pandas時系列データ（基本）

- 「Pythonによるデータ分析入門」P347～より抜粋

標準ライブラリ関連

datetime

```
from datetime import datetime

now = datetime.now()
now
>>datetime.datetime(2021, 5, 8, 9, 49, 16, 657431)

now.year, now.month, now.day
>>(2021, 5, 8)

delta = datetime(2021, 5, 8) - datetime(2021, 1, 1)
delta
>>datetime.timedelta(days=127)

delta = datetime(2021, 5, 8, 9, 36) -datetime(2021,1,1)
delta
>>datetime.timedelta(days=127, seconds=34560)

delta.days
>>127
```

timedelta

```
from datetime imort timedelta
start = datetime(2021, 5,8)
start + timedelta(60)
>>datetime.datetime(2021, 7, 7, 0, 0)
```

strftime, strptime

```
stamp = datetime(2021, 5, 8)
stamp
>>datetime.datetime(2021, 5, 8, 0, 0)
str(stamp)
>>'2021-05-08 00:00:00'

stamp.strftime('%Y%m%d')
>>'20210508'

stamp.strftime('%y-%m-%d')
>>'21-05-08'

# 文字列をdate型へ型を合わせてパース
value = '20210508'
datetime.strptime(value, '%Y%m%d')
>>datetime.datetime(2021, 5, 8, 0, 0)

datestrs = ['2021/4/1', '2021/5/1']
[datetime.strptime(x, '%Y/%m/%d) for x in datestrs]
>>[datetime.datetime(2021, 4, 1, 0, 0), datetime.datetime(2021, 5, 1, 0, 0)]
```

Datetimeフォーマット一覧

型	説明
%Y	yyyy
%y	yy
%m	月mm
%d	日dd
%H	時間(24時間)
%I	時間(12時間)
%M	2桁の分
%S	2桁の秒
%w	曜日(0:日曜～)
%F	%Y-%M-%dの短縮形2021-5-8
%D	%m%d%yの短縮形 08/05/21

dateutil.parser > parse

```
from dateutil.parser import parse

# 自動でパースしてくれる
parse(‘2021/5/8’)
>>datetime.datetime(2021, 5, 8, 0, 0)

parse(‘2021/5/8 10:15’)
>>datetime.datetime(2021, 5, 8, 10, 15)

# これは無理なのでstrptimeでパースがよい
parse(‘210508’)
>>datetime.datetime(2008, 5, 21, 0, 0)

# これも無理、年は4桁がよいのかも
parse(‘21/5/8’)
>>datetime.datetime(2008, 5, 21, 0, 0)
```

Pandas関連

pd.to_datetime()

- 日時データに変換
- pandasの場合、標準の日付であればパース可能

```
 datestrs = ['2021-05-08 10:20:00', '2021-05-09 10:20:00']

 pd.to_datetime(datestrs)

 >>DatetimeIndex(['2021-05-08 10:20:00', '2021-05-09 10:20:00'],
dtype='datetime64[ns]', freq=None)
```

pd.date_range()

- 日時データ範囲を生成

```
 pd.date_range('2021/5/1', '2021/5/10')
 >>DatetimeIndex(['2021-05-01', '2021-05-02', '2021-05-03',～
'2021-05-10'], dtype='datetime64[ns]', freq='D')

 pd.date_range(start='2021/5/1', periods=10)
 >>DatetimeIndex(['2021-05-01', '2021-05-02', '2021-05-03',～
'2021-05-10'], dtype='datetime64[ns]', freq='D')

 pd.date_range(end='2021/5/28', periods=5)
 >>DatetimeIndex(['2021-05-24', '2021-05-25', '2021-05-26',～
'2021-05-28'],dtype='datetime64[ns]', freq='D')

 # タイムスタンプとしては午前零時に標準化したい場合
 pd.date_range('2021-05-09 12:24:28', periods=5, normalize=True)
 >>DatetimeIndex(['2021-05-09', '2021-05-10', '2021-05-11',～
'2021-05-13'],dtype='datetime64[ns]', freq='D')
```

```
# 毎月1日
pd.date_range('2021/5/1', periods=5, freq='MS')
>>DatetimeIndex(['2021-05-01', '2021-06-01', '2021-07-01',～
'2021-09-01'],dtype='datetime64[ns]', freq='MS')

# 毎月末
pd.date_range('2021/5/1', periods=5, freq='M')
>>DatetimeIndex(['2021-05-31', '2021-06-30', '2021-07-31',～
'2021-09-30'],dtype='datetime64[ns]', freq='M')
```

