Time Series

Time series data

- Time series data consists of a sequence of observations gathered over a period. It can be categorized into two types:
  - Regular interval, such as hourly or daily observations.
  - Irregular intervals, such as transaction times.

Date	Covid case
5/31/2021	344
5/30/2021	185
5/29/2021	199
5/28/2021	347
5/27/2021	384
5/26/2021	363
…	…

Time	Temperature
2022-09-20 12:00	15.3
2022-09-20 13:00	14.5
2022-09-20 14:00	14.5
2022-09-20 15:00	13.8
2022-09-20 16:00	13.0
2022-09-20 17:00	12.7
…	…

Time	Transaction ID	Amount
2022-03-10 15:14:32	363211	450
2022-03-10 16:05:45	363212	1260
2022-03-10 20:29:08	363213	3140
2022-03-10 20:51:27	363214	250
2022-03-11 01:33:18	363215	980
2022-03-11 07:22:23	363216	740
…	…	

Store Date/Time as string type

	date	weekly_sales
0	17/05/2021	238
1	10/05/2021	214
2	03/05/2021	195
3	27/04/2021	208
4	20/04/2021	220
5	13/04/2021	206

```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   date          6 non-null      object
 1   weekly_sales  6 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 224.0+ bytes
```

- Limitation:

```
sales_df.sort_values("date")
```

	date	weekly_sales
2	03/05/2021	195
1	10/05/2021	214
5	13/04/2021	206
0	17/05/2021	238
4	20/04/2021	220
3	27/04/2021	208

Unable to sort data by date.

```
sales_df[sales_df.date < "20/04/2021"]
```

	date	weekly_sales
0	17/05/2021	238
1	10/05/2021	214
2	03/05/2021	195
5	13/04/2021	206

Cannot select data by date.

Outline

- Python Datetime module
  - Datetime object
    - Conversion between string and datetime
  - Timedelta object

- Pandas
  - Function to_datetime()
  - DatetimeIndex
    - Subset selection
    - Information extraction
    - Method resample()

Python datetime module

- Python build-in datetime module includes different data types.

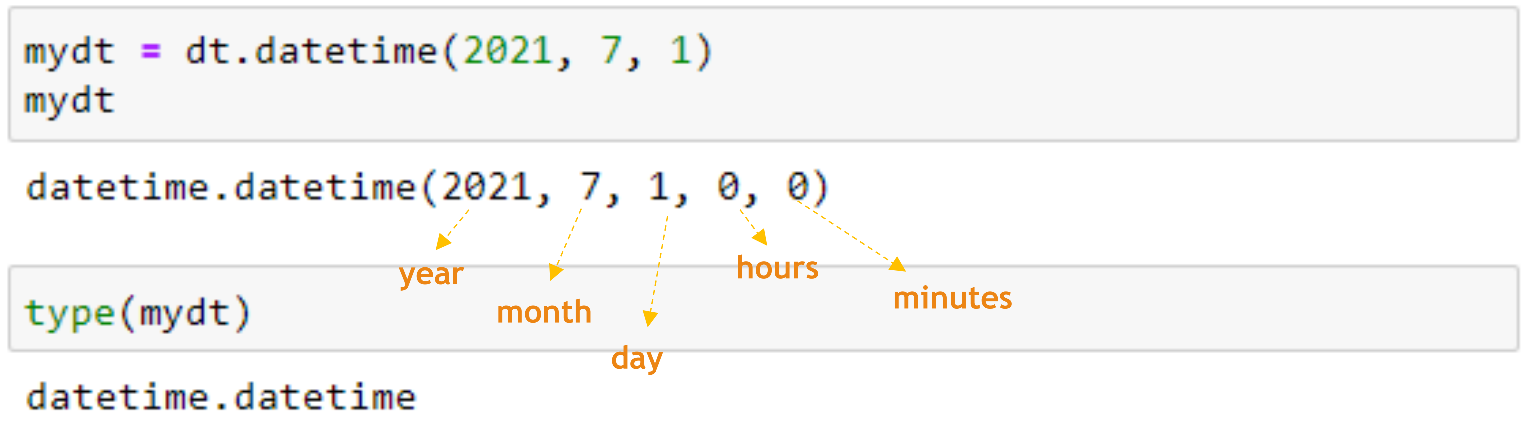
```
import datetime as dt
```

Data type (class)	Description
`date`	Stores calendar date(year, month, day).
`time`	Stores time of day as hours, minutes, seconds, and microseconds.
`datetime`	Store both date and time.
`timedelta`	Represents the difference between two datetime values.
`Tzinfo`	Base type for storing time zone information.

https://docs.python.org/3/library/datetime.html#datetime.datetime

Datetime object

- Use `datetime()` with three arguments `year`, `month` and `day` to create a datetime object.



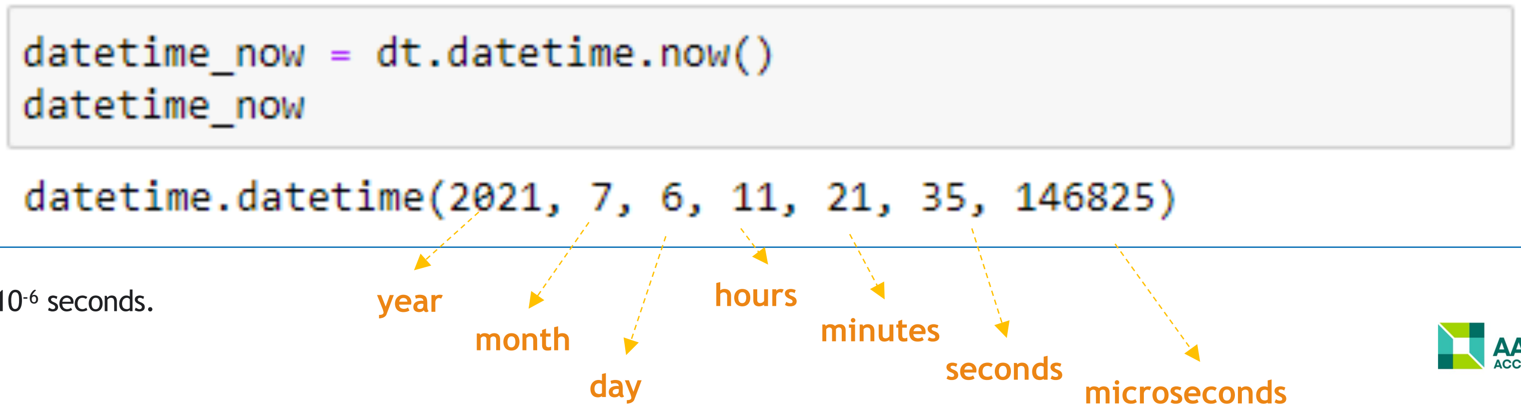
```
mydt = dt.datetime(2021, 7, 1)
mydt
```

```
datetime.datetime(2021, 7, 1, 0, 0)
```

```
type(mydt)
```

```
datetime.datetime
```

- Use method `now()` to create a datetime object.



