Часові ряди

Часові ряди – важливий різновид структурованих даних. Вони зустрічаються в багатьох областях. Будь-які результати спостережень чи вимірювань у різні моменти часу утворюють часовий ряд. Для багатьох часових рядів характерна фіксована частота, тобто інтервали між сусідніми точками однакові.

Часові ряди можуть бути нерегулярними, коли інтервали часу між сусідніми точками розрізняються.

Часові ряди

Дані про дату та час можуть бути представлені по-різному:

- часові мітки, конкретні моменти; Timestamp в Pandas;
- фіксовані періоди; Period в Pandas;
- часові інтервали, що позначаються мітками початку та кінця; періоди вважатимуться окремими випадками інтервалів;
- час експерименту або час, що минув; кожна часові мітка вимірює час, що минув з початкового моменту. Timedelta в Pandas.

Часові ряди

У стандартній бібліотеці Python є типи даних для представлення дати та часу, а також засоби для роботи з календарем. Зокрема це модулі datetime, time та calendar.

```
from datetime import datetime

datetime.datetime(year, month, day, hour=0, minute=0, second=0, microsecond=0, tzinfo=None, *, fold=0)

dt=datetime(year=2021, month=8, day=31)

datetime.datetime(2021, 8, 31, 0, 0)
dt.day
31
```

Часові ряди

```
dt.weekday()
1
dt.strftime('%A’)
'Tuesday'
dt.strftime('%B’)
'August’
```

```
from dateutil import parser
date = parser.parse("2021-09-04")
date
datetime.datetime(2021, 9, 4, 0, 0)
```

```
parser.parse('02/11/2021', dayfirst=True)
datetime.datetime(2021, 11, 2, 0, 0)
```

Часові ряди

Бібліотека Pandas надає об'єкт Timestamp. Вона може створювати з кількох таких об'єктів Timestamp об'єкт класу DatetimeIndex, який можна використовувати для індексації даних в об'єктах Series або DataFrame.

```
date = pd.to_datetime('2021-09-04’)
```
Timestamp('2021-09-04 00:00:00’)

```
date = pd.to_datetime('1st of January 2021’)
```
Timestamp('2021-01-01 00:00:00’)

```
date.strftime('%A’)
```
'Friday'

Часові ряди

```
dates = pd.to_datetime([datetime(2021, 10, 2), '1st of January 2021', '2021-09-04', '02/11/2021’])
```

```
DatetimeIndex(['2021-10-02', '2021-01-01', '2021-09-04', '2021-02-11'], dtype='datetime64[ns]', freq=None)
```

```
dates.to_period('D’)
```

```
PeriodIndex(['2021-10-02', '2021-01-01', '2021-09-04', '2021-02-11'], dtype='period[D]', freq='D’)
```

```
dates - dates[2]
```

```
TimedeltaIndex(['28 days', '-246 days', '0 days', '-205 days'], dtype='timedelta64[ns]', freq=None)
```

Часові ряди

Щоб полегшити створення регулярних послідовностей, бібліотека Pandas надає кілька функцій: pd.date_range() — для відліків дати/часу, pd.period_range() — для періодів часу та pd.timedelta_range() — для дельт.

Функція pd.date_range() створює регулярну послідовність дат, приймаючи на вході початкову дату, кінцеву дату та необов'язковий період. За замовчуванням період дорівнює одному дню:

pd.date_range('20210801', '20210808')

```
DatetimeIndex(['2021-08-01', '2021-08-02', '2021-08-03', '2021-08-04',
               '2021-08-05', '2021-08-06', '2021-08-07', '2021-08-08'],
              dtype='datetime64[ns]', freq='D')
```

Часові ряди

pd.date_range('2021-08-01', periods=5)

```
DatetimeIndex(['2021-08-01', '2021-08-02', '2021-08-03', '2021-08-04',
               '2021-08-05'],
              dtype='datetime64[ns]', freq='D')
```

pd.date_range('2021-08-01', periods=5, freq='M’)

```
DatetimeIndex(['2021-08-31', '2021-09-30', '2021-10-31', '2021-11-30',
               '2021-12-31'],
              dtype='datetime64[ns]', freq='M')
```

Можливі значення параметру freq:
D –день; B – робочі дні; W – тиждень; M – місяць; A, Y – рік; H – година; T, min – хвилина; S – секунда.
Додавання S в кінці означає початок. Наприклад, YS – початок року.
За замовчуванням використовується кінець періоду.

Часові ряди

pd.date_range('2021-08-01', periods=14, freq='B’)

```
DatetimeIndex(['2021-08-02', '2021-08-03', '2021-08-04', '2021-08-05',
               '2021-08-06', '2021-08-09', '2021-08-10', '2021-08-11',
               '2021-08-12', '2021-08-13', '2021-08-16', '2021-08-17',
               '2021-08-18', '2021-08-19'],
              dtype='datetime64[ns]', freq='B')
```

pd.date_range('2021-08-01', periods=10, freq='A’)

```
DatetimeIndex(['2021-12-31', '2022-12-31', '2023-12-31', '2024-12-31',
               '2025-12-31', '2026-12-31', '2027-12-31', '2028-12-31',
               '2029-12-31', '2030-12-31'],
              dtype='datetime64[ns]', freq='A-DEC')
```

pd.date_range('2021-08-01', periods=10, freq='2M')

```
DatetimeIndex(['2021-08-31', '2021-10-31', '2021-12-31', '2022-02-28',
               '2022-04-30', '2022-06-30', '2022-08-31', '2022-10-31',
               '2022-12-31', '2023-02-28'],
              dtype='datetime64[ns]', freq='2M')
```

