pd.date_range("2022-12-1",periods=7,freq="H"))#"H" represents that period is 1 hour
#for timeperiod objects we use period_range
print(pd.period_range("2022-05",periods=5,freq="M"))#using period_range for timeperiod object and setting frequency as month
```

```
DatetimeIndex(['2022-12-01', '2022-12-02', '2022-12-03', '2022-12-04',
               '2022-12-05', '2022-12-06', '2022-12-07', '2022-12-08',
               '2022-12-09', '2022-12-10', '2022-12-11', '2022-12-12',
               '2022-12-13', '2022-12-14', '2022-12-15', '2022-12-16',
               '2022-12-17', '2022-12-18', '2022-12-19', '2022-12-20',
               '2022-12-21', '2022-12-22', '2022-12-23', '2022-12-24',
               '2022-12-25', '2022-12-26', '2022-12-27', '2022-12-28',
               '2022-12-29', '2022-12-30', '2022-12-31'],
              dtype='datetime64[ns]', freq='D')
DatetimeIndex(['2022-12-01', '2022-12-02', '2022-12-03', '2022-12-04',
               '2022-12-05', '2022-12-06', '2022-12-07'],
              dtype='datetime64[ns]', freq='D')
DatetimeIndex(['2022-12-01 00:00:00', '2022-12-01 01:00:00',
               '2022-12-01 02:00:00', '2022-12-01 03:00:00',
               '2022-12-01 04:00:00', '2022-12-01 05:00:00',
               '2022-12-01 06:00:00'],
              dtype='datetime64[ns]', freq='H')
PeriodIndex(['2022-05', '2022-06', '2022-07', '2022-08', '2022-09'], dtype='period[M]')
```

In [84]:
```
#frequency and offsets
#D Calendar day
print(pd.date_range("2022-12-1",periods=7,freq="D"))
#W Weekly
print(pd.date_range("2022-12-1",periods=7,freq="W"))
#M Month end
print(pd.date_range("2022-12-1",periods=7,freq="M"))
#Q Quarter end
print(pd.date_range("2022-12-1",periods=7,freq="Q"))
#A Year end
print(pd.date_range("2022-12-1",periods=7,freq="A"))
#H Hours
print(pd.date_range("2022-12-1",periods=7,freq="H"))
#T Minutes
print(pd.date_range("2022-12-1",periods=7,freq="T"))
#S Seconds
print(pd.date_range("2022-12-1",periods=7,freq="S"))
#L Milliseonds
#U Microseconds
#N nanoseconds
#BH Business hours
#BM Business month end
#BQ Business quarter end
#BA Business year end
#adding S at the start of month,quarter,year indicates frequency is calculated from start rather than end
#to mark annual or quarterly period we can add three letter month code Q-JAN,BQ-FEB,QS-MAR
print(pd.date_range("2022-12-1",periods=7,freq="Q-FEB"))
#to mark weekly frequency we add three letter code of day W-SUN,W-MON
print(pd.date_range("2022-12-1",periods=7,freq="W-MON"))
#codes can be combined with numbers
print(pd.date_range("2022-12-1",periods=7,freq="2H30T"))#frequency is set as 2 and half hours
```