時系列の基準頻度（P359より抜粋）

文字	オフセットクラス	説明
B	BusinessDay	毎営業日
M	MonthEnd	暦通りの月末ごと
BM	BusinessMonthEnd	月の最終営業日
MS	MonthBegin	暦通りの月初ごと
BMS	BusinessMonthBegin	月の営業開始日ごと
W-MON,W-TUE,...	Week	毎週指定した曜日ごと

```
# 時系列のサンプル作成
index = pd.date_range('2021/5/1', periods=100)
ts = pd.DataFrame(np.random.randn(100), index=index, columns=['VALUE'])


# 5月だけを取り出す
ts['2021/5']

# 以降を取り出す
ts['2021/5/25':]

# 一定期間を取り出す
ts['2021/5/25':'2021/6/12']
```

文字列⇄日時の型変換

- note.mknk.meより
- pd.read_csv等で取り込んだ場合、すべて文字列になっている

	A	B
0	2017-11-01 12:24	2017年11月1日 12時24分
1	2017-11-18 23:00	2017年11月18日 23時00分
2	2017-12-05 5:05	2017年12月5日 5時05分
3	2017-12-22 8:54	2017年12月22日 8時54分
4	2018-01-08 14:20	2018年1月8日 14時20分
5	2018-01-19 20:01	2018年1月19日 20時01分

1.読込み後：文字列→datetime型に変換（読込み後）

```
df.dtypes
>>A object
>>B object
>>dtype: object

type(df['A'][0])
>>str

# 文字列→datetime型へ変換（デフォルト）
pd.to_datetime(df['A'])
>>0   2017-11-01 12:24:00
>>1   2017-11-18 23:00:00
>>2   2017-12-05 05:05:00
～|
>>Name: A, dtype: datetime64[ns]

# 文字列→datetime型へ変換（フォーマット指定。上記でうまくいかない場合）
pd.to_datetime(df['A'], format='%Y年%m月%d日% %H時%M分')
>>0   2017-11-01 12:24:00
>>1   2017-11-18 23:00:00
>>2   2017-12-05 05:05:00
～
>>Name: B, dtype: datetime64[ns]

# 重要：列を新規追加（datetime型）（オリジナルを変えず列を追加するのがよい）
df['X'] = pd.to_datetime(df['A'],～)
```

	A	B	X
0	2017-11-01 12:24	2017年11月1日 12時24分	2017-11-01 12:24:00
1	2017-11-18 23:00	2017年11月18日 23時00分	2017-11-18 23:00:00
2	2017-12-05 5:05	2017年12月5日 5時05分	2017-12-05 05:05:00
3	2017-12-22 8:54	2017年12月22日 8時54分	2017-12-22 08:54:00
4	2018-01-08 14:20	2018年1月8日 14時20分	2018-01-08 14:20:00
5	2018-01-19 20:01	2018年1月19日 20時01分	2018-01-19 20:01:00

```
df.dtypes
>>A             object
>>B             object
>>X    datetime64[ns]
>>dtype: object
```

```
# 年の抽出
df['X'].dt.year   ←month, dayでも同様
0    2017
1    2017
2    2017
3    2017
4    2018
5    2018
Name: X, dtype: int64
```

```
# 曜日(6:日曜日)を表示
df['X'].dt.dayofweek
0    2
1    5
2    1
3    4
4    0
5    4
Name: X, dtype: int64
```

```
# 月曜日だけを表示
df[df['X'].dt.dayofweek == 0]
```