Часові ряди

Також можна використовувати різні поєднання та коди.

pd.date_range('2021-08-01', periods=10, freq='A-APR’)

```
DatetimeIndex(['2022-04-30', '2023-04-30', '2024-04-30', '2025-04-30',
               '2026-04-30', '2027-04-30', '2028-04-30', '2029-04-30',
               '2030-04-30', '2031-04-30'],
              dtype='datetime64[ns]', freq='A-APR')
```

pd.date_range('2021-08-01', periods=10, freq='W-MON’)

```
DatetimeIndex(['2021-08-02', '2021-08-09', '2021-08-16', '2021-08-23',
               '2021-08-30', '2021-09-06', '2021-09-13', '2021-09-20',
               '2021-09-27', '2021-10-04'],
              dtype='datetime64[ns]', freq='W-MON')
```

pd.date_range('2021-08-01', periods=10, freq='2D3H')

```
DatetimeIndex(['2021-08-01 00:00:00', '2021-08-03 03:00:00',
               '2021-08-05 06:00:00', '2021-08-07 09:00:00',
               '2021-08-09 12:00:00', '2021-08-11 15:00:00',
               '2021-08-13 18:00:00', '2021-08-15 21:00:00',
               '2021-08-18 00:00:00', '2021-08-20 03:00:00'],
              dtype='datetime64[ns]', freq='51H')
```

Часові ряди

pd.date_range('2021-08-01', periods=10, freq='WOM-2SUN')

```
DatetimeIndex(['2021-08-08', '2021-09-12', '2021-10-10', '2021-11-14',
               '2021-12-12', '2022-01-09', '2022-02-13', '2022-03-13',
               '2022-04-10', '2022-05-08'],
              dtype='datetime64[ns]', freq='WOM-2SUN')
```

Часові ряди

Для створення регулярних послідовностей значень періодів або часових інтервалів можна скористатися функціями pd.period_range() та pd.timedelta_range()

pd.period_range('2021-08-01', periods=10, freq='A’)

```
PeriodIndex(['2021', '2022', '2023', '2024', '2025', '2026', '2027', '2028',
             '2029', '2030'],
            dtype='period[A-DEC]', freq='A-DEC')
```

pd.timedelta_range(start='1 day', end='10 days’)

```
TimedeltaIndex([ '2 days',  '3 days',  '4 days',  '5 days',  '6 days',
                 '7 days',  '8 days',  '9 days', '10 days'],
               dtype='timedelta64[ns]', freq='D')
```

pd.timedelta_range(start='1 day', end='10 days',closed='left')

```
TimedeltaIndex(['1 days', '2 days', '3 days', '4 days', '5 days', '6 days',
                '7 days', '8 days', '9 days'],
               dtype='timedelta64[ns]', freq='D')
```

Часові ряди

Можна створити об’єкт Series з індексацією у вигляді дат:

```
ind = pd.DatetimeIndex(['2020-08-01', '2020-09-01', '2020-10-01','2021-10-01','2021-11-02'])
dt  = pd.Series([100, 200, 250,150,300], index=ind)
dt
```

```
2020-08-01    100
2020-09-01    200
2020-10-01    250
2021-10-01    150
2021-11-02    300
dtype: int64
```

dt.loc['2020-09-01':'2021-10-01’]

```
2020-09-01    200
2020-10-01    250
2021-10-01    150
dtype: int64
```

dt.loc['2020']

```
2020-08-01    100
2020-09-01    200
2020-10-01    250
dtype: int64
```

Часові ряди

dt.loc[‘2020-09-15’:’2021-10-15’]

```
2020-10-01    250
2021-10-01    150
dtype: int64
```

ind = pd.DatetimeIndex([‘2020-08-01’, ‘2020-09-01’, ‘2020-10-01’,’2020-10-07’,’2020-10-23’])

dt = pd.Series([100, 200, 250,150,300], index=ind)

```
2020-08-01    100
2020-09-01    200
2020-10-01    250
2020-10-07    150
2020-10-23    300
dtype: int64
```

dt.loc['2020-10']

```
2020-10-01    250
2020-10-07    150
2020-10-23    300
dtype: int64
```