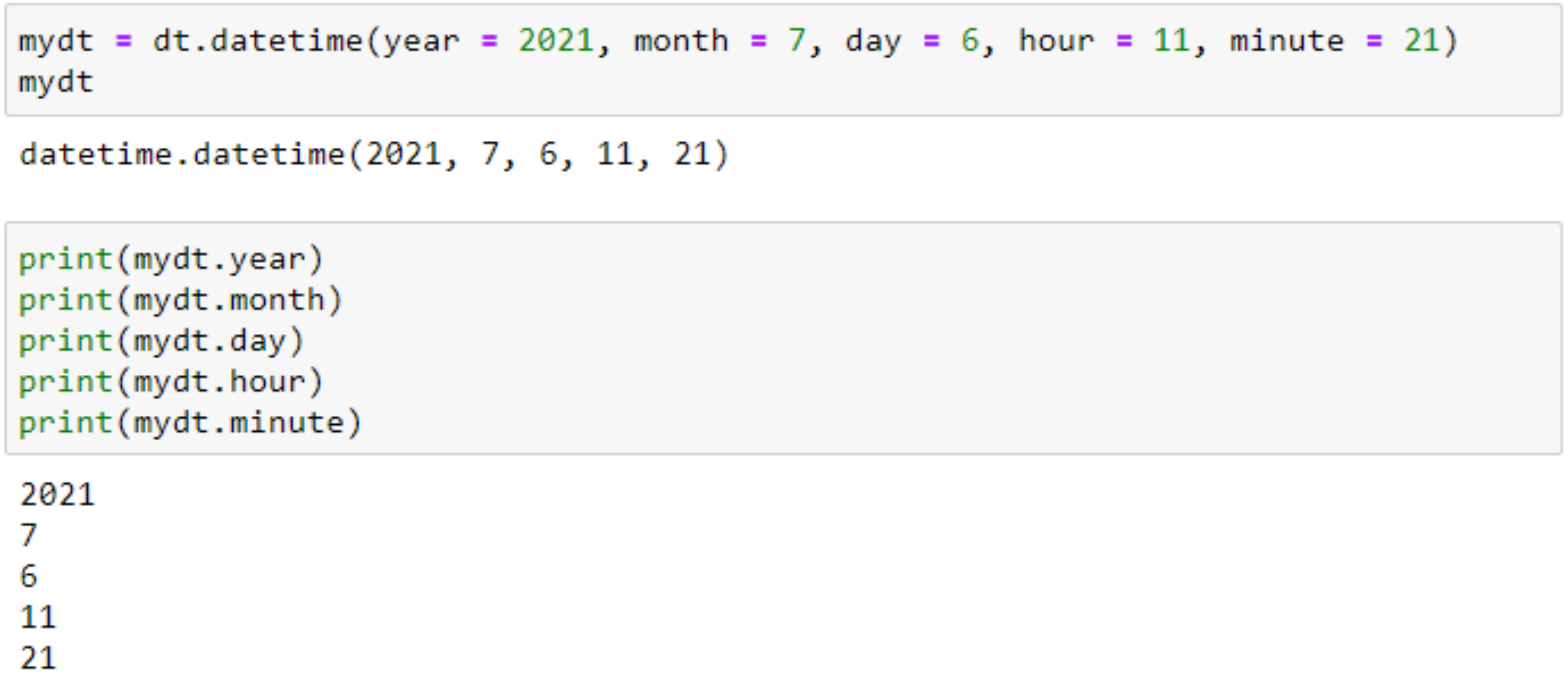
```
datetime_now = dt.datetime.now()
datetime_now
```

```
datetime.datetime(2021, 7, 6, 11, 21, 35, 146825)
```

1 microsecond = 1 × 10^{-6} seconds.

Datetime object - attributes

- Access attributes



```
mydt = dt.datetime(year = 2021, month = 7, day = 6, hour = 11, minute = 21)
mydt
```

```
datetime.datetime(2021, 7, 6, 11, 21)
```

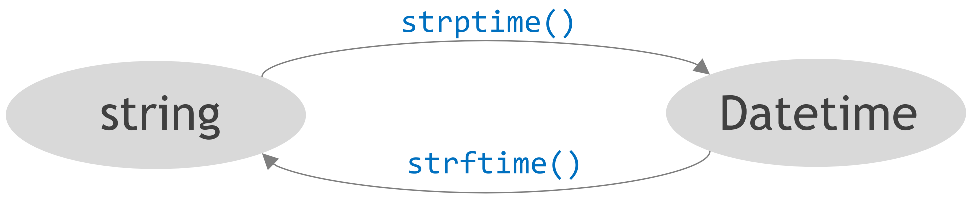
```
print(mydt.year)
print(mydt.month)
print(mydt.day)
print(mydt.hour)
print(mydt.minute)
```

```
2021
7
6
11
21
```

Datetime object - methods

- Some datetime methods

Methods	Description
`isocalendar()`	Returns a tuple year, week, and weekday
`isoweekday()`	Returns the day of the week as integer where Monday is 1 and Sunday is 7
`ctime()`	Returns a string representation of date and time
`strptime()`	Returns a DateTime object corresponding to the date string
`strftime()`	Returns a string representation of the DateTime object with the given format



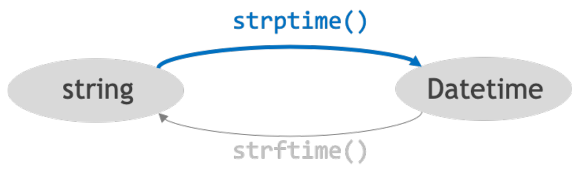
- Datetime methods: https://www.geeksforgeeks.org/python-datetime-datetime-class/
- `strptime` stands for string-parse-time.
- `strftime` stands for string-format-time.

Common datetime formats

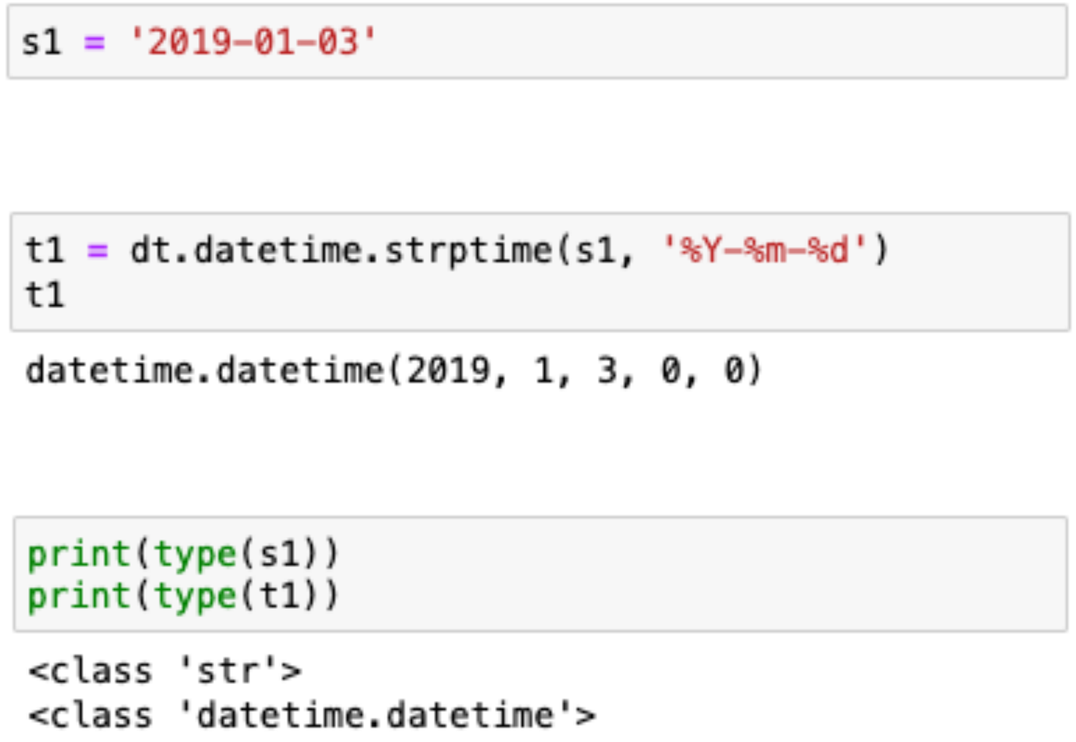
- Examples

General form	Example
`mm/dd/yy`	03/28/21
`dd/mm/yy`	28/03/21
`dd.mm.yyyy`	28.03.2021
`dd-mmm-yyyy`	28-Mar-2021
`hh:mm`	01:02
`hh:mm:ss.s`	01:02:34.75
`yyyy-mm-dd hh:mm`	2021-03-28 01:02
`yyyy-mm-dd hh:mm:ss.s`	2021-03-28 01:02:34.7

Datetime object – strptime()



- Convert a string to a datetime object by using `strptime()`.



```
s1 = '2019-01-03'
```

```
t1 = dt.datetime.strptime(s1, '%Y-%m-%d')
t1
```

```
datetime.datetime(2019, 1, 3, 0, 0)
```

```
print(type(s1))
print(type(t1))
```

```
<class 'str'>
<class 'datetime.datetime'>
```

