Kungfu Pandas

Lê Huỳnh Đ c

2021-06-03

Kungfu Pandas

Lê Huỳnh Đ c

2021-06-03

Contents

L i nói đ u

Gi i thi u cu n sách

Cài đ t Jupyter Lab

Cài đ t Pandas

C u trúc và ki u d li u

0.1 Series

Trong Pandas, `Series` là m ng 1 chi u bao g m m t danh sách giá tri, và m t m ng ch a index c a các giá tri. Trong d li u d ng b ng, m i Series đ c xem nh là m t c t c a b ng đó. Cách đ n gi n đ tạo 1 series nh sau

```
s = pd.Series(data, index=None, name=None)
```

Trong đó `data` có th có dạng:

- `numpy.ndarray`, `List`
- Python `dict`
- `Scalar`

`index` có th truy n ho c không, tùy vào dạng c a `data` mà `index` s đ c định nghĩa khác nhau. `name` là tên c a `Series`, giá tri này cũng không nh t thi t ph i truy n vào.

0.1.1 Các cách kh i tạo

Kh i tạo Series b ng array

Khi không truy n giá tri `index`, `Series` s m c định index c a nó là 1 m ng s nguyên t `0` đ n `len(data) - 1`

```
In [1]: pd.Series(data=[0, 1, 2], index=["a", "b", "c"], name="meow")
Out[1]:
a    0
b    1
c    2
Name: meow, dtype: int64
```

Khởi tạo Series bằng dict

```
In [1]: pd.Series({"b": 1, "a":0, "c": 2})
Out[1]:
b    1
a    0
c    2
dtype: int64
```

Lưu ý: Trong trường hợp bạn truyền vào `index`, `Series` sẽ đánh index dựa vào thứ tự trong `index`, và chỉ chấp nhận các giá trị của dict có key nằm trong `index`. Với các giá trị trong biến `index` không có trong keys của dict, `Series` sẽ tạo ra các giá trị bị thiếu là `NaN`.

```
In [1]: pd.Series({"a": 0, "b": 1, "c": 2, "e": 4}, index=["b", "c", "d", "a"])
Out[1]:
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

Lưu ý: `NaN` là giá trị mặc định cho dữ liệu bị thiếu trong pandas và giá trị này có kiểu là `float64` nên kiểu dữ liệu của `Series` cũng là `float64` khác với `int64` ví dụ trước đó.

Khởi tạo Series bằng một giá trị (Scalar)

```
In [1]: pd.Series(data=1, index=["a", "b", "c"])
Out[1]:
a    1
b    1
c    1
dtype: int64
```

0.1.2 Một số thao tác cơ bản

Thao tác trên `Series` cũng giống với thao tác trên `numpy.array`. Ngoài ra chúng ta còn có thể tác với Series dựa vào index

Ví dụ:

```
In [1]: s = pd.Series(data=[0, 1, 2, 3, 4, 5], index=[”a”, ”b”, ”c”, ”d”, ”e”, ”f”])
```

Hi n thi ṭoàn b giá tri ̣c a Series Ta goị thu c tính `.values`

```
In [1]: s.values
Out[1]:
array([0, 1, 2, 3, 4, 5])
```

L y theo indice

```
In [2]: s[2]
Out[2]: 2
```

L y theo index

```
In [3]: s[”c”]
Out[3]: 2
```

Slice indice

```
In [4]: s[1:3]
Out[4]:
b    1
d    2
dtype: int64
```

Slice index

```
In [5]: s[”b”:”c”]
Out[5]:
b    1
c    2
dtype: int64
```

List indice

```
In [6]: s[[1, 2, 4]]
Out[6]:
b    1
c    2
e    4
dtype: int64
```

List index

```
In [7]: s[[”b”, ”c”, ”e”]]
Out[7]:
b    1
c    2
e    4
dtype: int64
```

Đi u ki n

