BI

Time Series

Time series data

- Time series refers to a collection of data points observed and recorded over time.
- Anything measured at multiple points in time forms a time series, such as every 10 minutes, once per day, or once per month.

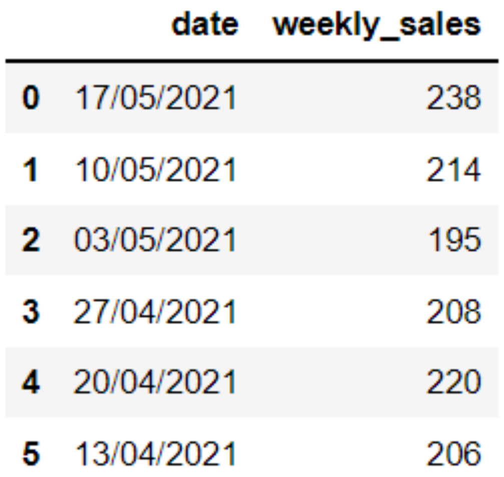
Year	Crude oil price
2021	$62.42
2020	$39.68
2019	$56.99
2018	$65.23
2017	$50.80
2016	$43.29
…	…

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
…	…

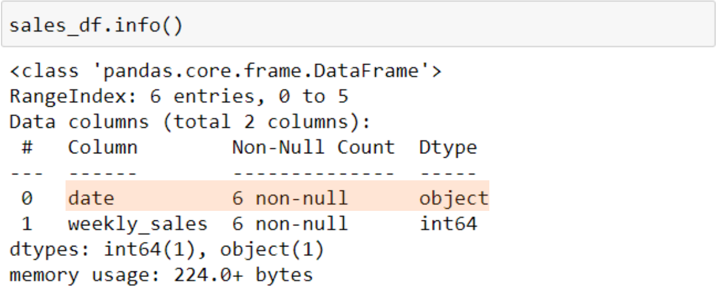
Month	# of word “trade” in headline
2015-11	22
2015-12	20
2016-01	10
2016-02	9
2016-03	6
2016-04	11
…	…

Store Date/Time as string type

- Limits:

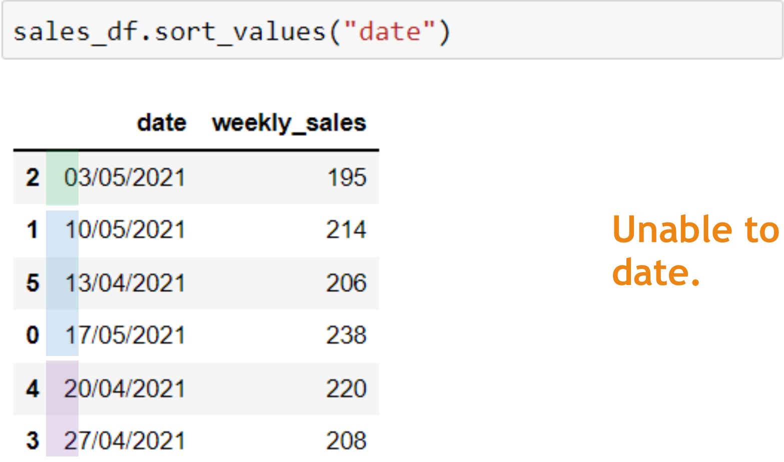


	date	weekly_sales
0	17/05/2021	238
1	10/05/2021	214
2	03/05/2021	195
3	27/04/2021	208
4	20/04/2021	220
5	13/04/2021	206



```
sales_df.info()
```

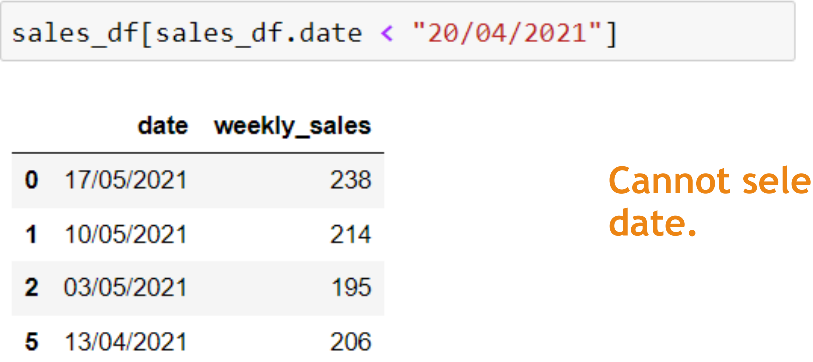
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   date          6 non-null      object
 1   weekly_sales  6 non-null      int64 
dtypes: int64(1), object(1)
memory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



```
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.

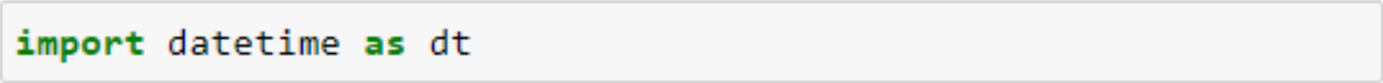
Outline

- Python Datetime module
  - Datetime objects
  - Conversion between string and datetime

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

Python datetime module

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



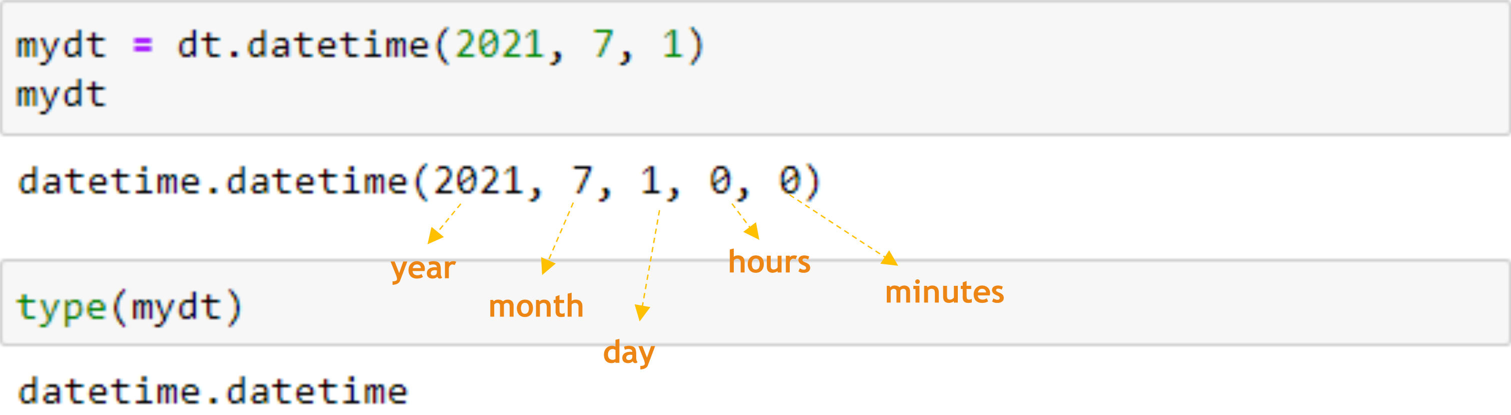
```
import datetime as dt
```

Data type	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.



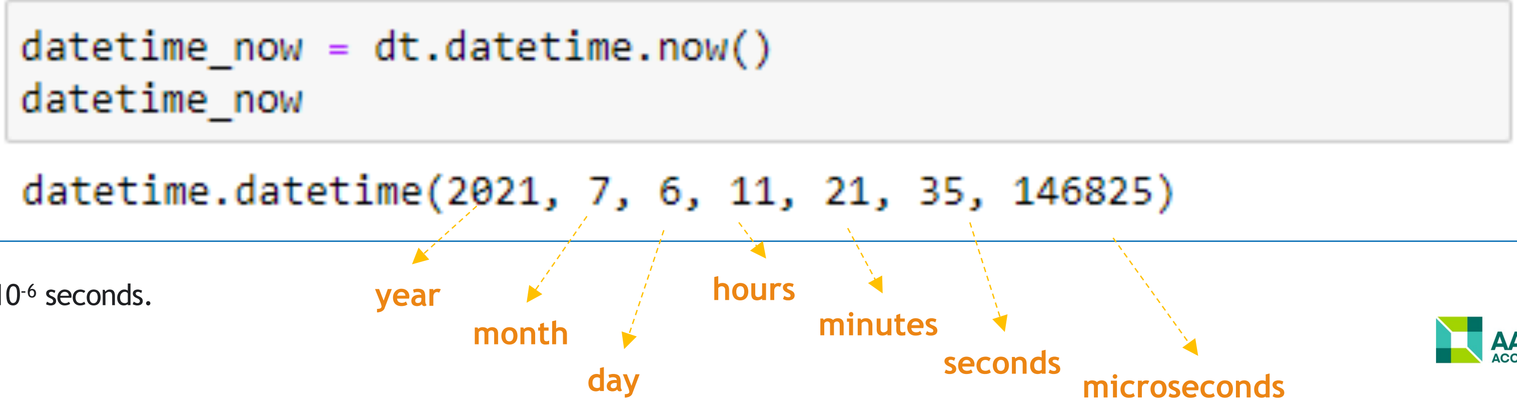
```
mydt = dt.datetime(2021, 7, 1)
mydt
```

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

```
type(mydt)
```

```
datetime.datetime
```

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



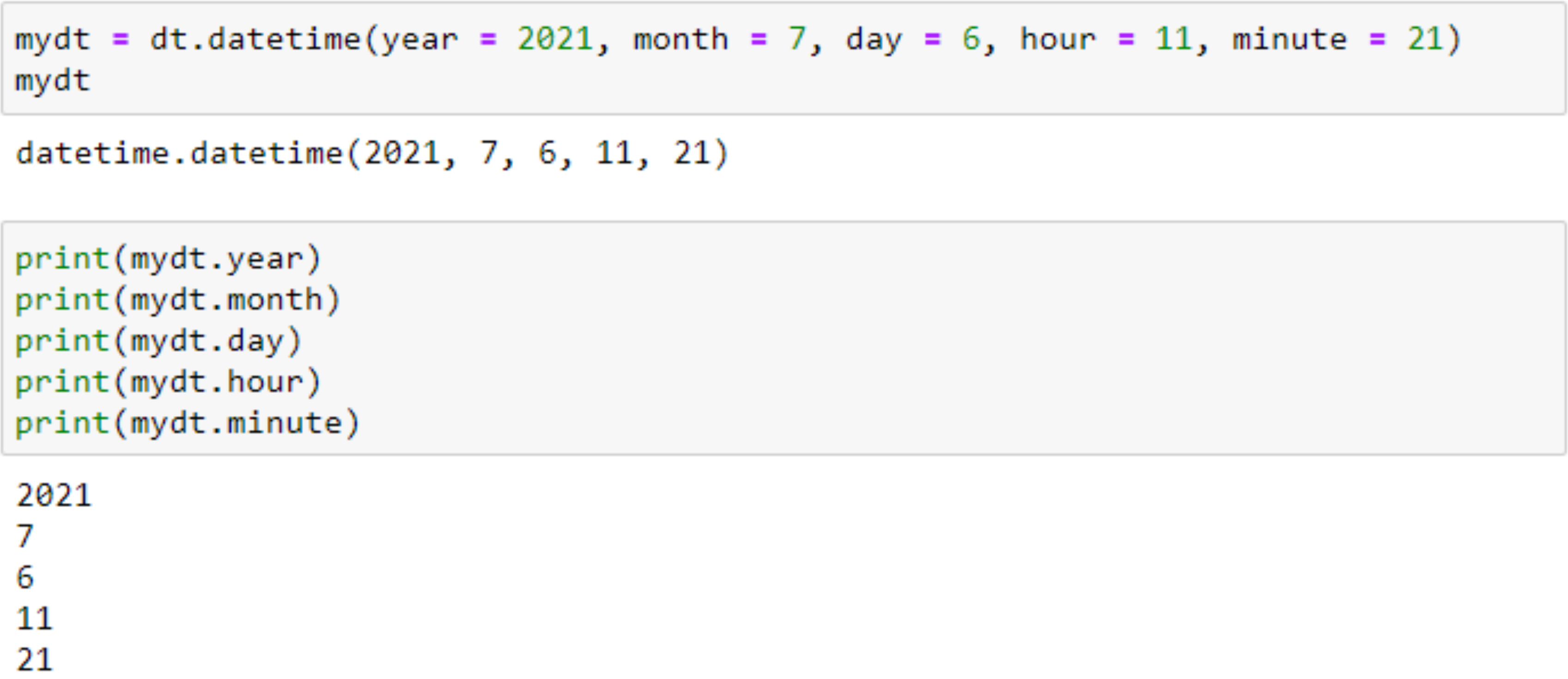
```
datetime_now = dt.datetime.now()
datetime_now
```

```
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)
```

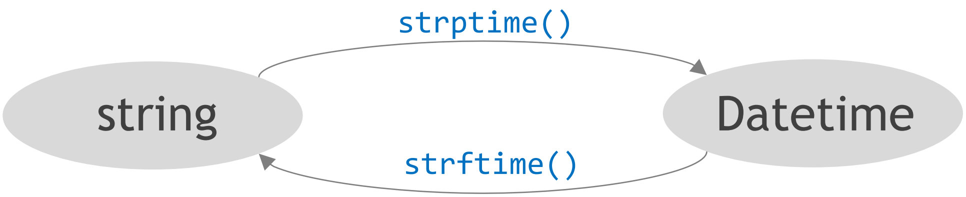
```
print(mydt.year)
print(mydt.month)
print(mydt.day)
print(mydt.hour)
print(mydt.minute)
```

```
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



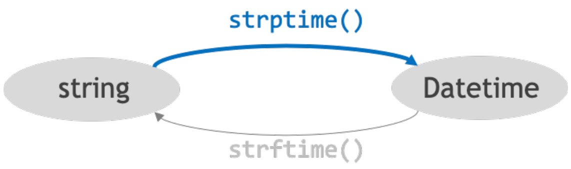
- Datetime methods: https://www.geeksforgeeks.org/python-datetime-datetime-class/
- `strptime` stands for string-parse-time.
- `strftime` stands for string-format-time.

Common datetime format

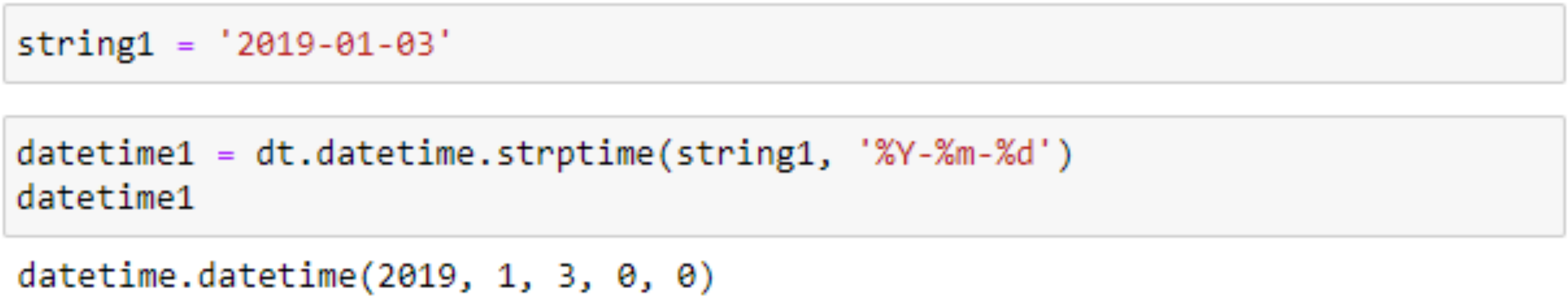
- Examples

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()`.



```
string1 = '2019-01-03'
```

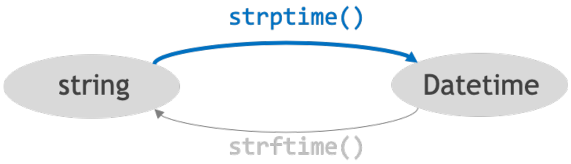
```
datetime1 = dt.datetime.strptime(string1, '%Y-%m-%d')
datetime1
```

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

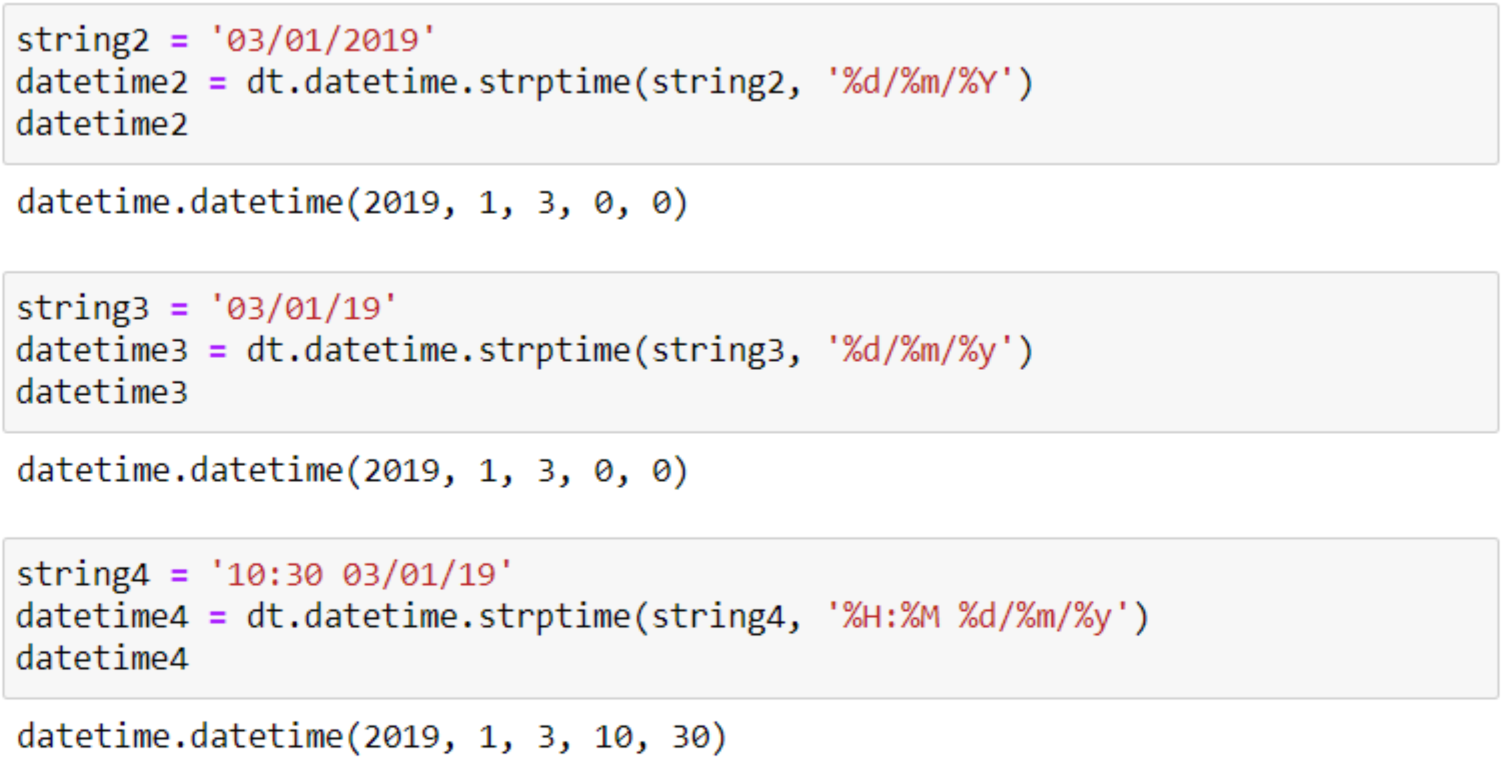
Directive	Meaning
%Y	Four-digit year
%y	Two-digit year
%m	Two-digit month [01,12]
%d	Two-digit day [01,31]
%H	Hour (24-hour clock) [00,23]
%M	Two-digit minute [00,59]
%S	Two-digit minute [00,59]

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

Datetime object – strptime()



- Other formats



```
string2 = '03/01/2019'
datetime2 = dt.datetime.strptime(string2, '%d/%m/%Y')
datetime2
```

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

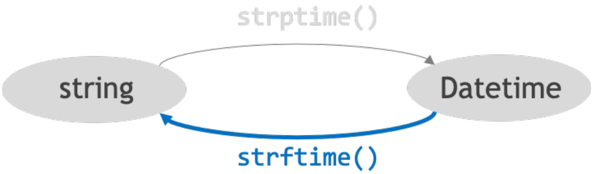
```
string3 = '03/01/19'
datetime3 = dt.datetime.strptime(string3, '%d/%m/%y')
datetime3
```

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

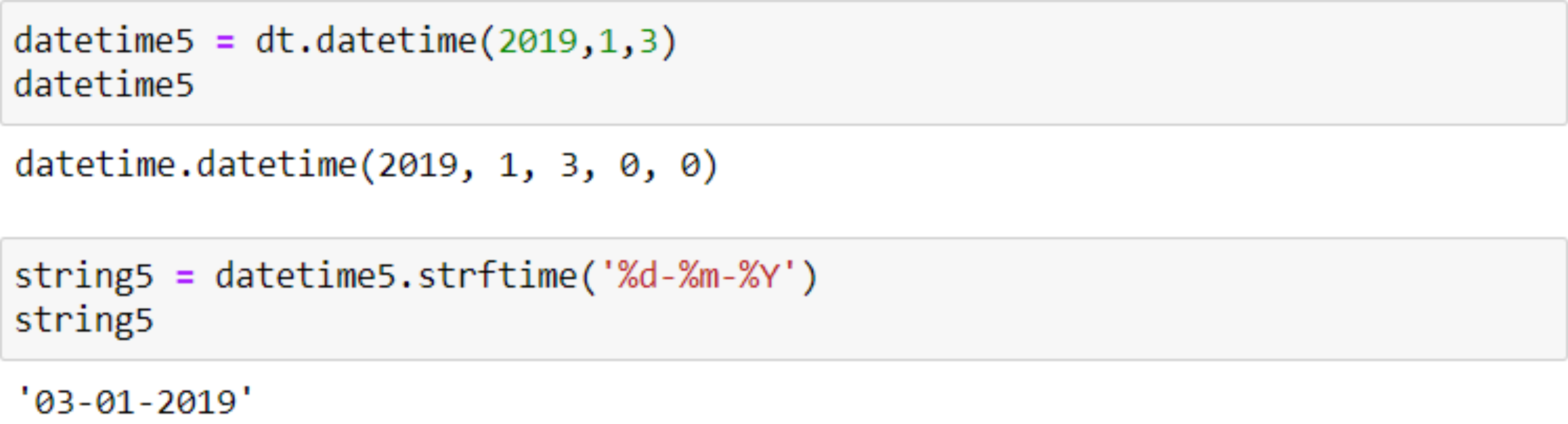
```
string4 = '10:30 03/01/19'
datetime4 = dt.datetime.strptime(string4, '%H:%M %d/%m/%y')
datetime4
```

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

Datetime object – strftime()



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



```
datetime5 = dt.datetime(2019,1,3)
datetime5
```

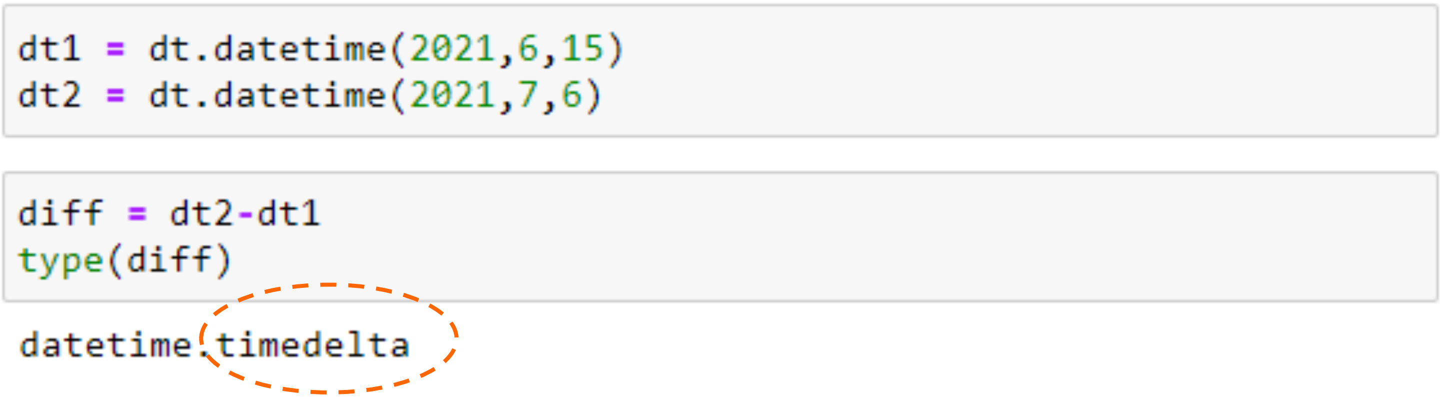
```
datetime.datetime(2019, 1, 3, 0, 0)
```

```
string5 = datetime5.strftime('%d-%m-%Y')
string5
```

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

Timedelta object

- Calculate the difference between two datetime object.

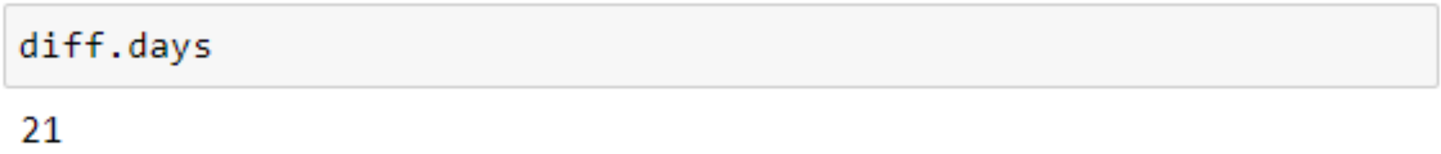


```
dt1 = dt.datetime(2021,6,15)
dt2 = dt.datetime(2021,7,6)
```

```
diff = dt2-dt1
type(diff)
```

```
datetime.timedelta
```

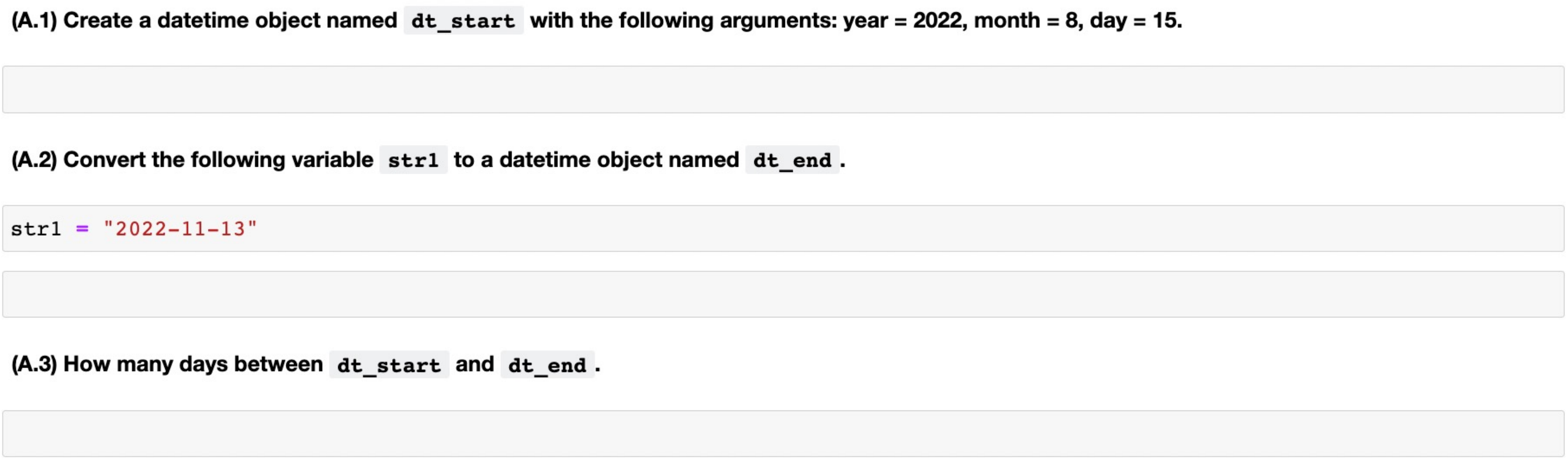
- Attribute: days



```
diff.days
```

```
21
```

Exercise



(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 objects
  - Conversion between string and datetime

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

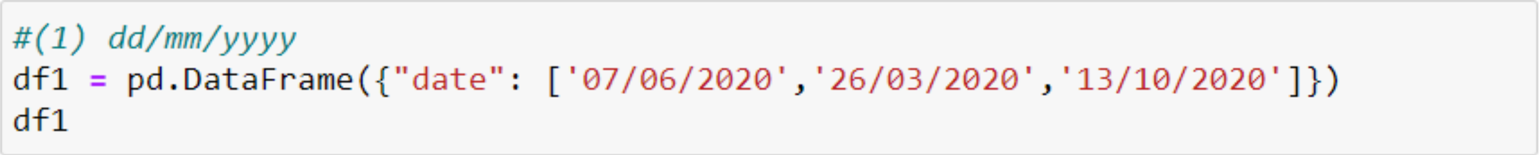
Pandas data types

- Pandas `dtype` mapping

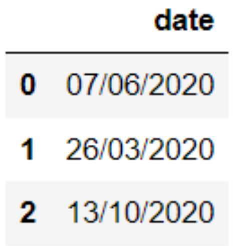
Pandas dtype	Python type	Usage
int64	int	Integer numbers
float64	float	Floating point numbers
object	str or mixed	Text or mixed numeric and non-numeric values
bool	bool	True/False values
datetime	datetime	Date and time values
timedelta	timedelta	Differences between two datetimes

Pandas – to_datetime

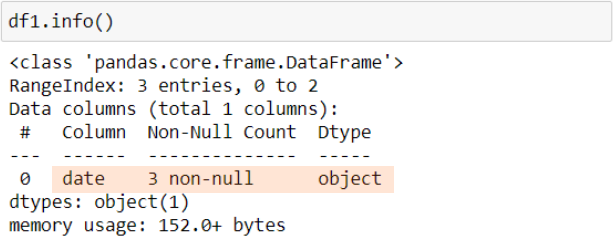
- Pandas `to_datetime()` function parses many different types of date and time formats.
- Example-1:



```
#(1) dd/mm/yyyy
df1 = pd.DataFrame({"date": ['07/06/2020','26/03/2020','13/10/2020']})
df1
```

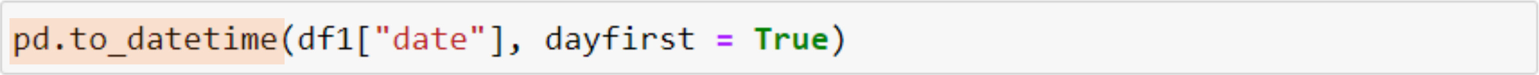


	date
0	07/06/2020
1	26/03/2020
2	13/10/2020

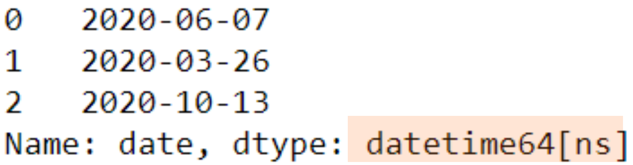


```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   date    3 non-null      object
dtypes: object(1)
memory usage: 152.0+ bytes
```



```
pd.to_datetime(df1["date"], dayfirst = True)
```



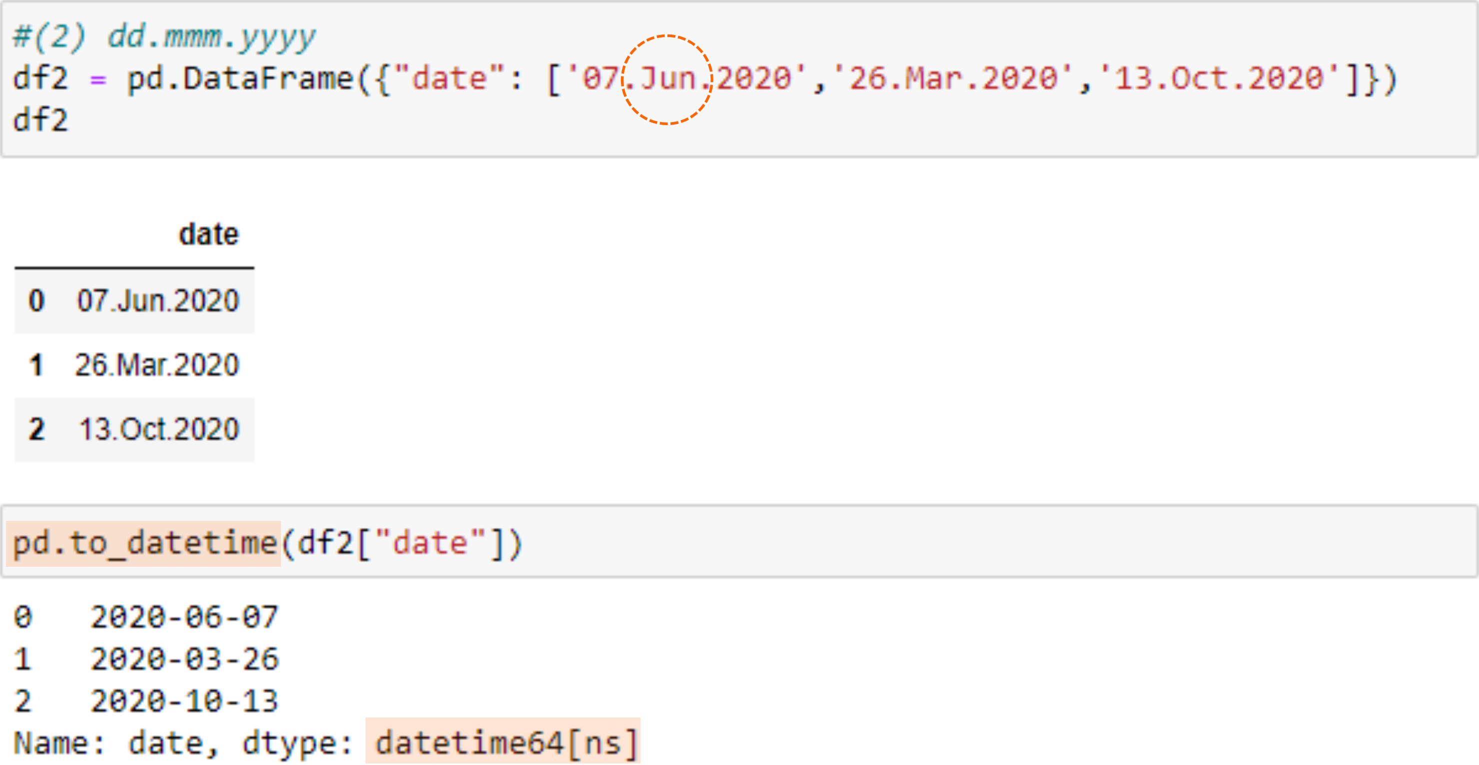
```
0    2020-06-07
1    2020-03-26
2    2020-10-13
Name: date, dtype: datetime64[ns]
```

If `dayfirst = True`, parses dates with the day first.
e.g., 07/06/2020 is parsed as 2020-06-07.

- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html
- https://numpy.org/doc/stable/reference/arrays.datetime.html
- datetime64[ns] indicates that the underlying data is stored as 64-bit integers, in units of nanoseconds (ns).

Pandas – to_datetime

- Example-2:



```
#(2) dd.mmm.yyyy
df2 = pd.DataFrame({"date": ['07.Jun.2020','26.Mar.2020','13.Oct.2020']})
df2
```

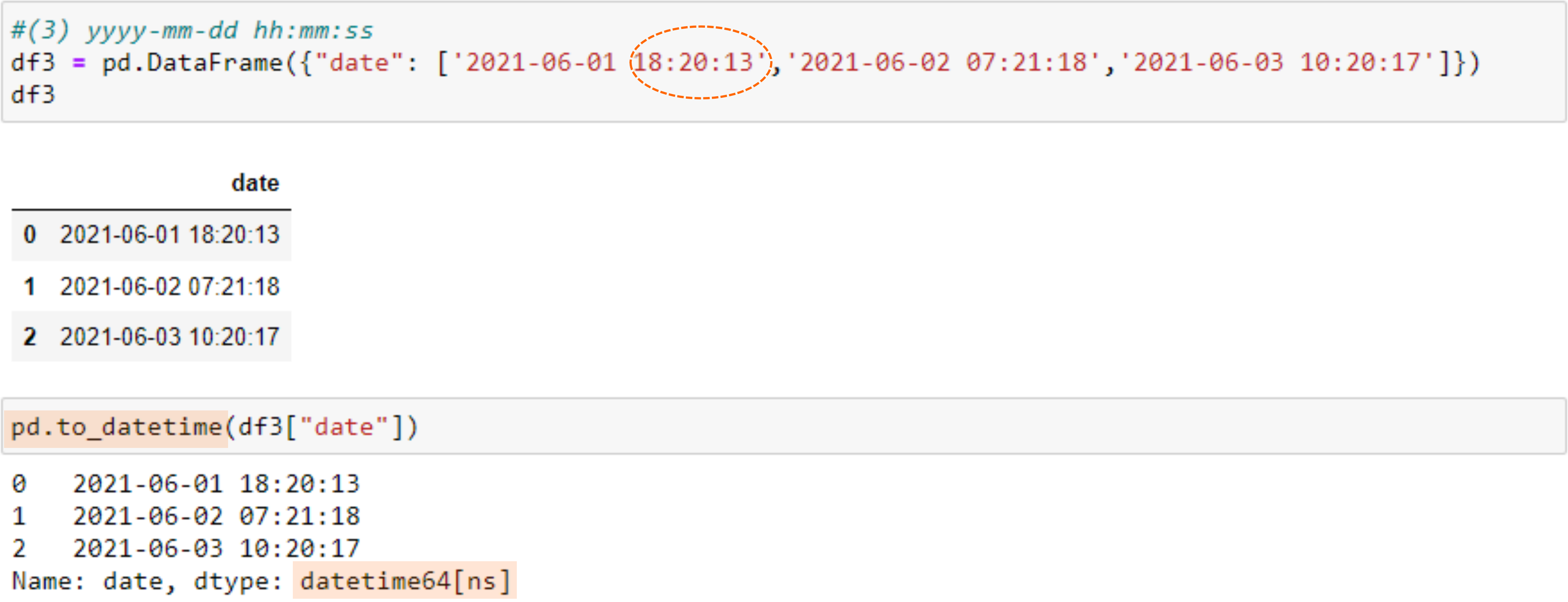
	date
0	07.Jun.2020
1	26.Mar.2020
2	13.Oct.2020

```
pd.to_datetime(df2["date"])
```

```
0    2020-06-07
1    2020-03-26
2    2020-10-13
Name: date, dtype: datetime64[ns]
```

Pandas – to_datetime

- Example-3:



```
#(3) yyyy-mm-dd hh:mm:ss
df3 = pd.DataFrame({"date": ['2021-06-01 18:20:13','2021-06-02 07:21:18','2021-06-03 10:20:17']})
df3
```

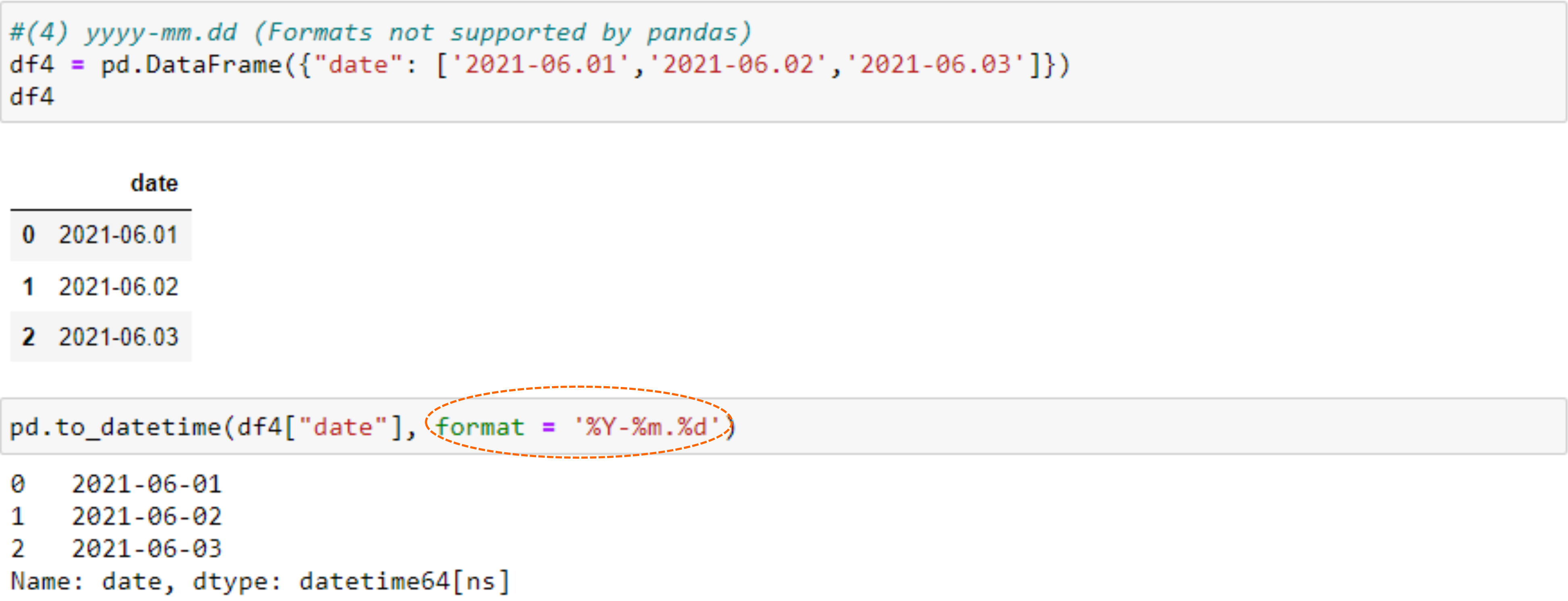
	date
0	2021-06-01 18:20:13
1	2021-06-02 07:21:18
2	2021-06-03 10:20:17

```
pd.to_datetime(df3["date"])
```

```
0    2021-06-01 18:20:13
1    2021-06-02 07:21:18
2    2021-06-03 10:20:17
Name: date, dtype: datetime64[ns]
```

Pandas – to_datetime

- Formats not supported by pandas



```
#(4) yyyy-mm.dd (Formats not supported by pandas)
df4 = pd.DataFrame({"date": ['2021-06.01','2021-06.02','2021-06.03']})
df4
```

	date
0	2021-06.01
1	2021-06.02
2	2021-06.03

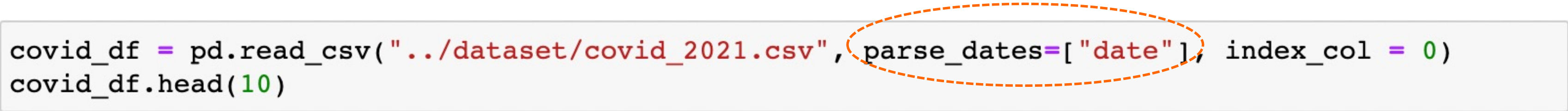
```
pd.to_datetime(df4["date"], format = '%Y-%m.%d')
```

```
0   2021-06-01
1   2021-06-02
2   2021-06-03
Name: date, dtype: datetime64[ns]
```

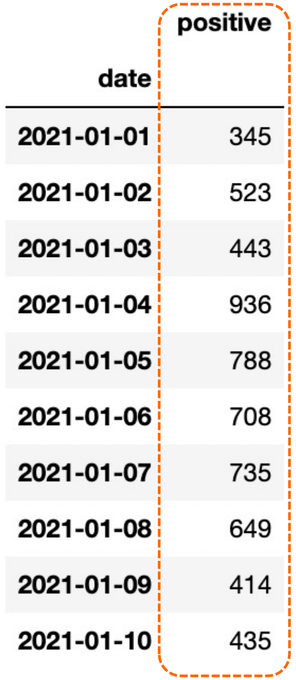
Format codes: https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes

DatetimeIndex

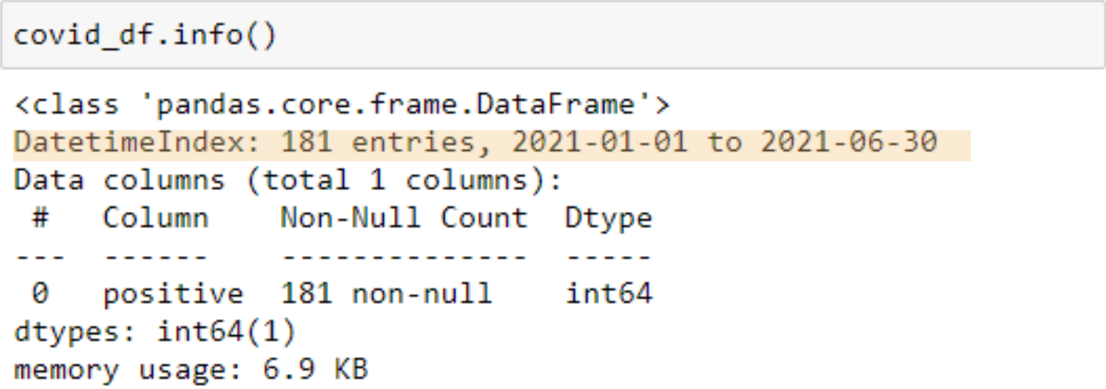
- Use argument `parse_dates()` to parse the column as `DataTime`.



```
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

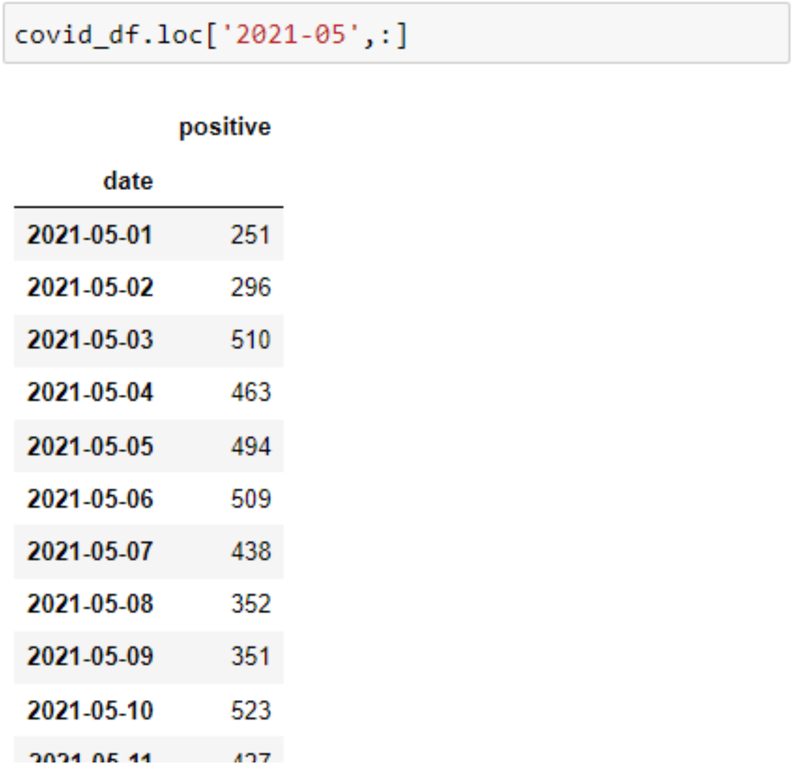


```
covid_df.info()
```

```
<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

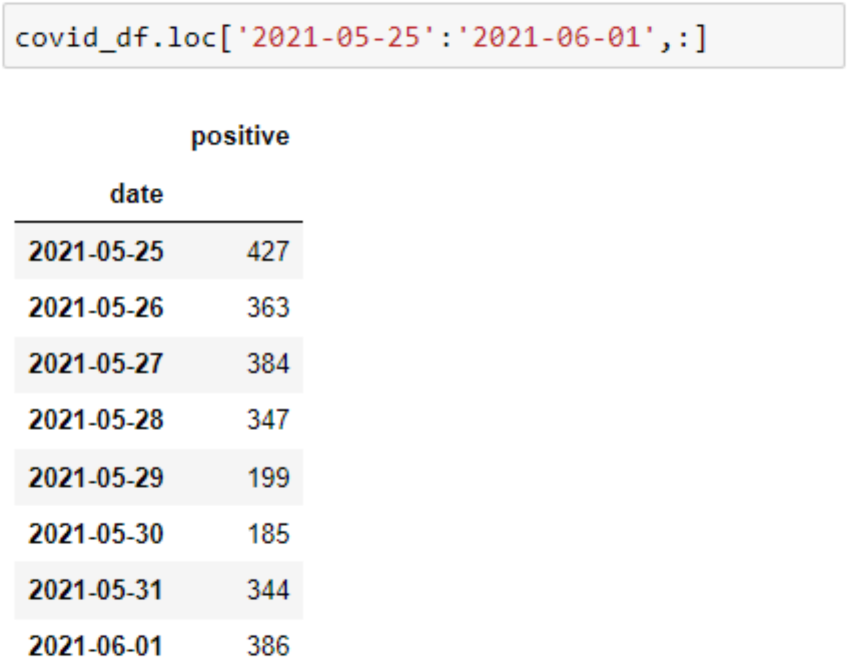
- A `DatetimeIndex` contains date-related properties and supports convenient slicing.
  - Select a subset by month



```
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

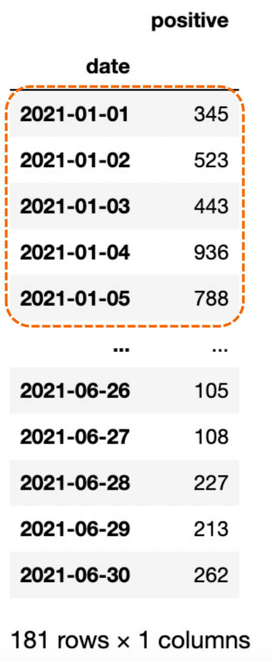


```
covid_df.loc['2021-05-25':'2021-06-01',:]
```

	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

Slice data using iloc and loc

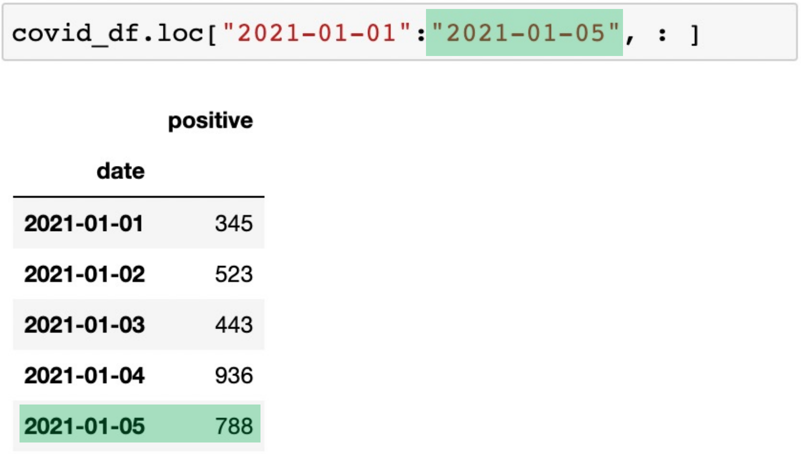
- Use `loc`: Data for "end_index" will be included.
- Use `iloc`: Data for "end_index" will not be included.



date	positive
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

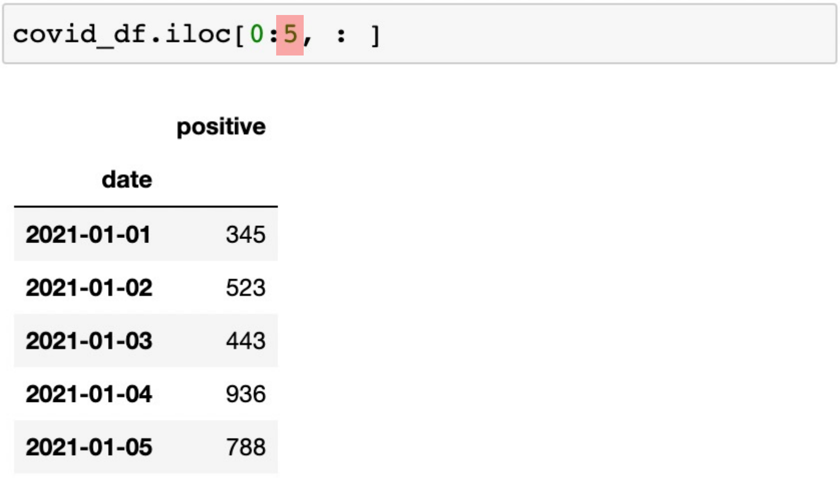
```
df.loc[start_indx : end_index]
```



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

date	positive
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]
```

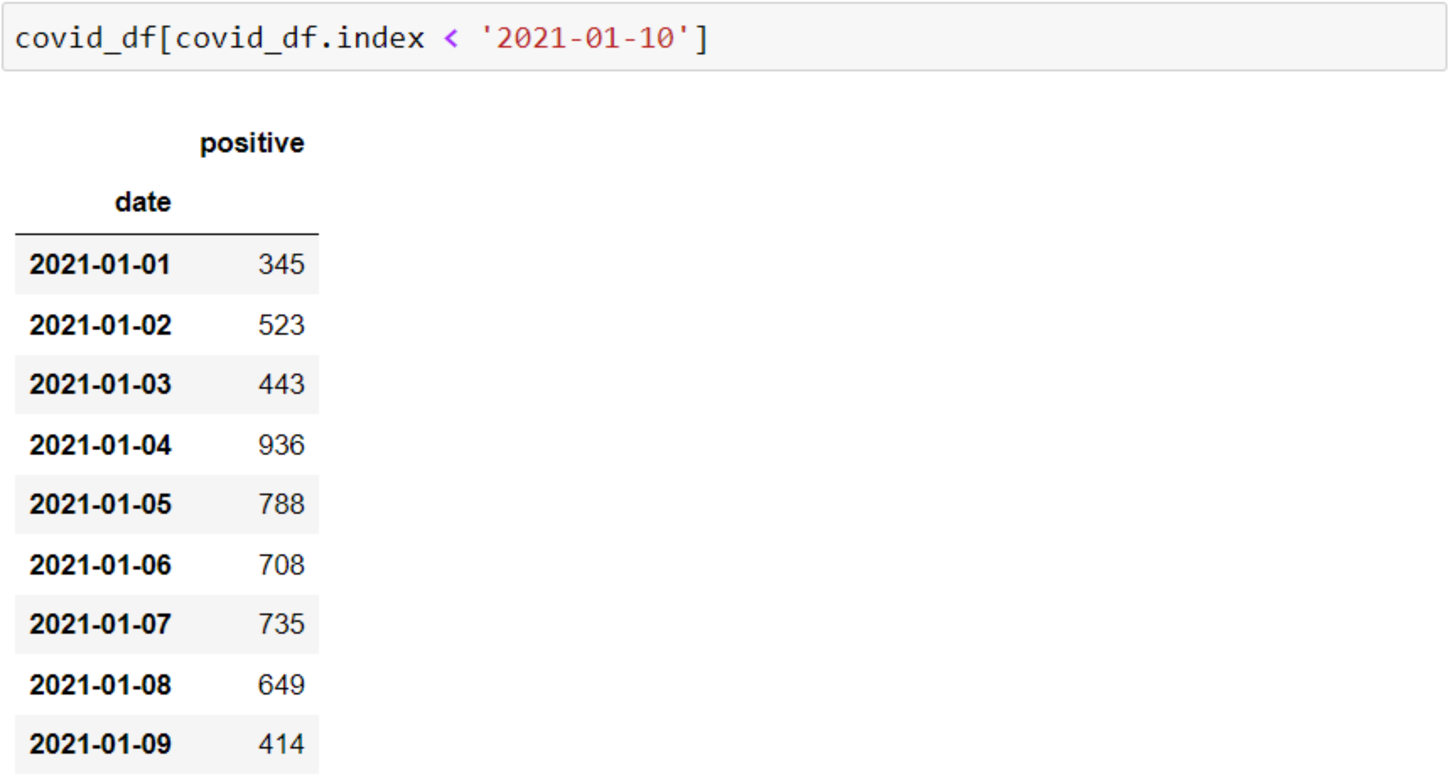


```
covid_df.iloc[0:5, : ]
```

	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
5		

DatetimeIndex - subset selection

- Select a subset by condition

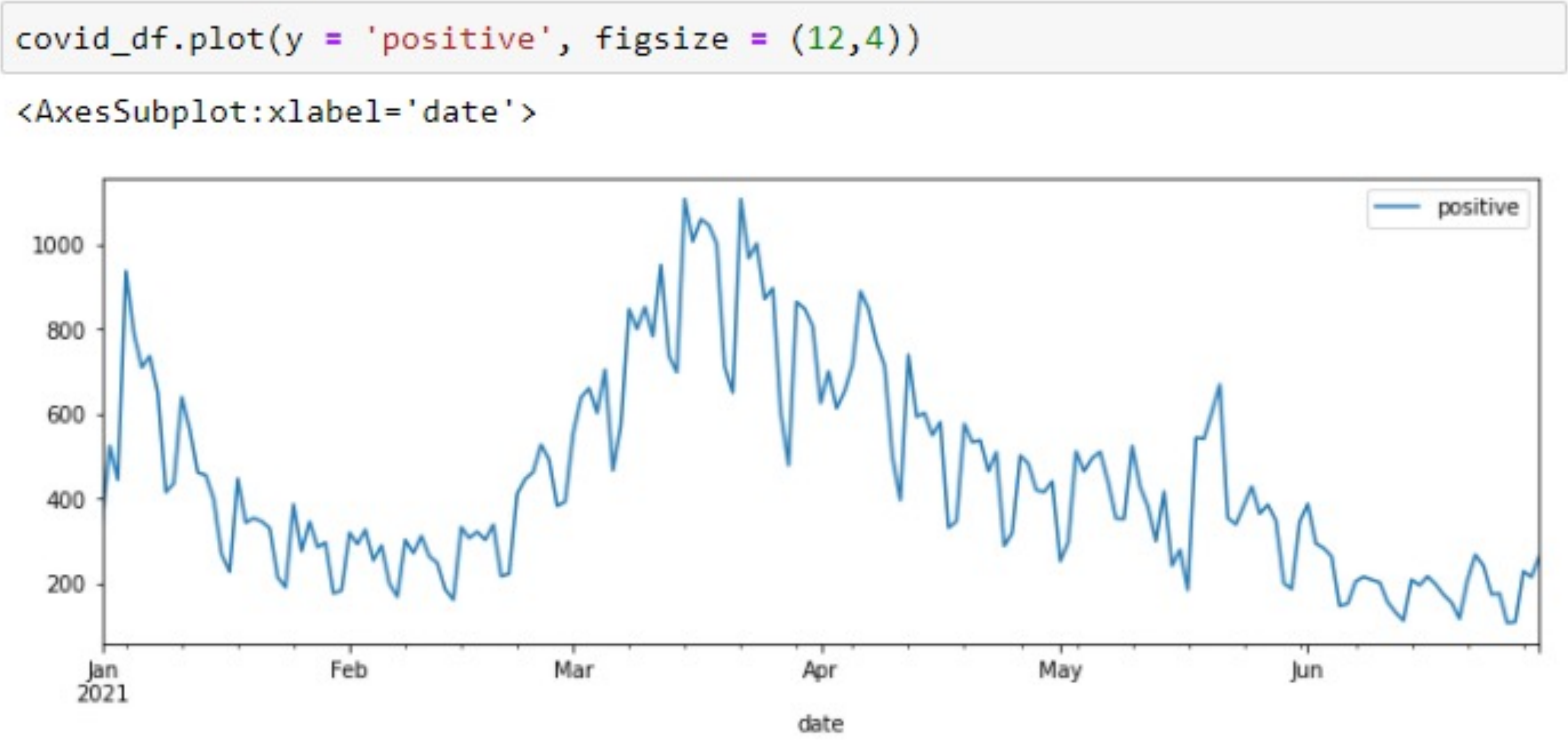


```python
covid_df[covid_df.index < '2021-01-10']
```

	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

DatetimeIndex – line chart

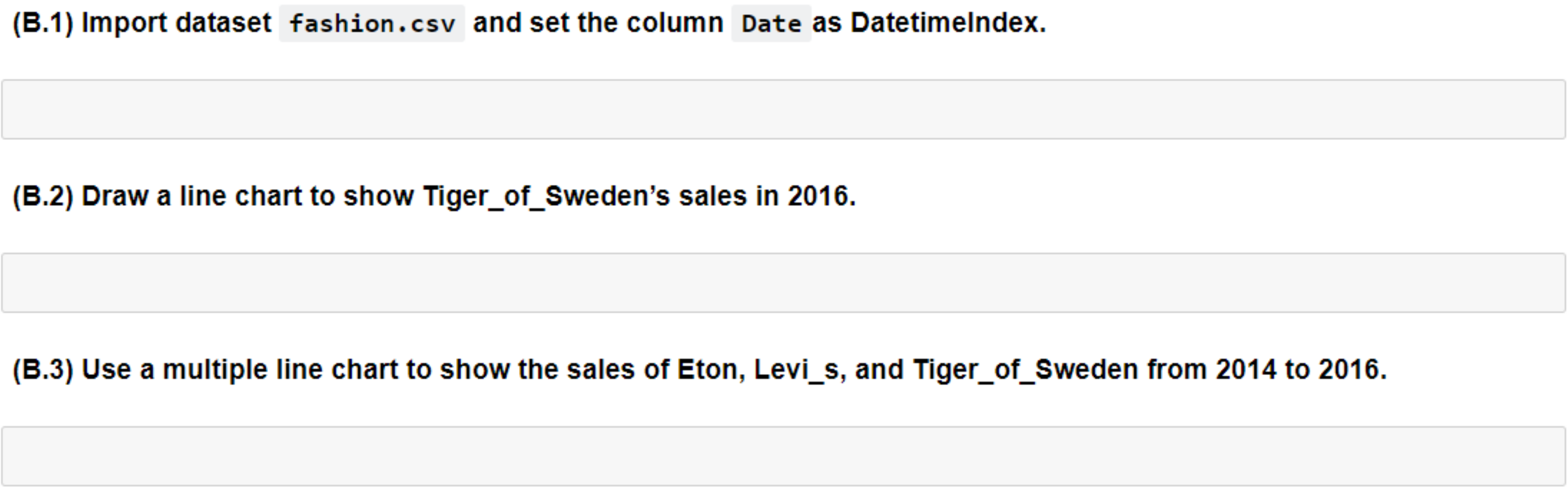
- Use `plot()` to plot a line chart.



```
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.

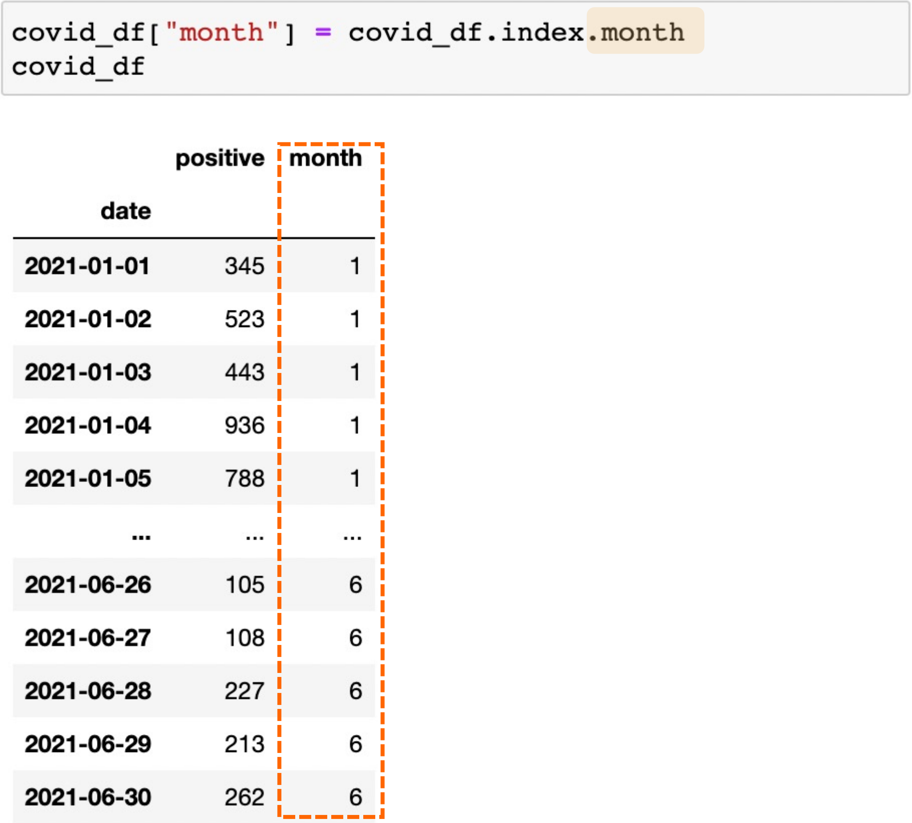
- Methods

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

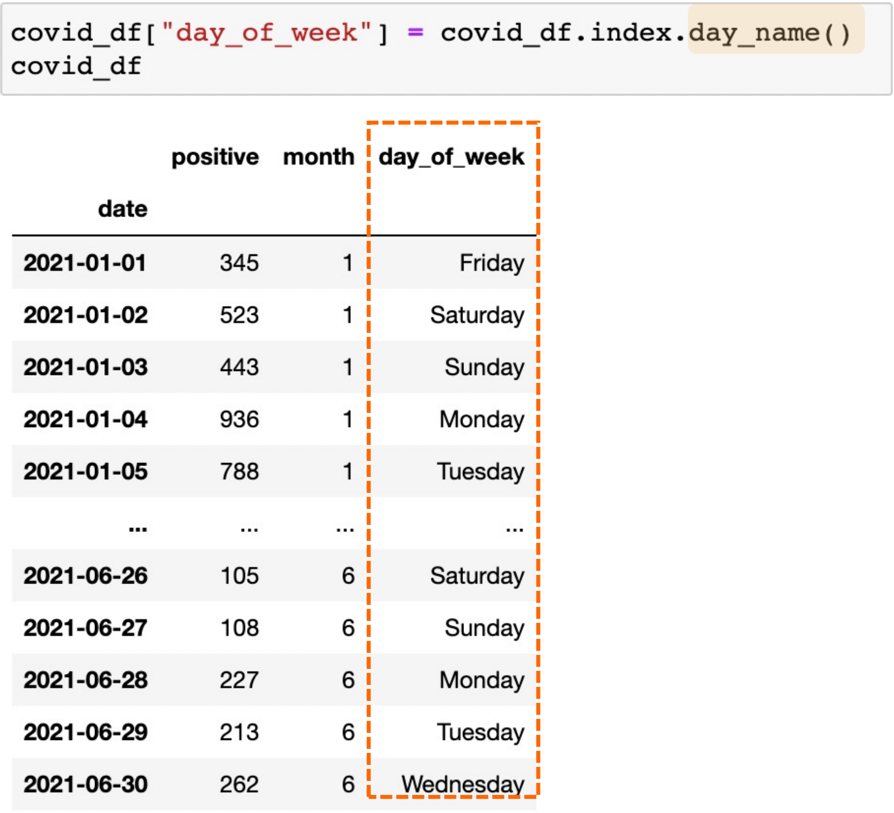
Information extraction

- 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

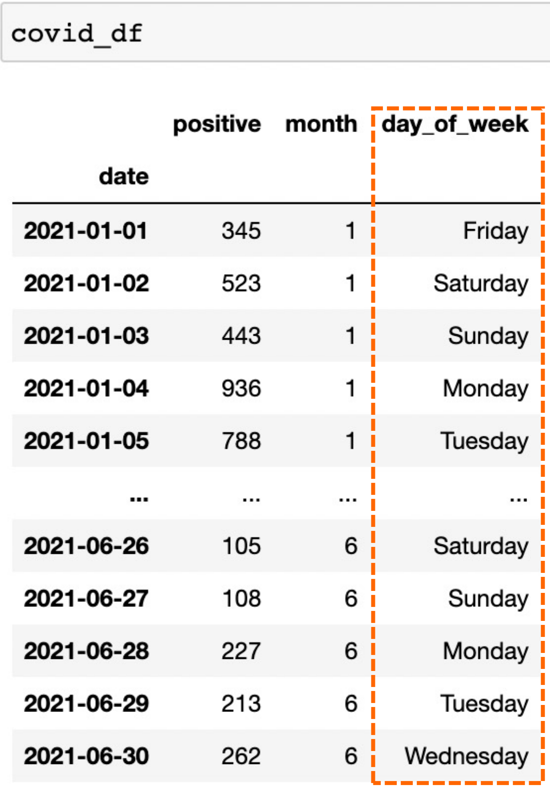


```
covid_df["day_of_week"] = covid_df.index.day_name()
covid_df
```

	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

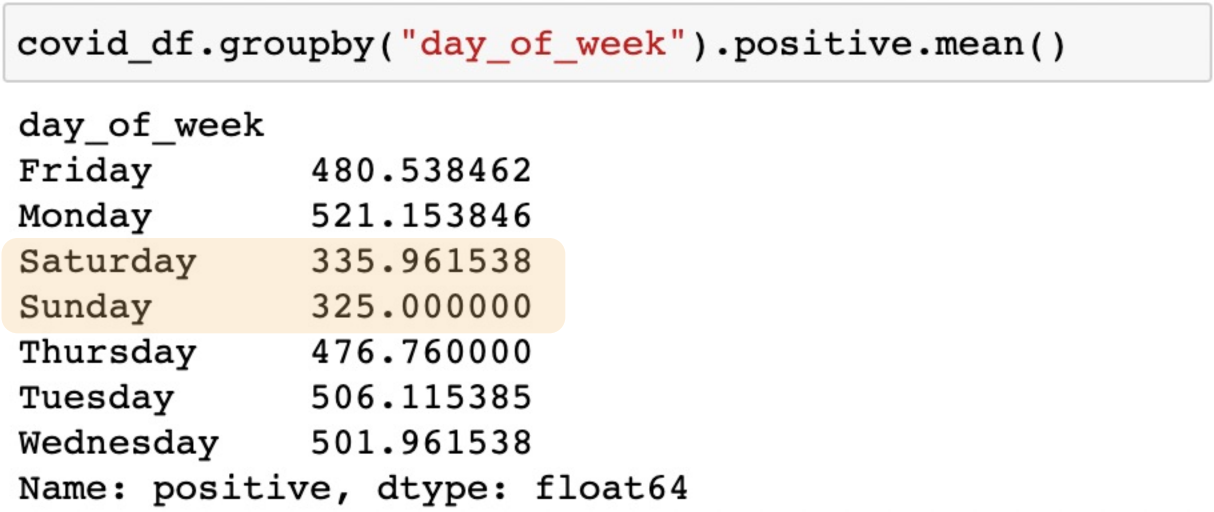
Information extraction

- 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



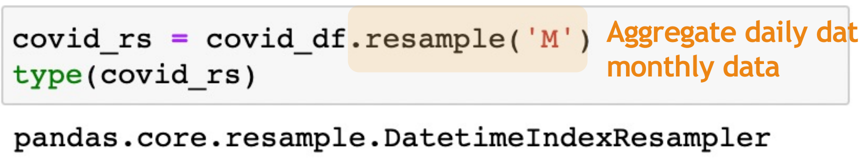
```
covid_df.groupby("day_of_week").positive.mean()
```

```
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
```

Resampling

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

Step1: Get a Resampler object

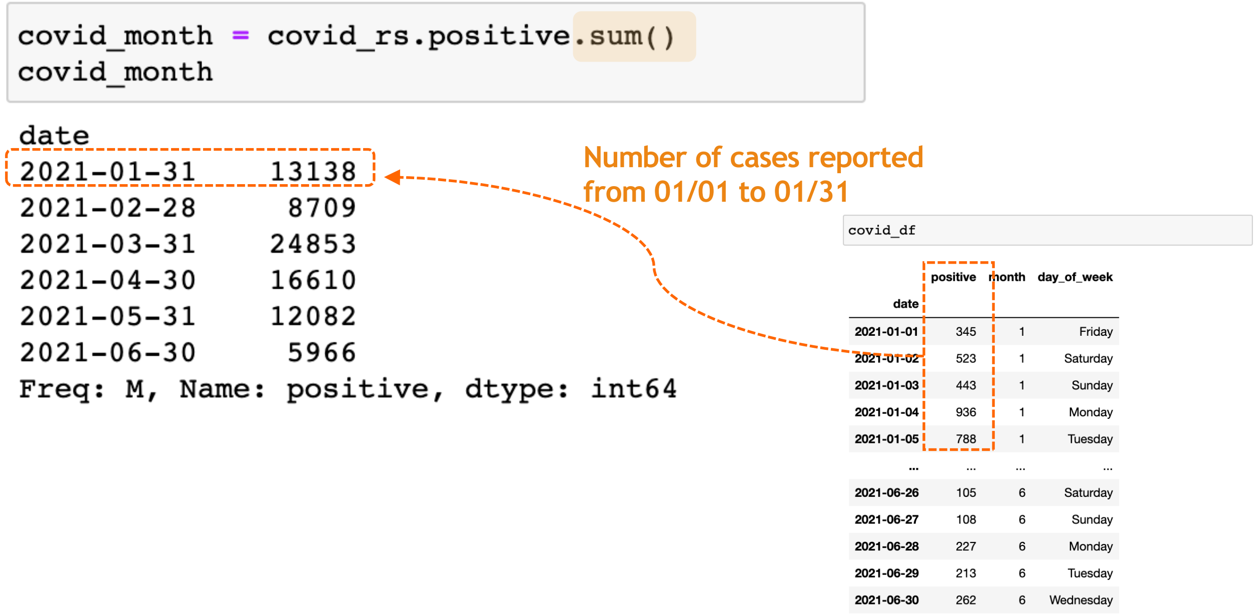


```
covid_rs = covid_df.resample('M')
type(covid_rs)
```

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

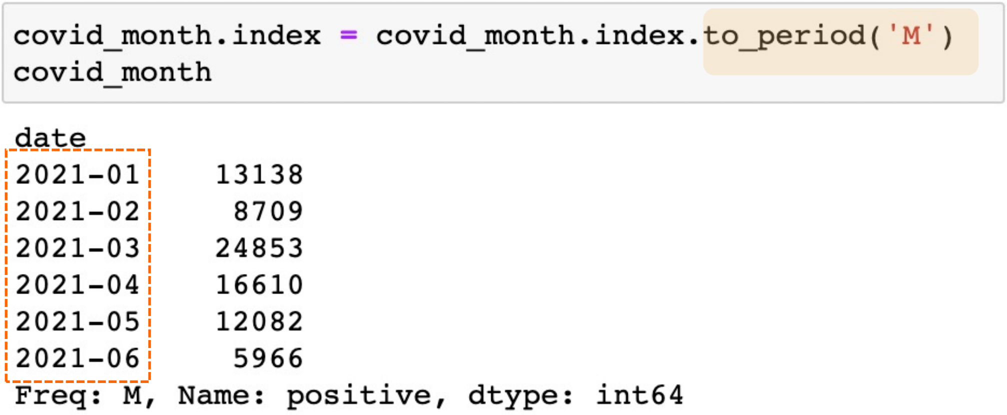
```
covid_df
```

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

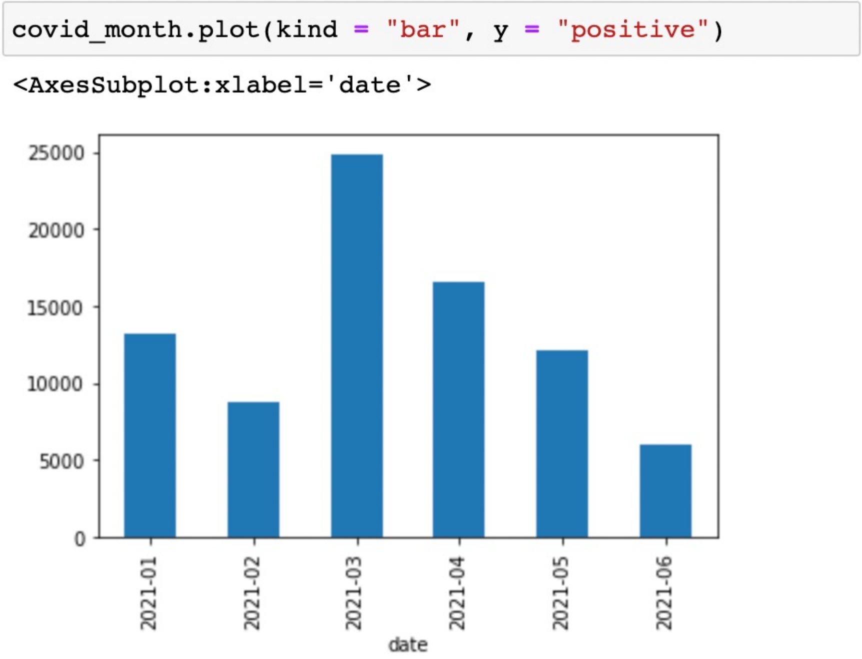
Resampling

- 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

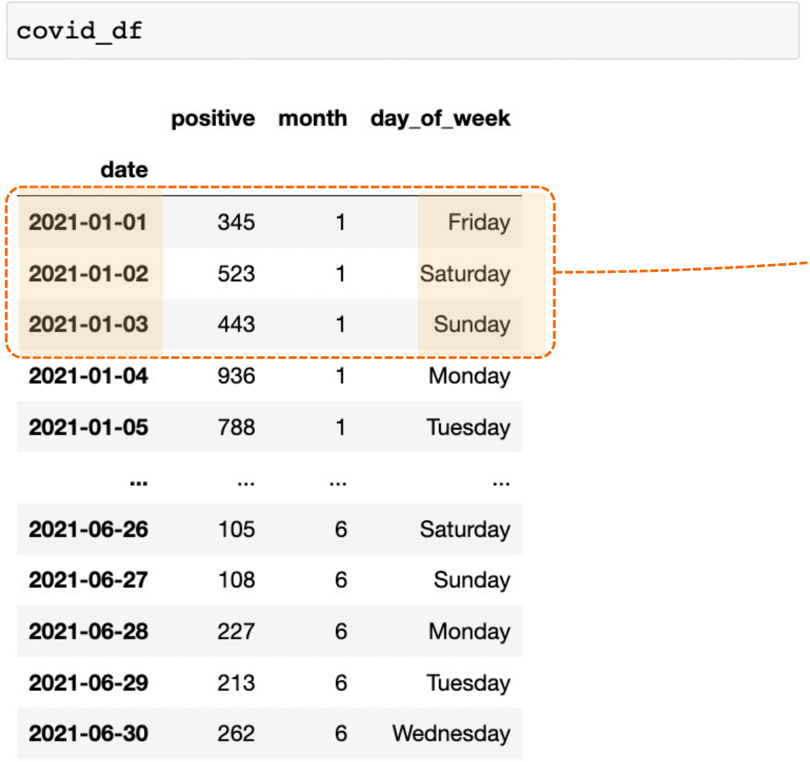
Resampling - frequencies

- Examples

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

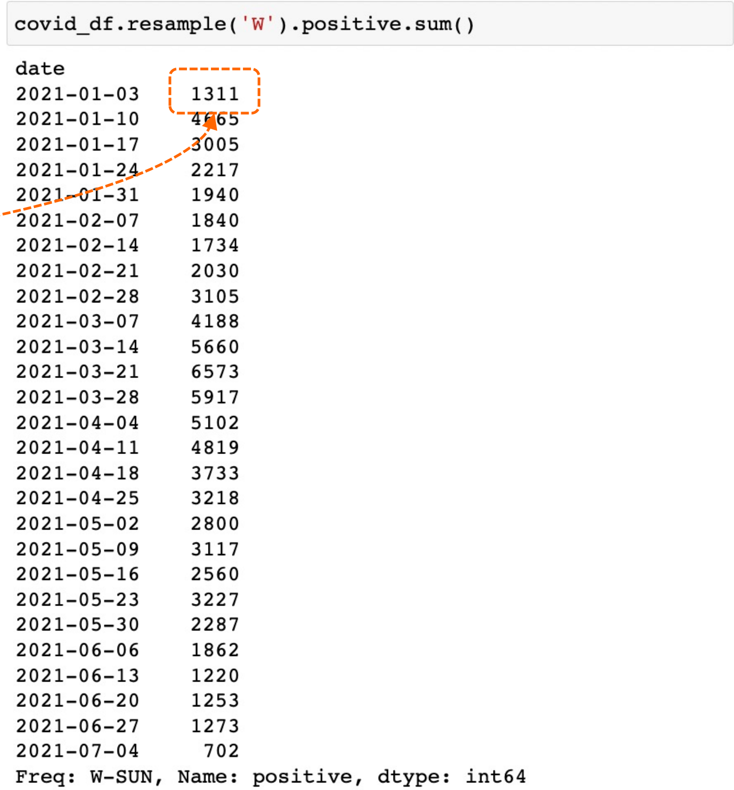
Resampling

- Aggregate daily data to weekly data



```
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



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

```
date
2021-01-03     1311
2021-01-10     4665
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
```

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”).sum()
```

Date	Sales
07/01	850
14/01	910

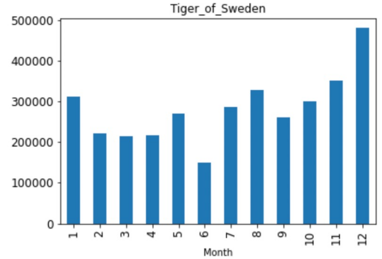
```
Dataframe.groupby(“day_of_week”).sum()
```

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

Exercise

(C.1) Use the dataframe `fashion_df` in Exercise.B. Extract the month information from the DatetimeIndex and add it to a new column named `Month`.

(C.2) Calculate the average monthly sales of `Tiger_of_Sweden` using the column obtained in (C.1). Display the results in a bar chart.
Hint: `groupby()`



(C.3) 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()`

(C.4) Use the year as the index of `year_df`.
Hint: `to_period()`

(C.5) Display the result obtained in (C.4) with a heatmap, excluding the `Month` column.