morepanda

July 31, 2024

Pandas Tutorial

```
[ ]: import pandas as pd
     import numpy as np
```

```
[ ]: ser = pd.Series(np.random.rand(34))
```

```
[ ]: type(ser)
```

```
[ ]: pandas.core.series.Series
```

```
[ ]: df = pd.DataFrame(np.random.rand(334,5),index=np.arange(334))
```

```
[ ]: type(df)
```

```
[ ]: pandas.core.frame.DataFrame
```

```
[ ]: df.describe()
```

```
[ ]:                 0           1           2           3           4
     count  334.000000  334.000000  334.000000  334.000000  334.000000
     mean     0.491573    0.485999    0.525196    0.513215    0.503712
     std      0.289900    0.298295    0.290110    0.290163    0.287810
     min      0.000752    0.009000    0.001205    0.005760    0.001818
     25%      0.231860    0.225245    0.279434    0.263333    0.238387
     50%      0.505285    0.467467    0.522584    0.543708    0.497589
     75%      0.724706    0.736435    0.790835    0.747561    0.762810
     max      0.996598    0.998575    0.998722    0.998581    0.993607
```

```
[ ]: df.dtypes
```

```
[ ]: 0    float64
     1    float64
     2    float64
     3    float64
     4    float64
     dtype: object
```

```
[ ]: df[0][0] = 'harry'
```

```
/tmp/ipykernel_2882/2521509939.py:1: FutureWarning: ChainedAssignmentError:
behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in
certain cases, but when using Copy-on-Write (which will become the default
behaviour in pandas 3.0) this will never work to update the original DataFrame
or Series, because the intermediate object on which we are setting values will
behave as a copy.
A typical example is when you are setting values in a column of a DataFrame,
like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in
a single step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

  df[0][0] = 'harry'
/tmp/ipykernel_2882/2521509939.py:1: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value 'harry' has dtype incompatible with float64, please explicitly
cast to a compatible dtype first.
  df[0][0] = 'harry'
```

```
[ ]: df.dtypes
```

```
[ ]: 0     object
     1    float64
     2    float64
     3    float64
     4    float64
     dtype: object
```

```
[ ]: df
```

```
[ ]:             0         1         2         3         4
     0       harry  0.510084  0.150709  0.320018  0.248735
     1    0.809194  0.831657  0.228068  0.468826  0.235885
     2    0.547153  0.108468  0.391120  0.083873  0.763309
     3    0.834332  0.679575  0.885339  0.276047  0.857763
     4    0.564907  0.379267  0.673159  0.829365  0.363084
     ..        …         …         …         …         …
     329  0.951285  0.112800  0.198844  0.632691  0.793102
     330  0.302888  0.269980  0.844576  0.404042  0.501014
```

```
331  0.575431  0.377791  0.576765  0.390397  0.071100
332  0.642299  0.983762  0.055172  0.985133  0.644982
333  0.627437  0.592703  0.612667  0.685590  0.983377

[334 rows x 5 columns]
```

```
[ ]: df.index
```

```
[ ]: Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
              …
             324, 325, 326, 327, 328, 329, 330, 331, 332, 333],
            dtype='int64', length=334)
```

```
[ ]: df.columns
```

```
[ ]: RangeIndex(start=0, stop=5, step=1)
```

```
[ ]: df.to_numpy()
```

```
[ ]: array([['harry', 0.5100841366465932, 0.15070850244617706,
                0.32001766574369916, 0.24873456067656008],
               [0.8091938237675353, 0.8316574062428087, 0.2280676994652584,
                0.4688255590206196, 0.23588513498079444],
               [0.547152859116426, 0.10846791180495285, 0.39111966320969305,
                0.08387282957635211, 0.7633094374406972],
               …,
               [0.5754309685886226, 0.37779087024643043, 0.5767649970792196,
                0.3903966562813168, 0.07110013366652279],
               [0.6422985422720803, 0.9837624779795211, 0.05517172577054863,
                0.9851326468433558, 0.644982337644408],
               [0.6274365910549311, 0.5927028286138861, 0.6126665397535713,
                0.6855904123520477, 0.9833768542878983]], dtype=object)
```

```
[ ]: df[0][0] = 0.3
```

```
[ ]: df.head()
```

```
[ ]:          0         1         2         3         4
     0       0.3  0.510084  0.150709  0.320018  0.248735
     1  0.809194  0.831657  0.228068  0.468826  0.235885
     2  0.547153  0.108468  0.391120  0.083873  0.763309
     3  0.834332  0.679575  0.885339  0.276047  0.857763
     4  0.564907  0.379267  0.673159  0.829365  0.363084
```

```
[ ]: df.to_numpy()
[ ]: array([[0.3, 0.5100841366465932, 0.15070850244617706,
             0.32001766574369916, 0.24873456067656008],
            [0.8091938237675353, 0.8316574062428087, 0.2280676994652584,
             0.4688255590206196, 0.23588513498079444],
            [0.547152859116426, 0.10846791180495285, 0.39111966320969305,
             0.08387282957635211, 0.7633094374406972],
            …,
            [0.5754309685886226, 0.37779087024643043, 0.5767649970792196,
             0.3903966562813168, 0.07110013366652279],
            [0.6422985422720803, 0.9837624779795211, 0.05517172577054863,
             0.9851326468433558, 0.644982337644408],
            [0.6274365910549311, 0.5927028286138861, 0.6126665397535713,
             0.6855904123520477, 0.9833768542878983]], dtype=object)
```

```
[ ]: df.T
```

```
[ ]:         0         1         2         3         4         5         6   \
     0       0.3  0.809194  0.547153  0.834332  0.564907  0.076853  0.563853
     1  0.510084  0.831657  0.108468  0.679575  0.379267  0.502644  0.404504
     2  0.150709  0.228068   0.39112  0.885339  0.673159  0.495401  0.164453
     3  0.320018  0.468826  0.083873  0.276047  0.829365  0.911103  0.311066
     4  0.248735  0.235885  0.763309  0.857763  0.363084  0.265242  0.514102

             7         8         9   …       324       325       326       327  \
     0  0.099642  0.096209  0.597923  …  0.697949   0.64429  0.433084  0.747222
     1  0.249668  0.387712  0.704734  …  0.224643  0.029566  0.422667  0.462275
     2  0.876684  0.696324  0.791604  …  0.988446  0.425504  0.922757  0.306572
     3  0.777345  0.689398  0.554072  …  0.890257  0.575022  0.194906  0.970747
     4  0.180809  0.482911  0.599289  …  0.063264  0.776611   0.18693  0.477916

             328       329       330       331       332       333
     0  0.262762  0.951285  0.302888  0.575431  0.642299  0.627437
     1   0.72837    0.1128   0.26998  0.377791  0.983762  0.592703
     2  0.951156  0.198844  0.844576  0.576765  0.055172  0.612667
     3  0.655687  0.632691  0.404042  0.390397  0.985133   0.68559
     4  0.169175  0.793102  0.501014    0.0711  0.644982  0.983377

     [5 rows x 334 columns]
```

```
[ ]: df.sort_index(axis=1,ascending=False)
```

```
[ ]:           4         3         2         1         0
     0    0.248735  0.320018  0.150709  0.510084       0.3
     1    0.235885  0.468826  0.228068  0.831657  0.809194
     2    0.763309  0.083873  0.391120  0.108468  0.547153
     3    0.857763  0.276047  0.885339  0.679575  0.834332
     4    0.363084  0.829365  0.673159  0.379267  0.564907
```

```
..         …         …         …         …         …
329  0.793102  0.632691  0.198844  0.112800  0.951285
330  0.501014  0.404042  0.844576  0.269980  0.302888
331  0.071100  0.390397  0.576765  0.377791  0.575431
332  0.644982  0.985133  0.055172  0.983762  0.642299
333  0.983377  0.685590  0.612667  0.592703  0.627437

[334 rows x 5 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
          0         1         2         3         4
0       0.3  0.510084  0.150709  0.320018  0.248735
1  0.809194  0.831657  0.228068  0.468826  0.235885
2  0.547153  0.108468  0.391120  0.083873  0.763309
3  0.834332  0.679575  0.885339  0.276047  0.857763
4  0.564907  0.379267  0.673159  0.829365  0.363084
```

[ ]:
```
type(df[0])
```

[ ]:
```
pandas.core.series.Series
```

[ ]:
```
new = df
```

[ ]:
```
new[0][0] = 9783
```

```
/tmp/ipykernel_2882/1098291267.py:1: FutureWarning: ChainedAssignmentError:
behaviour will change in pandas 3.0!
You are setting values through chained assignment. Currently this works in
certain cases, but when using Copy-on-Write (which will become the default
behaviour in pandas 3.0) this will never work to update the original DataFrame
or Series, because the intermediate object on which we are setting values will
behave as a copy.
A typical example is when you are setting values in a column of a DataFrame,
like:

df["col"][row_indexer] = value

Use `df.loc[row_indexer, "col"] = values` instead, to perform the assignment in
a single step and ensure this keeps updating the original `df`.

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

  new[0][0] = 9783
/tmp/ipykernel_2882/1098291267.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  new[0][0] = 9783
```

[ ]:
```
new
```

[ ]:
```
            0         1         2         3         4
0        9783  0.510084  0.150709  0.320018  0.248735
1    0.809194  0.831657  0.228068  0.468826  0.235885
2    0.547153  0.108468  0.391120  0.083873  0.763309
3    0.834332  0.679575  0.885339  0.276047  0.857763
4    0.564907  0.379267  0.673159  0.829365  0.363084
..        …         …         …         …         …
329  0.951285  0.112800  0.198844  0.632691  0.793102
330  0.302888  0.269980  0.844576  0.404042  0.501014
331  0.575431  0.377791  0.576765  0.390397  0.071100
332  0.642299  0.983762  0.055172  0.985133  0.644982
333  0.627437  0.592703  0.612667  0.685590  0.983377

[334 rows x 5 columns]
```

[ ]:
```
df
```

[ ]:
```
            0         1         2         3         4
0        9783  0.510084  0.150709  0.320018  0.248735
1    0.809194  0.831657  0.228068  0.468826  0.235885
2    0.547153  0.108468  0.391120  0.083873  0.763309
3    0.834332  0.679575  0.885339  0.276047  0.857763
4    0.564907  0.379267  0.673159  0.829365  0.363084
..        …         …         …         …         …
329  0.951285  0.112800  0.198844  0.632691  0.793102
330  0.302888  0.269980  0.844576  0.404042  0.501014
331  0.575431  0.377791  0.576765  0.390397  0.071100
332  0.642299  0.983762  0.055172  0.985133  0.644982
333  0.627437  0.592703  0.612667  0.685590  0.983377

[334 rows x 5 columns]
```

[ ]:
```
df.loc[0,0] = 654
```

[ ]:
```
df
```

[ ]:
```
            0         1         2         3         4
0         654  0.510084  0.150709  0.320018  0.248735
1    0.809194  0.831657  0.228068  0.468826  0.235885
2    0.547153  0.108468  0.391120  0.083873  0.763309
```

```
3    0.834332  0.679575  0.885339  0.276047  0.857763
4    0.564907  0.379267  0.673159  0.829365  0.363084
..        …         …         …         …         …
329  0.951285  0.112800  0.198844  0.632691  0.793102
330  0.302888  0.269980  0.844576  0.404042  0.501014
331  0.575431  0.377791  0.576765  0.390397  0.071100
332  0.642299  0.983762  0.055172  0.985133  0.644982
333  0.627437  0.592703  0.612667  0.685590  0.983377

[334 rows x 5 columns]
```

```
[ ]: df.drop(4,axis=1)
```

```
[ ]:         0         1         2         3
0         654  0.510084  0.150709  0.320018
1    0.809194  0.831657  0.228068  0.468826
2    0.547153  0.108468  0.391120  0.083873
3    0.834332  0.679575  0.885339  0.276047
4    0.564907  0.379267  0.673159  0.829365
..        …         …         …         …
329  0.951285  0.112800  0.198844  0.632691
330  0.302888  0.269980  0.844576  0.404042
331  0.575431  0.377791  0.576765  0.390397
332  0.642299  0.983762  0.055172  0.985133
333  0.627437  0.592703  0.612667  0.685590

[334 rows x 4 columns]
```

```
[ ]: df
```

```
[ ]:         0         1         2         3         4
0         654  0.510084  0.150709  0.320018  0.248735
1    0.809194  0.831657  0.228068  0.468826  0.235885
2    0.547153  0.108468  0.391120  0.083873  0.763309
3    0.834332  0.679575  0.885339  0.276047  0.857763
4    0.564907  0.379267  0.673159  0.829365  0.363084
..        …         …         …         …         …
329  0.951285  0.112800  0.198844  0.632691  0.793102
330  0.302888  0.269980  0.844576  0.404042  0.501014
331  0.575431  0.377791  0.576765  0.390397  0.071100
332  0.642299  0.983762  0.055172  0.985133  0.644982
333  0.627437  0.592703  0.612667  0.685590  0.983377

[334 rows x 5 columns]
```

```
[ ]:
```