Часові ряди

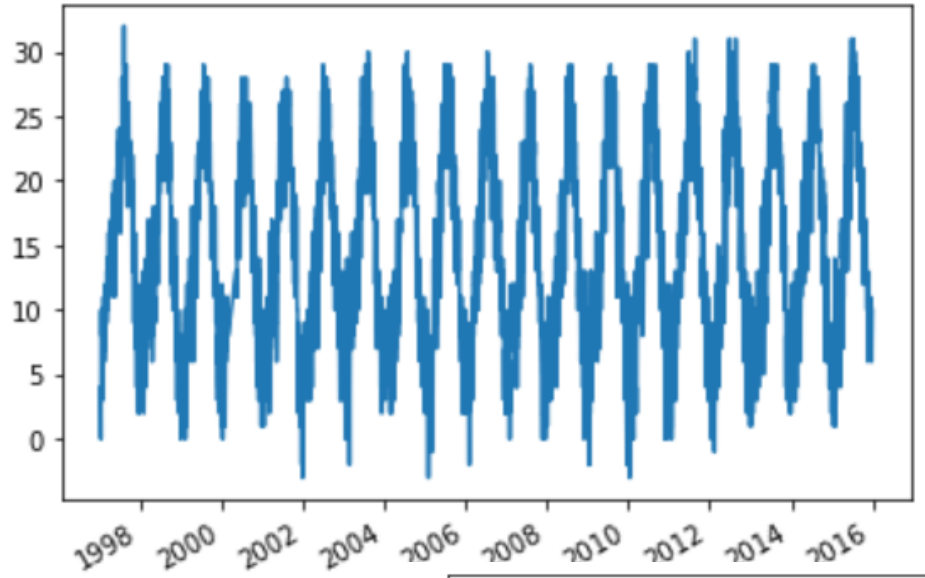
Візьмемо набір даних про погоду та виділимо середні температури. При цьому при зчитуванні файлу потрібно встановити параметр parse_dates.

df=pd.read_csv('weather.csv',index_col=['CET'],parse_dates=True)

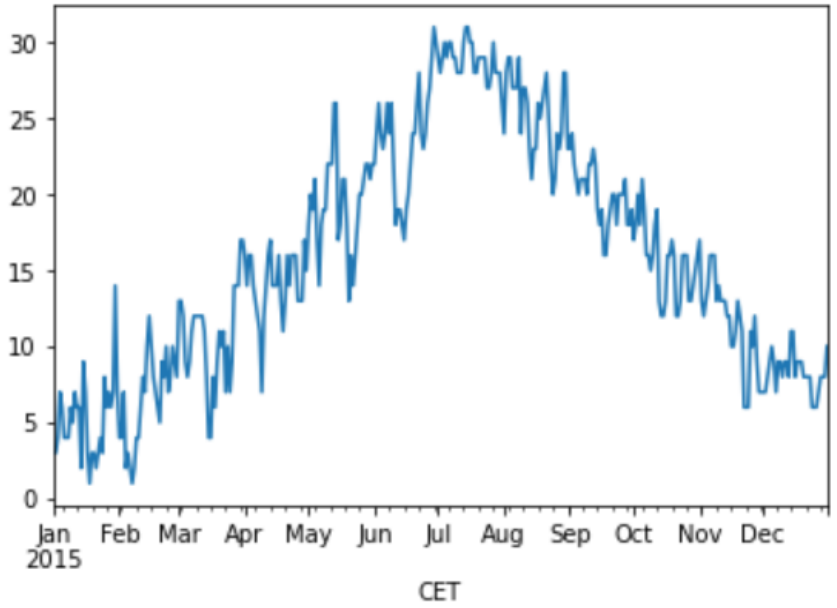
```
1997-1-1       4.0
1997-1-2       3.0
1997-1-3       3.0
1997-1-4       3.0
1997-1-5       0.0
               ...
2015-12-27     7.0
2015-12-28     8.0
2015-12-29     8.0
2015-12-30     8.0
2015-12-31    10.0
Name: Mean TemperatureC, Length: 6812, dtype: float64
```

Часові ряди

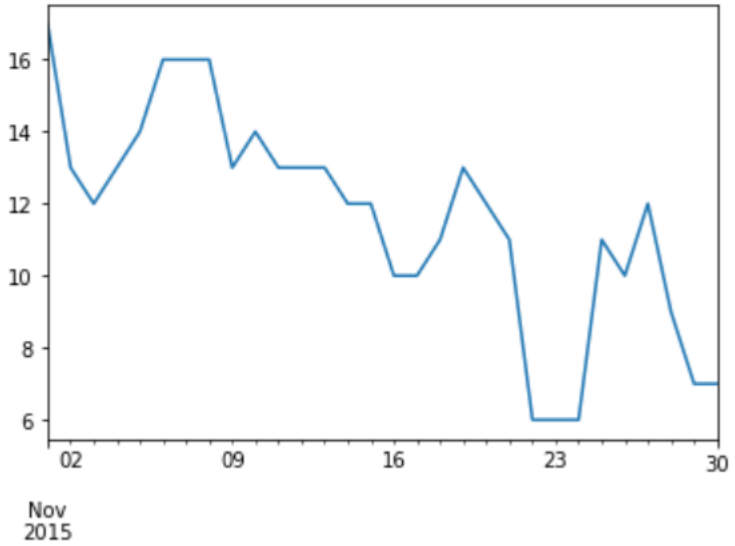
df.plot()



df.loc['2015'].plot()



df.loc['2015-11'].plot()



Часові ряди

df.loc['2013'].first('1W’)

```
2013-01-01    6.0
2013-01-02    6.0
2013-01-03    7.0
2013-01-04    4.0
2013-01-05    4.0
2013-01-06    3.0
```

df.loc['2013-11'].first('1W’)

```
2013-11-01    10.0
2013-11-02    12.0
2013-11-03    12.0
```

df.loc['2014-12'].last('1W')

```
CET
2014-12-29    5.0
2014-12-30    3.0
2014-12-31    2.0
```

Часові ряди

При роботі з даними часових рядів часто буває необхідно перерозбити дані з використанням інтервалів інший періодичності. Зробити це можна за допомогою методу resample() або набагато простішого методу asfreq(). Основна різниця між ними полягає в тому, що resample() виконує агрегування даних, а asfreq() - вибірку даних.

df.resample('M').mean()

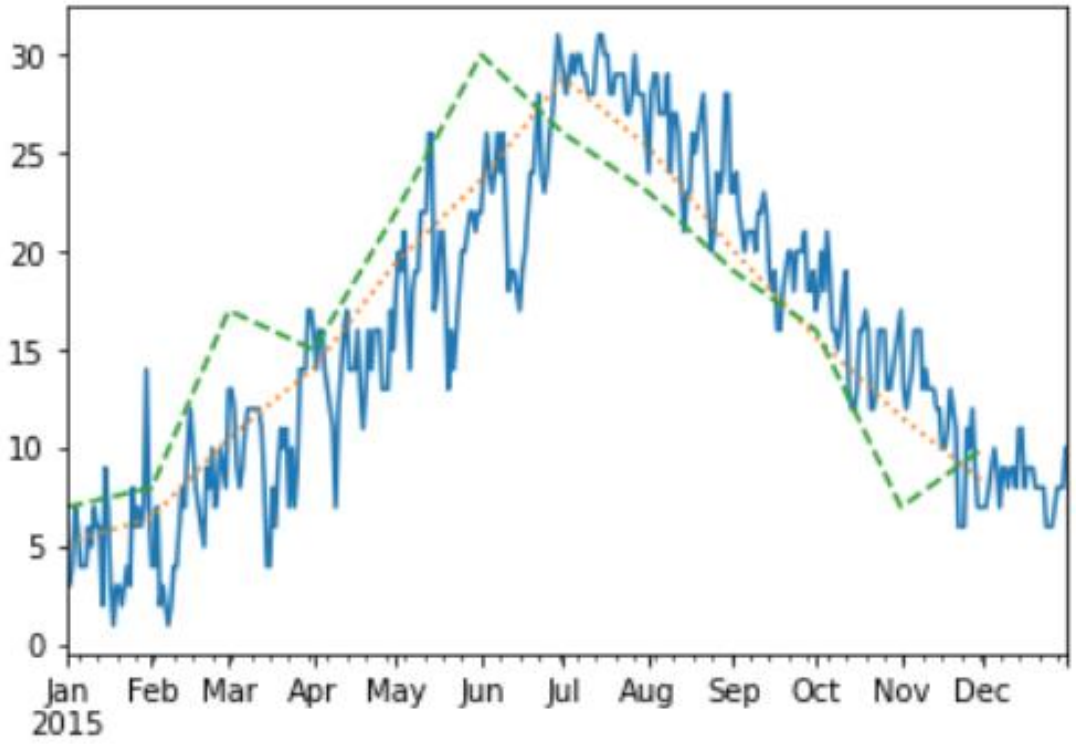
```
CET
1997-01-31     6.967742
1997-02-28     9.607143
1997-03-31    12.967742
1997-04-30    15.500000
1997-05-31    16.709677
```

df.asfreq('M')

```
CET
1997-01-31     9.0
1997-02-28    11.0
1997-03-31    14.0
1997-04-30    19.0
1997-05-31    17.0
```

Часові ряди

df.loc[‘2015’].plot(style=‘-’)
df.loc['2015'].resample('M').mean().plot(style=':’)
df.loc['2015'].asfreq('M').plot(style='--')

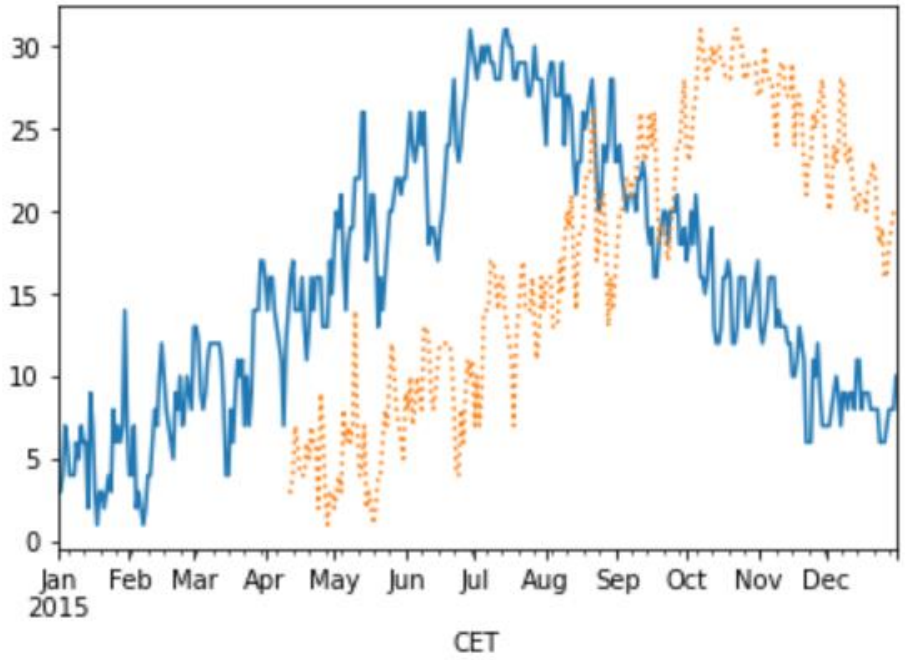


Часові ряди

Ще одна поширена операція з часовими рядами - зсув даних у часі. У бібліотеці Pandas є метод для подібних обчислень - shift(), який виконує зсув даних. Зсув задається кратним періодом. За замовчуванням відбувається зсув на один період

df.loc['2015'].plot(style='-’)
df.loc['2015'].shift(100).plot(style=':')