	A	B	X
4	2018-01-08 14:20	2018年1月8日 14時20分	2018-01-08 14:20:00

2.読込み後：datetime型→文字列に変換

```
# datetime型→文字列に変換（デフォルト）
df['X'].astype(str)
>>0    2017-11-01 12:24:00
>>1    2017-11-18 23:00:00
>>2    2017-12-05 05:05:00
>>~
>>Name: X, dtype: object

# datetime型→文字列に変換（フォーマット指定）
df['X'].dt.strftime('%Y年%m月%d日')
>>0    2017年11月01日
>>1    2017年11月18日
>>2    2017年12月05日
>>~
>>Name: X, dtype: object

# 列を新規追加
df['日付'] = df['X'].dt.strftime('%Y年%m月%d日')
```

3.読込み時：文字列→datetime型に変換

```
url =
'https://raw.githubusercontent.com/nkmk/python-snippets/8501c6f55aa9b1c5ecfc940f18ba4aef23325cd8/notebook/data/src/sample_datetime_multi.csv'
```

```
# parse_datesに、datetime型へ変換したい列（列：0）を渡す
df_csv = pd.read_csv(url, parse_dates=[0])
df_csv
```

	A	B
0	2017-11-01 12:24:00	2017年11月1日 12時24分
1	2017-11-18 23:00:00	2017年11月18日 23時00分
2	2017-12-05 05:05:00	2017年12月5日 5時05分
3	2017-12-22 08:54:00	2017年12月22日 8時54分
4	2018-01-08 14:20:00	2018年1月8日 14時20分
5	2018-01-19 20:01:00	2018年1月19日 20時01分

```
df_csv.dtypes
>>A    datetime64[ns]
>>B            object
>>dtype: object
```

```
# 標準的な書式でない場合は「date_parser」に読み込む書式を渡して読み込む
# Lambda箇所はわかりにくい（dateはデフォルト？）
df_csv2 = pd.read_csv(url,
                      parse_dates=[1],
                      date_parser=lambda date: pd.to_datetime(date, \
                              format='%Y年%m月%d日　%H時%M分'))
```

	A	B
0	2017-11-01 12:24	2017-11-01 12:24:00
1	2017-11-18 23:00	2017-11-18 23:00:00
2	2017-12-05 5:05	2017-12-05 05:05:00
3	2017-12-22 8:54	2017-12-22 08:54:00
4	2018-01-08 14:20	2018-01-08 14:20:00
5	2018-01-19 20:01	2018-01-19 20:01:00

```
df_csv2.dtypes
>>A            object
>>B    datetime64[ns]
>>dtype: object
```

→仮にdate_parserなしで読み込むと、Bはobjectのまま読み込まれる

```
# indexに指定した列をdatetime型へ変換する場合(parse_dates=True)
df_csv3 = pd.read_csv(url,
                      index_col=1,
                      parse_dates=True,
                      date_parser=lambda date: pd.to_datetime(date,\
                         format='%Y年%m月%d日 %H時%M分'))
df_csv3
```

B	A
2017-11-01 12:24:00	2017-11-01 12:24
2017-11-18 23:00:00	2017-11-18 23:00
2017-12-05 05:05:00	2017-12-05 5:05
2017-12-22 08:54:00	2017-12-22 8:54
2018-01-08 14:20:00	2018-01-08 14:20
2018-01-19 20:01:00	2018-01-19 20:01

```
# pd.read_excel()でもparse_dates, date_parser を利用可能
```

DatetimeIndex

- datetime64[ns]型のDataFrameで日付をindexにするとDatetimeIndexになる
- いろんな機能が使えるらしいがよくわかっていない

```
df_i = df.set_index('X')
df_i
```

X	A	B
2017-11-01 12:24:00	2017-11-01 12:24	2017年11月1日 12時24分
2017-11-18 23:00:00	2017-11-18 23:00	2017年11月18日 23時00分
2017-12-05 05:05:00	2017-12-05 5:05	2017年12月5日 5時05分
2017-12-22 08:54:00	2017-12-22 8:54	2017年12月22日 8時54分
2018-01-08 14:20:00	2018-01-08 14:20	2018年1月8日 14時20分
2018-01-19 20:01:00	2018-01-19 20:01	2018年1月19日 20時01分

```
df_i.index.strftime('%Y%m%d')
>>Index(['20171101', '20171118', '20171205', '20171222', '20180108',
>>'20180119'], dtype='object', name='X')
```