```
In [5]: s[s > s.mean()]
Out[5]:
d    3
e    4
f    5
dtype: int64
```

0.2 DataFrame

`DataFrame` là c u trúc d li u chính và cũng là đ c tr ng c a pandas. Cũng gi ng nh SQL Table, `DataFrame` là m t b ng g m m t hay nhi u c t d li u. Ho c có th nói rõ h n là DataFrame là t p h p các Series lại v i nhau.

Cách kh i tạo DataFrame nh sau {#cach-khoi-tao-data-frame}

```
df = pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

Cũng gi ng nh Series, `data` c a DataFrame có nhi u cách kh i tạo khác nhau nh :

- `dict` c a Series, `dict` c a `numpy.array`/`List`
- M ng 2 chi u `numpy.ndarray`, `List` c a `List`
- M ng có c u trúc[1]
- T 1 `Series`
- T `DataFrame` khác

Tùy vào c u trúc c a `data` mà chúng ta có th b qua bi n `index`. Bi n `columns` th hi n tên c a các `Series`. `dtype` s định nghĩa các ki u d li u c a d li u, chúng ta s th o lu n v nó ph n k ti p c a ch ng này. `copy` dùng đ tạo b n sao t d li u `data`, nó ch nh h ng khi `data` là DataFrame khác ho c numpy.ndarray, vi c copy này s tránh tr ng h p 2 bi n cùng tr v cùng 1 b nh .

[1] `https://numpy.org/doc/stable/user/basics.rec.html`

0.2.1 Các cách kh i tạo

Kh i tạo DataFrame t dict c a Series

Khi không truy n bi n `index` vào, thì index c a `DataFrame` s là h p gi a 2 index c a `Series` và chúng s đ c s p x p theo th t t v ng. N u ta không truy n `columns` thì các c t c a `DataFrame` s đ c s p x p theo th t truy n vào các keys c a dict.

Khi truy n bi n `index` vào, t ng t nh Series, ch nh ng index n m trong `index` m i đ c chọn, còn nh ng index bi ̣thi u s đ c đi n giá tri ̣`NaN`

Khi truy n giá tri ̣`columns`, DataFrame s chọn nh ng `Series` thu c dict có key thu c `columns`, giá tri ̣trong `columns` không có trong key c a dict s đ c gán `NaN`

```
In [1]: d = {
            "one": pd.Series([1, 2, 3], index=["c", "b", "a"]),
            "two": pd.Series([1, 2, 3, 4], index=["c", "a", "b", "d"])
        }
In [2]: pd.DataFrame(d)
Out[2]:
   one  two
a  3.0    2
b  2.0    3
c  1.0    1
d  NaN    4

In [3]: pd.DataFrame(d, index=["d", "b", "a"])
Out[3]:
   one  two
d  NaN    4
b  2.0    3
a  3.0    2

In [4]: pd.DataFrame(d, index=["d", "b", "a"], columns=["two", "three"])
Out[4]:
   two  three
d    4    NaN
b    3    NaN
a    2    NaN
```

Kh i tạo DataFrame t dict c a numpy.ndarray/List

Đ i v i vi c kh i tạo này, b t bu c các m ng ph i có cùng đ dài. Khi không truy n `index` vào thì index c a DataFrame s đ c tạo t `0` đ n `len(n) - 1` trong đó `n` là đ dài c a m ng. Khi truy n giá tri ̣`columns`, DataFrame s chọn nh ng key thu c dict và cũng thu c `columns`, giá tri ̣trong `columns` không có trong key c a dict s đ c gán `NaN`

```
In [1]: d = {
            ”one”: [1, 2, 3, 4],
            ”two”: [1, 2, 3, 4],
            ”three”: [1, 2, 3, 4]
        }
In [2]: pd.DataFrame(data=d,
                     index=[”a”, ”b”, ”c”, ”d”],
                     columns=[”one”, ”two”, ”four”])
Out[2]:
   one  two four
a    1    1   NaN
b    2    2   NaN
c    3    3   NaN
d    4    4   NaN
```