Directive	Meaning
%Y	Four-digit year
%y	Two-digit year
%m	Two-digit month [01,12]
%d	Two-digit day [01,31]
%H	Hour (24-hour clock) [00,23]
%M	Two-digit minute [00,59]
%S	Two-digit minute [00,59]

Format codes: https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

$$\begin{tikzcd}\text{string} \arrow[r, bend left, "\texttt{strptime()}"] & \text{Datetime} \arrow[l, bend left, "\texttt{strftime()}"]\end{tikzcd}$$

Datetime object – strptime()

- Other formats

```
s2 = '03/01/2019'
t2 = dt.datetime.strptime(s2, '%d/%m/%Y')
t2
```

```
datetime.datetime(2019, 1, 3, 0, 0)
```

```
s3 = '03/01/19'
t3 = dt.datetime.strptime(s3, '%d/%m/%y')
t3
```

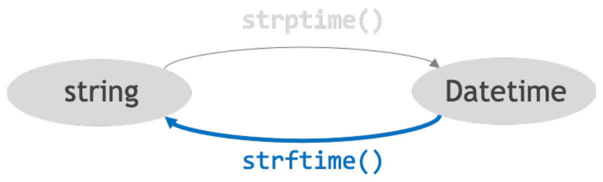
```
datetime.datetime(2019, 1, 3, 0, 0)
```

```
s4 = '10:30 03/01/19'
t4 = dt.datetime.strptime(s4, '%H:%M %d/%m/%y')
t4
```

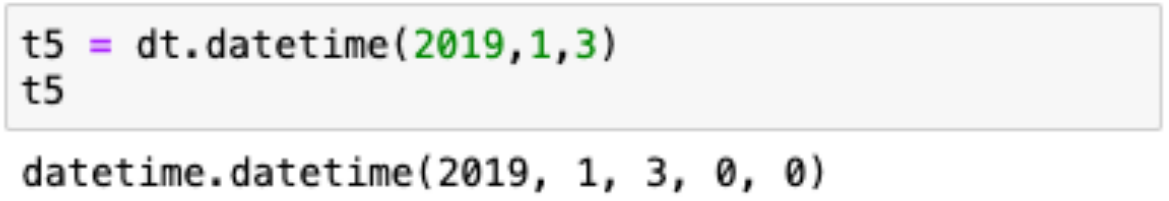
```
datetime.datetime(2019, 1, 3, 10, 30)
```

Directive	Meaning
%Y	Four-digit year
%y	Two-digit year
%m	Two-digit month [01,12]
%d	Two-digit day [01,31]
%H	Hour (24-hour clock) [00,23]
%M	Two-digit minute [00,59]
%S	Two-digit minute [00,59]

Datetime object – strftime()

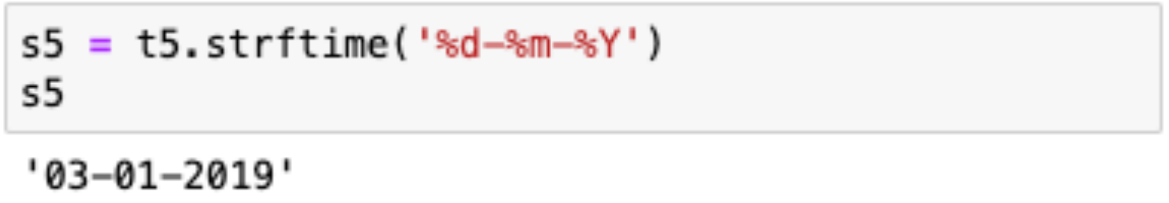


- Convert a datetime object to a string by using `strftime()`.



```
t5 = dt.datetime(2019,1,3)
t5
```

```
datetime.datetime(2019, 1, 3, 0, 0)
```



```
s5 = t5.strftime('%d-%m-%Y')
s5
```

```
'03-01-2019'
```

Timedelta object

- Timedelta is used to represent the duration between two timestamps.
- A timedelta object can be created by two datetime objects.

Example-1

```
t1 = dt.datetime(2021, 6, 15)
t2 = dt.datetime(2021, 7, 6)
```

```
diff_12 = t2 - t1
diff_12
```

```
datetime.timedelta(days=21)
```

```
diff_12.days
```

Attribute “days”

```
21
```

Example-2

```
t3 = dt.datetime(2021, 6, 15, 10, 12, 40)
t4 = dt.datetime(2021, 6, 15, 10, 13, 20)
```

```
diff_34 = t4 - t3
diff_34
```

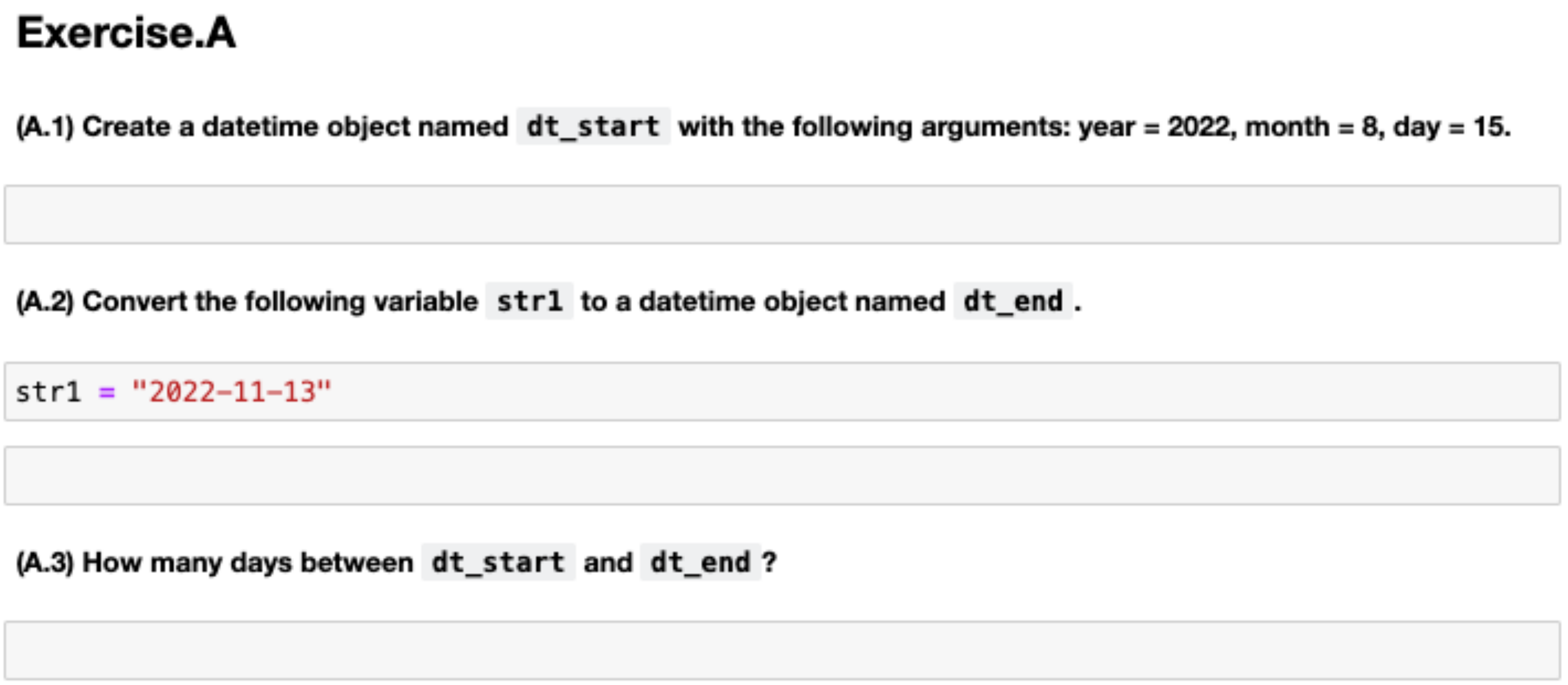
```
datetime.timedelta(seconds=40)
```

```
diff_34.seconds
```

