1. What is Datetime?

Datetime is python's datatype which can handle date and time functions with the help of datetime library.

In [ ]:

```
# necessary libraries
import pandas as pd
import datetime as dt
```

In [ ]:

```
dt1 = dt.time(4,30,10)
print(dt1)
print(type(dt1))
#Note - if we want to create time only
# format - h,m,s
```

```
04:30:10
<class 'datetime.time'>
```

In [ ]:

```
dt2 = dt.date(2022,7,7)
print(dt2)
print(type(dt2))
#Note - if we want to creat date only
# format -Y,m,d
```

```
2022-07-07
<class 'datetime.date'>
```

In [ ]:

```
#creating datetime
dt3 = dt.datetime(2023,7,4,4,30,00)
print(dt3)
print(type(dt3))
#Note - if we want to creat datetime
# format - Y,m,d,h,m,s
```

```
2023-07-04 04:30:00
<class 'datetime.datetime'>
```

2. What is Timestamp?

Timestamp is pandas datatype which can handle date and time functions

- different ways to create timestamp

In [ ]:

```
ts1 = pd.Timestamp("2022/06/05")
print(ts1)
print(type(ts1))
#note we are crating date although it taking  time by default
```

```
2022-06-05 00:00:00
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
```

In [ ]:

```
ts2=pd.Timestamp('2022 june 25 1:00pm')
print(ts2)
```

```
2022-06-25 13:00:00
```

In [ ]:

```
ts2=pd.Timestamp('2022 june 25 11:00am')
print(ts2)
```

```
2022-06-25 11:00:00
```

In [ ]:

```
ts3=pd.Timestamp('1:00:2pm')
print(ts3)

#we are creating time but it also taking date by default (today's)
```

```
2023-07-04 13:00:02
```

if we want only date or time component not datetime we can fetch them using date() and time() function

In [ ]:

```
ts4 = pd.Timestamp("2022/06/05").date()
print(ts4)
print(type(ts4))
```

```
2022-06-05
<class 'datetime.date'>
```

In [ ]:

```
ts5=pd.Timestamp('1:00:2pm').time()
print(ts5)
```

```
13:00:02
```

3. why there are two types of datetime what is difference between them?

- Timestamp object offers specialized features and integration with pandas over python datetime
- While Python's datetime module is powerful for general date and time operations, pandas' Timestamp object is optimized for handling time series data within the pandas ecosystem.

4. extracting different components from datetime object

In [ ]:

```
x = dt.datetime(2023,1,5,9,21,56)
print(x)
print(type(x))
print(x.year)
print(x.month)
print(x.date())
print(x.hour)
print(x.minute)
print(x.second)
```

```
2023-01-05 09:21:56
<class 'datetime.datetime'>
2023
1
2023-01-05
9
21
56
```

In [ ]:

```
y = pd.Timestamp("2023 June 25 12:10:5")
print(y)
print(type(y))

# fetching attributes
print(y.year)
print(y.month)
print(y.day)
print(y.hour)
print(y.minute)
print(y.second)
```

```
2023-06-25 12:10:05
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
2023
6
25
12
10
5
```

5. date_range (period and frequency)

- we can give maximum 3 parameters only at a time
- if we are using period we don't require end

In [ ]:

```
pd.date_range(start="2020",periods=10)
#starte with 2020 and give 10 dates
# by default feq will be 'D' daily
```

Out[ ]:

```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06', '2020-01-07', '2020-01-08',
               '2020-01-09', '2020-01-10'],
              dtype='datetime64[ns]', freq='D')
```

In [ ]:

```
pd.date_range(start="2020",periods=10, freq='M')
#Note: M --> month end freq
```

Out[ ]:

```
DatetimeIndex(['2020-01-31', '2020-02-29', '2020-03-31', '2020-04-30',
               '2020-05-31', '2020-06-30', '2020-07-31', '2020-08-31',
               '2020-09-30', '2020-10-31'],
              dtype='datetime64[ns]', freq='M')
```

In [ ]:

```
pd.date_range(start="2020", end="2022")
```

Out[ ]:

```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06', '2020-01-07', '2020-01-08',
               '2020-01-09', '2020-01-10',
               ...
               '2021-12-23', '2021-12-24', '2021-12-25', '2021-12-26',
               '2021-12-27', '2021-12-28', '2021-12-29', '2021-12-30',
               '2021-12-31', '2022-01-01'],
              dtype='datetime64[ns]', length=732, freq='D')
```

In [ ]:

```
pd.date_range(start="2020", end="2022", freq='MS')
#Note: M --> month start freq
```

Out[ ]:

```
DatetimeIndex(['2020-01-01', '2020-02-01', '2020-03-01', '2020-04-01',
               '2020-05-01', '2020-06-01', '2020-07-01', '2020-08-01',
               '2020-09-01', '2020-10-01', '2020-11-01', '2020-12-01',
               '2021-01-01', '2021-02-01', '2021-03-01', '2021-04-01',
               '2021-05-01', '2021-06-01', '2021-07-01', '2021-08-01',
               '2021-09-01', '2021-10-01', '2021-11-01', '2021-12-01',
               '2022-01-01'],
              dtype='datetime64[ns]', freq='MS')
```

Note: M --> month end freq

```
'D' -> 'day': Calendar day
'B' -> 'business day': Business day (excluding weekends)
'H' -> 'hour': Hourly frequency
'T' -> 'min': Minutely frequency
'S' -> 'sec': Secondly frequency
'W' -> 'week': Weekly frequency (on Sundays by default)
'M' -> 'month': Month end frequency
'MS' -> 'month': Month start frequency
'Q' -> 'quarter': Quarter end frequency
'A' -> 'year': Year end frequency
```

5. timedelta

In [ ]:

```
my_date = pd.Timestamp('2020 jan 31')
print(my_date)


my_date2 = my_date+pd.Timedelta(days=20)
print(my_date2)

print(my_date+pd.Timedelta(weeks=2))
print(my_date+pd.Timedelta(minutes=10))
print(my_date+pd.Timedelta(seconds=30))
print(my_date-pd.Timedelta(days=10))
print(my_date+pd.Timedelta(days=-10))
```

```
2020-01-31 00:00:00
2020-02-20 00:00:00
2020-02-14 00:00:00
2020-01-31 00:10:00
2020-01-31 00:00:30
2020-01-21 00:00:00
2020-01-21 00:00:00
```