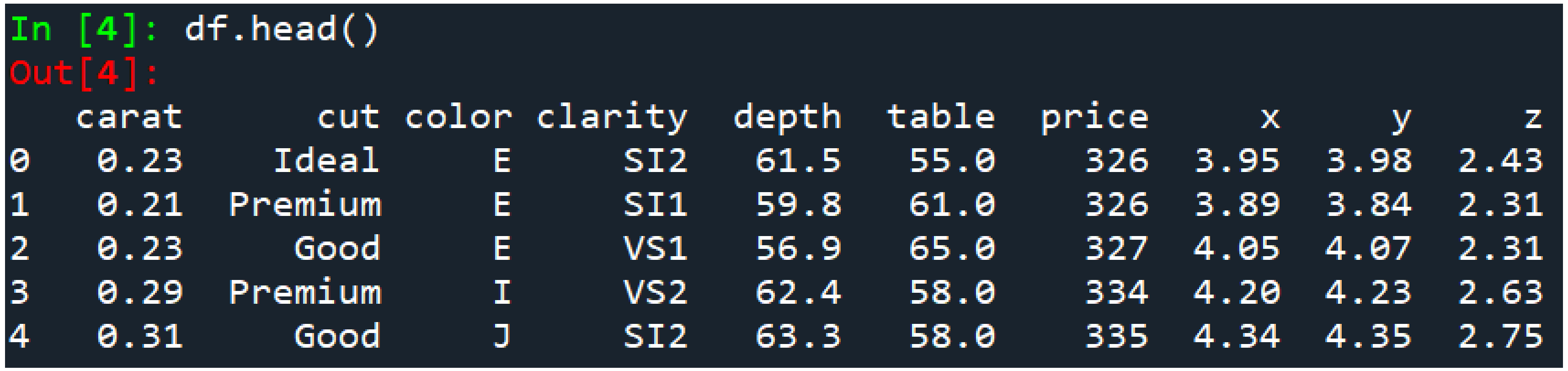
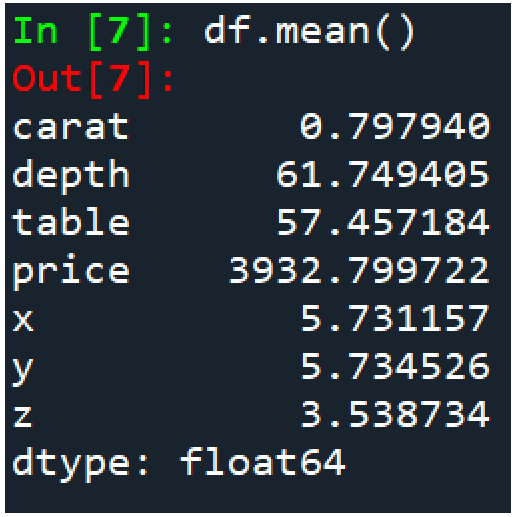
Lecture 12: Groupby, missing values, and strings

More Panda bears.

Grouping basics

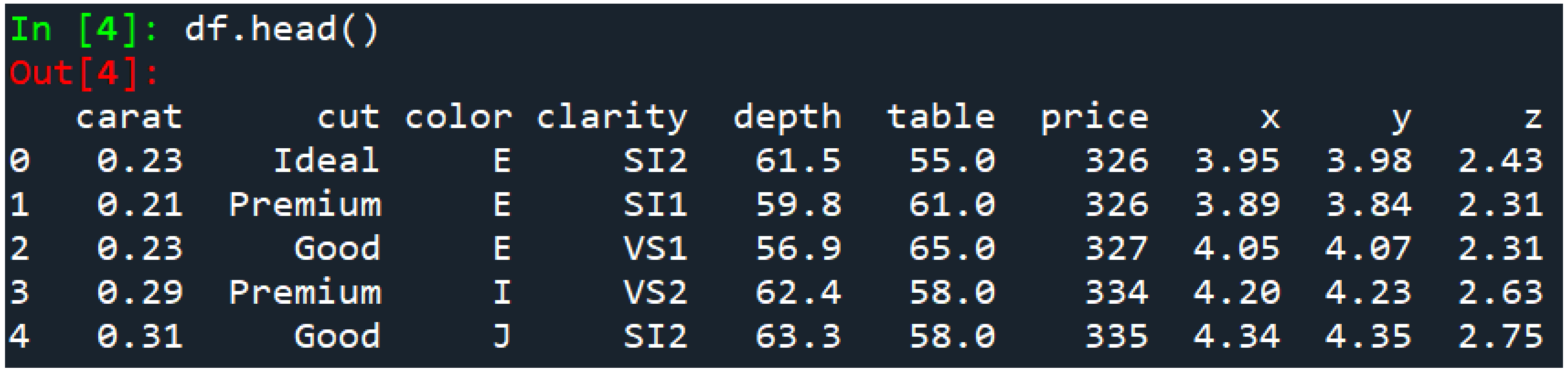


```
In [4]: df.head()
Out[4]:
   carat      cut color clarity  depth  table  price     x     y     z
0   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```

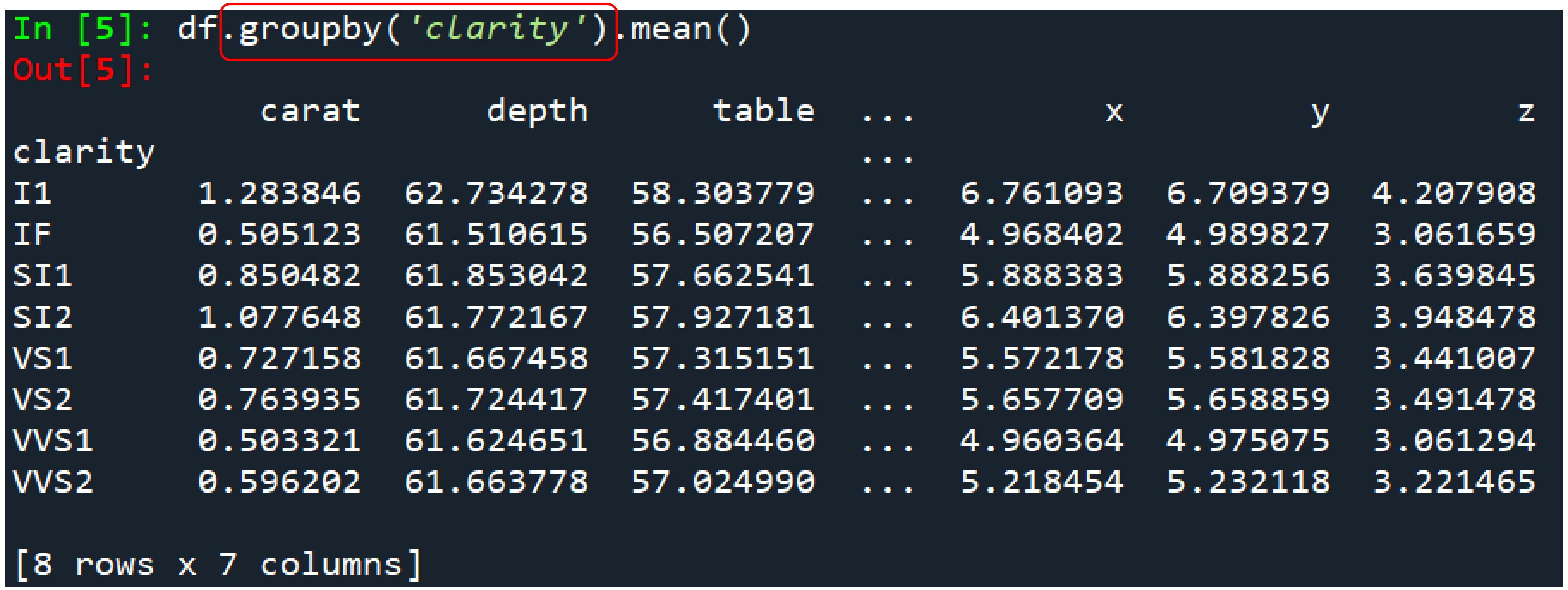


```
In [7]: df.mean()
Out[7]:
carat       0.797940
depth      61.749405
table      57.457184
price    3932.799722
x           5.731157
y           5.734526
z           3.538734
dtype: float64
```

Grouping basics

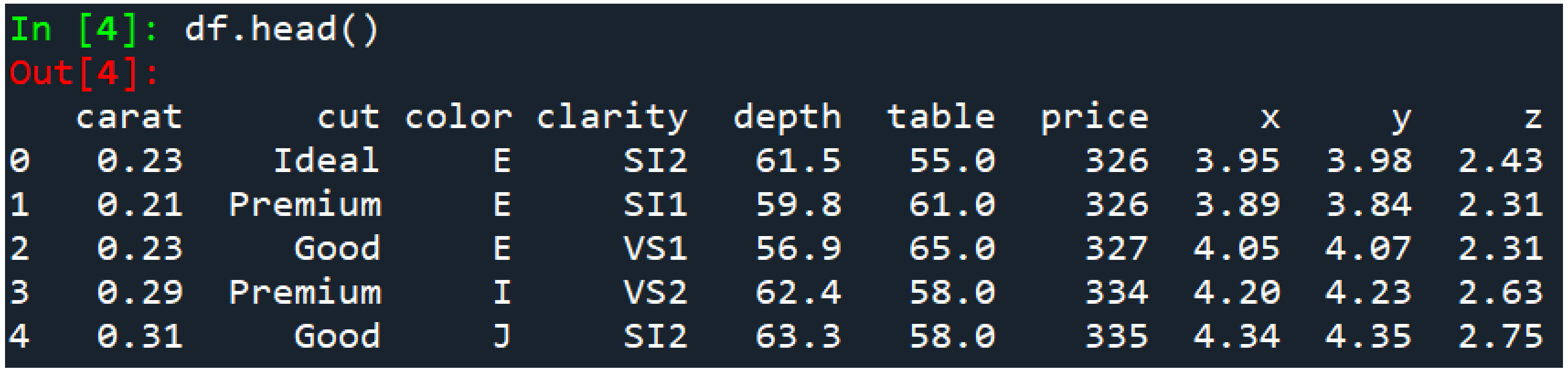


```
In [4]: df.head()
Out[4]:
   carat      cut color clarity  depth  table  price     x     y     z
0   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```

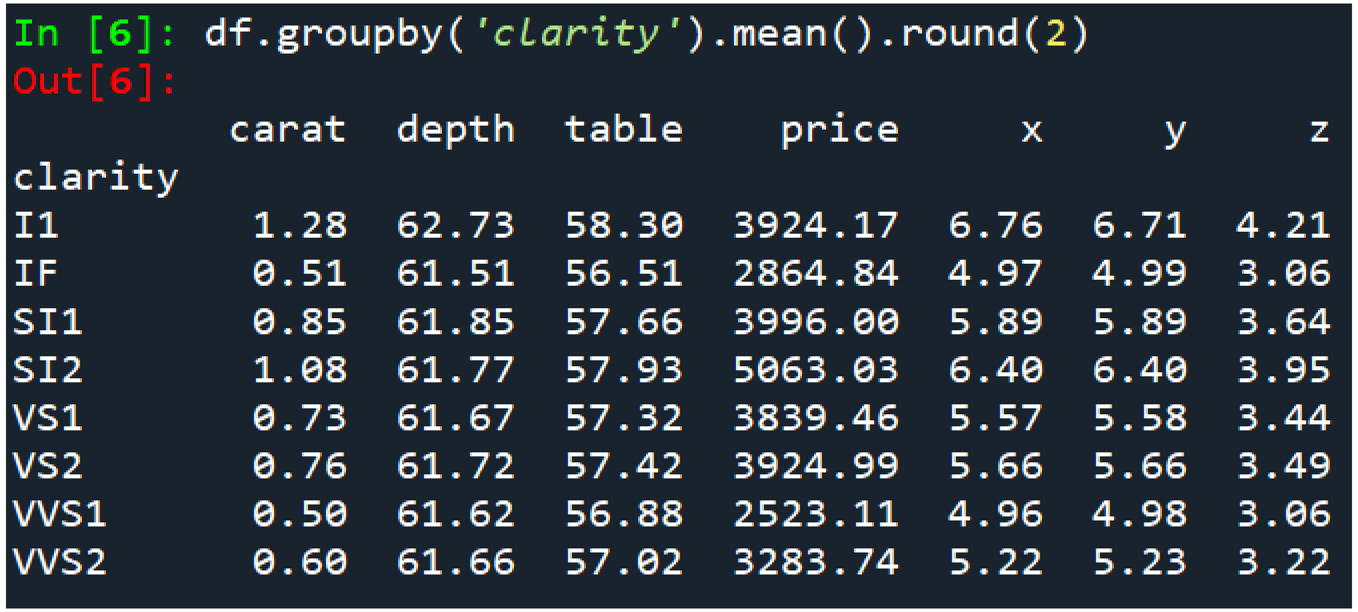


```
In [5]: df.groupby('clarity').mean()
Out[5]:
             carat      depth      table  ...         x         y         z
clarity                                   ...
I1        1.283846  62.734278  58.303779  ...  6.761093  6.709379  4.207908
IF        0.505123  61.510615  56.507207  ...  4.968402  4.989827  3.061659
SI1       0.850482  61.853042  57.662541  ...  5.888383  5.888256  3.639845
SI2       1.077648  61.772167  57.927181  ...  6.401370  6.397826  3.948478
VS1       0.727158  61.667458  57.315151  ...  5.572178  5.581828  3.441007
VS2       0.763935  61.724417  57.417401  ...  5.657709  5.658859  3.491478
VVS1      0.503321  61.624651  56.884460  ...  4.960364  4.975075  3.061294
VVS2      0.596202  61.663778  57.024990  ...  5.218454  5.232118  3.221465

[8 rows x 7 columns]
```

Grouping basics

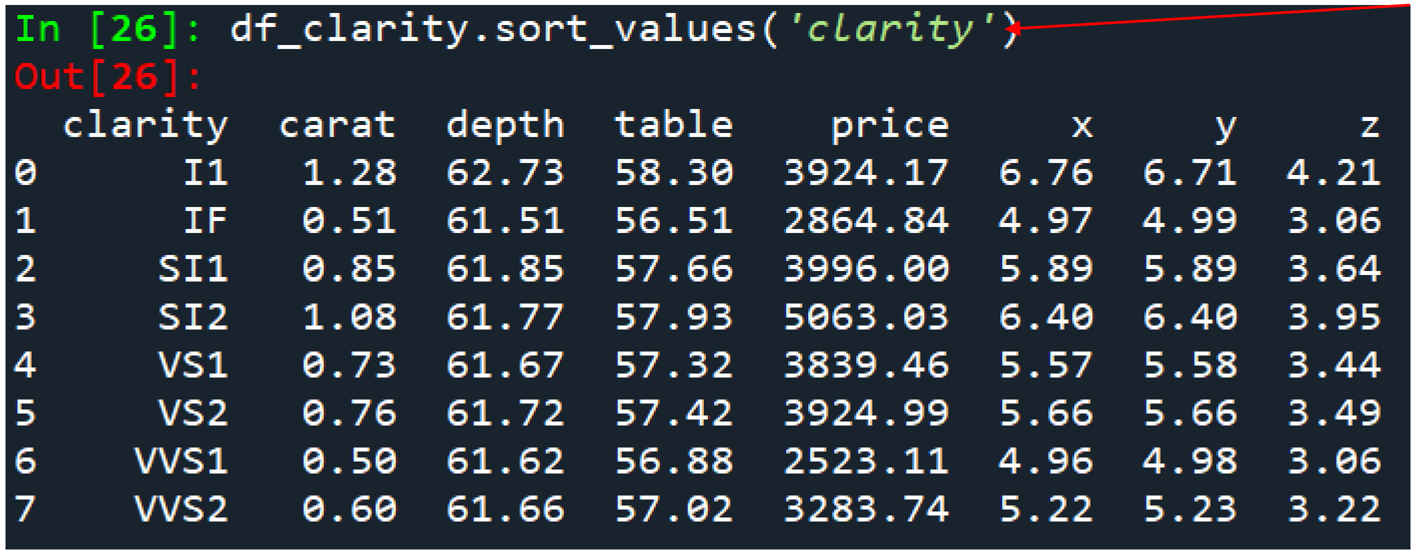


```
In [4]: df.head()
Out[4]:
   carat      cut color clarity  depth  table  price     x     y     z
0   0.23    Ideal     E     SI2   61.5   55.0    326  3.95  3.98  2.43
1   0.21  Premium     E     SI1   59.8   61.0    326  3.89  3.84  2.31
2   0.23     Good     E     VS1   56.9   65.0    327  4.05  4.07  2.31
3   0.29  Premium     I     VS2   62.4   58.0    334  4.20  4.23  2.63
4   0.31     Good     J     SI2   63.3   58.0    335  4.34  4.35  2.75
```



```
In [6]: df.groupby('clarity').mean().round(2)
Out[6]:
         carat  depth  table    price     x     y     z
clarity
I1        1.28  62.73  58.30  3924.17  6.76  6.71  4.21
IF        0.51  61.51  56.51  2864.84  4.97  4.99  3.06
SI1       0.85  61.85  57.66  3996.00  5.89  5.89  3.64
SI2       1.08  61.77  57.93  5063.03  6.40  6.40  3.95
VS1       0.73  61.67  57.32  3839.46  5.57  5.58  3.44
VS2       0.76  61.72  57.42  3924.99  5.66  5.66  3.49
VVS1      0.50  61.62  56.88  2523.11  4.96  4.98  3.06
VVS2      0.60  61.66  57.02  3283.74  5.22  5.23  3.22
```

Aside: sorting by values

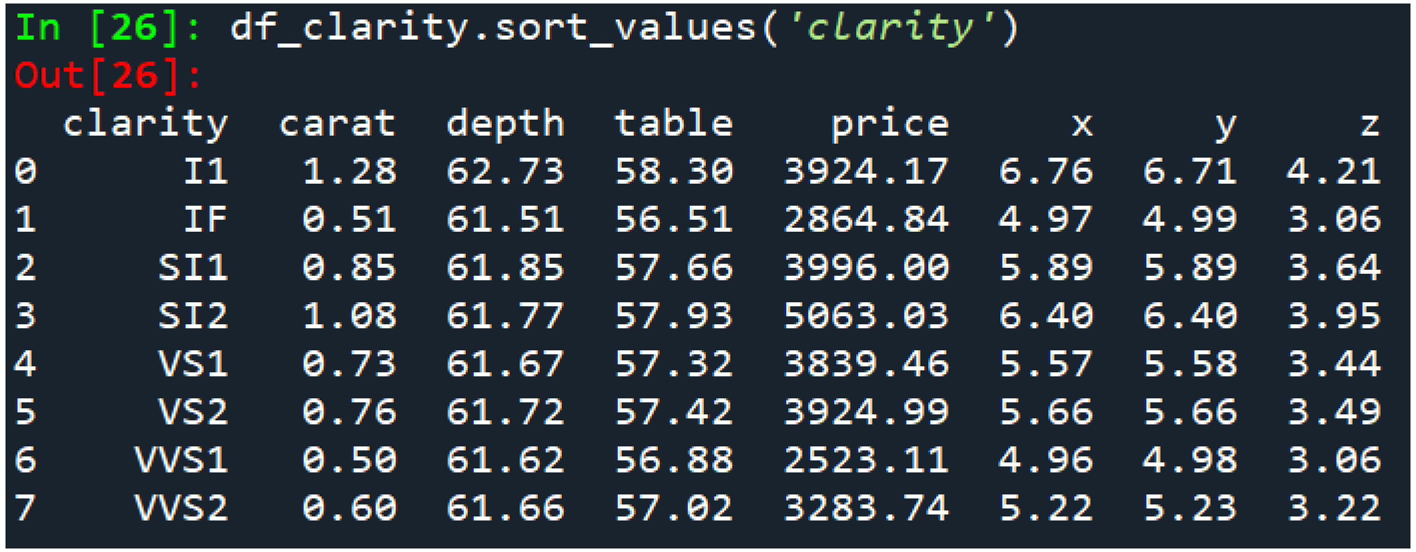


```
In [26]: df_clarity.sort_values('clarity')
Out[26]:
  clarity  carat  depth  table    price     x     y     z
0      I1   1.28  62.73  58.30  3924.17  6.76  6.71  4.21
1      IF   0.51  61.51  56.51  2864.84  4.97  4.99  3.06
2     SI1   0.85  61.85  57.66  3996.00  5.89  5.89  3.64
3     SI2   1.08  61.77  57.93  5063.03  6.40  6.40  3.95
4     VS1   0.73  61.67  57.32  3839.46  5.57  5.58  3.44
5     VS2   0.76  61.72  57.42  3924.99  5.66  5.66  3.49
6    VVS1   0.50  61.62  56.88  2523.11  4.96  4.98  3.06
7    VVS2   0.60  61.66  57.02  3283.74  5.22  5.23  3.22
```

ascending=True
key word argument

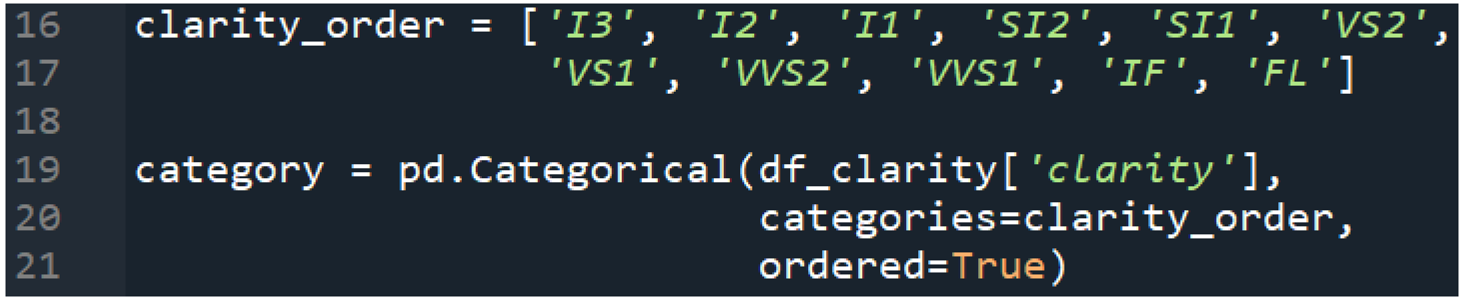
Aside: custom sorting order

But alphabetical order is not the correct order for diamond clarity!



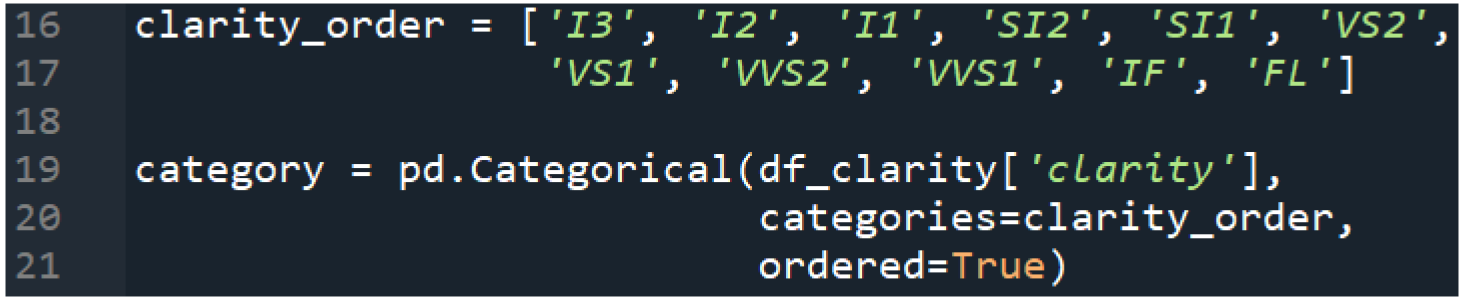
```
In [26]: df_clarity.sort_values('clarity')
Out[26]:
  clarity  carat  depth  table    price     x     y     z
0      I1   1.28  62.73  58.30  3924.17  6.76  6.71  4.21
1      IF   0.51  61.51  56.51  2864.84  4.97  4.99  3.06
2     SI1   0.85  61.85  57.66  3996.00  5.89  5.89  3.64
3     SI2   1.08  61.77  57.93  5063.03  6.40  6.40  3.95
4     VS1   0.73  61.67  57.32  3839.46  5.57  5.58  3.44
5     VS2   0.76  61.72  57.42  3924.99  5.66  5.66  3.49
6    VVS1   0.50  61.62  56.88  2523.11  4.96  4.98  3.06
7    VVS2   0.60  61.66  57.02  3283.74  5.22  5.23  3.22
```

Aside: custom sorting order

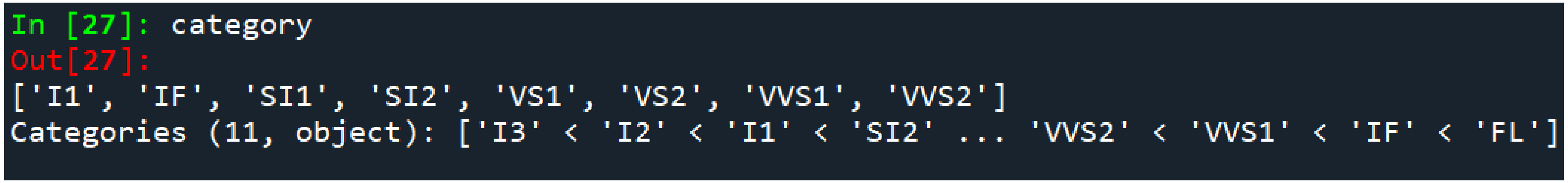


```
clarity_order = ['I3', 'I2', 'I1', 'SI2', 'SI1', 'VS2',
                 'VS1', 'VVS2', 'VVS1', 'IF', 'FL']

category = pd.Categorical(df_clarity['clarity'],
                          categories=clarity_order,
                          ordered=True)
```

Aside: custom sorting order

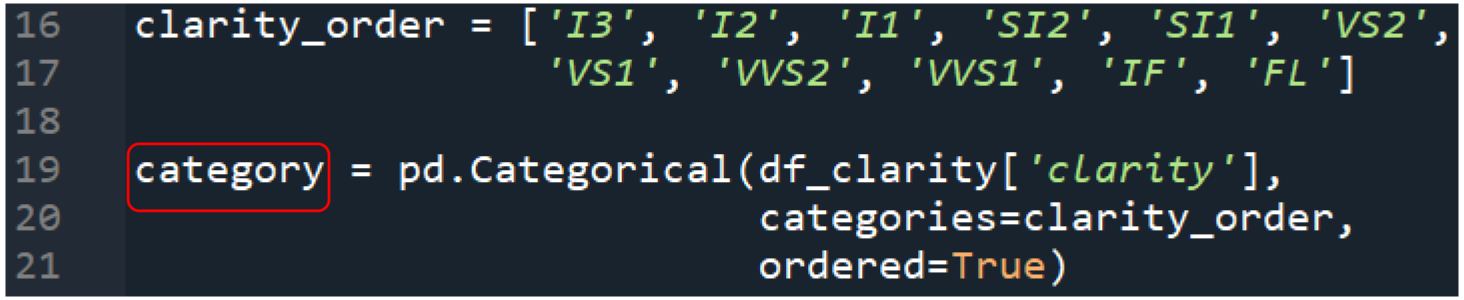


```python
clarity_order = ['I3', 'I2', 'I1', 'SI2', 'SI1', 'VS2',
                 'VS1', 'VVS2', 'VVS1', 'IF', 'FL']

category = pd.Categorical(df_clarity['clarity'],
                          categories=clarity_order,
                          ordered=True)
```

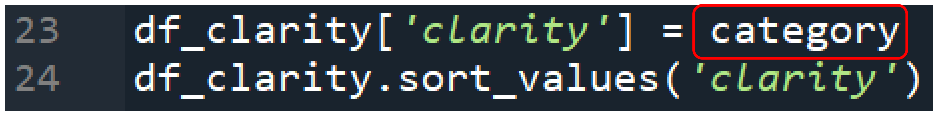


```
In [27]: category
Out[27]:
['I1', 'IF', 'SI1', 'SI2', 'VS1', 'VS2', 'VVS1', 'VVS2']
Categories (11, object): ['I3' < 'I2' < 'I1' < 'SI2' ... 'VVS2' < 'VVS1' < 'IF' < 'FL']
```

Aside: custom sorting order

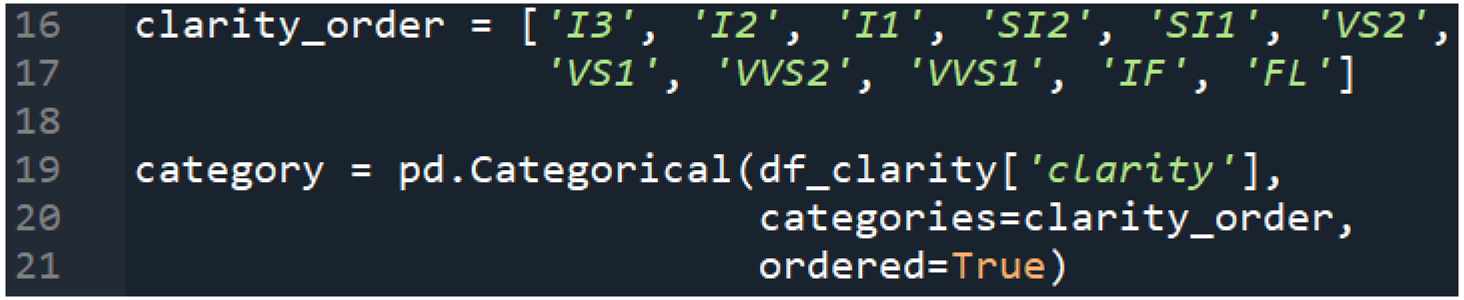


```
clarity_order = ['I3', 'I2', 'I1', 'SI2', 'SI1', 'VS2',
                 'VS1', 'VVS2', 'VVS1', 'IF', 'FL']

category = pd.Categorical(df_clarity['clarity'],
                          categories=clarity_order,
                          ordered=True)
```

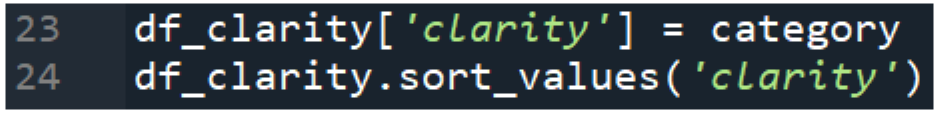


```
df_clarity['clarity'] = category
df_clarity.sort_values('clarity')
```

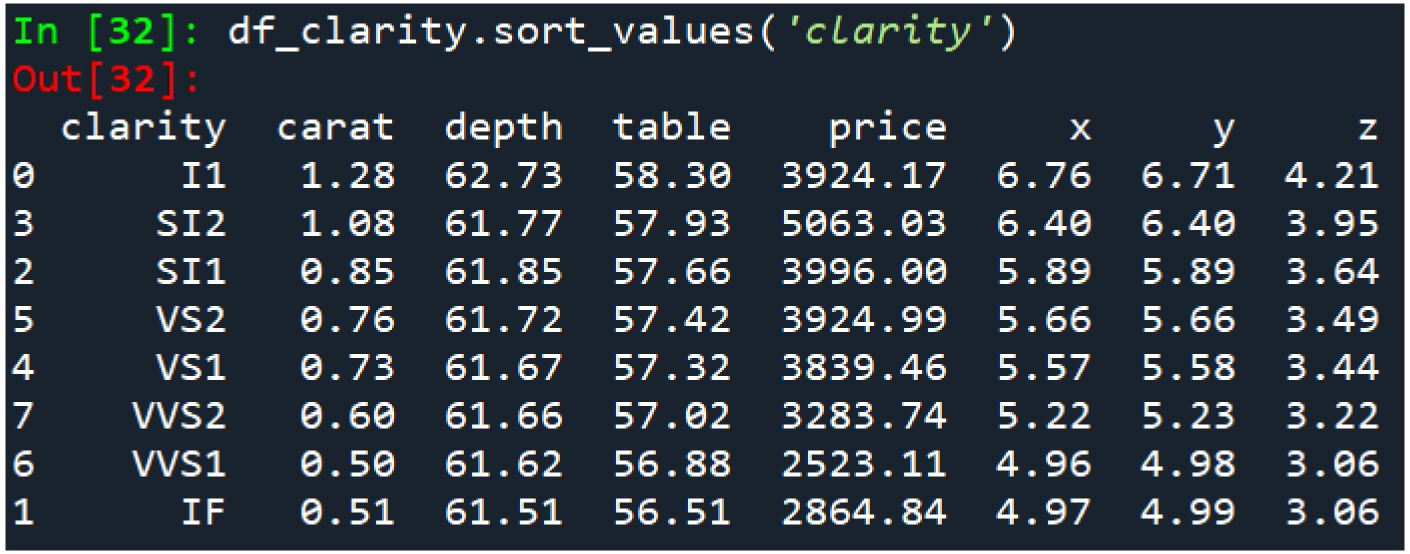
Aside: custom sorting order



```python
clarity_order = ['I3', 'I2', 'I1', 'SI2', 'SI1', 'VS2',
                 'VS1', 'VVS2', 'VVS1', 'IF', 'FL']

category = pd.Categorical(df_clarity['clarity'],
                          categories=clarity_order,
                          ordered=True)
```

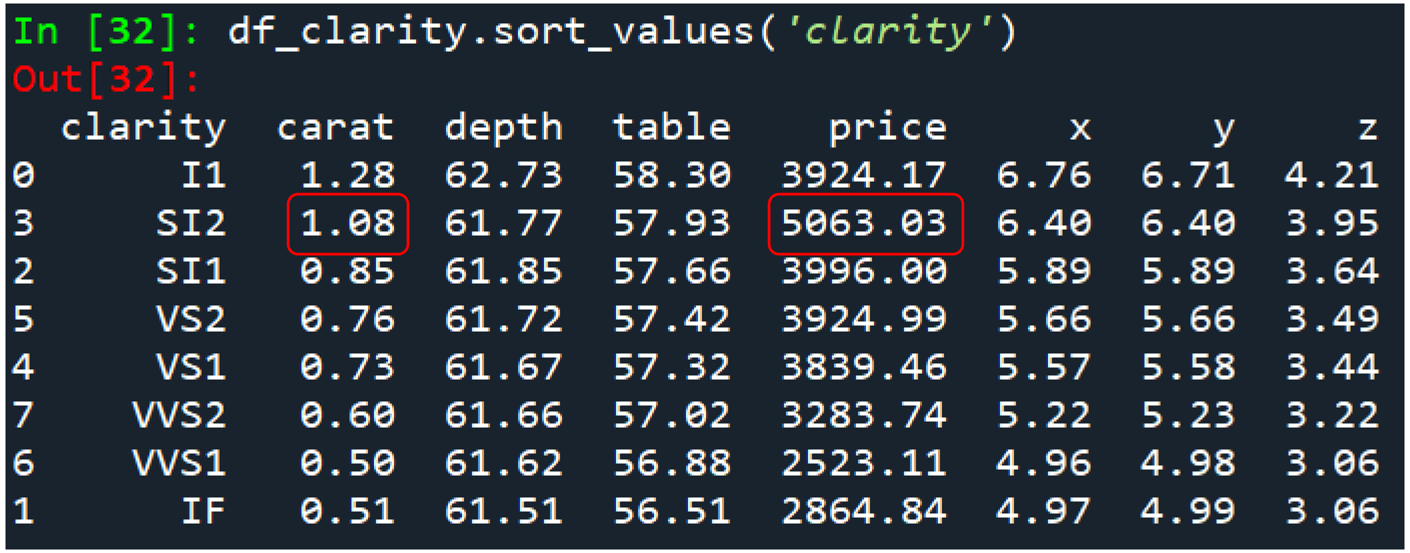


```python
df_clarity['clarity'] = category
df_clarity.sort_values('clarity')
```



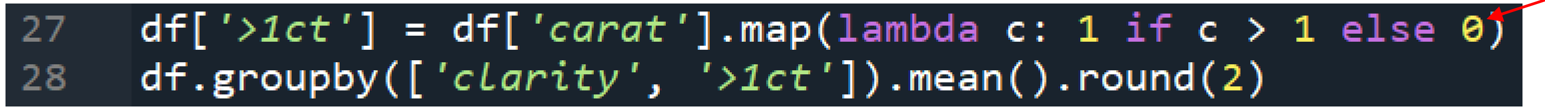
```
In [32]: df_clarity.sort_values('clarity')
Out[32]:
  clarity  carat  depth  table    price     x     y     z
0      I1   1.28  62.73  58.30  3924.17  6.76  6.71  4.21
3     SI2   1.08  61.77  57.93  5063.03  6.40  6.40  3.95
2     SI1   0.85  61.85  57.66  3996.00  5.89  5.89  3.64
5     VS2   0.76  61.72  57.42  3924.99  5.66  5.66  3.49
4     VS1   0.73  61.67  57.32  3839.46  5.57  5.58  3.44
7    VVS2   0.60  61.66  57.02  3283.74  5.22  5.23  3.22
6    VVS1   0.50  61.62  56.88  2523.11  4.96  4.98  3.06
1      IF   0.51  61.51  56.51  2864.84  4.97  4.99  3.06
```

Groupby multiple categories



```
In [32]: df_clarity.sort_values('clarity')
Out[32]:
  clarity  carat  depth  table    price     x     y     z
0      I1   1.28  62.73  58.30  3924.17  6.76  6.71  4.21
3     SI2   1.08  61.77  57.93  5063.03  6.40  6.40  3.95
2     SI1   0.85  61.85  57.66  3996.00  5.89  5.89  3.64
5     VS2   0.76  61.72  57.42  3924.99  5.66  5.66  3.49
4     VS1   0.73  61.67  57.32  3839.46  5.57  5.58  3.44
7    VVS2   0.60  61.66  57.02  3283.74  5.22  5.23  3.22
6    VVS1   0.50  61.62  56.88  2523.11  4.96  4.98  3.06
1      IF   0.51  61.51  56.51  2864.84  4.97  4.99  3.06
```

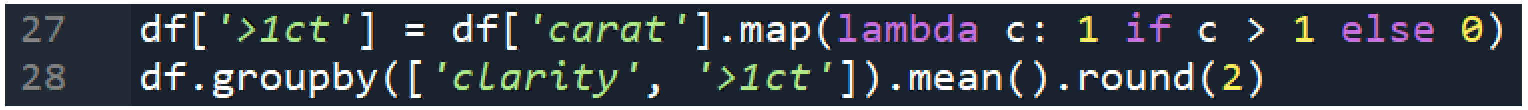
Groupby multiple categories



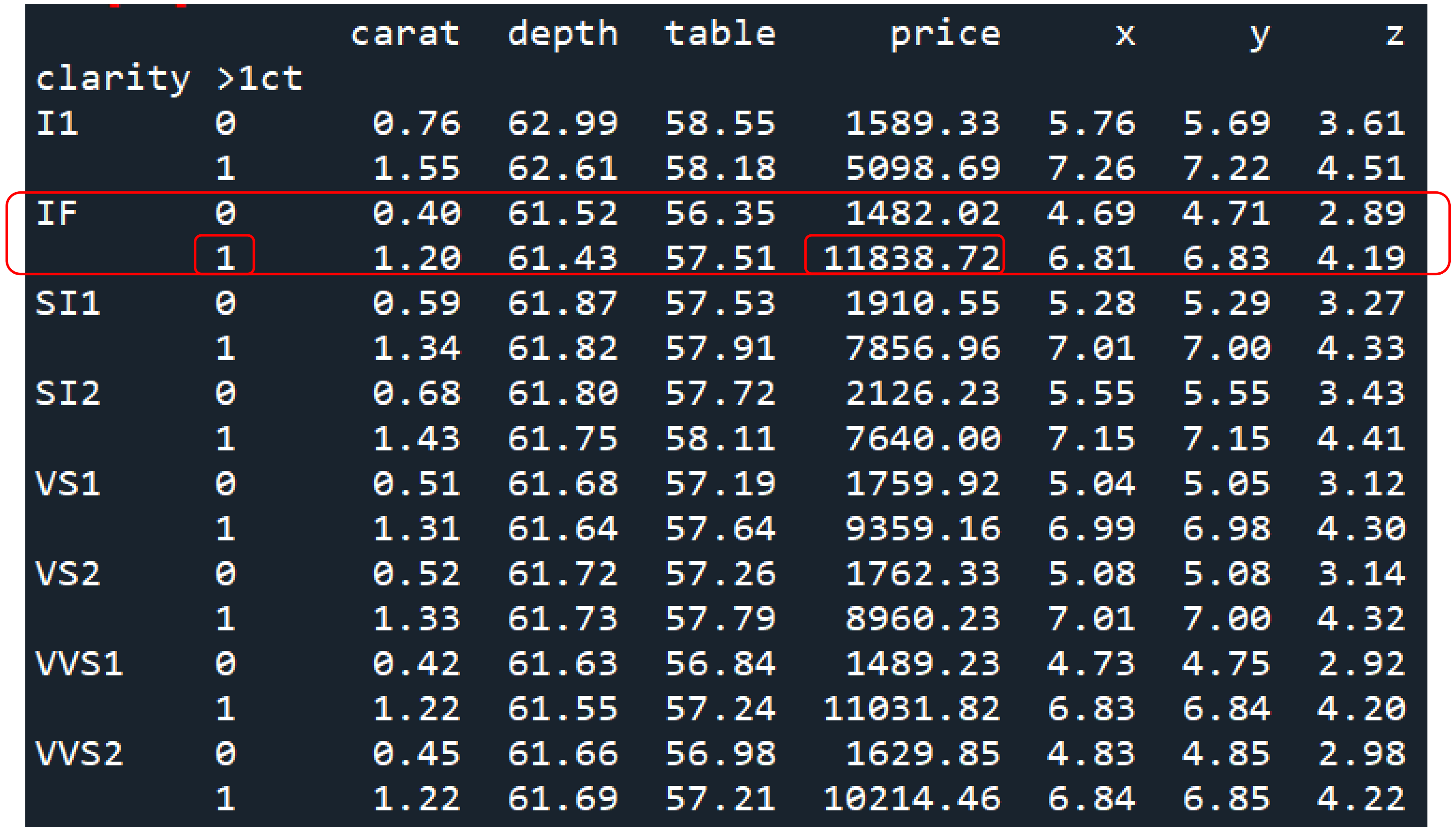
```
df['>1ct'] = df['carat'].map(lambda c: 1 if c > 1 else 0)
df.groupby(['clarity', '>1ct']).mean().round(2)
```

Special form for lambda functions

Groupby multiple categories

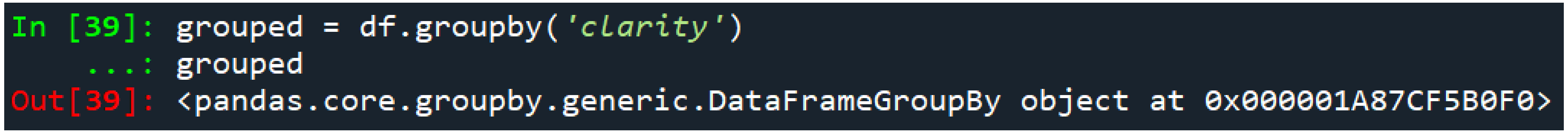


```python
df['>1ct'] = df['carat'].map(lambda c: 1 if c > 1 else 0)
df.groupby(['clarity', '>1ct']).mean().round(2)
```



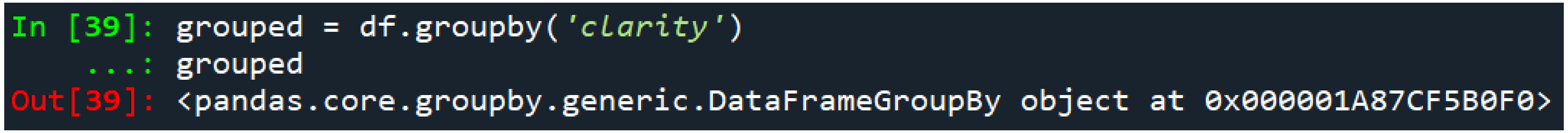
```
                carat  depth  table     price     x     y     z
clarity >1ct
I1      0        0.76  62.99  58.55   1589.33  5.76  5.69  3.61
        1        1.55  62.61  58.18   5098.69  7.26  7.22  4.51
IF      0        0.40  61.52  56.35   1482.02  4.69  4.71  2.89
        1        1.20  61.43  57.51  11838.72  6.81  6.83  4.19
SI1     0        0.59  61.87  57.53   1910.55  5.28  5.29  3.27
        1        1.34  61.82  57.91   7856.96  7.01  7.00  4.33
SI2     0        0.68  61.80  57.72   2126.23  5.55  5.55  3.43
        1        1.43  61.75  58.11   7640.00  7.15  7.15  4.41
VS1     0        0.51  61.68  57.19   1759.92  5.04  5.05  3.12
        1        1.31  61.64  57.64   9359.16  6.99  6.98  4.30
VS2     0        0.52  61.72  57.26   1762.33  5.08  5.08  3.14
        1        1.33  61.73  57.79   8960.23  7.01  7.00  4.32
VVS1    0        0.42  61.63  56.84   1489.23  4.73  4.75  2.92
        1        1.22  61.55  57.24  11031.82  6.83  6.84  4.20
VVS2    0        0.45  61.66  56.98   1629.85  4.83  4.85  2.98
        1        1.22  61.69  57.21  10214.46  6.84  6.85  4.22
```

The GroupBy object

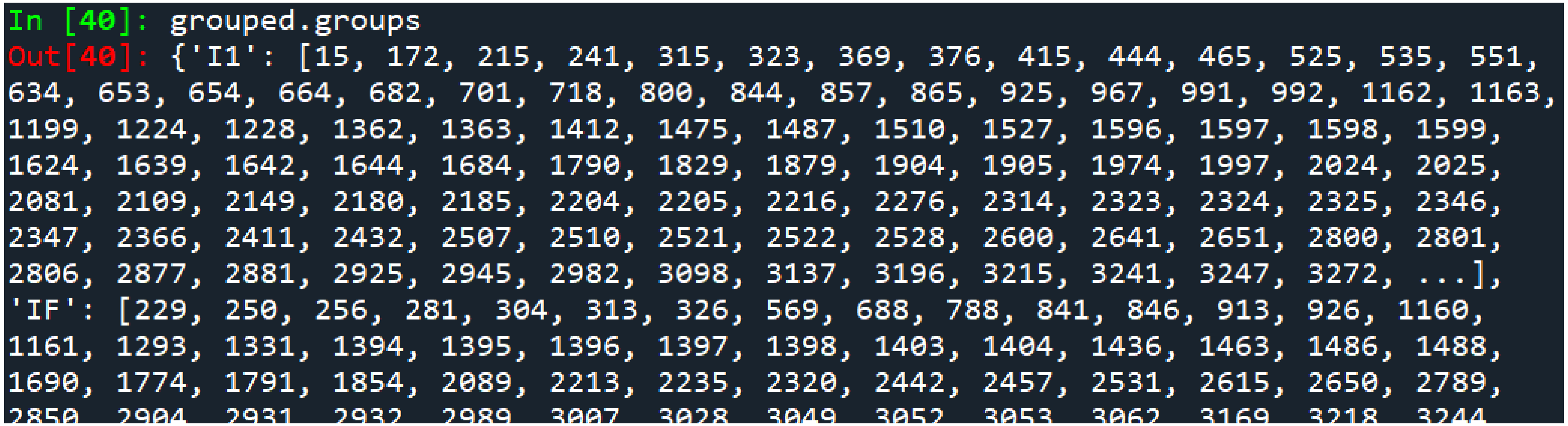


```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```

The GroupBy object

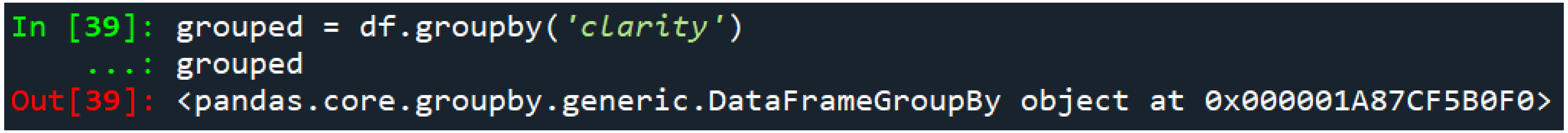


```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```

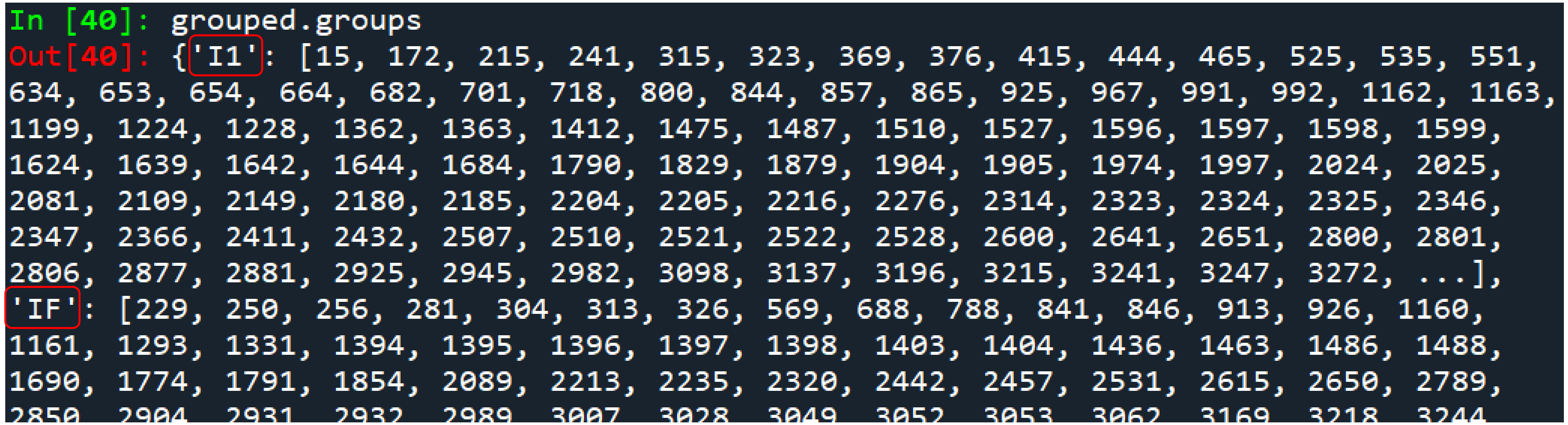


```
In [40]: grouped.groups
Out[40]: {'I1': [15, 172, 215, 241, 315, 323, 369, 376, 415, 444, 465, 525, 535, 551,
634, 653, 654, 664, 682, 701, 718, 800, 844, 857, 865, 925, 967, 991, 992, 1162, 1163,
1199, 1224, 1228, 1362, 1363, 1412, 1475, 1487, 1510, 1527, 1596, 1597, 1598, 1599,
1624, 1639, 1642, 1644, 1684, 1790, 1829, 1879, 1904, 1905, 1974, 1997, 2024, 2025,
2081, 2109, 2149, 2180, 2185, 2204, 2205, 2216, 2276, 2314, 2323, 2324, 2325, 2346,
2347, 2366, 2411, 2432, 2507, 2510, 2521, 2522, 2528, 2600, 2641, 2651, 2800, 2801,
2806, 2877, 2881, 2925, 2945, 2982, 3098, 3137, 3196, 3215, 3241, 3247, 3272, ...],
'IF': [229, 250, 256, 281, 304, 313, 326, 569, 688, 788, 841, 846, 913, 926, 1160,
1161, 1293, 1331, 1394, 1395, 1396, 1397, 1398, 1403, 1404, 1436, 1463, 1486, 1488,
1690, 1774, 1791, 1854, 2089, 2213, 2235, 2320, 2442, 2457, 2531, 2615, 2650, 2789,
2850  2904  2931  2932  2989  3007  3028  3049  3052  3053  3062  3169  3218  3244
```

The GroupBy object

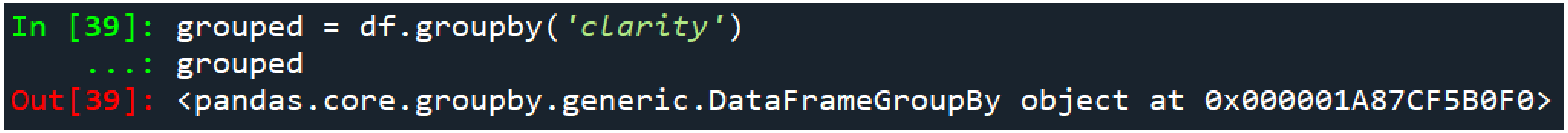


```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```

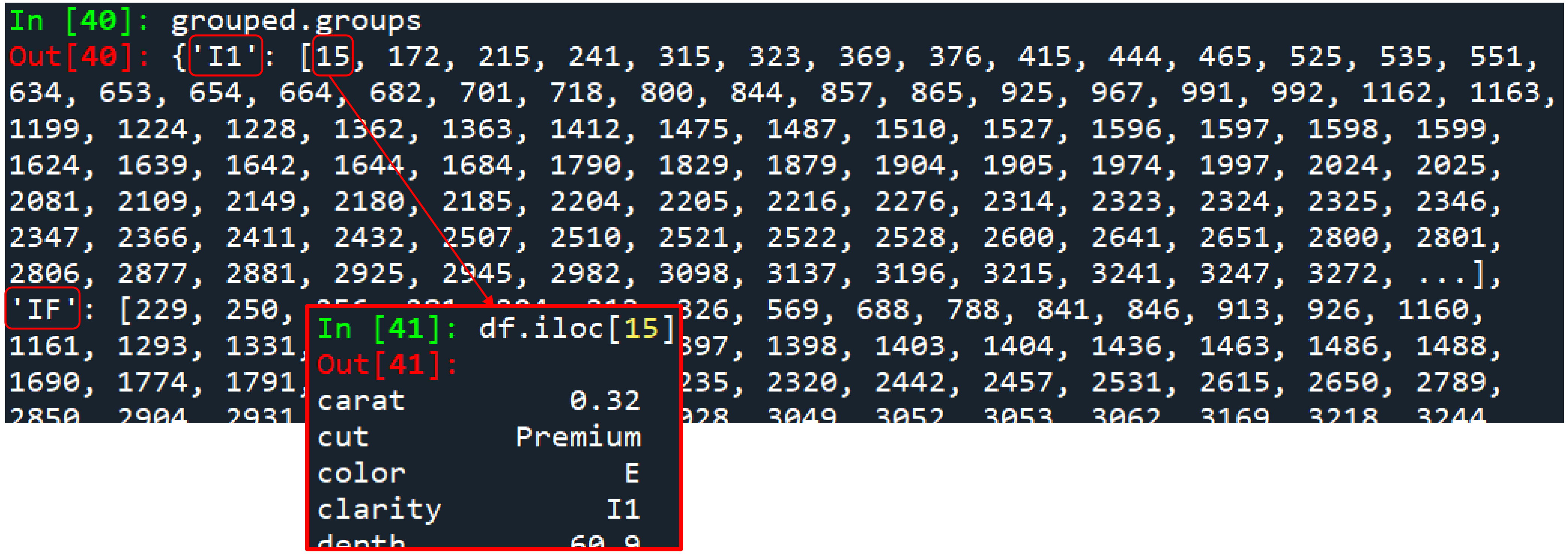


```
In [40]: grouped.groups
Out[40]: {'I1': [15, 172, 215, 241, 315, 323, 369, 376, 415, 444, 465, 525, 535, 551,
634, 653, 654, 664, 682, 701, 718, 800, 844, 857, 865, 925, 967, 991, 992, 1162, 1163,
1199, 1224, 1228, 1362, 1363, 1412, 1475, 1487, 1510, 1527, 1596, 1597, 1598, 1599,
1624, 1639, 1642, 1644, 1684, 1790, 1829, 1879, 1904, 1905, 1974, 1997, 2024, 2025,
2081, 2109, 2149, 2180, 2185, 2204, 2205, 2216, 2276, 2314, 2323, 2324, 2325, 2346,
2347, 2366, 2411, 2432, 2507, 2510, 2521, 2522, 2528, 2600, 2641, 2651, 2800, 2801,
2806, 2877, 2881, 2925, 2945, 2982, 3098, 3137, 3196, 3215, 3241, 3247, 3272, ...],
'IF': [229, 250, 256, 281, 304, 313, 326, 569, 688, 788, 841, 846, 913, 926, 1160,
1161, 1293, 1331, 1394, 1395, 1396, 1397, 1398, 1403, 1404, 1436, 1463, 1486, 1488,
1690, 1774, 1791, 1854, 2089, 2213, 2235, 2320, 2442, 2457, 2531, 2615, 2650, 2789,
2850  2904  2931  2932  2989  3007  3028  3049  3052  3053  3062  3169  3218  3244
```

The GroupBy object



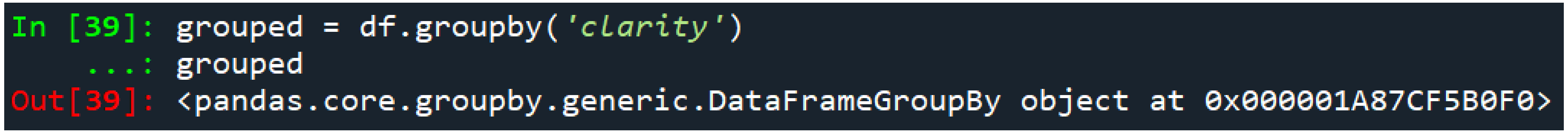
```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```



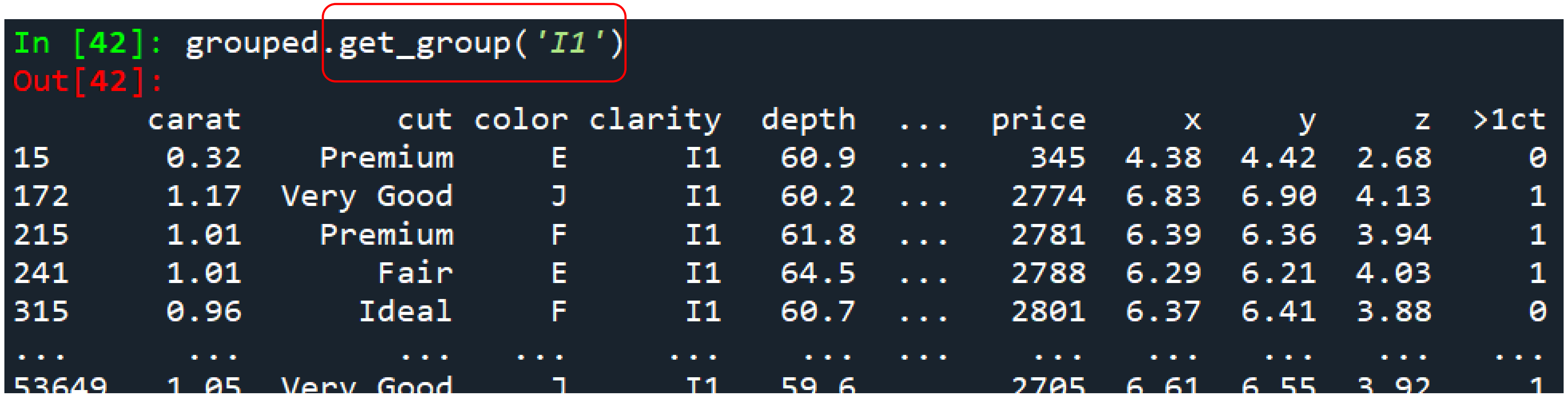
```
In [40]: grouped.groups
Out[40]: {'I1': [15, 172, 215, 241, 315, 323, 369, 376, 415, 444, 465, 525, 535, 551,
634, 653, 654, 664, 682, 701, 718, 800, 844, 857, 865, 925, 967, 991, 992, 1162, 1163,
1199, 1224, 1228, 1362, 1363, 1412, 1475, 1487, 1510, 1527, 1596, 1597, 1598, 1599,
1624, 1639, 1642, 1644, 1684, 1790, 1829, 1879, 1904, 1905, 1974, 1997, 2024, 2025,
2081, 2109, 2149, 2180, 2185, 2204, 2205, 2216, 2276, 2314, 2323, 2324, 2325, 2346,
2347, 2366, 2411, 2432, 2507, 2510, 2521, 2522, 2528, 2600, 2641, 2651, 2800, 2801,
2806, 2877, 2881, 2925, 2945, 2982, 3098, 3137, 3196, 3215, 3241, 3247, 3272, ...],
'IF': [229, 250,                     326, 569, 688, 788, 841, 846, 913, 926, 1160,
1161, 1293, 1331,                    397, 1398, 1403, 1404, 1436, 1463, 1486, 1488,
1690, 1774, 1791,                    235, 2320, 2442, 2457, 2531, 2615, 2650, 2789,
2850, 2904, 2931,                    028, 3049, 3052, 3053, 3062, 3169, 3218, 3244
```

```
In [41]: df.iloc[15]
Out[41]:
carat             0.32
cut            Premium
color                E
clarity             I1
depth             60.9
```

The GroupBy object



```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```



```
In [42]: grouped.get_group('I1')
Out[42]:
       carat        cut color clarity  depth  ...  price     x     y     z  >1ct
15      0.32    Premium     E      I1   60.9  ...    345  4.38  4.42  2.68     0
172     1.17  Very Good     J      I1   60.2  ...   2774  6.83  6.90  4.13     1
215     1.01    Premium     F      I1   61.8  ...   2781  6.39  6.36  3.94     1
241     1.01       Fair     E      I1   64.5  ...   2788  6.29  6.21  4.03     1
315     0.96      Ideal     F      I1   60.7  ...   2801  6.37  6.41  3.88     0
...      ...        ...   ...     ...    ...  ...    ...   ...   ...   ...   ...
53649   1.05  Very Good     J      I1   59.6  ...   2705  6.61  6.55  3.92     1
```

The GroupBy object

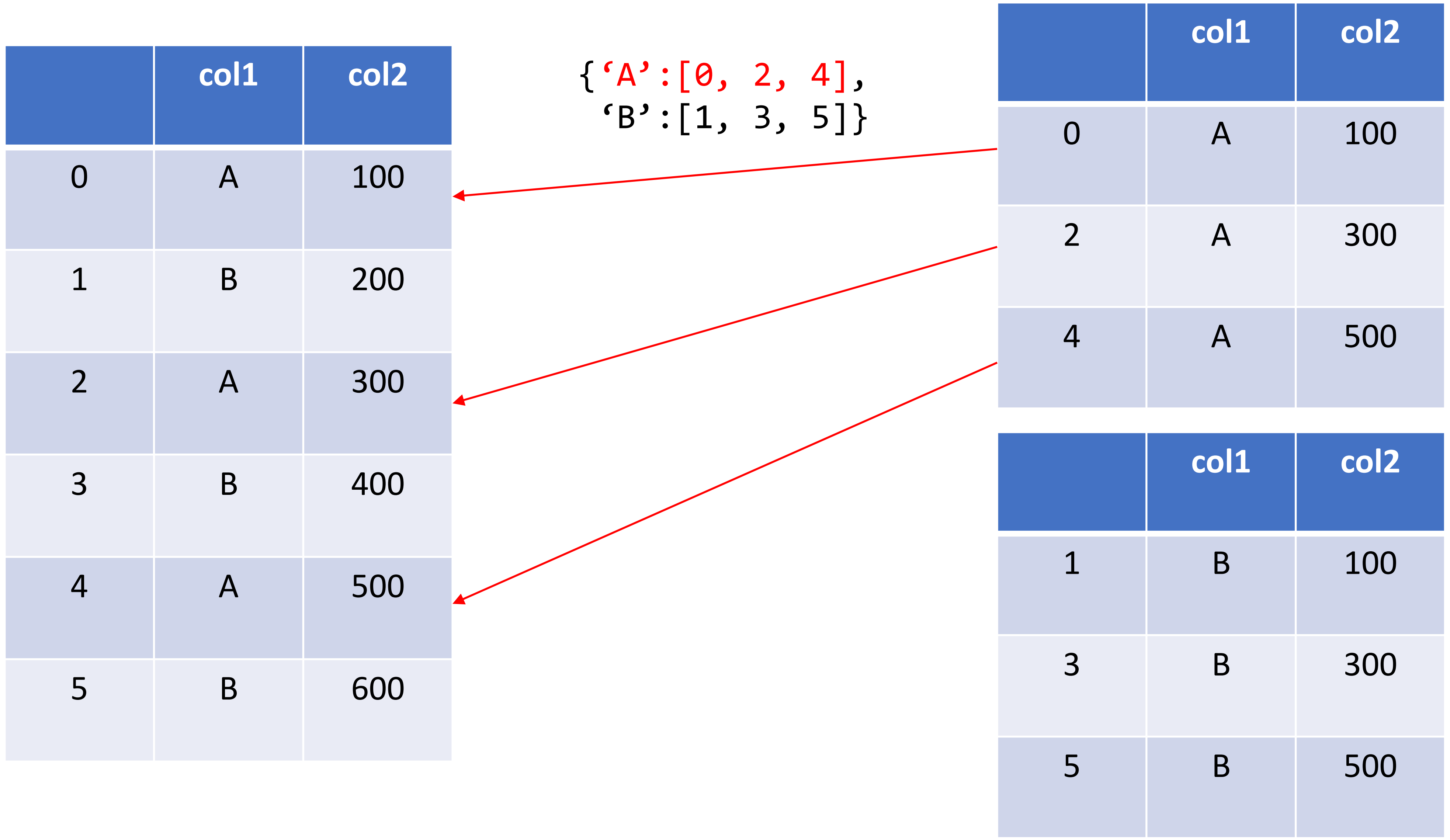
	col1	col2
0	A	100
1	B	200
2	A	300
3	B	400
4	A	500
5	B	600

```
df.groupby('col1')
```

	col1	col2
0	A	100
2	A	300
4	A	500

	col1	col2
1	B	100
3	B	300
5	B	500

The GroupBy object



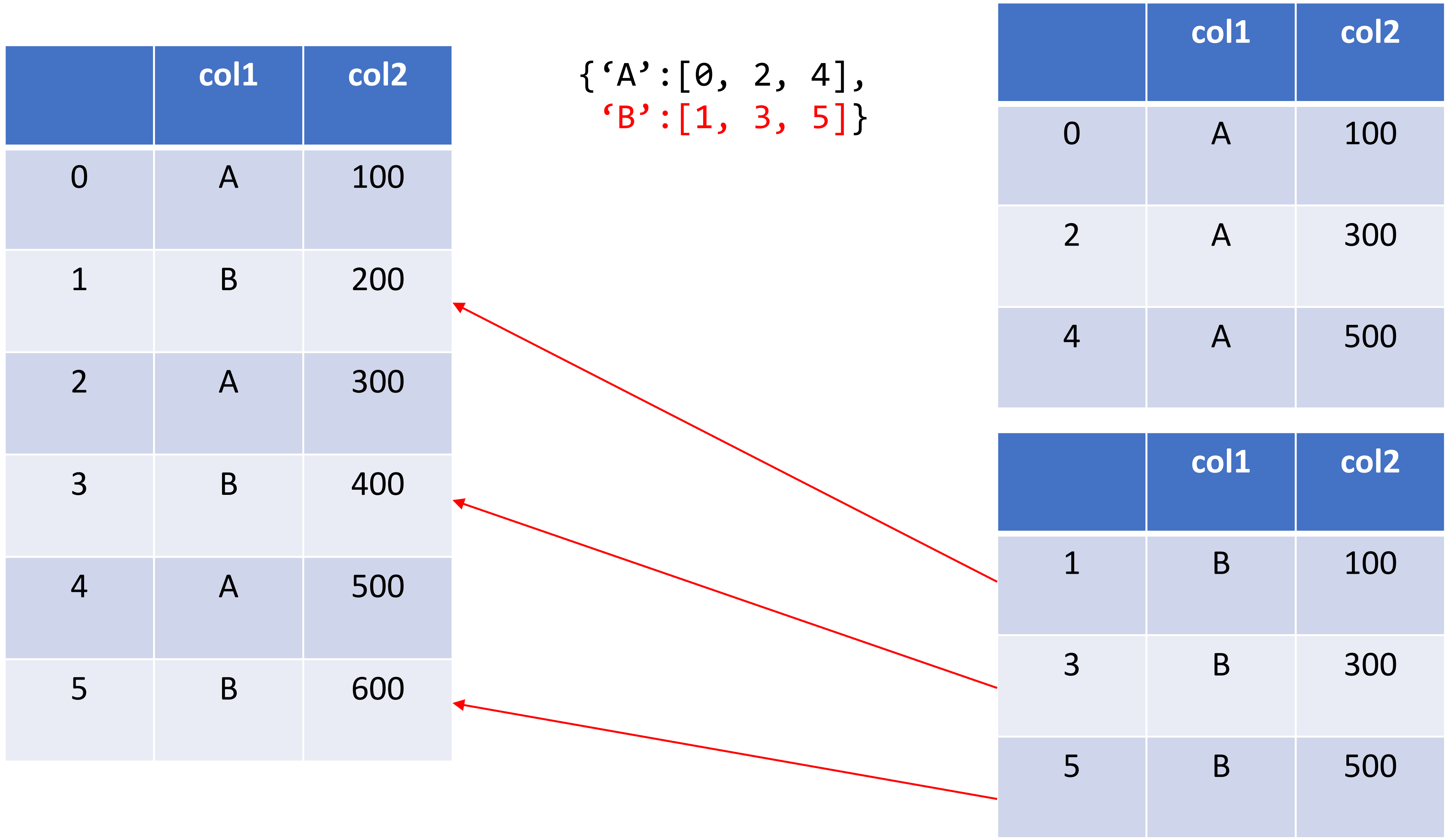
	col1	col2
0	A	100
1	B	200
2	A	300
3	B	400
4	A	500
5	B	600

```
{‘A’:[0, 2, 4],
 ‘B’:[1, 3, 5]}
```

	col1	col2
0	A	100
2	A	300
4	A	500

	col1	col2
1	B	100
3	B	300
5	B	500

The GroupBy object



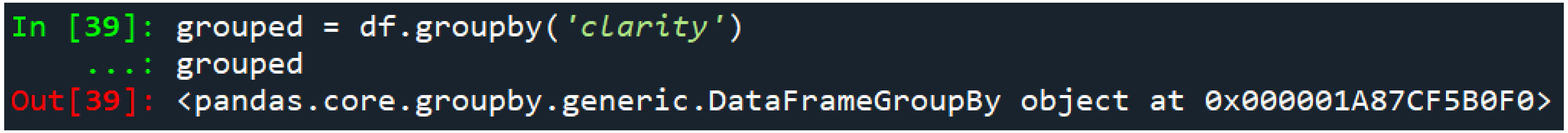
	col1	col2
0	A	100
1	B	200
2	A	300
3	B	400
4	A	500
5	B	600

```
{‘A’:[0, 2, 4],
 ‘B’:[1, 3, 5]}
```

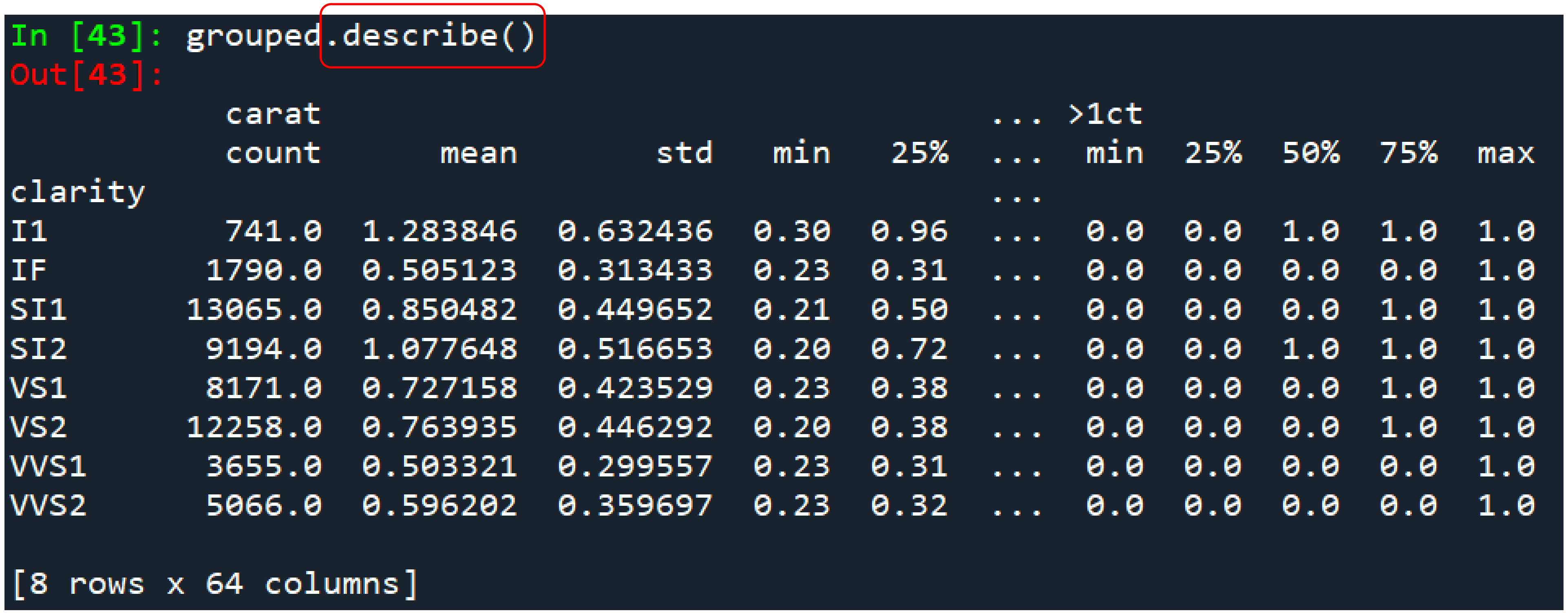
	col1	col2
0	A	100
2	A	300
4	A	500

	col1	col2
1	B	100
3	B	300
5	B	500

The GroupBy object

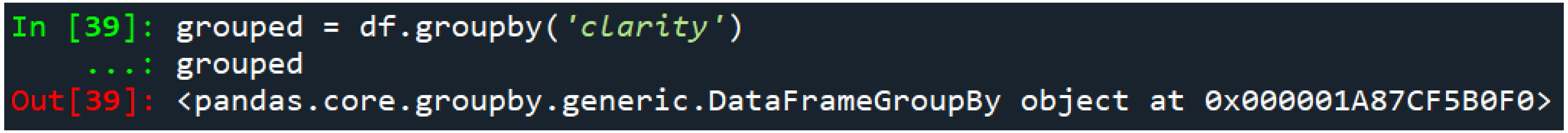


```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```

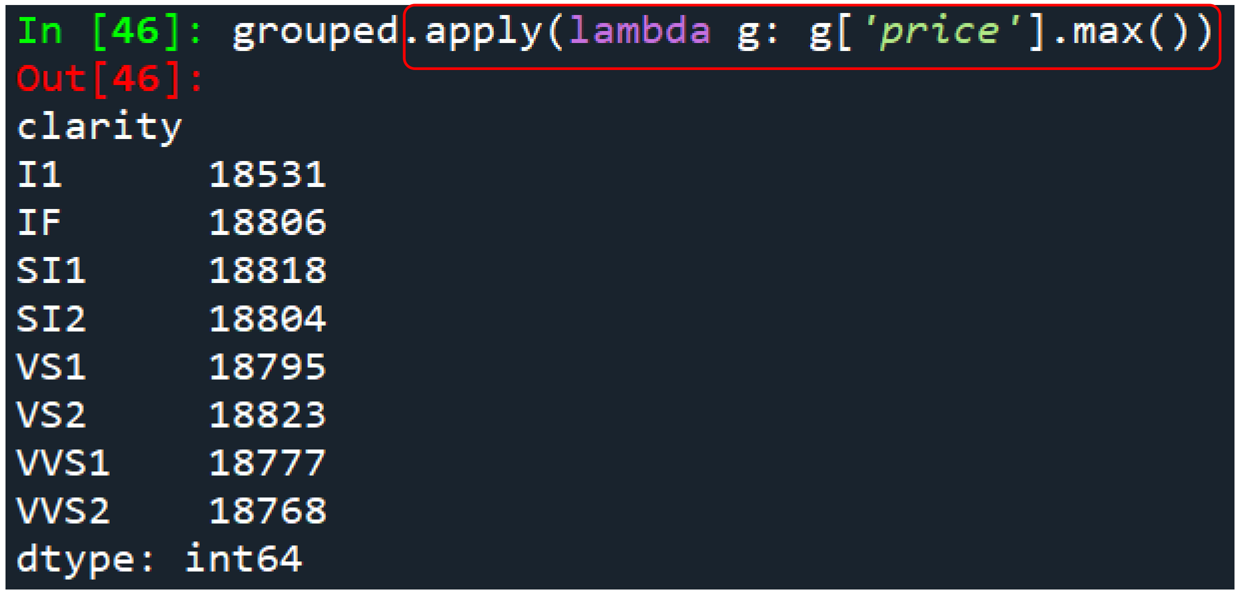


```
In [43]: grouped.describe()
Out[43]:
           carat                                         ... >1ct
           count      mean       std   min   25%  ...  min  25%  50%  75%  max
clarity                                                 ...
I1         741.0  1.283846  0.632436  0.30  0.96  ...  0.0  0.0  1.0  1.0  1.0
IF        1790.0  0.505123  0.313433  0.23  0.31  ...  0.0  0.0  0.0  0.0  1.0
SI1      13065.0  0.850482  0.449652  0.21  0.50  ...  0.0  0.0  0.0  1.0  1.0
SI2       9194.0  1.077648  0.516653  0.20  0.72  ...  0.0  0.0  1.0  1.0  1.0
VS1       8171.0  0.727158  0.423529  0.23  0.38  ...  0.0  0.0  0.0  1.0  1.0
VS2      12258.0  0.763935  0.446292  0.20  0.38  ...  0.0  0.0  0.0  1.0  1.0
VVS1      3655.0  0.503321  0.299557  0.23  0.31  ...  0.0  0.0  0.0  0.0  1.0
VVS2      5066.0  0.596202  0.359697  0.23  0.32  ...  0.0  0.0  0.0  0.0  1.0

[8 rows x 64 columns]
```

The GroupBy object



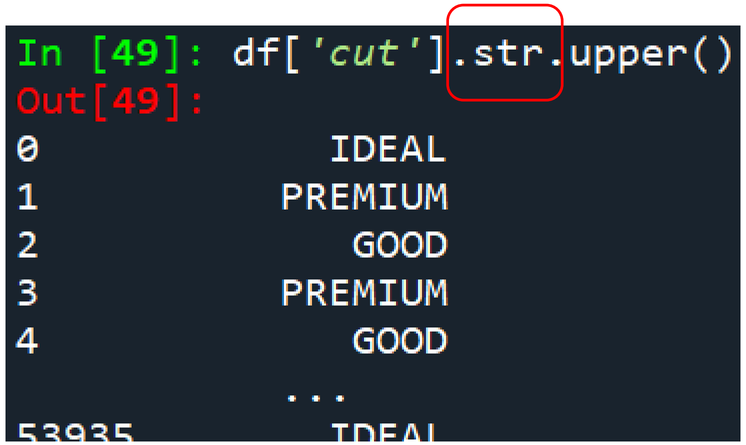
```
In [39]: grouped = df.groupby('clarity')
    ...: grouped
Out[39]: <pandas.core.groupby.generic.DataFrameGroupBy object at 0x000001A87CF5B0F0>
```



```
In [46]: grouped.apply(lambda g: g['price'].max())
Out[46]:
clarity
I1       18531
IF       18806
SI1      18818
SI2      18804
VS1      18795
VS2      18823
VVS1     18777
VVS2     18768
dtype: int64
```

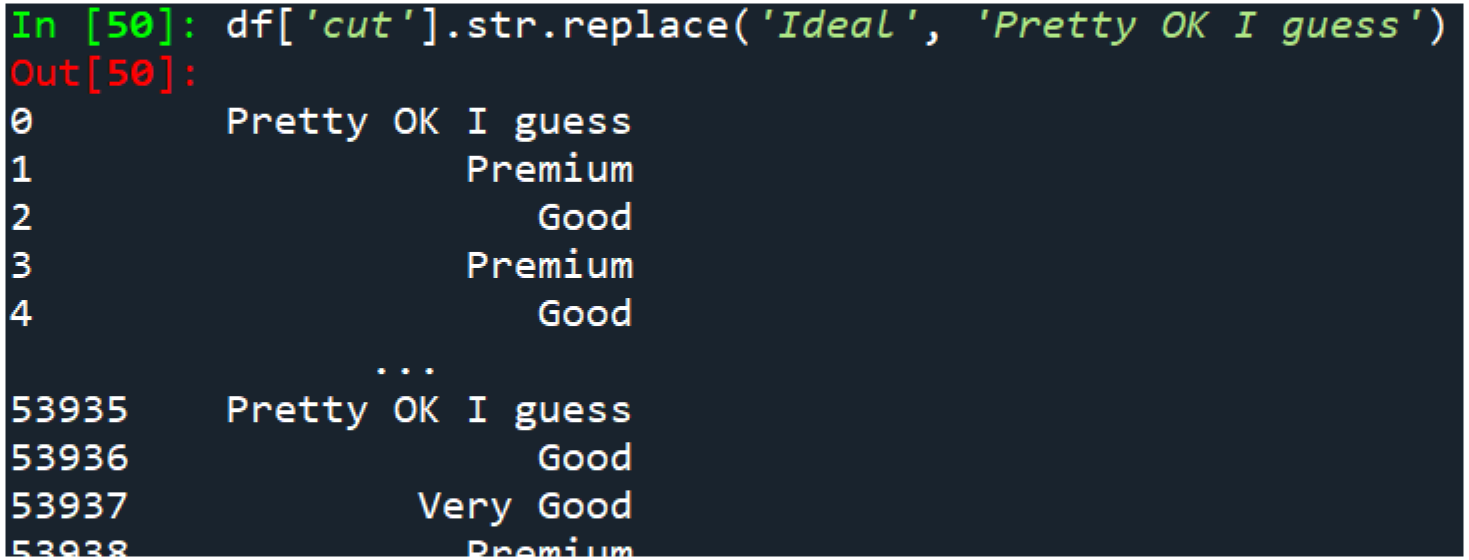
The *apply* method of GroupBy objects iterates over groups. The *apply* method for DataFrame objects iterates over columns.

Strings in DataFrames



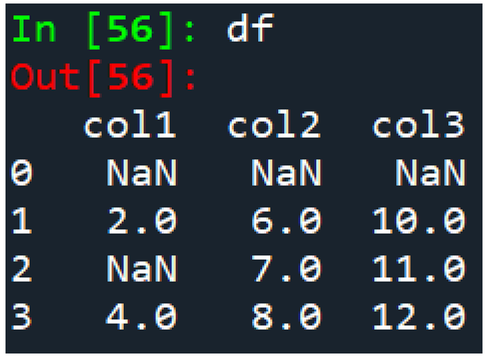
```
In [49]: df['cut'].str.upper()
Out[49]:
0            IDEAL
1          PREMIUM
2             GOOD
3          PREMIUM
4             GOOD
            ...
53935        IDEAL
```

Strings in DataFrames

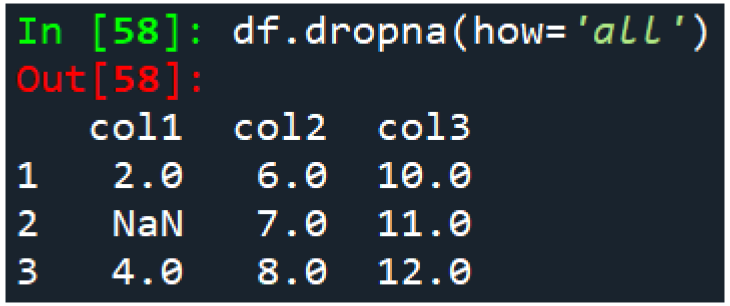


```
In [50]: df['cut'].str.replace('Ideal', 'Pretty OK I guess')
Out[50]:
0        Pretty OK I guess
1                  Premium
2                     Good
3                  Premium
4                     Good
               ...
53935    Pretty OK I guess
53936                 Good
53937            Very Good
```

Brief intro to missing values

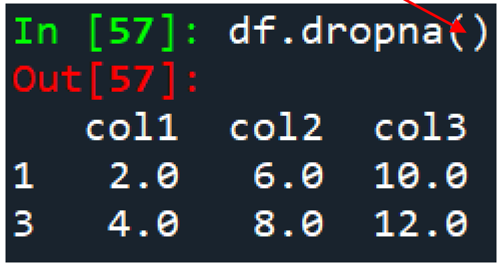


```
In [56]: df
Out[56]:
   col1  col2  col3
0   NaN   NaN   NaN
1   2.0   6.0  10.0
2   NaN   7.0  11.0
3   4.0   8.0  12.0
```

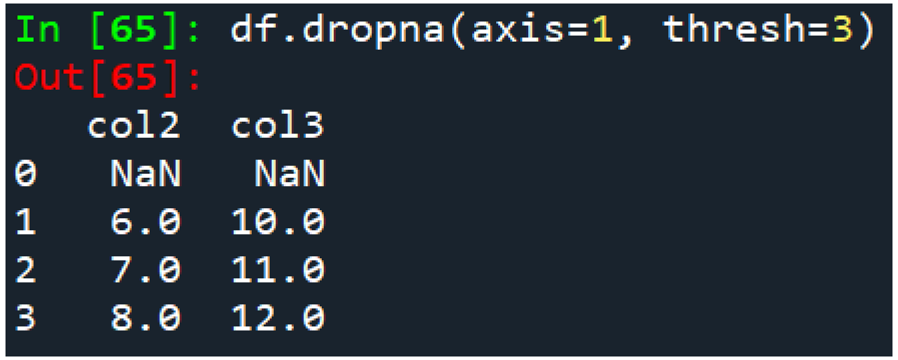


```
In [58]: df.dropna(how='all')
Out[58]:
   col1  col2  col3
1   2.0   6.0  10.0
2   NaN   7.0  11.0
3   4.0   8.0  12.0
```

Kwarg with default value: *how='any'*

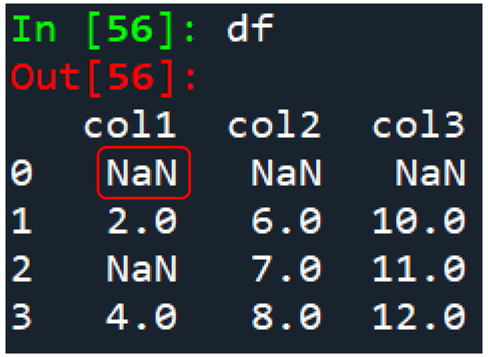


```
In [57]: df.dropna()
Out[57]:
   col1  col2  col3
1   2.0   6.0  10.0
3   4.0   8.0  12.0
```

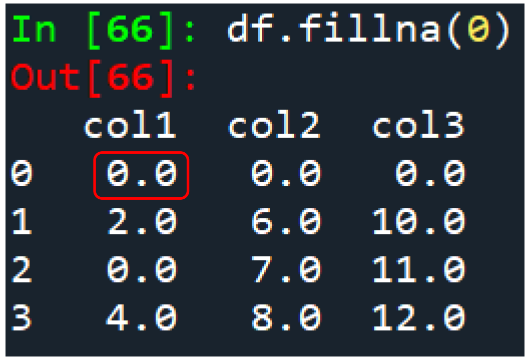


```
In [65]: df.dropna(axis=1, thresh=3)
Out[65]:
   col2  col3
0   NaN   NaN
1   6.0  10.0
2   7.0  11.0
3   8.0  12.0
```

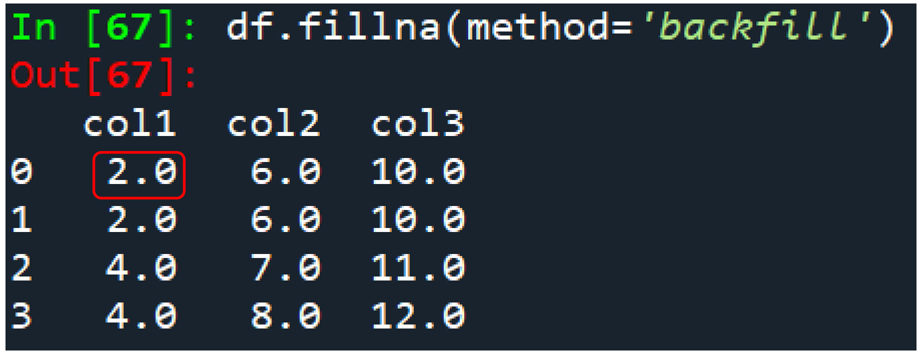
Brief intro to missing values



```
In [56]: df
Out[56]:
   col1  col2  col3
0   NaN   NaN   NaN
1   2.0   6.0  10.0
2   NaN   7.0  11.0
3   4.0   8.0  12.0
```

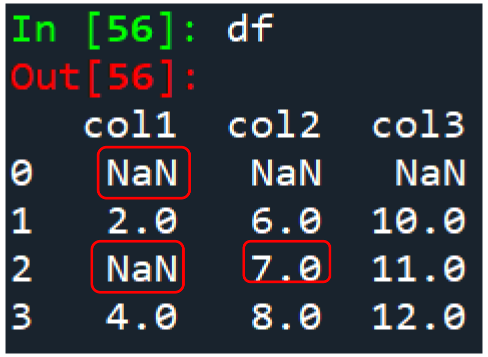


```
In [66]: df.fillna(0)
Out[66]:
   col1  col2  col3
0   0.0   0.0   0.0
1   2.0   6.0  10.0
2   0.0   7.0  11.0
3   4.0   8.0  12.0
```

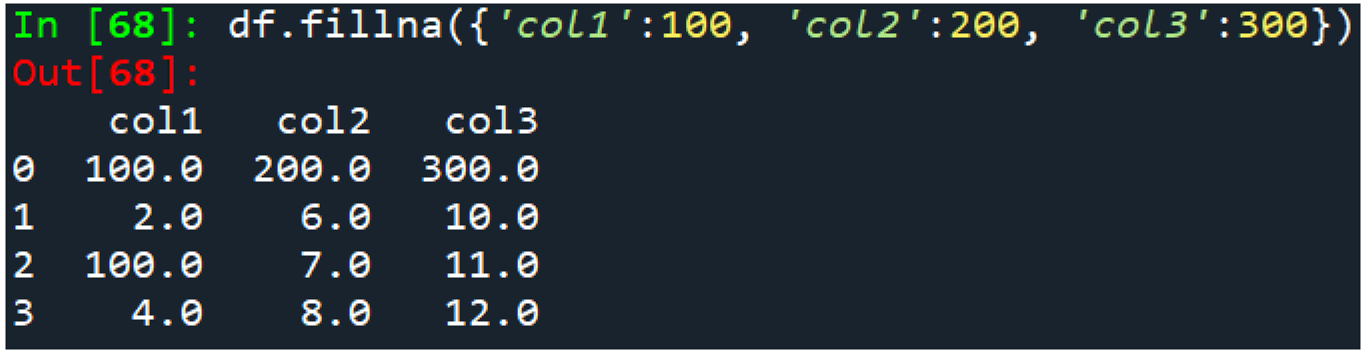


```
In [67]: df.fillna(method='backfill')
Out[67]:
   col1  col2  col3
0   2.0   6.0  10.0
1   2.0   6.0  10.0
2   4.0   7.0  11.0
3   4.0   8.0  12.0
```

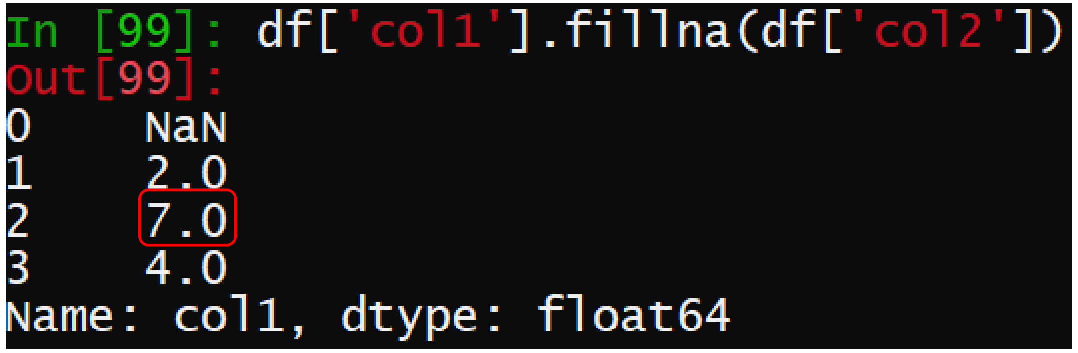
Brief intro to missing values



```
In [56]: df
Out[56]:
   col1  col2  col3
0   NaN   NaN   NaN
1   2.0   6.0  10.0
2   NaN   7.0  11.0
3   4.0   8.0  12.0
```



```
In [68]: df.fillna({'col1':100, 'col2':200, 'col3':300})
Out[68]:
    col1   col2   col3
0  100.0  200.0  300.0
1    2.0    6.0   10.0
2  100.0    7.0   11.0
3    4.0    8.0   12.0
```



```
In [99]: df['col1'].fillna(df['col2'])
Out[99]:
0      NaN
1      2.0
2      7.0
3      4.0
Name: col1, dtype: float64
```