Attribute “seconds”

```
40
```

Exercise



Exercise.A

(A.1) Create a datetime object named `dt_start` with the following arguments: year = 2022, month = 8, day = 15.

```
```

(A.2) Convert the following variable `str1` to a datetime object named `dt_end`.

```
str1 = "2022-11-13"
```

```
```

(A.3) How many days between `dt_start` and `dt_end` ?

```
```

Outline

- Python Datetime module
  - Datetime object
    - Conversion between string and datetime
  - Timedelta object

- Pandas
  - Function to_datetime()
  - DatetimeIndex
    - Subset selection
    - Information extraction
    - Method resample()

Pandas data types

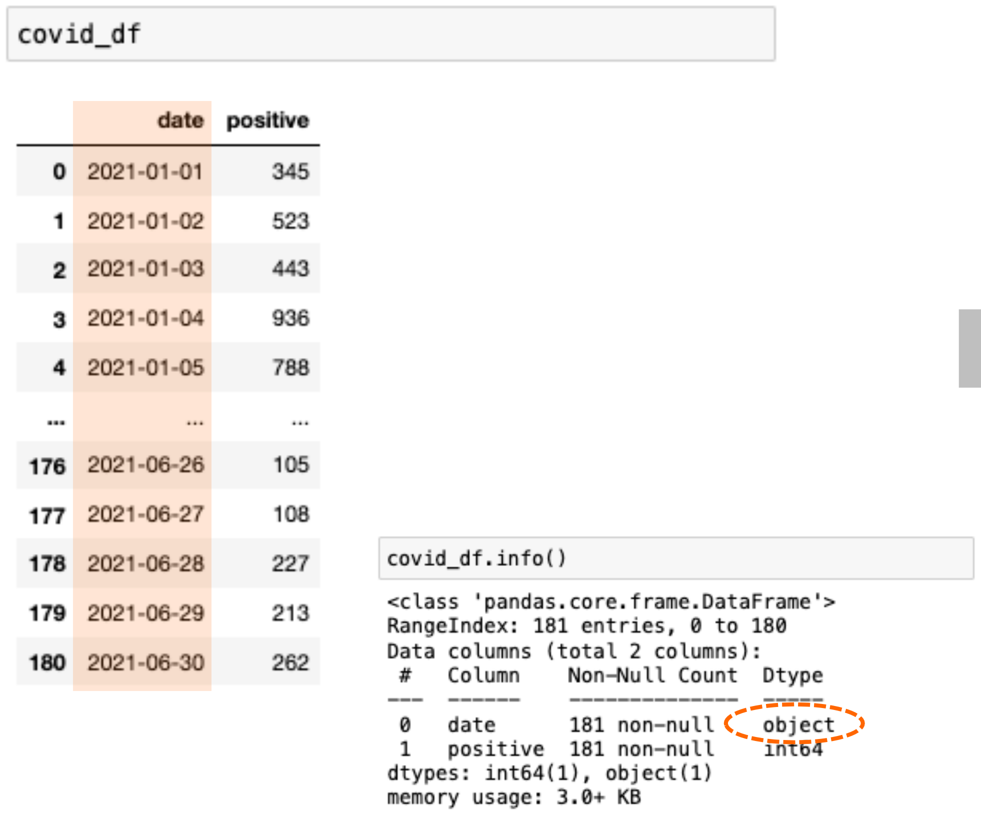
- Pandas `dtype` mapping

Pandas dtype	Python type	Usage
int64	int	Integer numbers
float64	float	Floating point numbers
object	str or mixed	Text or mixed numeric and non-numeric values
bool	bool	True/False values
datetime	datetime	A specific time point
timedelta	timedelta	Duration between two points in time
period	-	A time span

https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#overview

Pandas – to_datetime

- Use the function `to_datetime()` to convert a column to a datetime type.

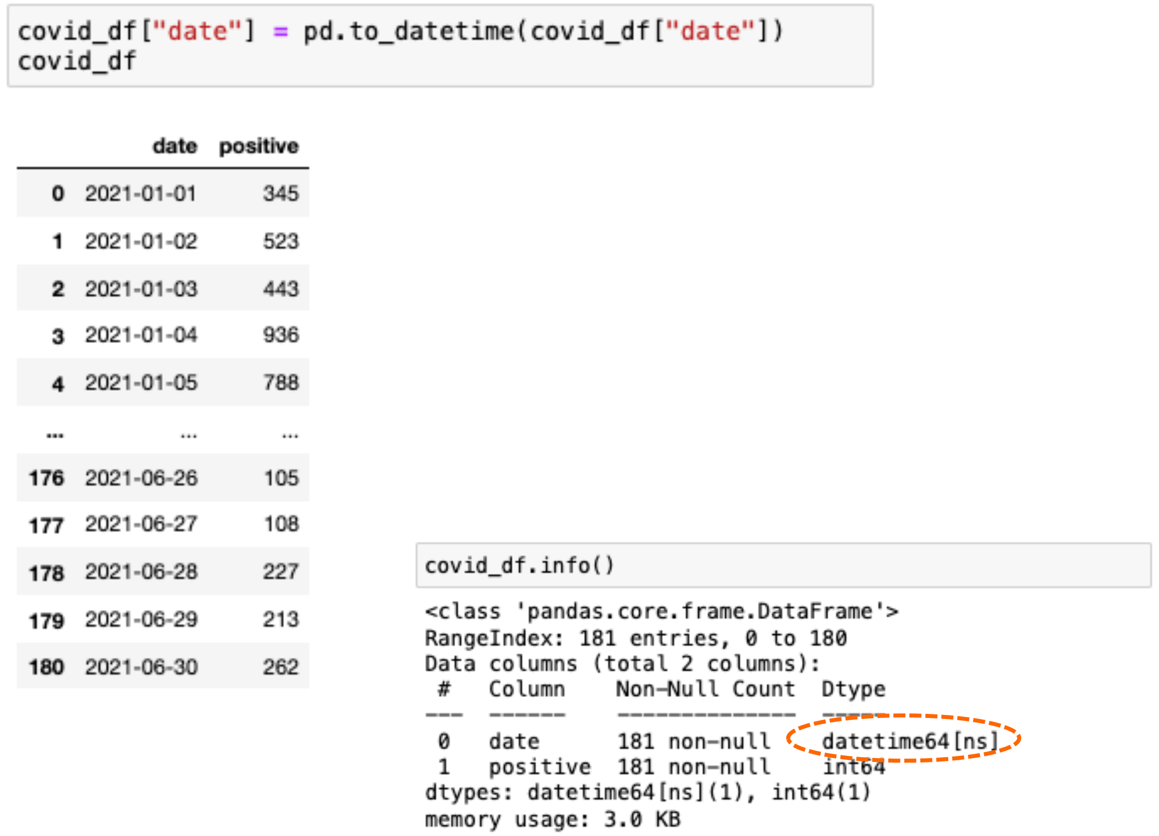


```
covid_df
```

	date	positive
0	2021-01-01	345
1	2021-01-02	523
2	2021-01-03	443
3	2021-01-04	936
4	2021-01-05	788
...	...	...
176	2021-06-26	105
177	2021-06-27	108
178	2021-06-28	227
179	2021-06-29	213
180	2021-06-30	262

```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181 entries, 0 to 180
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      181 non-null    object
 1   positive  181 non-null    int64
dtypes: int64(1), object(1)
memory usage: 3.0+ KB
```



```
covid_df["date"] = pd.to_datetime(covid_df["date"])
covid_df
```

	date	positive
0	2021-01-01	345
1	2021-01-02	523
2	2021-01-03	443
3	2021-01-04	936
4	2021-01-05	788
...	...	...
176	2021-06-26	105
177	2021-06-27	108
178	2021-06-28	227
179	2021-06-29	213
180	2021-06-30	262

```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 181 entries, 0 to 180
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   date      181 non-null    datetime64[ns]
 1   positive  181 non-null    int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 3.0 KB
```

