Time Series

1 Kiểu dữ liệu thời gian trong Python

1.1 Giới thiệu

Khi nhắc đến thời gian, người ta sẽ thường đề cập đến 3 dạng sau:

- Thời điểm: là một điểm cụ thể trong dòng thời gian. Ví dụ: `16/01/2020`, `11/11/2019`, ...
- Khoảng thời điểm: là một khoảng trong dòng thời gian, có điểm đầu điểm cuối xác định. Ví dụ: `năm 2019`, `quý 2 năm 2020`, ...
- Khoảng thời gian: chỉ độ dài của một khoảng trong dòng thời gian. Ví dụ: `86400 giây`, `1 năm 2 tháng 20 ngày`, ...

Sau đây, chúng ta cùng tìm hiểu một số cách phổ biến để làm việc với thời gian trong `python`

1.2 Thư viện `datetime` và `dateutil`

Thư viện `datetime` và `dateutil` là hai thư viện đi kèm khi cài dặt python. Hai thư viện này cung cấp rất nhiều công cụ để làm việc với thời gian.

Ví dụ như bạn có thể tạo ra một thời điểm bằng đối tượng `datetime` của thư viện `datetime`

[1]:
```
from datetime import datetime

# tạo ra thời điểm
d1 = datetime(year = 2019,
     month = 11,
     day = 1,
     hour = 12,
     minute = 34,
     second = 56)
d1
```

[1]:
```
datetime.datetime(2019, 11, 1, 12, 34, 56)
```

[2]:
```
# hoặc lấy thời điểm hiện tại
d2 = datetime.now()
```

```
d2
```

[2]: datetime.datetime(2022, 11, 9, 10, 34, 16, 440501)

Bạn cũng đó thể dịch chuỗi ngày tháng thành kiểu thời gian bằng thư viện `dateutil`

[3]:
```
from dateutil import parser

d3 = parser.parse('20201208')
d3
```

[3]: datetime.datetime(2020, 12, 8, 0, 0)

Khi đã có đối tượng `datetime`, bạn có thể làm được nhiều điều như in ra thứ của đối tượng `datetime` đó bằng phương thức `.strftime()`

[4]:
```
d3 = datetime.now()
d3.strftime('%Y/%m/%d')
```

[4]: '2022/11/09'

Phương thức `.strftime(<format>)` hỗ trợ chuyển đổi đối tượng `datetime` thành `str` theo định dang truyền vào. Quy ước về định dạng bạn có thể tham khảo tại đây

`datetime` đơn giản nhưng mạnh mẽ, bạn có thể làm gần như mọi thứ bằng các đối tượng dựng sẵn của `datetime` và `dateutil`.

Tuy nhiên, khi làm việc với mảng thời gian lớn thì các kiểu này không được hiệu quả. Vì vậy, chúng ta cùng đến với thư viện `numpy`.

1.3 Thư viện numpy

Thư viện `numpy` hỗ trợ lưu trữ thời gian bằng đối tượng `datetime64`.

[5]:
```
import numpy as np

# tạo thời điểm, chính xác đến giây
d4 = np.datetime64('2019-11-19', 's')
d4
```

[5]: numpy.datetime64('2019-11-19T00:00:00')

Để tạo ra một mảng thời gian, chúng ta làm như sau

[6]:
```
# tạo mảng thời điểm trong numpy
# phải chỉ rõ dtype là một kiểu thời gian mà numpy hỗ trợ
da1 = np.array(
    ['2019-10-01', '2019-11-03'],
    dtype = np.datetime64)
da1
```

[6]:
```
array(['2019-10-01', '2019-11-03'], dtype='datetime64[D]')
```

Bạn cũng có thể sinh ra một mảng thời gian bằng `np.arange()` như sau:

[7]:
```
# để tạo mảng thời điểm bằng np.arange(), cũng phải chỉ ra dtype
da2 = np.arange('2020-11', '2020-12', dtype = 'datetime64[D]')
da2
```

[7]:
```
array(['2020-11-01', '2020-11-02', '2020-11-03', '2020-11-04',
       '2020-11-05', '2020-11-06', '2020-11-07', '2020-11-08',
       '2020-11-09', '2020-11-10', '2020-11-11', '2020-11-12',
       '2020-11-13', '2020-11-14', '2020-11-15', '2020-11-16',
       '2020-11-17', '2020-11-18', '2020-11-19', '2020-11-20',
       '2020-11-21', '2020-11-22', '2020-11-23', '2020-11-24',
       '2020-11-25', '2020-11-26', '2020-11-27', '2020-11-28',
       '2020-11-29', '2020-11-30'], dtype='datetime64[D]')
```

Tuy nhiên, đối tượng `numpy.datetime64` lại không có phương thức tương tự `.strftime()`.

Các bạn có thể đọc thêm về đối tượng `numpy.datetime64` tại đây

1.4 Thư viện pandas

`pandas` kết hợp ưu điểm của những cách trên thành đối tượng `Timestamp`.

[8]:
```
import pandas as pd

# tạo Timestamp
d5 = pd.Timestamp('02-01-03 04:05:06')
d5
```

[8]:
```
Timestamp('2003-02-01 04:05:06')
```

[9]:
```
# chuyển chuỗi thời gian thành Timestamp
d6 = pd.to_datetime(['Nov 18th, 2019', 'Dec 24th, 2020'])
d6
```

[9]:
```
DatetimeIndex(['2019-11-18', '2020-12-24'], dtype='datetime64[ns]', freq=None)
```

[10]:
```
# chuyển Timestamp thành chuỗi ký tự
d5.strftime('%A %d/%m/%y %H:%M:%S')
```

[10]:
```
'Saturday 01/02/03 04:05:06'
```

[11]:
```
# tạo mảng thời điểm
da3 = pd.date_range(
    start = '2019-10-01',
    end = '2019-10-15',
    freq = 'H')
da3
```

```
[11]: DatetimeIndex(['2019-10-01 00:00:00', '2019-10-01 01:00:00',
                     '2019-10-01 02:00:00', '2019-10-01 03:00:00',
                     '2019-10-01 04:00:00', '2019-10-01 05:00:00',
                     '2019-10-01 06:00:00', '2019-10-01 07:00:00',
                     '2019-10-01 08:00:00', '2019-10-01 09:00:00',
                     …
                     '2019-10-14 15:00:00', '2019-10-14 16:00:00',
                     '2019-10-14 17:00:00', '2019-10-14 18:00:00',
                     '2019-10-14 19:00:00', '2019-10-14 20:00:00',
                     '2019-10-14 21:00:00', '2019-10-14 22:00:00',
                     '2019-10-14 23:00:00', '2019-10-15 00:00:00'],
                    dtype='datetime64[ns]', length=337, freq='H')
```

2 pandas Time Series

`Time Series` là một `Series` có `index` là một mảng thời điểm (có kiểu `DatetimeIndex`).

2.1 Khởi tạo

Để khởi tạo một `Time Series`, bạn cần hai thành phần:

- Dữ liệu.
- `DatetimeIndex`: có thể tạo ra bằng `pandas.to_datetime()`, `pandas.date_range()`, ...

```
[12]: datetime_index = pd.to_datetime(['20201207', '20201208'])
      datetime_index
```

```
[12]: DatetimeIndex(['2020-12-07', '2020-12-08'], dtype='datetime64[ns]', freq=None)
```

```
[13]: t1 = pd.Series(data = [1, 2], index = datetime_index)

      t2 = pd.Series(data = [1, 2], index = ['2012-12-07', '2012-12-08'])
      print(type(t1.index)) # index có kiểu DatetimeIndex
      print(type(t2.index)) # index có kiểu bình thường
```

```
<class 'pandas.core.indexes.datetimes.DatetimeIndex'>
<class 'pandas.core.indexes.base.Index'>
```

```
[14]: print(type(t1)) # kiểu của Time Series t1 vẫn là Series
```

```
<class 'pandas.core.series.Series'>
```

2.2 Các thao tác trên `Time Series`

Về cơ bản, các thao tác trên `Series` đều có thể thực hiện trên `Time Series`.

Tuy nhiên, `Time Series` cũng có những điểm riêng của mình.

2.2.1 Truy xuất dữ liệu

Với `Time Series`, bạn có thể truy xuất bằng khoảng thời điểm.

[15]:
```
# tạo dữ liệu ngẫu nhiên
data_1 = np.random.randint(10000, size = 365)
# tạo DatetimeIndex
index_1 = pd.date_range(start = '2019-01-01', periods = 365, freq = 'D')
# tạo Time Series
ts1 = pd.Series(data = data_1, index = index_1)
ts1
```

[15]:
```
2019-01-01     144
2019-01-02    4976
2019-01-03    2703
2019-01-04    2061
2019-01-05    9211
              ...
2019-12-27    6326
2019-12-28    5084
2019-12-29    6853
2019-12-30    7408
2019-12-31     325
Freq: D, Length: 365, dtype: int32
```

[16]:
```
# lấy dữ liệu trong tháng 05 năm 2019
ts1['2019-05']
```

[16]:
```
2019-05-01      28
2019-05-02     697
2019-05-03    4634
2019-05-04     500
2019-05-05    8440
2019-05-06    1394
2019-05-07     171
2019-05-08     619
2019-05-09    7800
2019-05-10    6503
2019-05-11    3571
2019-05-12    5420
2019-05-13    6642
2019-05-14    7079
2019-05-15    5356
2019-05-16    2104
2019-05-17    3334
2019-05-18     383
2019-05-19     907
2019-05-20    5652
```

```
2019-05-21    7878
2019-05-22    2813
2019-05-23    1180
2019-05-24    9284
2019-05-25    2881
2019-05-26     517
2019-05-27    5723
2019-05-28    6795
2019-05-29    4811
2019-05-30    4447
2019-05-31    5663
Freq: D, dtype: int32
```

Chúng ta có thể dùng phương thức `.asfreq()` để tạo ra một `Time Series` mới với dữ liệu từ `Time Series` sẵn có với mốc thời điểm theo `phương pháp` nhất định.

```
<tên_Time_Series>.asfreq(<tên_phương_pháp>)
```

Trong đó, `<tên_phương_pháp>` được quy ước như sau :

Tên phương thức	Ý nghĩa	Tên phương thức	Ý nghĩa
D	Ngày trong năm	B	Ngày làm việc
W	Chủ nhật hàng tuần		
M	Ngày cuối tháng	MS	Ngày đầu tháng
Q	Ngày cuối quý	QS	Ngày đầu quý
A	Ngày cuối năm	AS	Ngày đầu năm

Nói thêm:

- `W` sẽ lấy dữ liệu là của các ngày chủ nhật, nếu muốn đổi thứ khác, có thể dùng `W-MON`, `W-TUE`, ..., `W-SAT`.
- Bạn có thể thay đổi cách tính của một quý bằng cách chỉ rõ tháng bắt đầu (kết thúc). Ví dụ `Q-FEB` nghĩa là ngày cuối của quý là ngày cuối tháng hai. Như vậy, `Q` = `Q-MAR`

[17]:
```
# lấy dữ liệu của ngày đầu tháng
ts1.asfreq('M')
```

[17]:
```
2019-01-31    3044
2019-02-28    5595
2019-03-31    4656
2019-04-30    9514
2019-05-31    5663
2019-06-30    7892
2019-07-31    5597
2019-08-31     992
2019-09-30    8889
2019-10-31    5356
```

```
2019-11-30    8056
2019-12-31     325
Freq: M, dtype: int32
```

2.2.2 Thêm dữ liệu mới

Tương tự như `Series`, bạn có thể thêm dữ liệu mới bằng cú pháp :

`<tên_Time_Series>.[<label>] = <dữ_liệu>`

Tuy nhiên, cần lưu ý là `<label>` phải là `Timestamp`, hoặc có thể chuyển đổi được thành `Timestamp`.

[18]:
```
# tạo Time Series
data_2 = [1, 2, 3]
index_2 = pd.to_datetime(['2019-01-01', '2019-02-01', '2019-03-01'])
ts2 = pd.Series(data = data_2, index = index_2)
ts2
```

[18]:
```
2019-01-01    1
2019-02-01    2
2019-03-01    3
dtype: int64
```

[19]:
```
# thêm dữ liệu mới
ts2[pd.Timestamp('2019-02-28')] = 4
ts2
```

[19]:
```
2019-01-01    1
2019-02-01    2
2019-03-01    3
2019-02-28    4
dtype: int64
```

2.2.3 Nhóm dữ liệu

`Time Series` còn cho phép bạn nhóm dữ liệu lại trước khi thực hiện các thao tác thông kê bằng phương thức `.resample()`.

Cú pháp :

`<tên_Time_Series>.resample(<cách_nhóm>)`

Một số cách nhóm đơn giản như:

- `W`: theo tuần.
- `M`: theo tháng.
- `A`: theo năm

[20]:
```
# tính tổng theo tháng
ts1.resample('MS').max()
```

[20]:
```
2019-01-01    9868
2019-02-01    9900
2019-03-01    9958
2019-04-01    9810
2019-05-01    9284
2019-06-01    9601
2019-07-01    9999
2019-08-01    9428
2019-09-01    9879
2019-10-01    9796
2019-11-01    9827
2019-12-01    9881
Freq: MS, dtype: int32
```

[21]:
```
# tính trung bình theo quý
ts1.resample('Q').mean()
```

[21]:
```
2019-03-31    5212.355556
2019-06-30    4877.549451
2019-09-30    5138.489130
2019-12-31    4945.815217
Freq: Q-DEC, dtype: float64
```