Kh i tạo DataFrame t M ng 2 chi u/ 2-d numpy.ndarray

Khi không truy n `index` vào thì index c a `DataFrame` s đ c tạo t `0` đ n `len(n) - 1` trong đó `n` là s l ng List con ho c là s dòng hay `shape[0]` c a `numpy.ndarray`. Khi không truy n `columns` thì tên columns s đ c tạo t `0` đ n `len(n) - 1` v i `n` là đ dài l n nh t c a List con ho c `shape[1]` c a `numpy.ndarray`

```
In [1]: pd.DataFrame(data=[[1, 2], [3, 4, 5]],
                     index=[”a”, ”b”],
                     columns=['one','two','three'])
Out[1]:
   one  two  three
a    1    2    NaN
b    3    4    5.0

In [2]: pd.DataFrame(data=np.random.rand(2,3),
                     index=[”a”, ”b”],
                     columns=['one','two','three']))
Out[2]:
        one       two     three
a  0.662008  0.085735  0.331281
b  0.115360  0.358092  0.862477
```

Kh i tạo DataFrame t danh sách các dict

cách kh i tạo này, bạn hãy t ng t ng r ng m i dict là m t dòng c a DataFrame v i các key là tên c t và value là giá tri tại c t đó. Vi c truy n thêm ho c không truy n `index` cũng gi ng nh các tr ng h p kh i tạo trên.

L u ý: Trong tr ng h p này, n u bạn truy n `columns` vào thì `columns` b t bu c ph i ch a t t c các key c a dict

Trong ví du ̣d i đây, `columns` ph i ch a toàn b keys `["one", "two", "three"]`, n u thi u 1 trong 3 s phát sinh l i.

```
In [1]: d = [{"one": 1, "two": 2}, {"one": 4, "two": 5, "three": 6}]
In [2]: pd.DataFrame(d, index=["a", "b"], columns=["one", "two", "three", "four"])
Out[2]:
   one  two  three  four
a    1    2    NaN   NaN
b    4    5    6.0   NaN
```

Kh i tạo DataFrame t M ng có c u trúc

M ng có c u trúc là m ng mà các ph n t c a nó là m t c u trúc, bao g m các thành ph n nh h n, các thành ph n này đ c đ t tên và khai báo ki u d li u. D i đây là m t ví du ̣M ng có c u trúc trong numpy

```
In [1]: data = np.array([('pikachu', 9, 27.0), ('mewtwo', 3, 81.0)],
                        dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')])
In [2]: pd.DataFrame(data)
Out[2]:
       name  age  weight
0   pikachu    9    27.0
1    mewtwo    3    81.0
```

Kh i tạo DataFrame t namedtuple

Các tr ng trong `nametuple` s đ c gán thành tên các columns trong `DataFrame`. Nh ng giá tri ̣c a `namedtuple` s đ c xem là 1 dòng trong `DataFrame`. S l ng c t c a `DataFrame` s phu ̣thu c vào s l ng giá tri ̣c a ph n t `namedtuple` đ u tiên. N u các ph n t phía sau có s l ng giá tri ́t h n thì s đ c đi n `NaN` và ng c lại s tr ra l i n u s l ng giá tri ̣c a `namedtuple` l n h n s l ng giá tri ̣c a ph n t `namedtuple` đ u tiên.

Ví du ̣v cách tạo namedtuple

```
from collections import namedtuple
Point2D = namedtuple("Point2D", "x y")
Point3D = namedtuple("Point3D", "x y z")
```

Tạo DataFrame t namedtuple `Point2D`

```
In [1]: pd.DataFrame([Point2D(0, 0), Point2D(0, 1), Point2D(0, 2)])
Out[1]:
   x  y
0  0  0
1  0  1
2  0  2
```

Tạo DataFrame t namedtuple c `Point2D` và `Point3D`

```
In [1]: pd.DataFrame([Point3D(0, 0, 0), Point2D(0, 1), Point3D(0, 2, 3)])
Out[1]:
   x  y    z
0  0  0  0.0
1  0  1  NaN
2  0  2  3.0
```

Nh ta th y, tại ph n t th 2 ch có 2 giá tri, trong khi ph n t th nh t có 3 giá tri, v y nên ph n t bi thi u tại c t z s đ c gán `NaN`

Kh i tạo DataFrame t Series

```
In [1]: s = pd.Series(data=[0, 1, 2], index=["a", "b", "c"], name="meow")
In [2]: pd.DataFrame(s)
Out[2]:
   meow
a     0
b     1
c     2
```

`name` c a Series s là tên c t c a DataFrame và `index` c a Series s là index c a DataFrame n u ta không truy n các bi n `index`, `columns` khi kh i tạo `pd.DataFrame`

0.2.2 Các hàm kh i tạo thay th

DataFrame.from_dict

Cách kh i tạo

```
pd.DataFrame.from_dict(data, orient='columns', dtype=None, columns=None)
```

`data` truy n vào là 1 dict, `orient` có 2 giá tri có th đ a vào là `{"columns", "index"}`, `columns` là danh sách tên các c t c a DataFrame.

Lưu ý: Chỉ được truyền `columns` khi `orient="index"`. Khi `orient="columns"` sẽ báo lỗi.

Ví dụ tạo DataFrame khi `orient="columns"`. Với cách khởi tạo này tên các cột của DataFrame sẽ là key của dict

```
In [1]: data = {"col_1": [3, 2, 1, 0], "col_2": ["a", "b", "c", "d"]}
In [2]: pd.DataFrame.from_dict(data)
Out[2]:
   col_1 col_2
0      3     a
1      2     b
2      1     c
3      0     d
```

Ví dụ tạo DataFrame khi `orient="index"`. Với cách khởi tạo này index của DataFrame sẽ là key của dict.

```
In [1]: data = {"col_1": [3, 2, 1, 0], "col_2": ["a", "b", "c", "d"]}
In [2]: pd.DataFrame.from_dict(data, orient="index",
                               columns=["one", "two", "three", "four"])
Out[2]:
   col_1 col_2
0      3     a
1      2     b
2      1     c
3      0     d
```

DataFrame.from_records

Cách khởi tạo

```
pd.DataFrame.from_records(data)
```

`data` truyền vào có thể là một mảng hoặc có cấu trúc

```
In [1]: data = np.array([('Rex', 9, 81.0), ('Fido', 3, 27.0)],
                        dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')])
In [2]: pd.DataFrame.from_records(data, index=["a", "b"])
Out[2]:
   name  age  weight
a   Rex    9    81.0
b  Fido    3    27.0
```

Dữ liệu có thể là một danh sách các namedtuple

```
from collections import namedtuple
Point2D = namedtuple(”Point2D”, ”x y”)
Point3D = namedtuple(”Point3D”, ”x y z”)
pd.DataFrame.from_records([Point3D(0, 0, 0), Point2D(0, 1), Point3D(0, 2, 3)],
                          columns=[”x”,”y”,”z”], index=[”a”, ”b”, ”c”])
```

```
   x  y    z
a  0  0  0.0
b  0  1  NaN
c  0  2  3.0
```

Ho c 1 danh sách các dict

```
In [1]: d = [{”one”: 1, ”two”: 2}, {”one”: 4, ”two”: 5, ”three”: 6}]
In [2]: pd.DataFrame.from_records(d, index=[”a”, ”b”], columns=[”one”, ”two”, ”three”, ”four”])
Out[2]:
   one  two  three  four
a    1    2    NaN   NaN
b    4    5    6.0   NaN
```

0.3 Data type trong pandas

Đ ki m tra ki u d li u c a `Series` hay `DataFrame` bạn có th goị thu c tính `dtypes`. Các ki u d li u th ng g p c a Pandas đ c mô t theo b ng d i đây:

Các ki u d li u ph bi n	Numpy/Pandas object	Hi n thi.
Boolean	np.bool	*bool*
Integer	np.int, np.uint	*int uint*
Float	np.float	*float*
Object	np.object	*O, object*
Datetime	np.datetime64, pd.Timestamp	*datetime64*
Timedelta	np.timedelta64, pd.Timedelta	*timedelta64*
Category	pd.Categorical	*category*
Complex	np.complex	*complex*

Ví du.