- Pandas to_datetime() function parses many different types of date and time formats.
- https://pandas.pydata.org/docs/reference/api/pandas.to_datetime.html

DatetimeIndex

- Use `parse_dates()` to specify a list of column names that should be parsed as dates.

```
covid_df = pd.read_csv("../dataset/covid_2021.csv", parse_dates=["date"], index_col = 0)
covid_df.head(10)
```

	positive
date	
2021-01-01	345
2021-01-02	523
2021-01-03	443
2021-01-04	936
2021-01-05	788
2021-01-06	708
2021-01-07	735
2021-01-08	649
2021-01-09	414
2021-01-10	435

```
covid_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 181 entries, 2021-01-01 to 2021-06-30
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   positive  181 non-null    int64
dtypes: int64(1)
memory usage: 6.9 KB
```

DatetimeIndex - subset selection

- A `DatetimeIndex` contains date-related properties and supports convenient slicing.
  - Select a subset by month

```
covid_df.loc['2021-05',:]
```

	positive
date	
2021-05-01	251
2021-05-02	296
2021-05-03	510
2021-05-04	463
2021-05-05	494
2021-05-06	509
2021-05-07	438
2021-05-08	352
2021-05-09	351
2021-05-10	523
2021-05-11	427

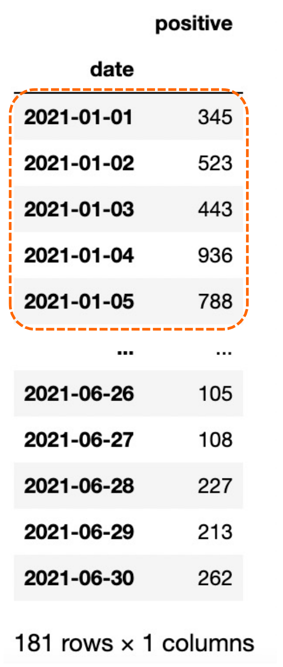
  - Select a subset by a range

```
covid_df.loc['2021-05-25':'2021-06-01',:]
```

	positive
date	
2021-05-25	427
2021-05-26	363
2021-05-27	384
2021-05-28	347
2021-05-29	199
2021-05-30	185
2021-05-31	344
2021-06-01	386

DatetimeIndex - slice data using iloc and loc

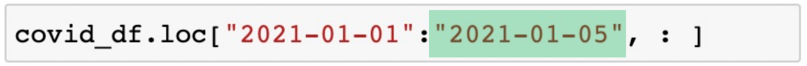
- Use loc: Data for "end_index" will be included.
- Use iloc: Data for "end_index" will not be included.



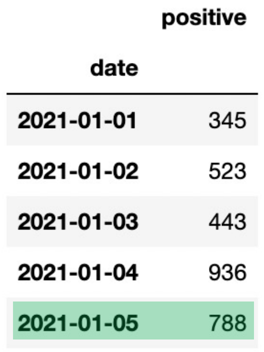
	positive
date	
2021-01-01	345
2021-01-02	523
2021-01-03	443
2021-01-04	936
2021-01-05	788
...	...
2021-06-26	105
2021-06-27	108
2021-06-28	227
2021-06-29	213
2021-06-30	262

181 rows × 1 columns

```
df.loc[start_indx : end_index]
```

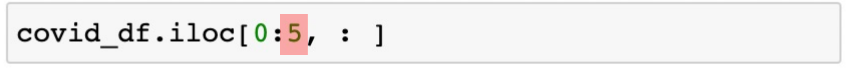


```
covid_df.loc["2021-01-01":"2021-01-05", : ]
```

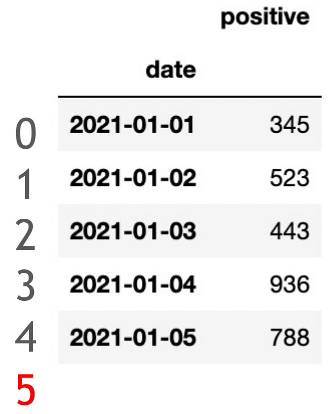


	positive
date	
2021-01-01	345
2021-01-02	523
2021-01-03	443
2021-01-04	936
2021-01-05	788

```
df.iloc[start_indx : end_index]
```



```
covid_df.iloc[0:5, : ]
```



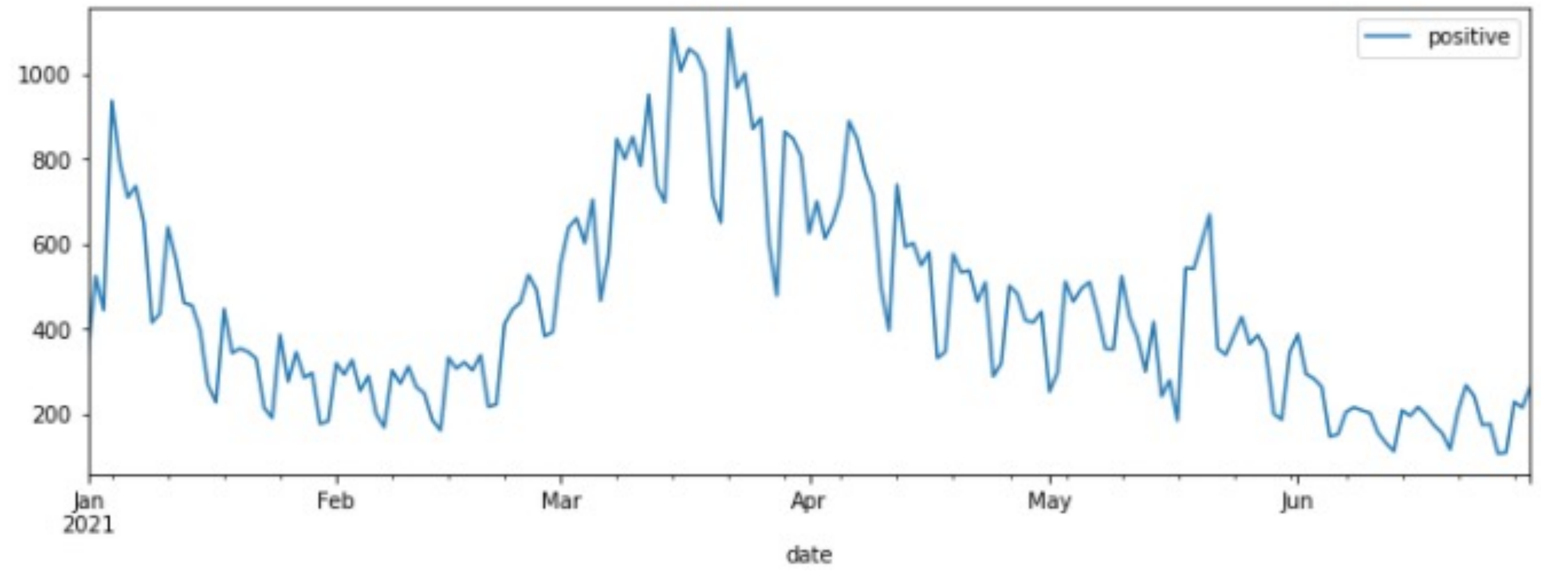
		positive
	date	
0	2021-01-01	345
1	2021-01-02	523
2	2021-01-03	443
3	2021-01-04	936
4	2021-01-05	788
5		

DatetimeIndex – line chart

- Use `plot()` to plot a line chart.

```
covid_df.plot(y = 'positive', figsize = (12,4))
```

```
<AxesSubplot:xlabel='date'>
```



Exercise

(B.1) Import dataset `fashion.csv` and set the column `Date` as DatetimeIndex.

(B.2) Draw a line chart to show Tiger_of_Sweden's sales in 2016.

(B.3) Use a multiple line chart to show the sales of Eton, Levi_s, and Tiger_of_Sweden from 2014 to 2016.

Information extraction

- Attributes

Attribute	Description
`year`	The year of the datetime.
`month`	The month as January=1, December=12.
`day`	The day of the datetime.
`hour`	The hours of the datetime.
`weekday`	The day of the week with Monday=0, Sunday=6.
`quarter`	The quarter of the date.