CET	Temperature	shift
1997-01-01	4.0	NaN
1997-01-02	3.0	4.0
1997-01-03	3.0	3.0
1997-01-04	3.0	3.0
1997-01-05	0.0	3.0



Часові ряди

df.diff()

CET	
1997-01-01	4.0
1997-01-02	3.0
1997-01-03	3.0
1997-01-04	3.0
1997-01-05	0.0

df-df.shift()

CET	
1997-01-01	NaN
1997-01-02	-1.0
1997-01-03	0.0
1997-01-04	0.0
1997-01-05	-3.0

CET	
1997-01-01	NaN
1997-01-02	-1.0
1997-01-03	0.0
1997-01-04	0.0
1997-01-05	-3.0

Часові ряди

Метод pct_change() розраховує зміну у відсотках за заданий період:
weather['change']=weather['Temperature'].pct_change()
weather.loc['2013-10'].head(7)

CET	Temperature	change
2013-10-01	22.0	0.157895
2013-10-02	21.0	-0.045455
2013-10-03	21.0	0.000000
2013-10-04	21.0	0.000000
2013-10-05	18.0	-0.142857
2013-10-06	18.0	0.000000
2013-10-07	18.0	0.000000

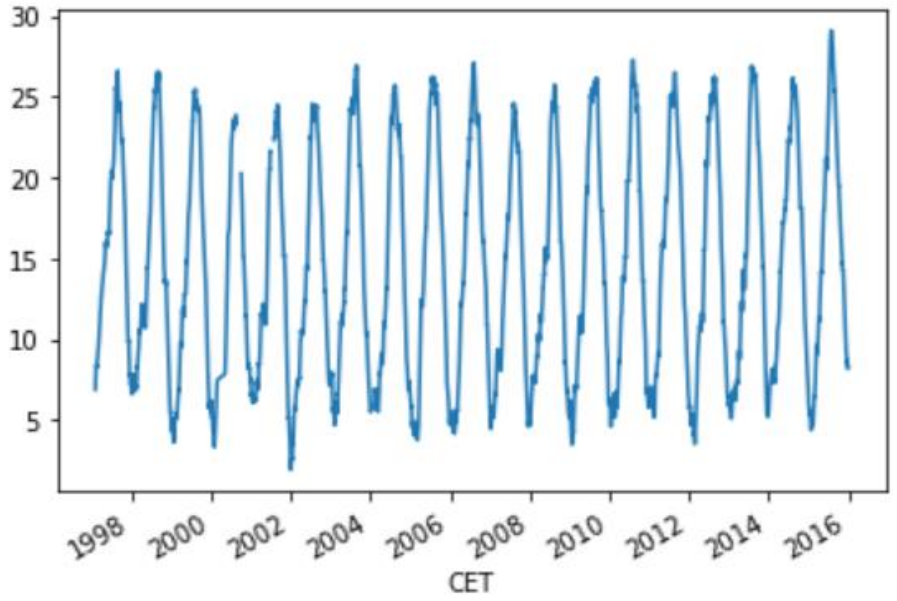
pct_change(periods=3)

CET	Temperature	change
2013-10-01	22.0	0.100000
2013-10-02	21.0	0.105263
2013-10-03	21.0	0.105263
2013-10-04	21.0	-0.045455
2013-10-05	18.0	-0.142857
2013-10-06	18.0	-0.142857
2013-10-07	18.0	-0.142857

Часові ряди

Ковзаючі статистичні показники - ще один з різновидів операцій, призначених для часових рядів. Працювати з ними можна за допомогою атрибуту rolling() об'єктів Series і DataFrame, що повертає представлення, схоже на виконання операції groupby. Далі можна застосовувати будь-яку агрегуючу функцію.

df.rolling(30).mean().plot()



Часові ряди

weather['rolling']=weather['Temperature'].rolling(2).mean()

CET	Temperature	rolling
1997-01-01	4.0	NaN
1997-01-02	3.0	3.5
1997-01-03	3.0	3.0
1997-01-04	3.0	3.0
1997-01-05	0.0	1.5
1997-01-06	3.0	1.5
1997-01-07	0.0	1.5

rolling(4).max()

CET	Temperature	rolling
1997-01-01	4.0	NaN
1997-01-02	3.0	NaN
1997-01-03	3.0	NaN
1997-01-04	3.0	4.0
1997-01-05	0.0	3.0
1997-01-06	3.0	3.0
1997-01-07	0.0	3.0

CET	Temperature	rolling
2015-09-01	23.0	28.0
2015-09-02	24.0	28.0
2015-09-03	22.0	24.0
2015-09-04	21.0	24.0
2015-09-05	20.0	24.0
2015-09-06	21.0	22.0
2015-09-07	21.0	21.0

Робота з файлами

Функції для читання файлів в Pandas:
pd.read_csv() – можна зчитуваті дані як з локального csv-файлу, так і за інтернет-посиланням. За замовчанням використовує кому як роздільник.
pd.read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data/2021/2021-10-19/pumpkins.csv’)

	id	place	weight_lbs	grower_name	city	state_prov	country	gpc_site	seed_mother	pollinator_father	ott	est_weight	pct_chart
0	2013-F	1	154.50	Ellenbecker, Todd & Sequoia	Gleason	Wisconsin	United States	Nekoosa Giant Pumpkin Fest	209 Werner	Self	184.0	129.00	20.0
1	2013-F	2	146.50	Razo, Steve	New Middletown	Ohio	United States	Ohio Valley Giant Pumpkin Growers Weigh-off	150.5 Snyder	NaN	194.0	151.00	-3.0
2	2013-F	3	145.00	Ellenbecker, Todd & Sequoia	Glenson	Wisconsin	United States	Mishicot Pumpkin Fest	209 Werner	103 Mackinnon	177.0	115.00	26.0
3	2013-F	4	140.80	Martin, Margaret & Glen	Combined Locks	Wisconsin	United States	Cedarburg Wine and Harvest Festival	109 Martin '12	209 Werner '12	194.0	151.00	-7.0

Робота з файлами

Основні параметри pd.read_csv():
sep – роздільник
header – заголовок, за замовчуванням header=0, тобто заголовком стає перший рядок.
names – заголовки стовпців
index_col – встановлює стовпець (ці) як індекси
dtype – типи даних
skiprows – які рядки пропустити, вказавши кількість або індекс
nrows – кількість рядків, які потрібно прочитати
na_values – які символи розпізнавати як відсутні значення

Робота з файлами

skip_blank_lines – чи пропускати порожні рядки
comment – вказує, з яких символів починається коментар
converters – словник функцій, які потрібно застосувати до певних стовпців
thousands – роздільник тисяч
decimal – роздільник цілої частини, за замовчанням крапка

Робота з файлами

pd.read_table() - за замовчанням використовує табуляцію як роздільник. Має аналогічні до pd.read_csv() параметри.

pd. read_fwf() – читає файли з фіксованою шириною стовпців, без роздільників

pd.read_clipboard () – читає дані з буферу обміну.

Робота з файлами

За замовчанням перший рядок файлу стає іменами стовпців, але не всі файли мають ці заголовки.

pd.read_csv('users.csv’)

	14	male	city	student	0	no
0	34	female	city	teacher	22000	no
1	42	male	countryside	banker	24000	yes

Можна вказати відсутність заголовка:

pd.read_csv('users.csv',header=None)

	0	1	2	3	4	5
0	14	male	city	student	0	no

Робота з файлами

Але можна і прописати заголовки самостійно за допомогою параметра names:

pd.read_csv('users.csv', names=['Age', 'Gender', 'Region', ' Occupation','Income','Laptop’])

	Age	Gender	Region	Occupation	Income	Laptop
0	14	male	city	student	0	no
1	34	female	city	teacher	22000	no
2	42	male	countryside	banker	24000	yes
3	30	male	countryside	teacher	25000	no

Робота з файлами

pd.read_csv('Laptop-Users.csv', names=['Вік', 'Стать', 'Місцевість', 'Професія','Дохід','Має ноут'],header=0)

	Вік	Стать	Місцевість	Професія	Дохід	Має ноут
1	14	male	city	student	0	no
2	34	female	city	teacher	22000	no
3	42	male	countryside	banker	24000	yes

pd.read_csv('Laptop-Users.csv', names=['Вік', 'Стать', 'Місцевість', 'Професія','Дохід','Має ноут'],skiprows=[0])

	Вік	Стать	Місцевість	Професія	Дохід	Має ноут
0	14	male	city	student	0	no
1	34	female	city	teacher	22000	no

Робота з файлами

Можна зробити індексом один зі стовпців або масив стовпців:
pd.read_csv('Laptop-Users.csv',index_col='Age’)

Age	Gender	Region	Occupation	Income	Has Laptop
14	male	city	student	0	no
34	female	city	teacher	22000	no
42	male	countryside	banker	24000	yes

pd.read_csv('Laptop-Users.csv',index_col=['Age','Gender'])

Age	Gender	Region	Occupation	Income	Has Laptop
14	male	city	student	0	no
34	female	city	teacher	22000	no
42	male	countryside	banker	24000	yes

Робота з файлами

Наприклад, потрібно зчитати файл з іншими розділювачами:

```
I  II III  IV
1 a  b  c   d
2  e    r    y   u
```

pd.read_table('text.txt’)

	I II III IV
0	1 a b c d
1	2 e r y u

df= pd.read_table('text.txt',sep='\s+’)

	I	II	III	IV
1	a	b	c	d
2	e	r	y	u

df['I']

```
1    a
2    e
Name: I, dtype: object
```

Робота з файлами

Відсутні значення зазвичай або взагалі пропущені (порожні рядки), або представлені спеціальними маркерами. За замовчуванням pandas використовується набір загальновживаних маркерів.

pd.read_table('text.txt',sep='\s+',na_values=['b','u'])

	I	II	III	IV
1	a	NaN	c	d
2	e		r	y	NaN

Робота з файлами

Для обробки великих файлів або для того, щоб визначити правильний набір аргументів, необхідних для обробки великого файлу, іноді потрібно прочитати невеликий фрагмент файлу або послідовно читати файл невеликими порціями.

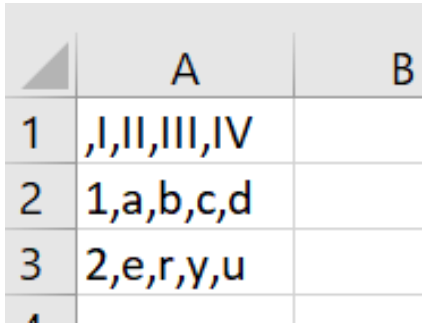
pd.read_csv('Laptop-Users.csv',nrows=3)

	Age	Gender	Region	Occupation	Income	Has Laptop
0	14	male	city	student	0	no
1	34	female	city	teacher	22000	no
2	42	male	countryside	banker	24000	yes

Робота з файлами

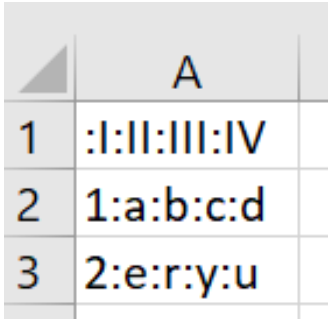
За допомогою методу to_csv об'єкта DataFrame можна вивести дані у файл csv:

```
df=pd.read_table('text.txt',sep='\s+')
df.to_csv('text.csv’)
```



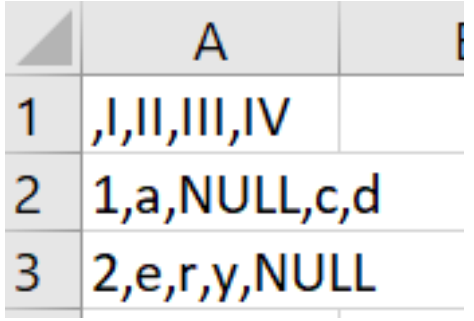
	A	B
1	,I,II,III,IV	
2	1,a,b,c,d	
3	2,e,r,y,u	

```
df.to_csv('text.csv',sep=':’)
```



	A
1	:I:II:III:IV
2	1:a:b:c:d
3	2:e:r:y:u

```
df.to_csv('text.csv',na_rep='NULL')
```



	A
1	,I,II,III,IV
2	1,a,NULL,c,d
3	2,e,r,y,NULL

Робота з файлами

Для читання файлу типу json використовується функція pd.read_json
pd.read_json('iris.json’)

	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

df.to_json('text.json’)

```
{"I  II III  IV":{"0":"1 a  b  c  d","1":"2  e   r   y  u"}}
```

df.to_json(orient='split')

```
'{"columns":["I  II III  IV"],"index":[0,1],"data":[["1 a  b  c  d"],["2  e   r   y  u"]]}'
```

Робота з файлами

df.to_json(orient='index’)

```
'{"0":{"I  II III  IV":"1 a  b  c   d"},"1":{"I  II III  IV":"2  e    r    y   u"}}'
```

df.to_json(orient='values')

```
[["1 a  b  c   d"],["2  e    r    y   u"]]
```

Параметр orient визначає формат рядків JSON. Основні його значення:

'split’ : словник {index -> [index], columns -> [columns], data -> [values]}

'records’ : список [{column -> value}, ... , {column -> value}]

'index’ : словник {index -> {column -> value}}

'columns’ : словник {column -> {index -> value}}

'values’ : масив значень

Робота з файлами

Для читання екселівських файлів використовується функція pd.read_excel

df=pd.read_excel(' Letters.xlsx ‘)

	Unnamed: 0	Letters
0	1	a
1	2	b
2	3	c

pd.read_excel('Letters.xlsx',index_col=0)

	Letters
1	a
2	b

Робота з файлами

Для читання html-файлів використовується функція pd.read_html

```
url = 'https://raw.githubusercontent.com/pandas-dev/pandas/master/pandas/tests/io/data/html/banklist.html'
pd.read_html(url)
```

```
[                                 Bank Name           City  ST   CERT  \
 0    Banks of Wisconsin d/b/a Bank of Kenosha       Kenosha  WI  35386
 1                      Central Arizona Bank     Scottsdale  AZ  34527
 2                             Sunrise Bank       Valdosta  GA  58185
 3                   Pisgah Community Bank      Asheville  NC  58701
 4                     Douglas County Bank   Douglasville  GA  21649
 ..                                    ...            ...  ..    ...
 501                  Superior Bank, FSB       Hinsdale  IL  32646
 502                 Malta National Bank          Malta  OH   6629
 503           First Alliance Bank & Trust Co.   Manchester  NH  34264
 504       National State Bank of Metropolis   Metropolis  IL   3815
 505                      Bank of Honolulu       Honolulu  HI  21029

                  Acquiring Institution       Closing Date       Updated Date
 0                North Shore Bank, FSB      May 31, 2013       May 31, 2013
```

Робота з файлами

url='https://en.wikipedia.org/wiki/TRAPPIST-1’
df = pd.read_html(url)

```
[                                                    0  \
 0   TRAPPIST-1 is within the red circle in the con...
 1          Observation data Epoch J2000 Equinox J2000
 2                                       Constellation
 3                                    Right ascension
 4                                        Declination
 5                                    Characteristics
 6                                 Evolutionary stage
 7                                       Spectral type
 8                           Apparent magnitude (V)
 9                           Apparent magnitude (R)
 10                          Apparent magnitude (I)
 11                          Apparent magnitude (J)
 12                          Apparent magnitude (H)
 13                          Apparent magnitude (K)
 14                                 V−R color index
```

len(df)
18

Робота з файлами

Нехай потрібна дана таблиця:

The TRAPPIST-1 planetary system[5][40][7]

Companion (in order from star)	Mass	Semimajor axis (AU)	Orbital period (days)	Eccentricity[40]	Inclination[7][41]	Radius
b	1.374 ± 0.069 $M_\oplus$	0.01154 ± 0.0001	1.51088432 ± 0.00000015	0.006 22 ± 0.003 04	89.56 ± 0.23°	$1.116^{+0.014}_{-0.012}$ $R_\oplus$
c	1.308 ± 0.056 $M_\oplus$	0.01580 ± 0.00013	2.42179346 ± 0.00000023	0.006 54 ± 0.001 88	89.70 ± 0.18°	$1.097^{+0.014}_{-0.012}$ $R_\oplus$
d	0.388 ± 0.012 $M_\oplus$	0.02227 ± 0.00019	4.04978035 ± 0.00000256	0.008 37 ± 0.000 93	$89.89^{+0.08}_{-0.15}$°	$0.778^{+0.011}_{-0.010}$ $R_\oplus$
e	0.692 ± 0.022 $M_\oplus$	0.02925 ± 0.00025	6.09956479 ± 0.00000178	0.005 10 ± 0.000 58	$89.736^{+0.053}_{-0.066}$°	$0.920^{+0.013}_{-0.012}$ $R_\oplus$
f	1.039 ± 0.031 $M_\oplus$	0.03849 ± 0.00033	9.20659399 ± 0.00000212	0.010 07 ± 0.000 68	$89.719^{+0.026}_{-0.039}$°	$1.045^{+0.013}_{-0.012}$ $R_\oplus$
g	1.321 ± 0.038 $M_\oplus$	0.04683 ± 0.0004	12.3535557 ± 0.00000341	0.002 08 ± 0.000 58	$89.721^{+0.019}_{-0.026}$°	$1.129^{+0.015}_{-0.013}$ $R_\oplus$
h	0.326 ± 0.020 $M_\oplus$	0.06189 ± 0.00053	18.7672745 ± 0.00001876	0.005 67 ± 0.001 21	89.796 ± 0.023°	0.775 ± 0.014 $R_\oplus$

Робота з файлами

planets = pd.read_html(url,match='The TRAPPIST-1 planetary system',header=0)

```
[  Companion(in order from star)           Mass Semimajor axis(AU)  \
 0                              b  1.374±0.069 M⊕     0.01154±0.0001
 1                              c  1.308±0.056 M⊕    0.01580±0.00013
 2                              d  0.388±0.012 M⊕    0.02227±0.00019
 3                              e  0.692±0.022 M⊕    0.02925±0.00025
 4                              f  1.039±0.031 M⊕    0.03849±0.00033
 5                              g  1.321±0.038 M⊕     0.04683±0.0004
 6                              h  0.326±0.020 M⊕    0.06189±0.00053

    Orbital period(days) Eccentricity[40]   Inclination[7][41]  \
 0  1.51088432±0.00000015  0.00622±0.00304          89.56±0.23°
 1  2.42179346±0.00000023  0.00654±0.00188          89.70±0.18°
 2  4.04978035±0.00000256  0.00837±0.00093     89.89+0.08−0.15°
 3  6.09956479±0.00000178  0.00510±0.00058  89.736+0.053−0.066°
 4  9.20659399±0.00000212  0.01007±0.00068  89.719+0.026−0.039°
 5  12.3535557±0.00000341  0.00208±0.00058  89.721+0.019−0.026°
 6  18.7672745±0.00001876  0.00567±0.00121        89.796±0.023°

                  Radius
 0  1.116+0.014−0.012 R⊕
 1  1.097+0.014−0.012 R⊕
```

Робота з файлами

Результат все одно буде списком з об’єктів DataFrame, навіть коли такий об’єкт один. Тому, щоб отримати сам об’єкт, потрібно обрати перший елемент списку:

df=planets[0]

	Companion(in order from star)	Mass	Semimajor axis(AU)	Orbital period(days)	Eccentricity[40]	Inclination[7][41]	Radius
0	b	1.374±0.069 M⊕	0.01154±0.0001	1.51088432±0.00000015	0.00622±0.00304	89.56±0.23°	1.116+0.014−0.012 R⊕
1	c	1.308±0.056 M⊕	0.01580±0.00013	2.42179346±0.00000023	0.00654±0.00188	89.70±0.18°	1.097+0.014−0.012 R⊕
2	d	0.388±0.012 M⊕	0.02227±0.00019	4.04978035±0.00000256	0.00837±0.00093	89.89+0.08−0.15°	0.778+0.011−0.010 R⊕
3	e	0.692±0.022 M⊕	0.02925±0.00025	6.09956479±0.00000178	0.00510±0.00058	89.736+0.053−0.066°	0.920+0.013−0.012 R⊕
4	f	1.039±0.031 M⊕	0.03849±0.00033	9.20659399±0.00000212	0.01007±0.00068	89.719+0.026−0.039°	1.045+0.013−0.012 R⊕
5	g	1.321±0.038 M⊕	0.04683±0.0004	12.3535557±0.00000341	0.00208±0.00058	89.721+0.019−0.026°	1.129+0.015−0.013 R⊕