```
DatetimeIndex(['2022-12-01', '2022-12-02', '2022-12-03', '2022-12-04',
               '2022-12-05', '2022-12-06', '2022-12-07'],
              dtype='datetime64[ns]', freq='D')
DatetimeIndex(['2022-12-04', '2022-12-11', '2022-12-18', '2022-12-25',
               '2023-01-01', '2023-01-08', '2023-01-15'],
              dtype='datetime64[ns]', freq='W-SUN')
DatetimeIndex(['2022-12-31', '2023-01-31', '2023-02-28', '2023-03-31',
               '2023-04-30', '2023-05-31', '2023-06-30'],
              dtype='datetime64[ns]', freq='M')
DatetimeIndex(['2022-12-31', '2023-03-31', '2023-06-30', '2023-09-30',
               '2023-12-31', '2024-03-31', '2024-06-30'],
              dtype='datetime64[ns]', freq='Q-DEC')
DatetimeIndex(['2022-12-31', '2023-12-31', '2024-12-31', '2025-12-31',
               '2026-12-31', '2027-12-31', '2028-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
DatetimeIndex(['2022-12-01 00:00:00', '2022-12-01 01:00:00',
               '2022-12-01 02:00:00', '2022-12-01 03:00:00',
               '2022-12-01 04:00:00', '2022-12-01 05:00:00',
               '2022-12-01 06:00:00'],
              dtype='datetime64[ns]', freq='H')
DatetimeIndex(['2022-12-01 00:00:00', '2022-12-01 00:01:00',
               '2022-12-01 00:02:00', '2022-12-01 00:03:00',
               '2022-12-01 00:04:00', '2022-12-01 00:05:00',
               '2022-12-01 00:06:00'],
              dtype='datetime64[ns]', freq='T')
DatetimeIndex(['2022-12-01 00:00:00', '2022-12-01 00:00:01',
               '2022-12-01 00:00:02', '2022-12-01 00:00:03',
               '2022-12-01 00:00:04', '2022-12-01 00:00:05',
               '2022-12-01 00:00:06'],
              dtype='datetime64[ns]', freq='S')
DatetimeIndex(['2023-02-28', '2023-05-31', '2023-08-31', '2023-11-30',
               '2024-02-29', '2024-05-31', '2024-08-31'],
              dtype='datetime64[ns]', freq='Q-FEB')
DatetimeIndex(['2022-12-05', '2022-12-12', '2022-12-19', '2022-12-26',
               '2023-01-02', '2023-01-09', '2023-01-16'],
              dtype='datetime64[ns]', freq='W-MON')
DatetimeIndex(['2022-12-01 00:00:00', '2022-12-01 02:30:00',
               '2022-12-01 05:00:00', '2022-12-01 07:30:00',
               '2022-12-01 10:00:00', '2022-12-01 12:30:00',
               '2022-12-01 15:00:00'],
              dtype='datetime64[ns]', freq='150T')
```

In [127]:
```
#resampling frequencies
df = pd.read_csv("apple.csv", parse_dates =["date"], index_col ="date")
print(df)
print("resampling frequency on close column=")
print(df["close"].resample("Y").mean())#freqeuncy si resampled to every year
print("resampling frequency on close column using asfreq=")
print(df["close"].asfreq("Y"))#freqeuncy at end of year is selected
#by default both leave up sampled points empty and hence we can use of method to fill nan values
print(df["close"].asfreq("Y",method="ffill"))#freqeuncy at end of year is selected
```

```
                     close         volume    open      high       low
date
2022-12-27 16:00:00  192.23     46,541,444  191.72  197.1800  191.4501
2018-11-13 00:00:00  192.23  46725710.0000  191.63  197.1800  191.4501
2018-11-12 00:00:00  194.17  50991030.0000  199.00  199.8500  193.7900
2018-11-09 00:00:00  204.47  34317760.0000  205.55  206.0100  202.2500
2018-11-08 00:00:00  208.49  25289270.0000  209.98  210.1200  206.7500
...                     ...            ...     ...       ...       ...
2017-11-17 00:00:00  170.15  21884010.0000  171.04  171.3900  169.6400
2017-11-16 00:00:00  171.10  23598650.0000  171.18  171.8700  170.3000
2017-11-15 00:00:00  169.08  28998220.0000  169.97  170.3197  168.3800
2017-11-14 00:00:00  171.34  24683350.0000  173.04  173.4800  171.1800
2017-11-13 00:00:00  173.97  16956290.0000  173.50  174.5000  173.4000

[254 rows x 5 columns]
resampling frequency on close column=
date
2017-12-31    171.970000
2018-12-31    191.635318
2019-12-31           NaN
2020-12-31           NaN
2021-12-31           NaN
2022-12-31    192.230000
Freq: A-DEC, Name: close, dtype: float64
resampling frequency on close column using asfreq=
date
2017-12-31   NaN
2018-12-31   NaN
2019-12-31   NaN
2020-12-31   NaN
2021-12-31   NaN
Freq: A-DEC, Name: close, dtype: float64
date
2017-12-31    172.26
2018-12-31    192.23
2019-12-31    192.23
2020-12-31    192.23
2021-12-31    192.23
Freq: A-DEC, Name: close, dtype: float64
```

In [134]:
```
#time shift
df = pd.read_csv("apple.csv", parse_dates =["date"], index_col ="date")
print(df)
df1=df.shift(2)#time is shifted by 2 days stock price at particular day is now stock price after two days later
print(df1)
#tshift will shift index,However tshift is deprecated and hence it is better to use shift
```

```
                     close         volume    open      high       low
date
2022-12-27 16:00:00  192.23     46,541,444  191.72  197.1800  191.4501
2018-11-13 00:00:00  192.23  46725710.0000  191.63  197.1800  191.4501
2018-11-12 00:00:00  194.17  50991030.0000  199.00  199.8500  193.7900
2018-11-09 00:00:00  204.47  34317760.0000  205.55  206.0100  202.2500
2018-11-08 00:00:00  208.49  25289270.0000  209.98  210.1200  206.7500
...                     ...            ...     ...       ...       ...
2017-11-17 00:00:00  170.15  21884010.0000  171.04  171.3900  169.6400
2017-11-16 00:00:00  171.10  23598650.0000  171.18  171.8700  170.3000
2017-11-15 00:00:00  169.08  28998220.0000  169.97  170.3197  168.3800
2017-11-14 00:00:00  171.34  24683350.0000  173.04  173.4800  171.1800
2017-11-13 00:00:00  173.97  16956290.0000  173.50  174.5000  173.4000

[254 rows x 5 columns]
                     close         volume    open      high       low
date
2022-12-27 16:00:00     NaN            NaN     NaN       NaN       NaN
2018-11-13 00:00:00     NaN            NaN     NaN       NaN       NaN
2018-11-12 00:00:00  192.23     46,541,444  191.72  197.1800  191.4501
2018-11-09 00:00:00  192.23  46725710.0000  191.63  197.1800  191.4501
2018-11-08 00:00:00  194.17  50991030.0000  199.00  199.8500  193.7900
...                     ...            ...     ...       ...       ...
2017-11-17 00:00:00  173.14  25047130.0000  170.78  173.7000  170.7800
2017-11-16 00:00:00  169.98  16041550.0000  170.29  170.5600  169.5600
2017-11-15 00:00:00  170.15  21884010.0000  171.04  171.3900  169.6400
2017-11-14 00:00:00  171.10  23598650.0000  171.18  171.8700  170.3000
2017-11-13 00:00:00  169.08  28998220.0000  169.97  170.3197  168.3800

[254 rows x 5 columns]
```

In [150]:
```
#rolling windows
df3=df["close"].rolling(5)#calculates mean from previous 5 values including itself hence first four values are nan
print(df3.mean())
print(df["open"].rolling(4).std())#calculates mean from previous 4 values including itself hence first three values are nan
```

```
date
2022-12-27 16:00:00        NaN
2018-11-13 00:00:00        NaN
2018-11-12 00:00:00        NaN
2018-11-09 00:00:00        NaN
2018-11-08 00:00:00    198.318
                        ...
2017-11-17 00:00:00    172.640
2017-11-16 00:00:00    171.866
2017-11-15 00:00:00    170.690
2017-11-14 00:00:00    170.330
2017-11-13 00:00:00    171.128
Name: close, Length: 254, dtype: float64
date
2022-12-27 16:00:00         NaN
2018-11-13 00:00:00         NaN
2018-11-12 00:00:00         NaN
2018-11-09 00:00:00    6.678705
2018-11-08 00:00:00    7.999446
                          ...
2017-11-17 00:00:00    1.364243
2017-11-16 00:00:00    0.391780
2017-11-15 00:00:00    0.583495
2017-11-14 00:00:00    1.275183
2017-11-13 00:00:00    1.643256
Name: open, Length: 254, dtype: float64
```