- Methods

Method	Description
`month_name()`	Return the month names
`day_name()`	Return the day of the week.

https://pandas.pydata.org/docs/reference/api/pandas.DatetimeIndex.html

DatetimeIndex - information extraction

- Add new columns.

```
covid_df["month"] = covid_df.index.month
covid_df
```

	positive	month
date		
2021-01-01	345	1
2021-01-02	523	1
2021-01-03	443	1
2021-01-04	936	1
2021-01-05	788	1
...	...	...
2021-06-26	105	6
2021-06-27	108	6
2021-06-28	227	6
2021-06-29	213	6
2021-06-30	262	6

```
covid_df["day_of_week"] = covid_df.index.day_name()
covid_df
```

	positive	month	day_of_week
date			
2021-01-01	345	1	Friday
2021-01-02	523	1	Saturday
2021-01-03	443	1	Sunday
2021-01-04	936	1	Monday
2021-01-05	788	1	Tuesday
...	...	...	...
2021-06-26	105	6	Saturday
2021-06-27	108	6	Sunday
2021-06-28	227	6	Monday
2021-06-29	213	6	Tuesday
2021-06-30	262	6	Wednesday

DatetimeIndex - information extraction

- Calculate group statistics

```
covid_df
```

date	positive	month	day_of_week
2021-01-01	345	1	Friday
2021-01-02	523	1	Saturday
2021-01-03	443	1	Sunday
2021-01-04	936	1	Monday
2021-01-05	788	1	Tuesday
...	...	...	...
2021-06-26	105	6	Saturday
2021-06-27	108	6	Sunday
2021-06-28	227	6	Monday
2021-06-29	213	6	Tuesday
2021-06-30	262	6	Wednesday

```
covid_df.groupby("day_of_week").positive.mean()
```

```
day_of_week
Friday         480.538462
Monday         521.153846
Saturday       335.961538
Sunday         325.000000
Thursday       476.760000
Tuesday        506.115385
Wednesday      501.961538
Name: positive, dtype: float64
```

➔ Group data by day of the week to compare patterns between weekdays and weekends.

DatetimeIndex - resampling

- The method `resample()` is used for frequency conversion of time series data.
- The method `resample()` will return a `Resampler` object, which contains functions to aggregate data.

Step1: Get a Resampler object

```
covid_rs = covid_df.resample('M')
type(covid_rs)
```

Aggregate daily data to monthly data

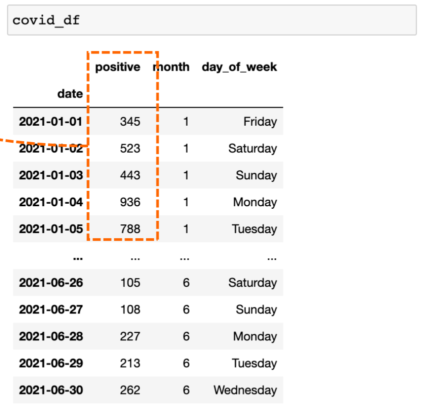
```
pandas.core.resample.DatetimeIndexResampler
```

Step2: Call an aggregate function

```
covid_month = covid_rs.positive.sum()
covid_month
```

```
date
2021-01-31    13138
2021-02-28     8709
2021-03-31    24853
2021-04-30    16610
2021-05-31    12082
2021-06-30     5966
Freq: M, Name: positive, dtype: int64
```

Number of cases reported from 01/01 to 01/31



```
covid_df
```

date	positive	month	day_of_week
2021-01-01	345	1	Friday
2021-01-02	523	1	Saturday
2021-01-03	443	1	Sunday
2021-01-04	936	1	Monday
2021-01-05	788	1	Tuesday
...	...	...	...
2021-06-26	105	6	Saturday
2021-06-27	108	6	Sunday
2021-06-28	227	6	Monday
2021-06-29	213	6	Tuesday
2021-06-30	262	6	Wednesday

https://pandas.pydata.org/pandas-docs/stable/reference/resampling.html

DatetimeIndex - resampling

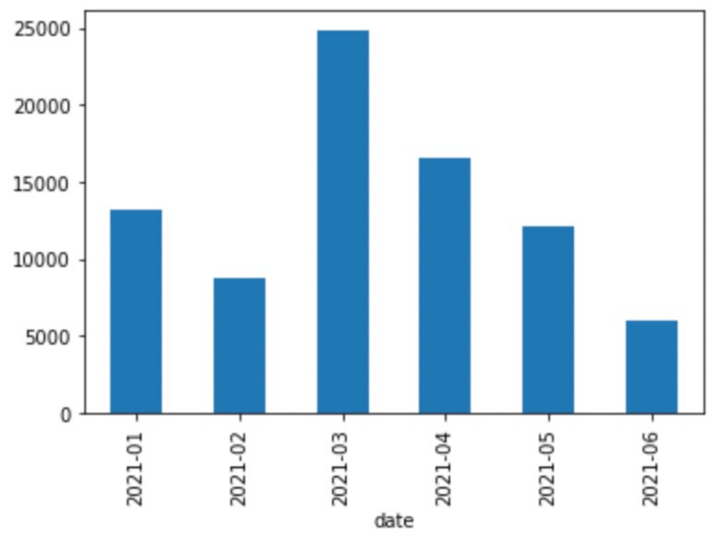
- Use `to_period()` to cast to index at a particular frequency.

```
covid_month.index = covid_month.index.to_period('M')
covid_month
```

```
date
2021-01    13138
2021-02     8709
2021-03    24853
2021-04    16610
2021-05    12082
2021-06     5966
Freq: M, Name: positive, dtype: int64
```

```
covid_month.plot(kind = "bar", y = "positive")
```

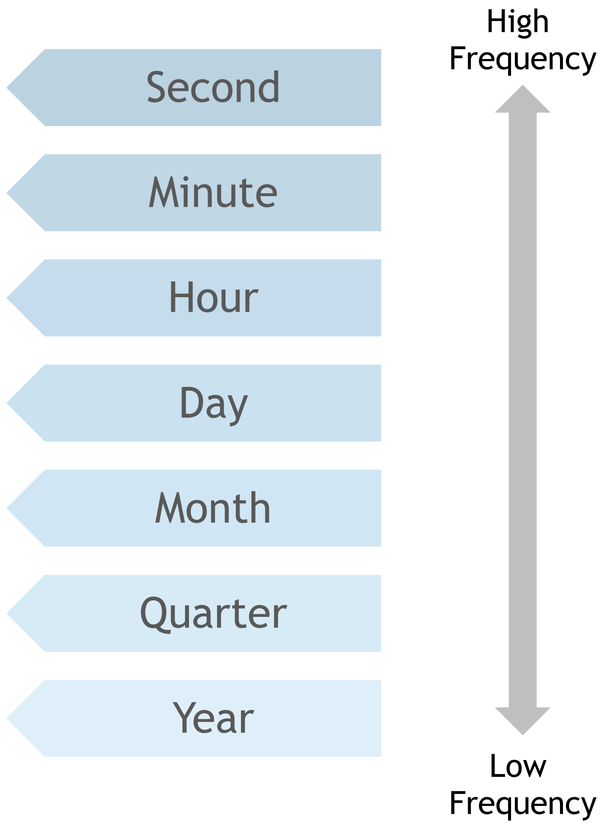
```
<AxesSubplot:xlabel='date'>
```



https://pandas.pydata.org/docs/reference/api/pandas.Series.dt.to_period.html

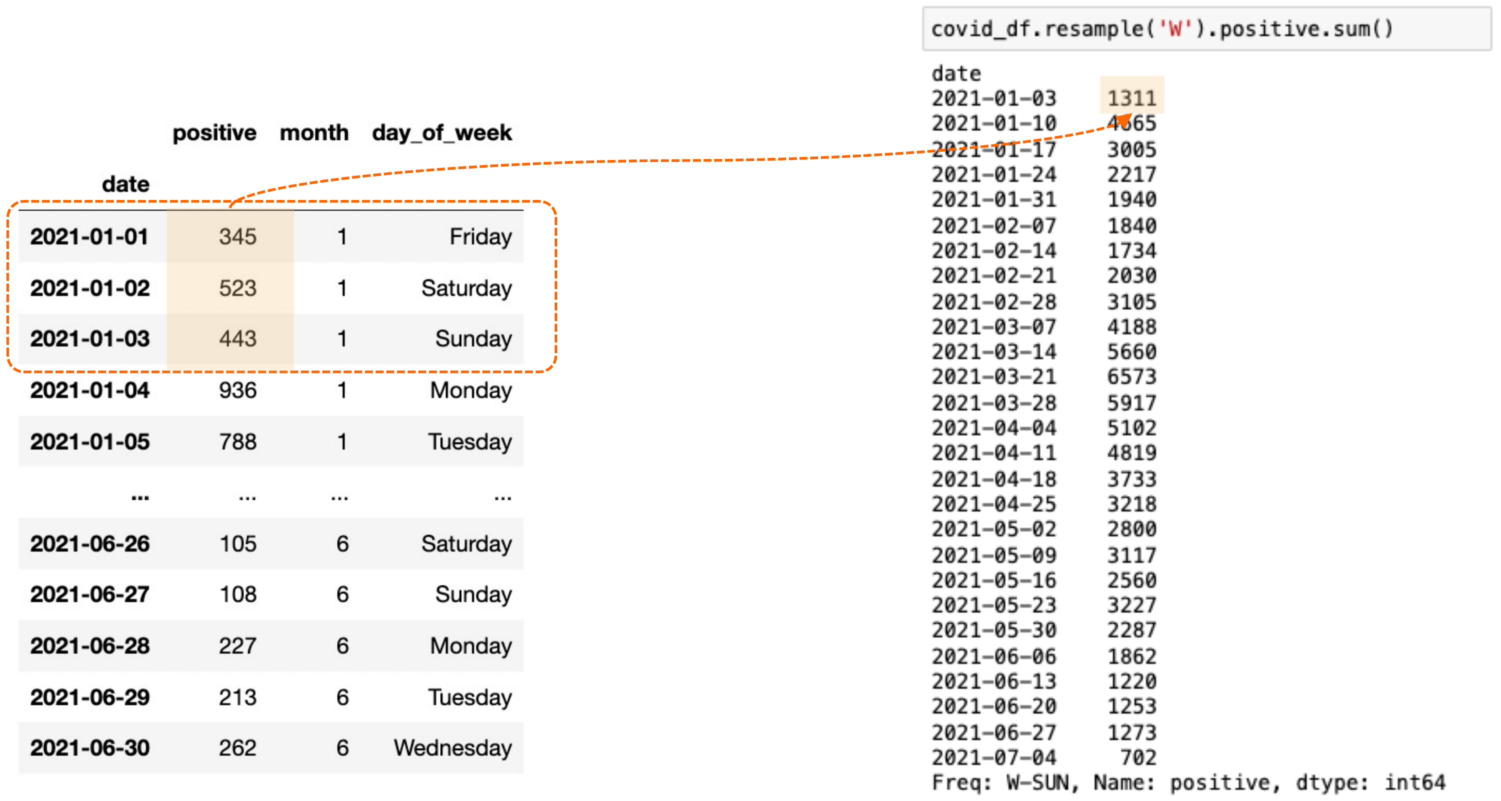
Resampling - frequencies

Alias	Description
H	hourly frequency
T or min	minutely frequency
S	secondly frequency
D	calendar day frequency
B	business day frequency
W	weekly frequency
M	month end frequency
MS	month start frequency
Q	quarter end frequency
QS	quarter start frequency
A	year end frequency
AS	year start frequency



Resampling – more examples

- Example-1: Calculate weekly total number of positive cases



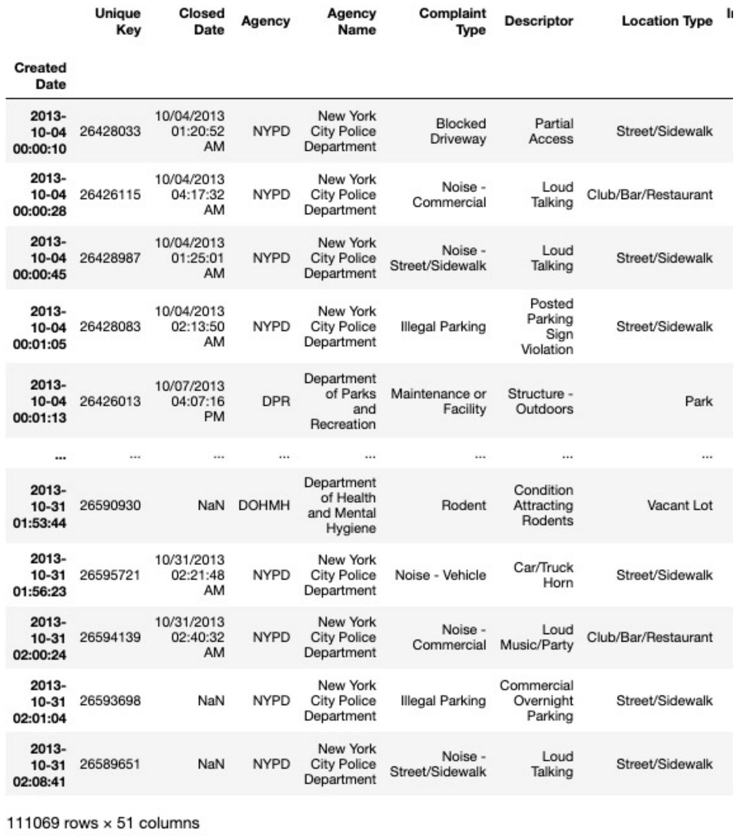
date	positive	month	day_of_week
2021-01-01	345	1	Friday
2021-01-02	523	1	Saturday
2021-01-03	443	1	Sunday
2021-01-04	936	1	Monday
2021-01-05	788	1	Tuesday
...	...	...	...
2021-06-26	105	6	Saturday
2021-06-27	108	6	Sunday
2021-06-28	227	6	Monday
2021-06-29	213	6	Tuesday
2021-06-30	262	6	Wednesday

```
covid_df.resample('W').positive.sum()
```

```
date
2021-01-03    1311
2021-01-10    4865
2021-01-17    3005
2021-01-24    2217
2021-01-31    1940
2021-02-07    1840
2021-02-14    1734
2021-02-21    2030
2021-02-28    3105
2021-03-07    4188
2021-03-14    5660
2021-03-21    6573
2021-03-28    5917
2021-04-04    5102
2021-04-11    4819
2021-04-18    3733
2021-04-25    3218
2021-05-02    2800
2021-05-09    3117
2021-05-16    2560
2021-05-23    3227
2021-05-30    2287
2021-06-06    1862
2021-06-13    1220
2021-06-20    1253
2021-06-27    1273
2021-07-04     702
Freq: W-SUN, Name: positive, dtype: int64
```

Resampling – more examples

- Example-2: Use `size()` to calculate the number of complaints per day

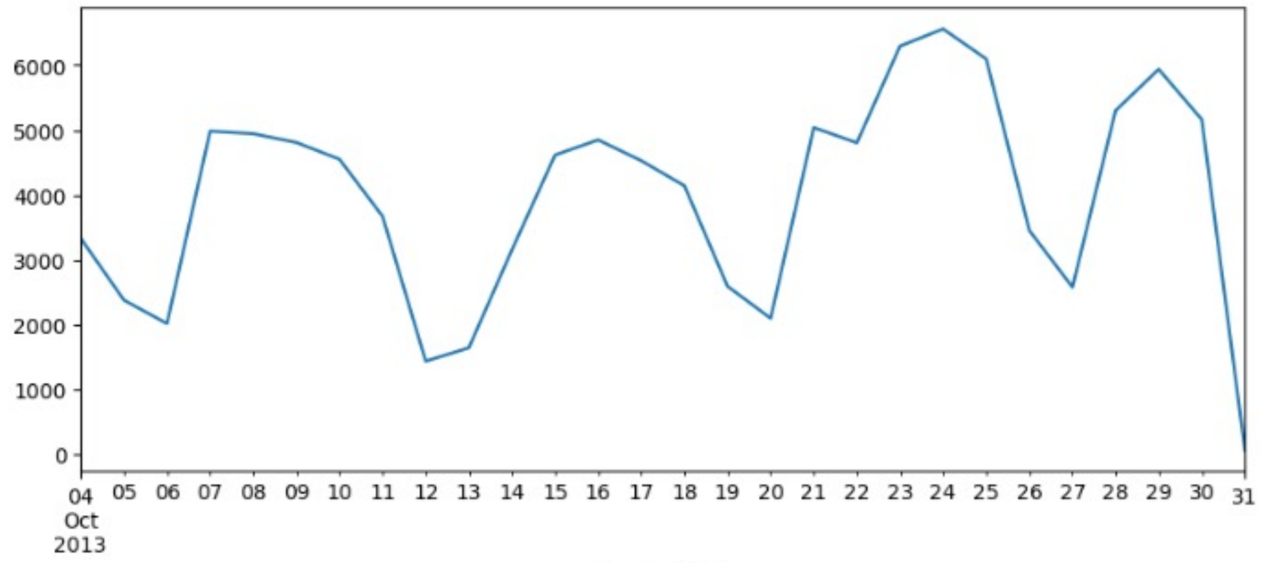


Created Date	Unique Key	Closed Date	Agency	Agency Name	Complaint Type	Descriptor	Location Type	I
2013-10-04 00:00:10	26428033	10/04/2013 01:20:52 AM	NYPD	New York City Police Department	Blocked Driveway	Partial Access	Street/Sidewalk	
2013-10-04 00:00:28	26426115	10/04/2013 04:17:32 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Talking	Club/Bar/Restaurant	
2013-10-04 00:00:45	26428987	10/04/2013 01:25:01 AM	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	
2013-10-04 00:01:05	26428083	10/04/2013 02:13:50 AM	NYPD	New York City Police Department	Illegal Parking	Posted Parking Sign Violation	Street/Sidewalk	
2013-10-04 00:01:13	26426013	10/07/2013 04:07:16 PM	DPR	Department of Parks and Recreation	Maintenance or Facility	Structure - Outdoors	Park	
...	...	...	...	...	...	...	...	
2013-10-31 01:53:44	26590930	NaN	DOHMH	Department of Health and Mental Hygiene	Rodent	Condition Attracting Rodents	Vacant Lot	
2013-10-31 01:56:23	26595721	10/31/2013 02:21:48 AM	NYPD	New York City Police Department	Noise - Vehicle	Car/Truck Horn	Street/Sidewalk	
2013-10-31 02:00:24	26594139	10/31/2013 02:40:32 AM	NYPD	New York City Police Department	Noise - Commercial	Loud Music/Party	Club/Bar/Restaurant	
2013-10-31 02:01:04	26593698	NaN	NYPD	New York City Police Department	Illegal Parking	Commercial Overnight Parking	Street/Sidewalk	
2013-10-31 02:08:41	26589651	NaN	NYPD	New York City Police Department	Noise - Street/Sidewalk	Loud Talking	Street/Sidewalk	

111069 rows × 51 columns

```
complain_df.resample("D").size().plot(figsize = (10, 4))
```

```
<Axes: xlabel='Created Date'>
```



Resample or groupby

Date	Sales	day_of_week
01/01	120	Monday
02/01	100	Tuesday
03/01	110	Wednesday
04/01	130	Thursday
05/01	120	Friday
06/01	150	Saturday
07/01	120	Sunday
08/01	130	Monday
09/01	120	Tuesday
10/01	160	Wednesday
11/01	120	Thursday
12/01	140	Friday
13/01	140	Saturday
14/01	100	Sunday

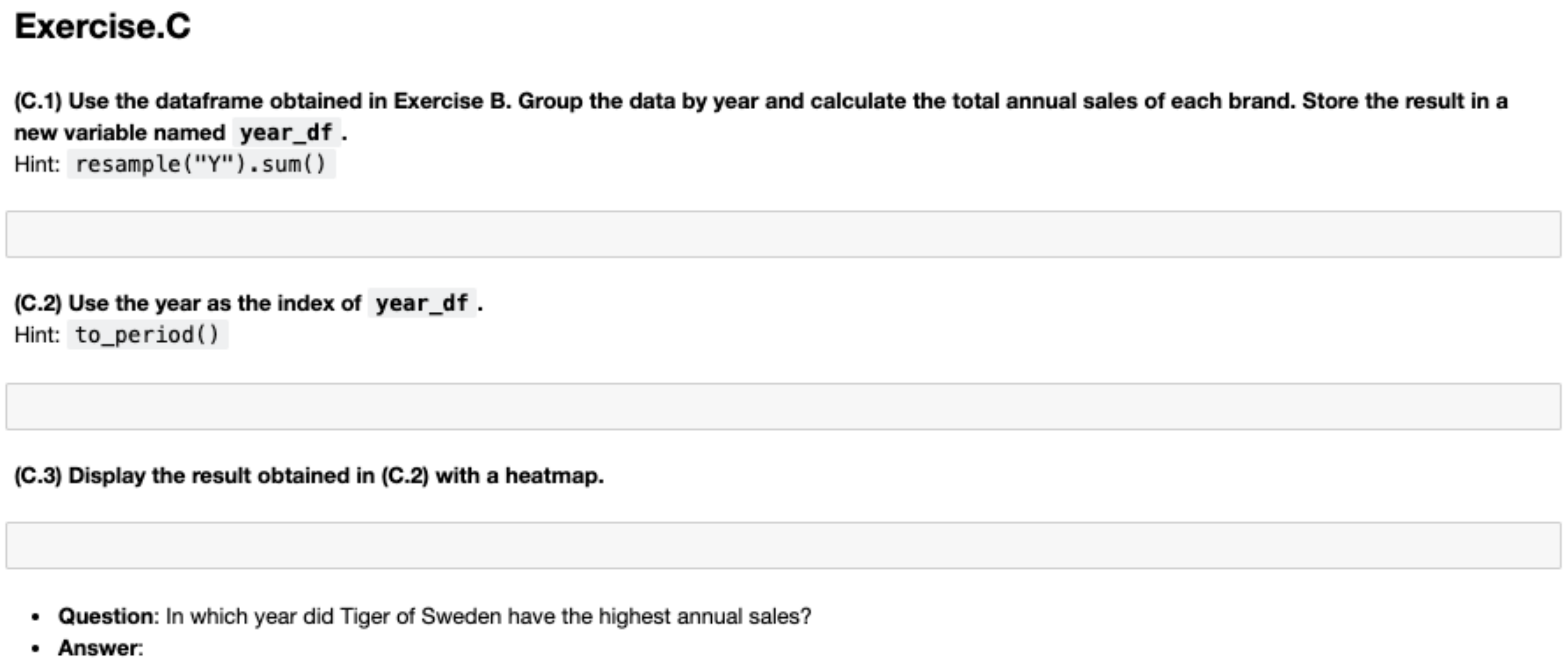
```
Dataframe.resample(“W”).Sales.sum()
```

Date	Sales
07/01	850
14/01	910

```
Dataframe.groupby(“day_of_week”).Sales.sum()
```

day_of_week	Sales
Monday	250
Tuesday	220
Wednesday	270
Thursday	250
Friday	260
Saturday	290
Sunday	220

Exercise



Exercise.C

(C.1) Use the dataframe obtained in Exercise B. Group the data by year and calculate the total annual sales of each brand. Store the result in a new variable named `year_df`.
Hint: `resample("Y").sum()`

(C.2) Use the year as the index of `year_df`.
Hint: `to_period()`

(C.3) Display the result obtained in (C.2) with a heatmap.

- **Question**: In which year did Tiger of Sweden have the highest annual sales?
- **Answer**: