KÖNIGSWEG

PyData Berlin 2017

Git with instructions: https://goo.gl/JdsqBe

Introduction to Data-Analysis with Pandas

90' tutorial

Alexander C. S. Hendorf

@hendorf

Alexander C. S. Hendorf

Königsweg GmbH

Strategic data consulting for startups and the industry.

EuroPython & PyConDE
Organisator + Programm Chair

mongoDB master, PSF managing member

Speaker mongoDB days, EuroPython, PyData...

🐦 @hendorf

Today

Introduction to Data-Analysis with Pandas

- Welcome & origins of Pandas
- Get ready to code along
- Reading and writing data across multiple formats
- DataSeries & DataFrames / NumPy
- Selecting data
- Operations
- Data visualisation
- Peek into statistical data analysis and aggregation
- How to mangle, reshape and pivot
- Ode to Indexes

Install Environment

https://github.com/alanderex/pandas-pydata-berlin-2017

***short-url*: https://goo.gl/JdsqBe**

Jupyter notebooks

pandas

numpy

matplotlib

xlsxwriter

Reading and writing data across multiple formats

- CSV
- Excel
- JSON
- Clipboard

- data
  - .info
  - .describe

Reading and writing data across multiple formats

- convention `import pandas as pd`
- *read:* `pd.read_csv/excel/json/…()`
- *write:* `pd.write_csv/excel/json/…()`
- *both:*
  - very flexible, highly customisable, often default setting just work fine
- preview data with `.head(#n)` and `.tail(#n)`

DataSeries & DataFrames / NumPy

Ode to NumPy

Definitions:

- Table
- Column
- Row
- Data-Series
- Data-Frame

Structure

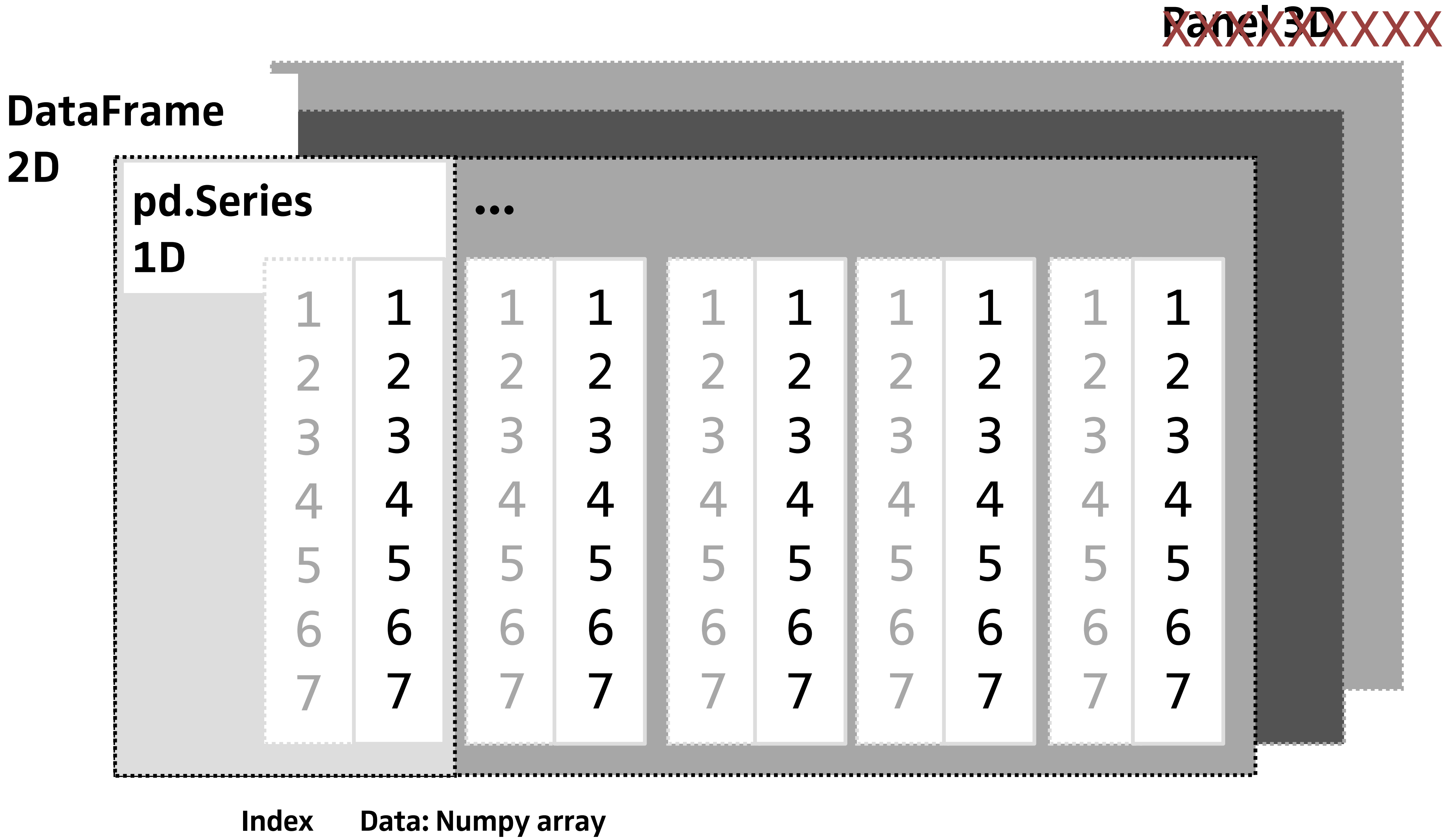
1
2
3
4
5
6
7

Data: Numpy array

Ode to NumPy

- Fundamental package needed for scientific computing with Python
- Powerful typed array object
- Broadcasting
- …

Structure



~~Panel 3D~~ XXXXXXX

DataFrame
2D

pd.Series
1D

...

1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7

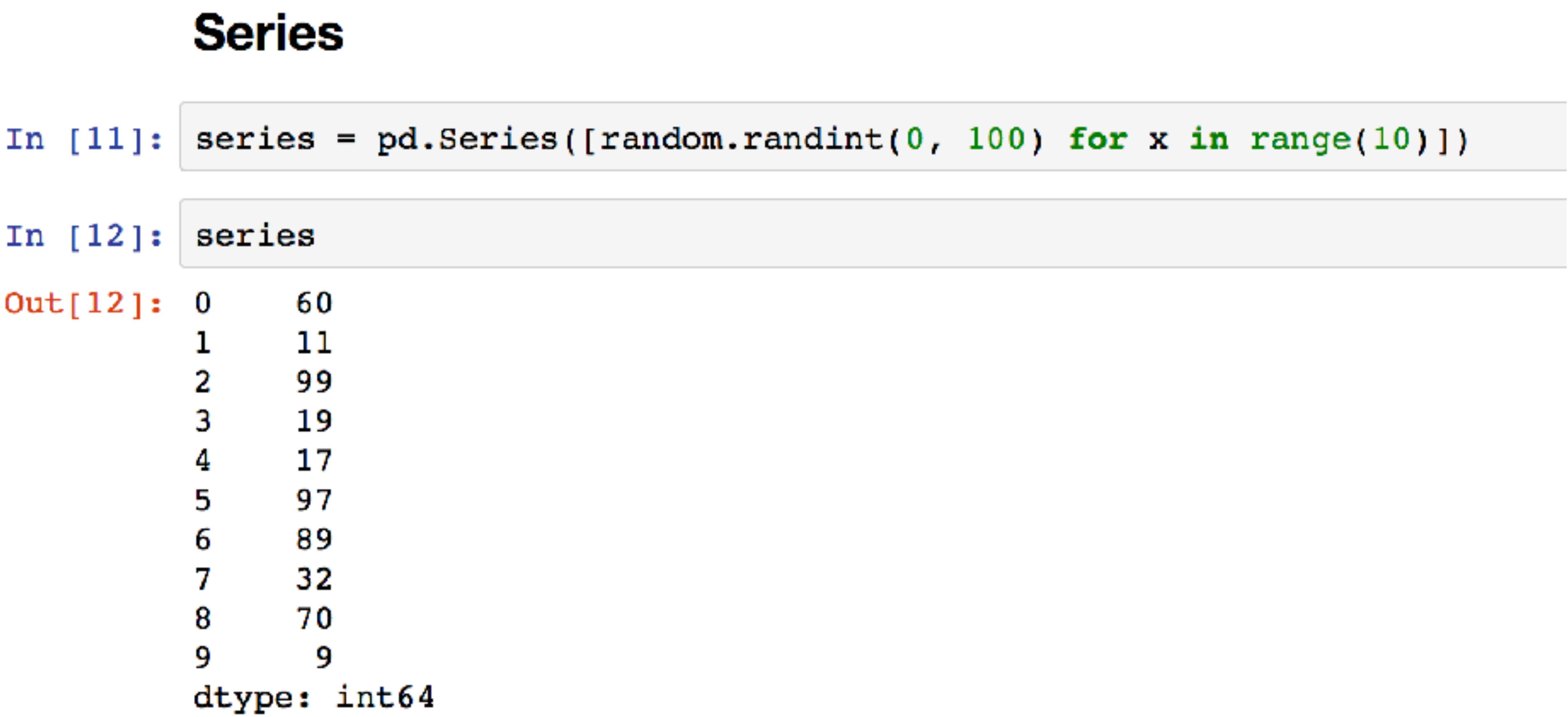
Index

Data: Numpy array

DataSeries & DataFrames / Numpy

Definitions:

- Table -> (2-D) Data-Frame
- Column -> Data-Series
- Row -> values @ same position in each Data-Series in Data-Frame



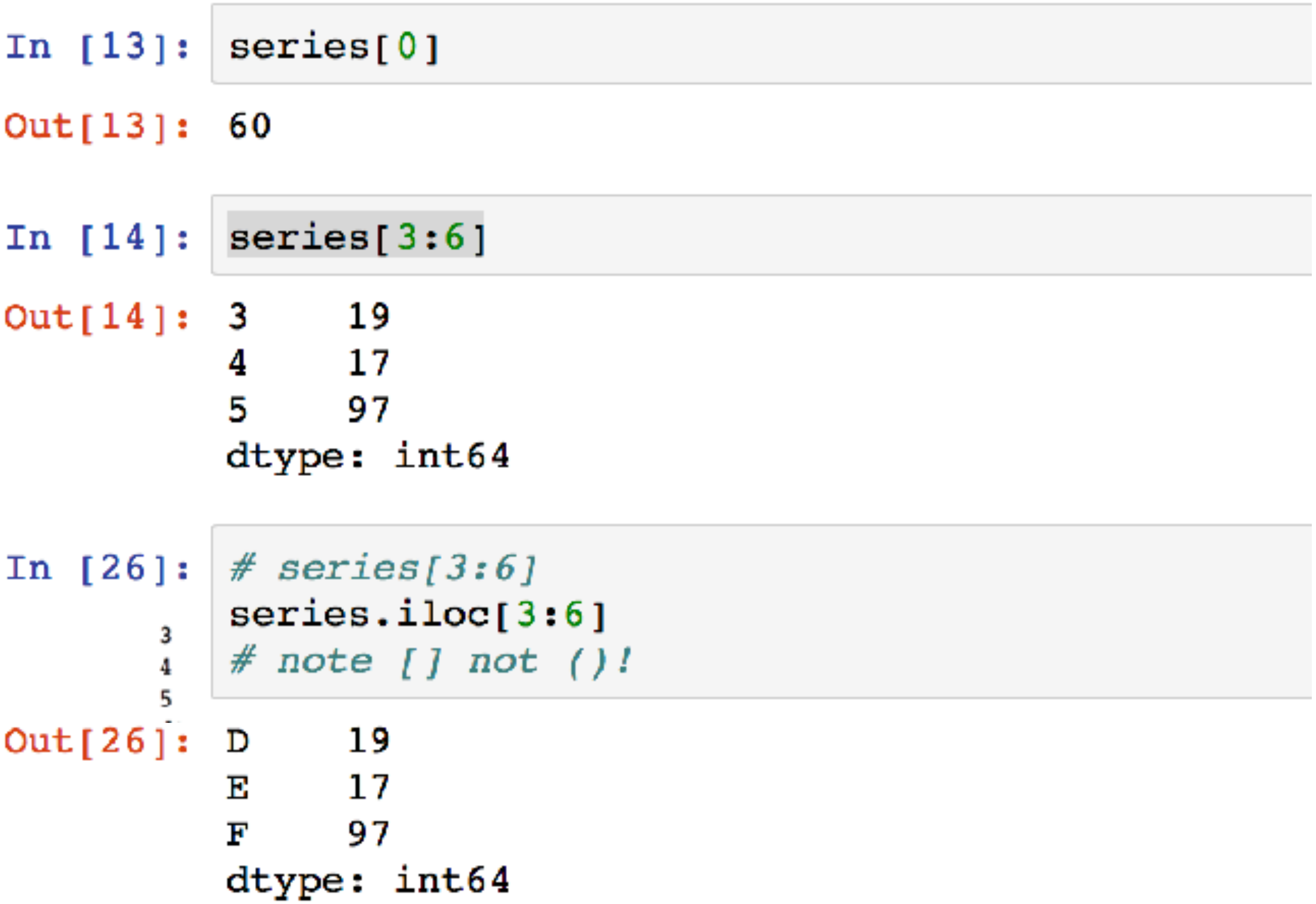
Series

```
In [11]: series = pd.Series([random.randint(0, 100) for x in range(10)])
```

```
In [12]: series
```

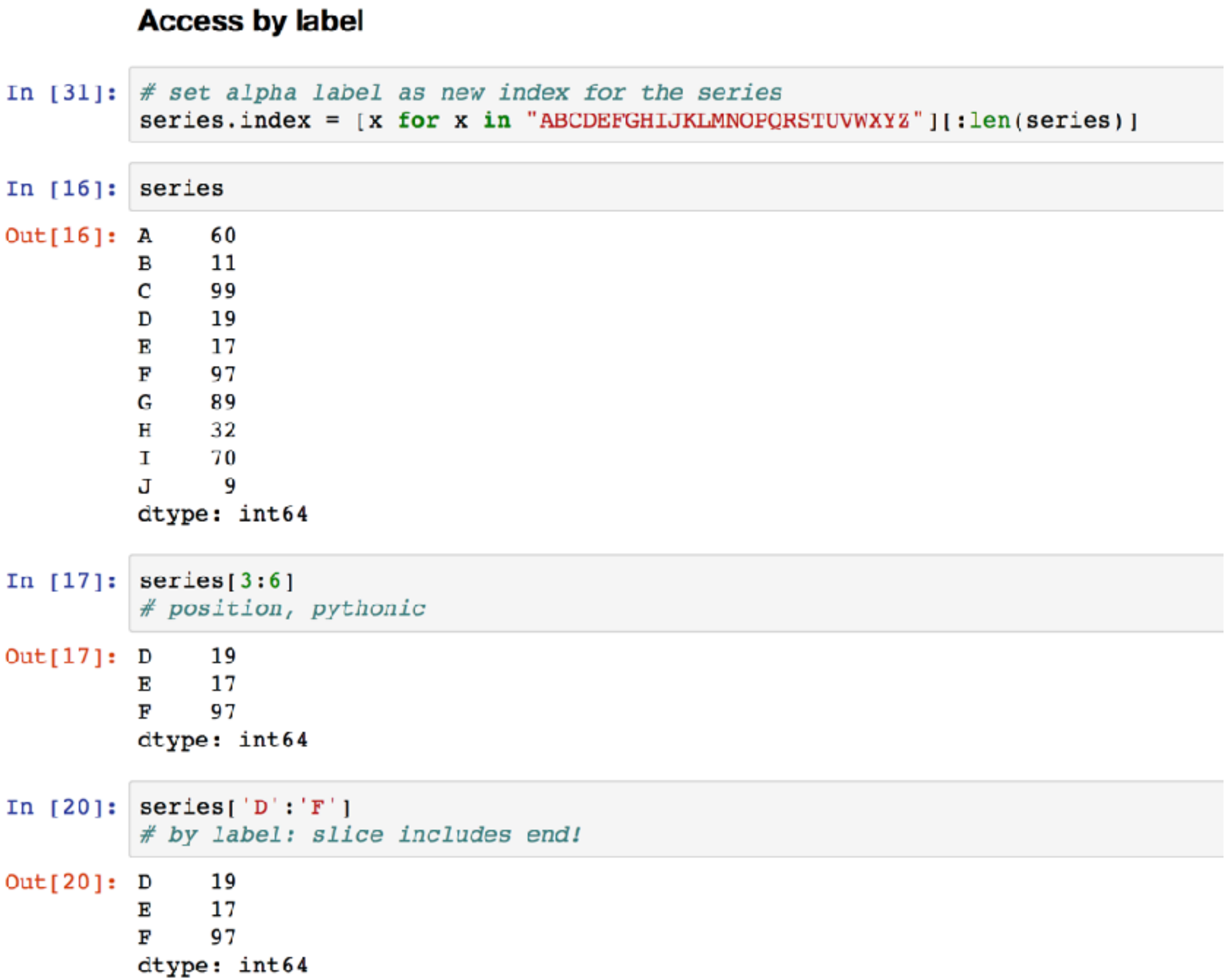
```
Out[12]: 0    60
         1    11
         2    99
         3    19
         4    17
         5    97
         6    89
         7    32
         8    70
         9     9
         dtype: int64
```

Access by Position / Slice



```
In [13]: series[0]

Out[13]: 60

In [14]: series[3:6]

Out[14]: 3     19
         4     17
         5     97
         dtype: int64

In [26]: # series[3:6]
         series.iloc[3:6]
      3  # note [] not ()!
      4
      5
Out[26]: D     19
         E     17
         F     97
         dtype: int64
```

```
0     60
1     11
2     99
3     19
4     17
5     97
6     89
7     32
8     70
9      9
```



Access by label

```
In [31]: # set alpha label as new index for the series
         series.index = [x for x in "ABCDEFGHIJKLMNOPQRSTUVWXYZ"][:len(series)]
```

```
In [16]: series
```

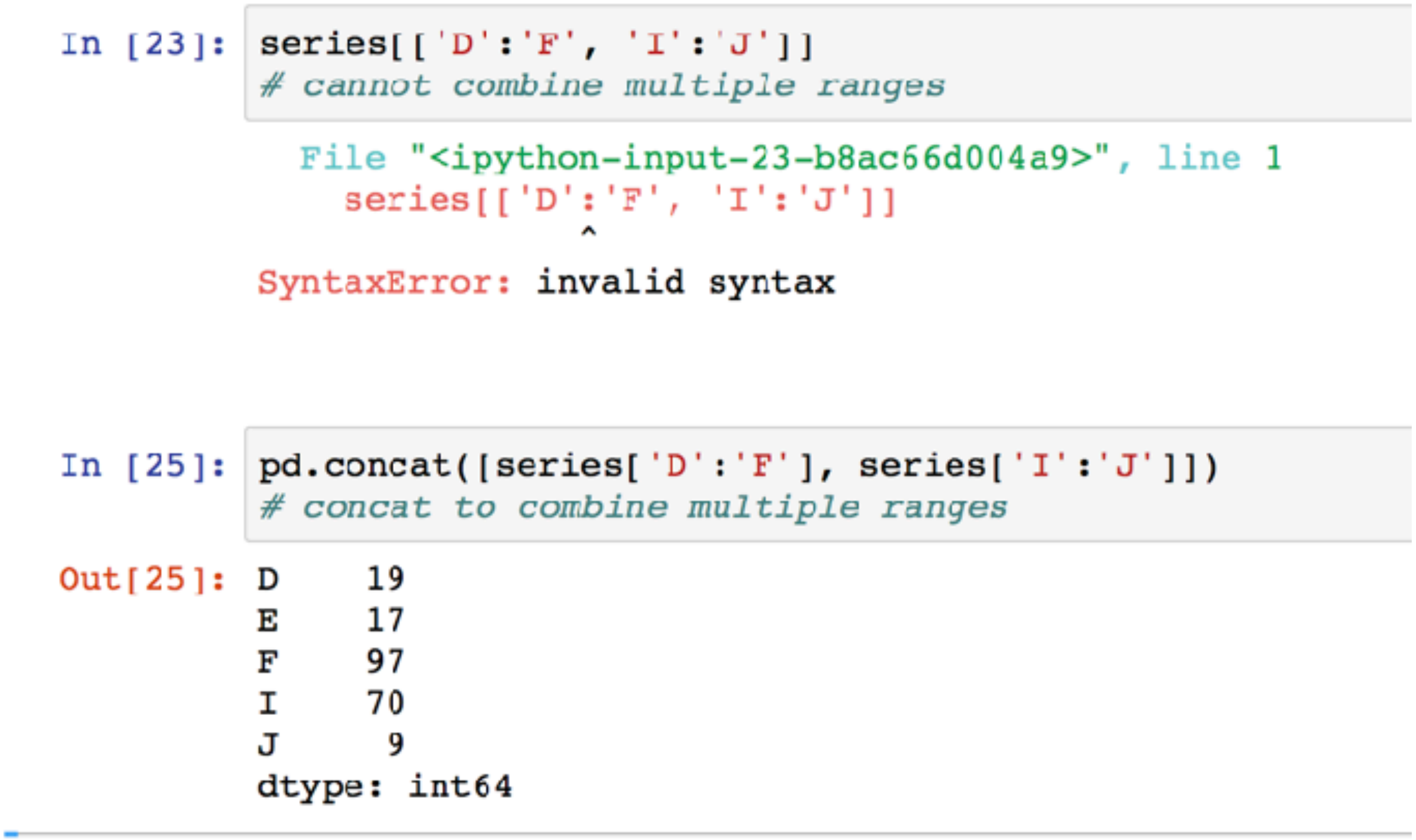
```
Out[16]: A    60
         B    11
         C    99
         D    19
         E    17
         F    97
         G    89
         H    32
         I    70
         J     9
         dtype: int64
```

```
In [17]: series[3:6]
         # position, pythonic
```

```
Out[17]: D    19
         E    17
         F    97
         dtype: int64
```

```
In [20]: series['D':'F']
         # by label: slice includes end!
```

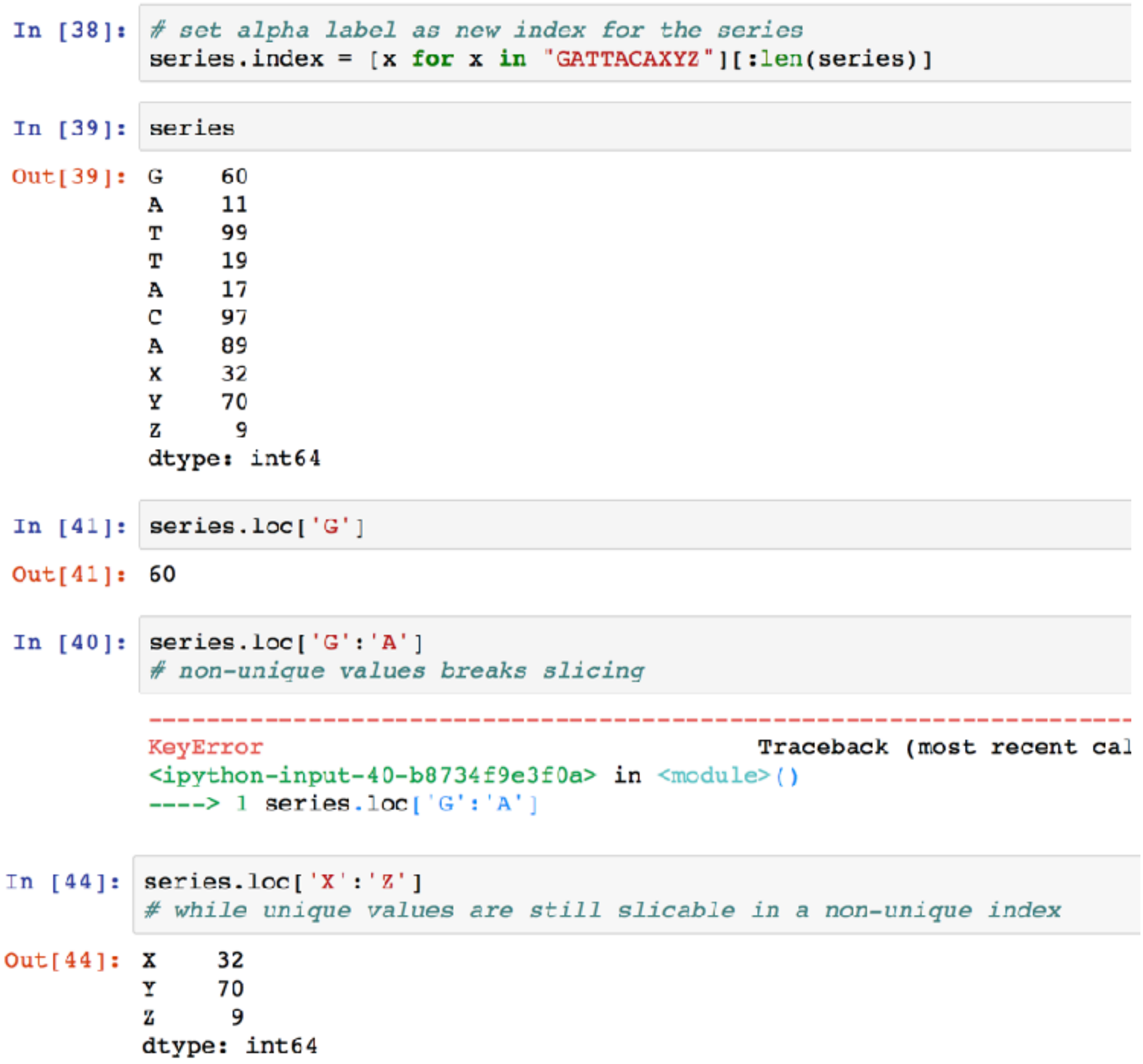
```
Out[20]: D    19
         E    17
         F    97
         dtype: int64
```



```
In [23]: series[['D':'F', 'I':'J']]
         # cannot combine multiple ranges

           File "<ipython-input-23-b8ac66d004a9>", line 1
             series[['D':'F', 'I':'J']]
                        ^
         SyntaxError: invalid syntax
```

```
In [25]: pd.concat([series['D':'F'], series['I':'J']])
         # concat to combine multiple ranges

Out[25]: D    19
         E    17
         F    97
         I    70
         J     9
         dtype: int64
```

```
A    60
B    11
C    99
D    19
E    17
F    97
G    89
H    32
I    70
J     9
```



```
In [38]: # set alpha label as new index for the series
         series.index = [x for x in "GATTACAXYZ"][:len(series)]

In [39]: series

Out[39]: G    60
         A    11
         T    99
         T    19
         A    17
         C    97
         A    89
         X    32
         Y    70
         Z     9
         dtype: int64

In [41]: series.loc['G']

Out[41]: 60

In [40]: series.loc['G':'A']
         # non-unique values breaks slicing

         ---------------------------------------------------------------------------
         KeyError                                  Traceback (most recent cal
         <ipython-input-40-b8734f9e3f0a> in <module>()
         ----> 1 series.loc['G':'A']

In [44]: series.loc['X':'Z']
         # while unique values are still slicable in a non-unique index

Out[44]: X    32
         Y    70
         Z     9
         dtype: int64
```

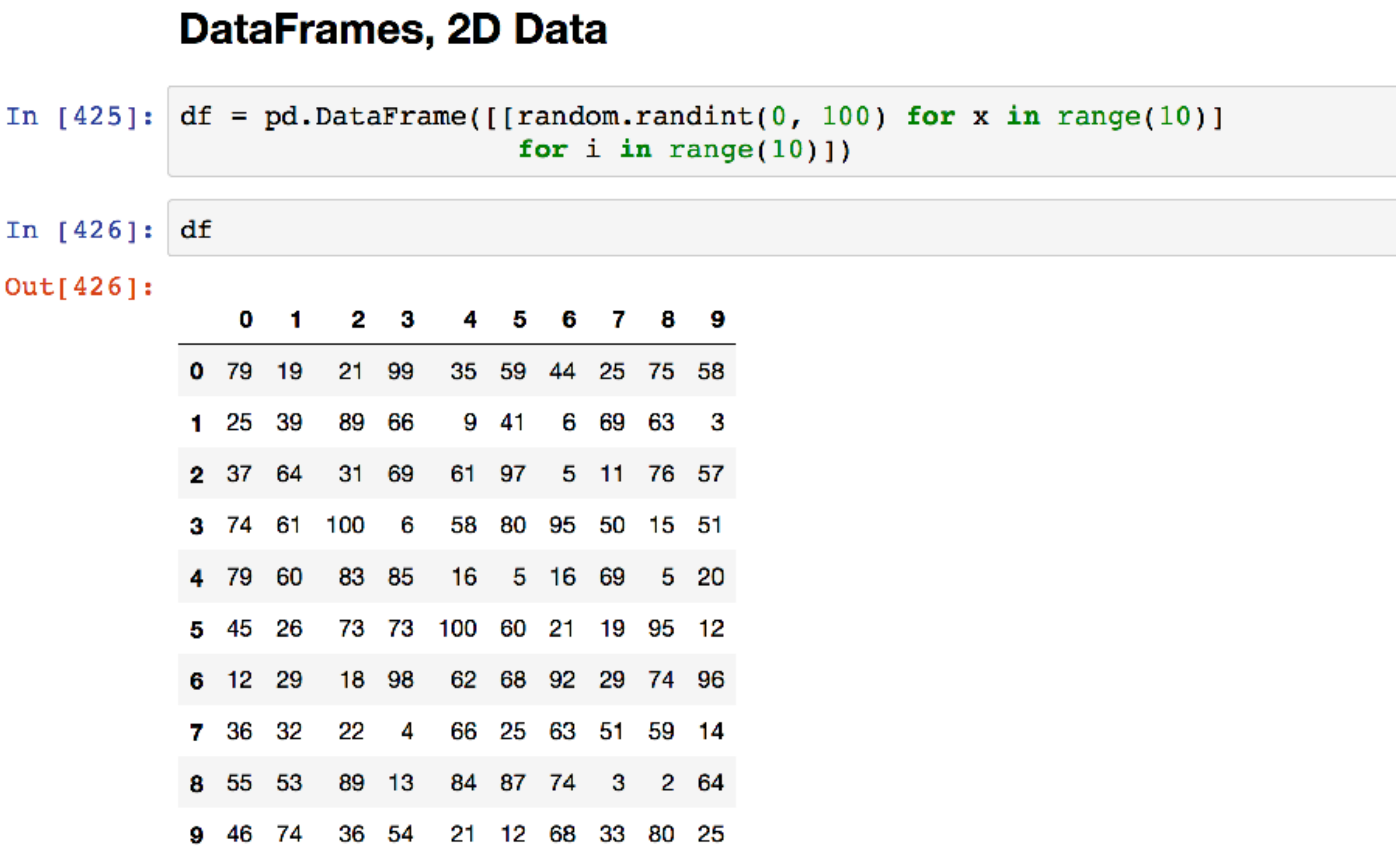
A	60
B	11
C	99
D	19
E	17
F	97
G	89
H	32
I	70
J	9

Structure: Index

- the label of a series is usually called index
- automatically created if not given
- can be reset or replaced
- immutable
- can only contain hashable objects
- one or more dimensions
- may contain a value more than once (NOT UNIQUE!)

Index Types

- Index
- MultiIndex
- DateTimeIndex
- TimeDelta
- IntervalIndex
- CategoricalIndex



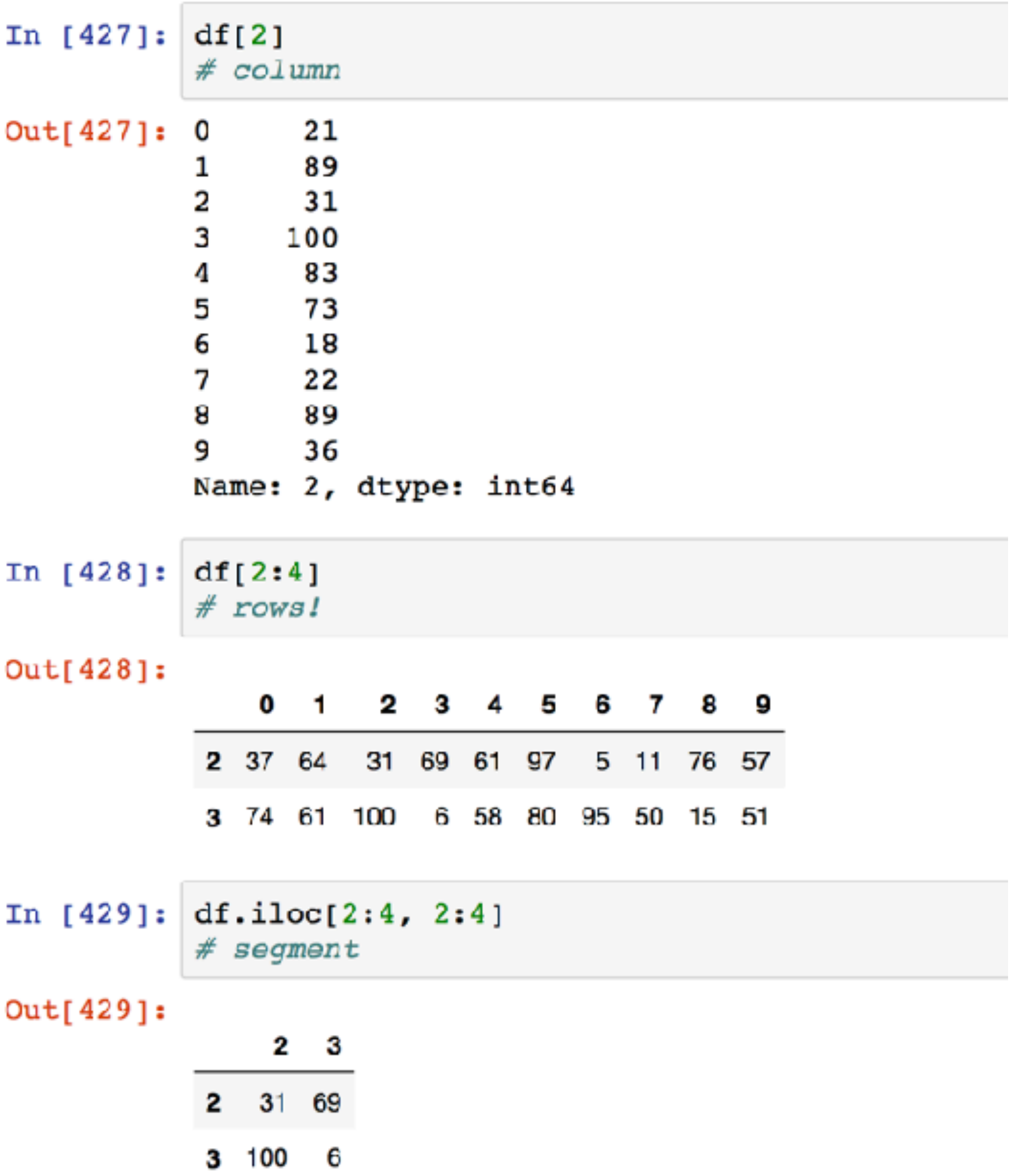
DataFrames, 2D Data

```
In [425]: df = pd.DataFrame([[random.randint(0, 100) for x in range(10)]
                                 for i in range(10)])
```

```
In [426]: df
```

Out[426]:

	0	1	2	3	4	5	6	7	8	9
0	79	19	21	99	35	59	44	25	75	58
1	25	39	89	66	9	41	6	69	63	3
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51
4	79	60	83	85	16	5	16	69	5	20
5	45	26	73	73	100	60	21	19	95	12
6	12	29	18	98	62	68	92	29	74	96
7	36	32	22	4	66	25	63	51	59	14
8	55	53	89	13	84	87	74	3	2	64
9	46	74	36	54	21	12	68	33	80	25



```
In [427]: df[2]
          # column

Out[427]: 0     21
          1     89
          2     31
          3    100
          4     83
          5     73
          6     18
          7     22
          8     89
          9     36
          Name: 2, dtype: int64
```

```
In [428]: df[2:4]
          # rows!
```

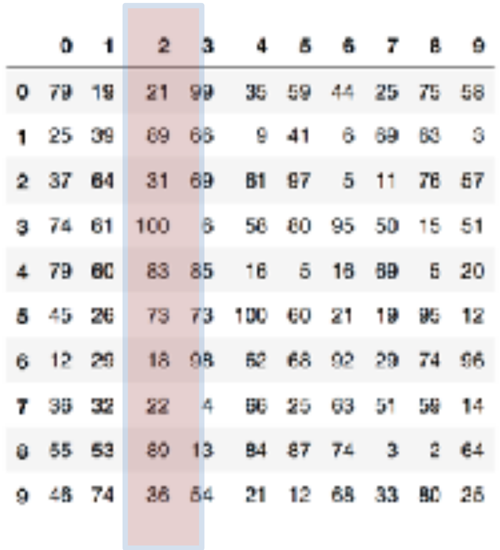
Out[428]:

	0	1	2	3	4	5	6	7	8	9
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51

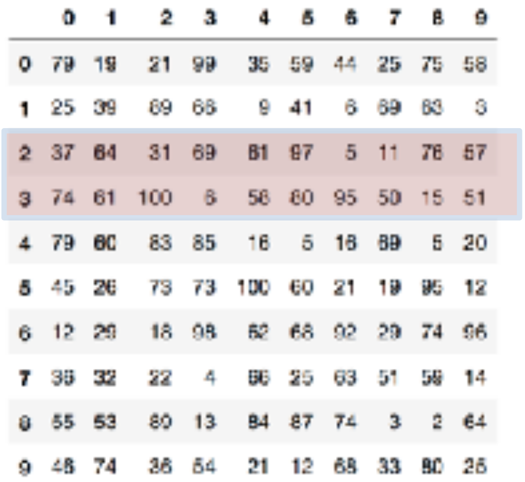
```
In [429]: df.iloc[2:4, 2:4]
          # segment
```

Out[429]:

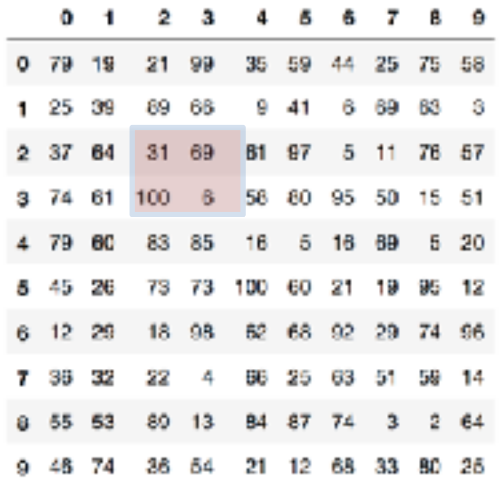
	2	3
2	31	69
3	100	6



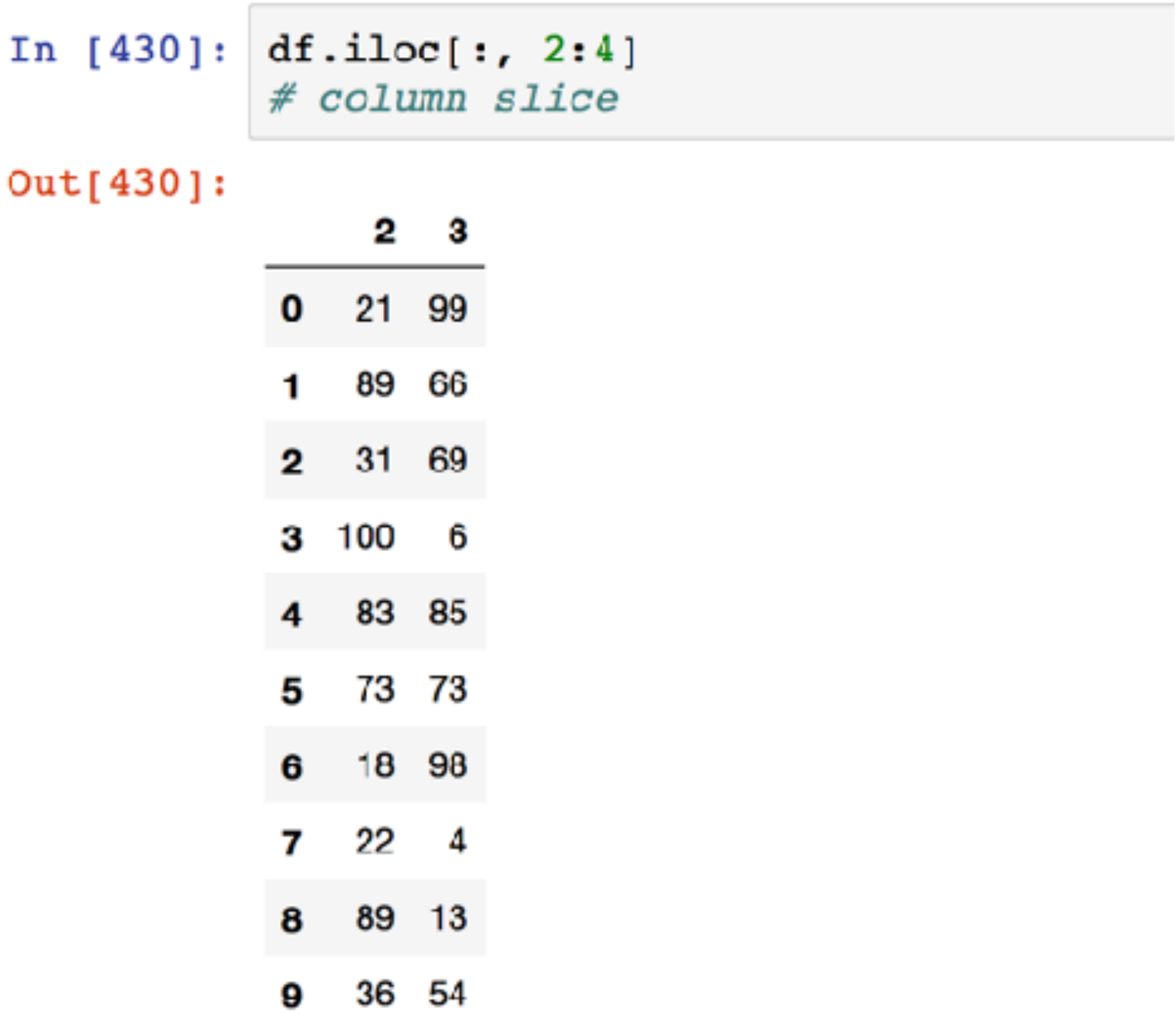
	0	1	2	3	4	5	6	7	8	9
0	79	19	21	99	35	59	44	25	75	58
1	25	39	89	66	9	41	6	89	83	3
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51
4	79	60	83	85	16	5	16	89	5	20
5	45	26	73	73	100	60	21	19	85	12
6	12	29	18	98	62	68	92	29	74	96
7	35	32	22	4	86	25	63	51	58	14
8	55	53	89	13	84	87	74	3	2	64
9	48	74	36	54	21	12	68	33	80	25



	0	1	2	3	4	5	6	7	8	9
0	79	19	21	99	35	59	44	25	75	58
1	25	39	89	66	9	41	6	89	83	3
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51
4	79	60	83	85	16	5	16	89	5	20
5	45	26	73	73	100	60	21	19	85	12
6	12	29	18	98	62	68	92	29	74	96
7	35	32	22	4	86	25	63	51	58	14
8	55	53	89	13	84	87	74	3	2	64
9	48	74	36	54	21	12	68	33	80	25



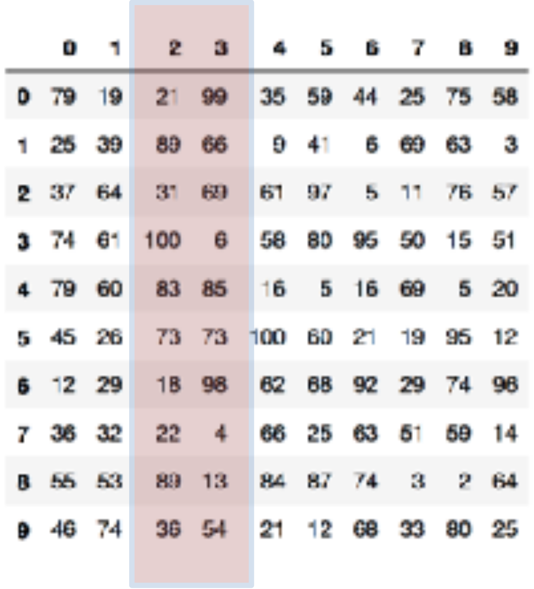
	0	1	2	3	4	5	6	7	8	9
0	79	19	21	99	35	59	44	25	75	58
1	25	39	89	66	9	41	6	89	83	3
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51
4	79	60	83	85	16	5	16	89	5	20
5	45	26	73	73	100	60	21	19	85	12
6	12	29	18	98	62	68	92	29	74	96
7	35	32	22	4	86	25	63	51	58	14
8	55	53	89	13	84	87	74	3	2	64
9	48	74	36	54	21	12	68	33	80	25



```
In [430]: df.iloc[:, 2:4]
          # column slice
```

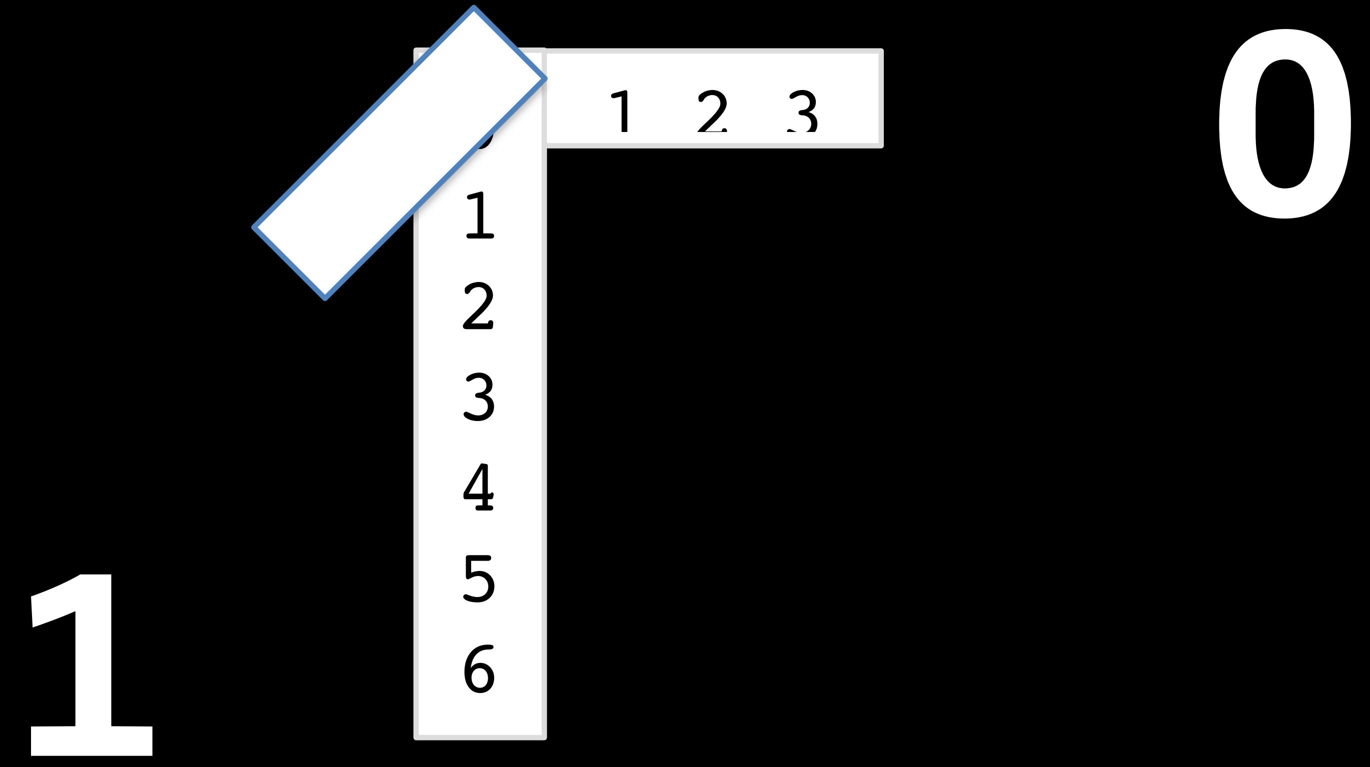
Out[430]:

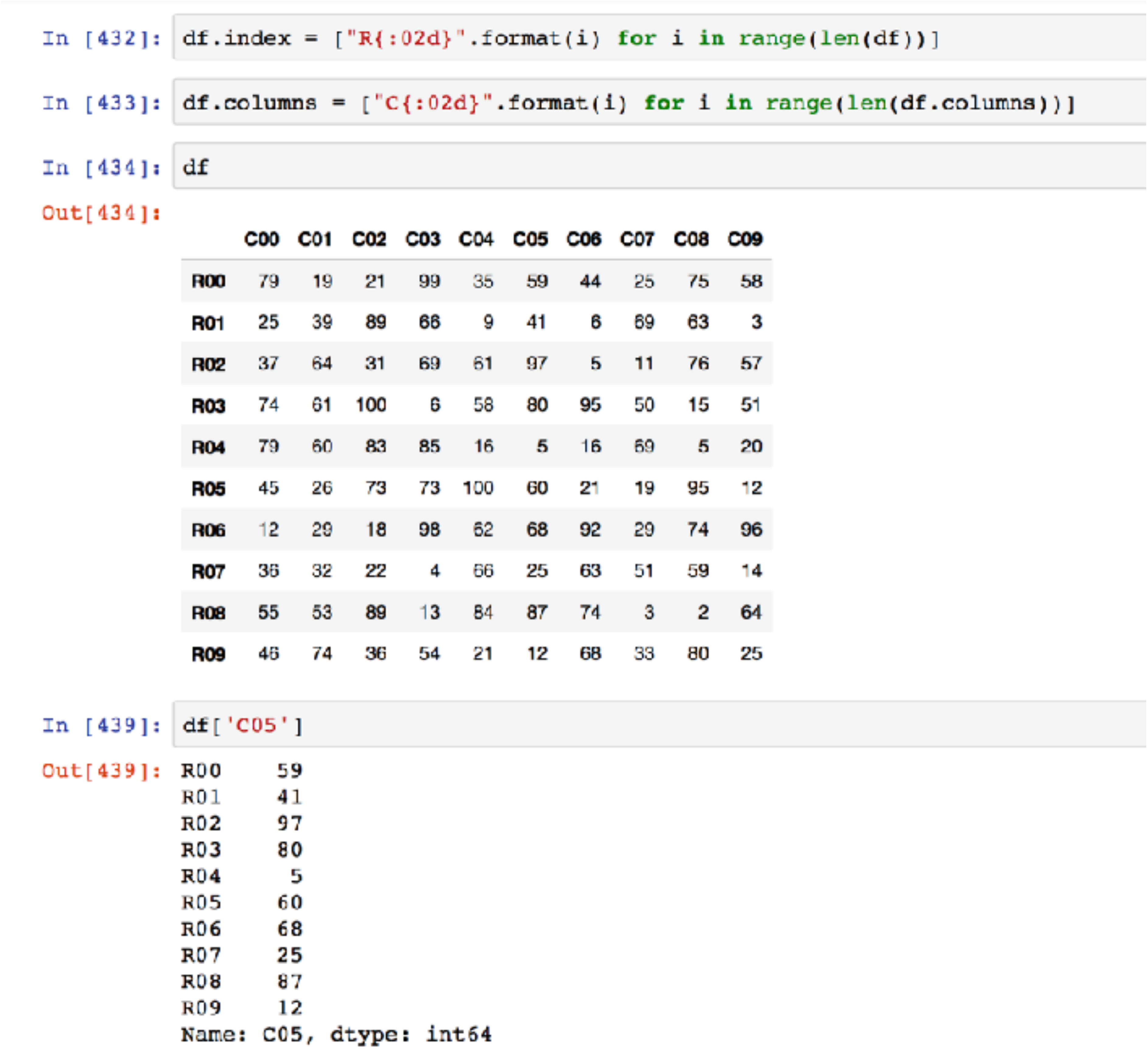
	2	3
0	21	99
1	89	66
2	31	69
3	100	6
4	83	85
5	73	73
6	18	98
7	22	4
8	89	13
9	36	54



	0	1	2	3	4	5	6	7	8	9
0	79	19	21	99	35	59	44	25	75	58
1	25	39	89	66	9	41	6	69	63	3
2	37	64	31	69	61	97	5	11	76	57
3	74	61	100	6	58	80	95	50	15	51
4	79	60	83	85	16	5	16	69	5	20
5	45	26	73	73	100	60	21	19	95	12
6	12	29	18	98	62	68	92	29	74	96
7	36	32	22	4	66	25	63	61	59	14
8	55	53	89	13	84	87	74	3	2	64
9	46	74	36	54	21	12	68	33	80	25

Axes





```
In [432]: df.index = ["R{:02d}".format(i) for i in range(len(df))]
```

```
In [433]: df.columns = ["C{:02d}".format(i) for i in range(len(df.columns))]
```

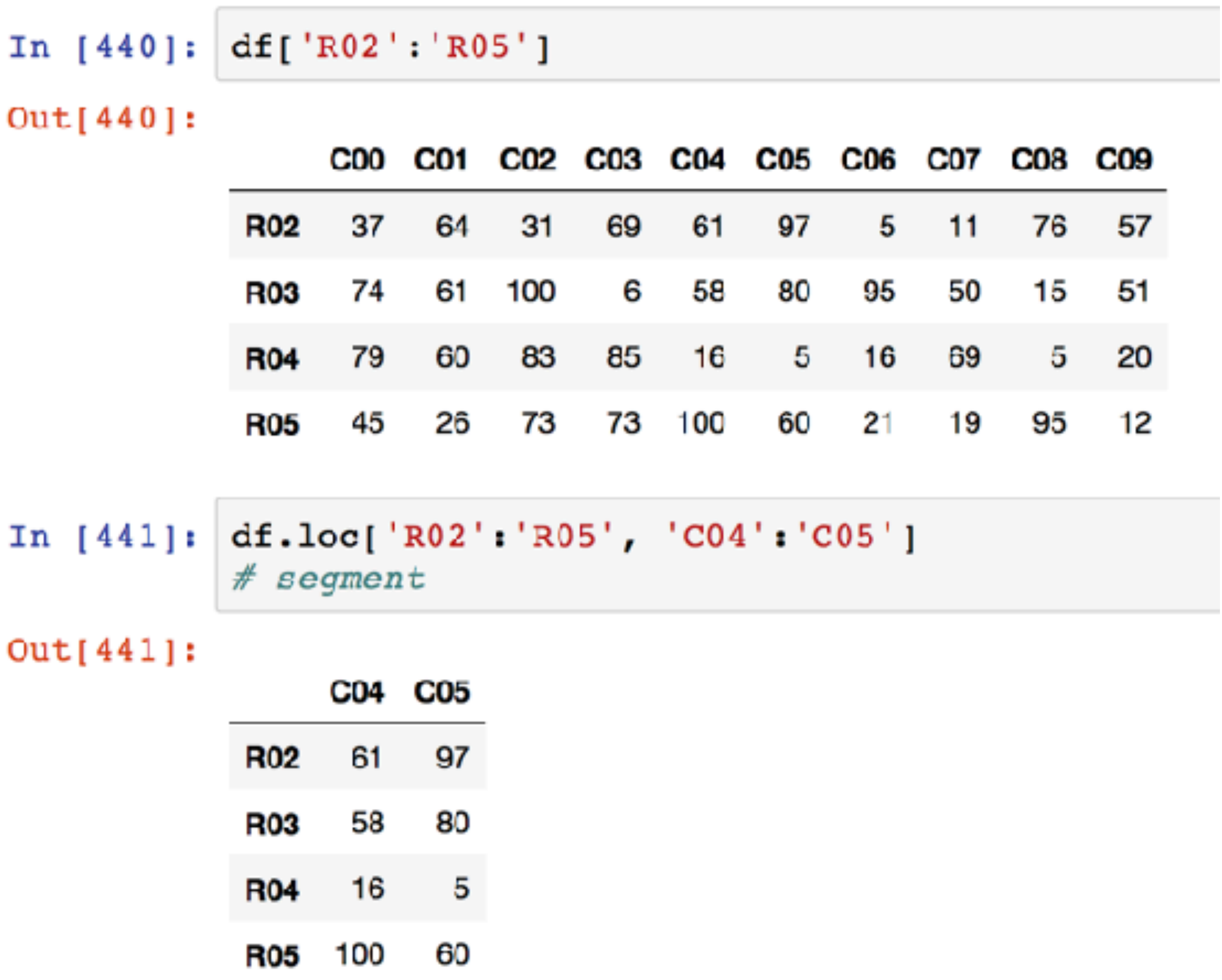
```
In [434]: df
```

Out[434]:

	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	19	21	99	35	59	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25

```
In [439]: df['C05']
```

```
Out[439]: R00    59
          R01    41
          R02    97
          R03    80
          R04     5
          R05    60
          R06    68
          R07    25
          R08    87
          R09    12
          Name: C05, dtype: int64
```



```
In [440]: df['R02':'R05']
```

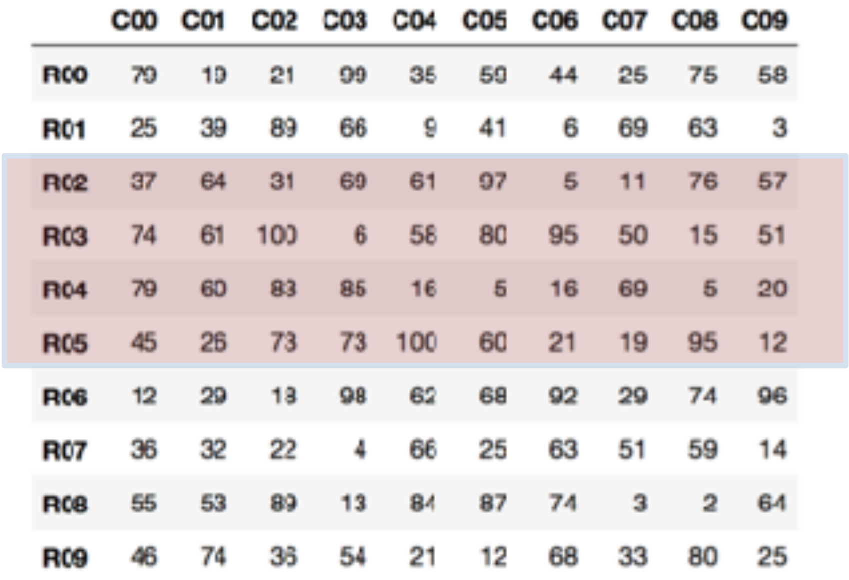
Out[440]:

	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12

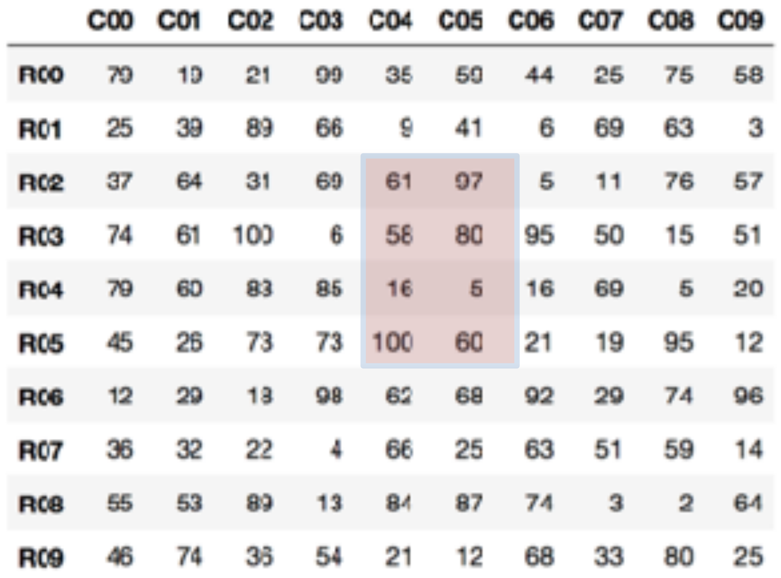
```
In [441]: df.loc['R02':'R05', 'C04':'C05']
          # segment
```

Out[441]:

	C04	C05
R02	61	97
R03	58	80
R04	16	5
R05	100	60

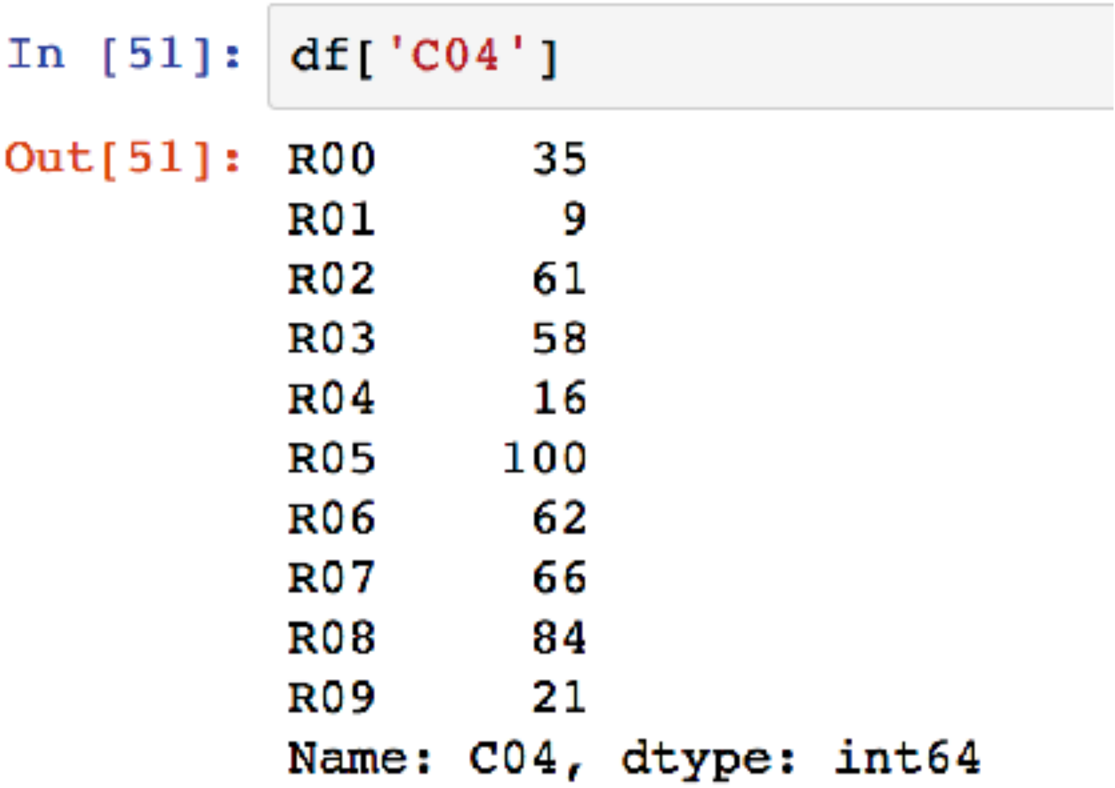


	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	10	21	99	35	50	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25

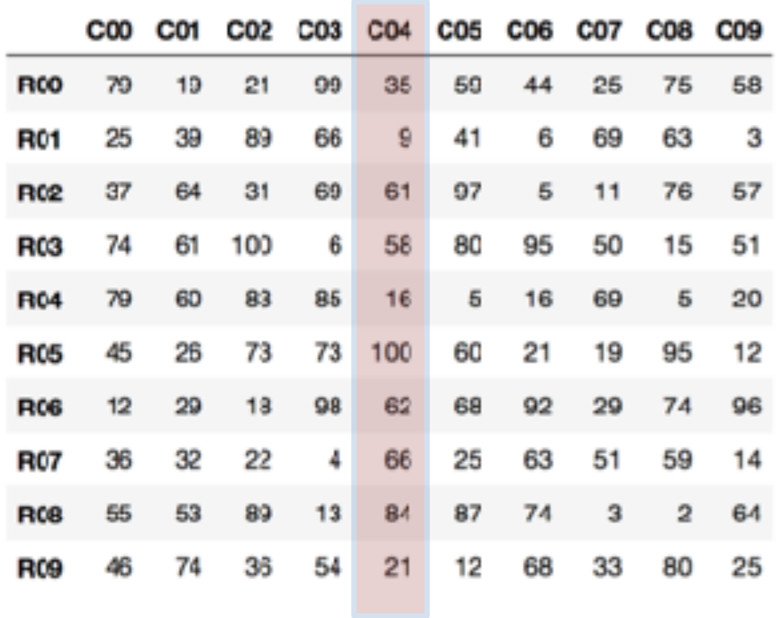


	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	10	21	99	35	50	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25

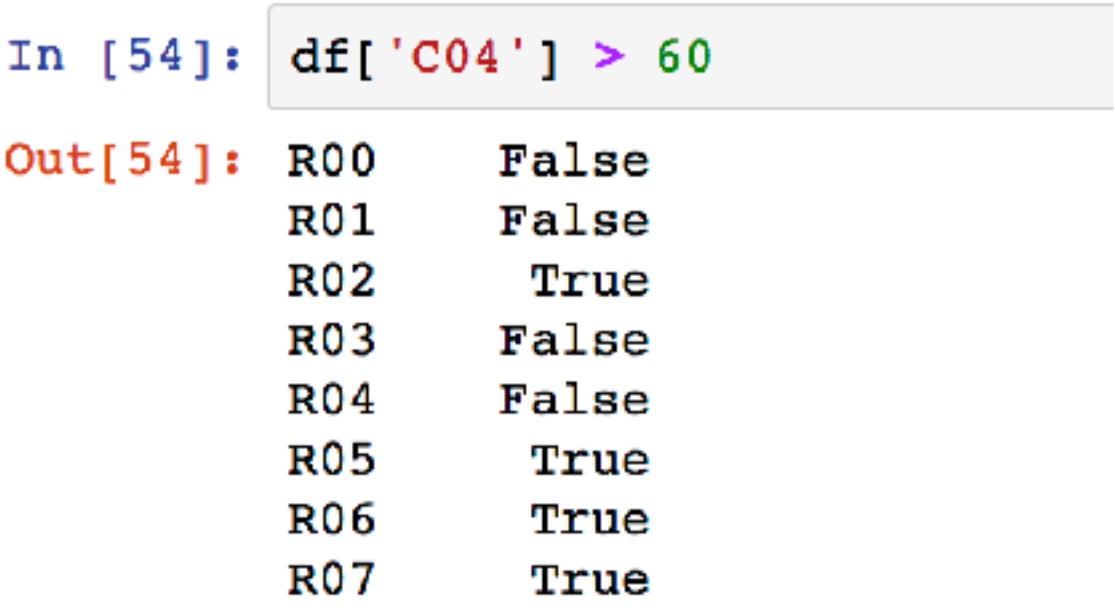
Boolean Indexing



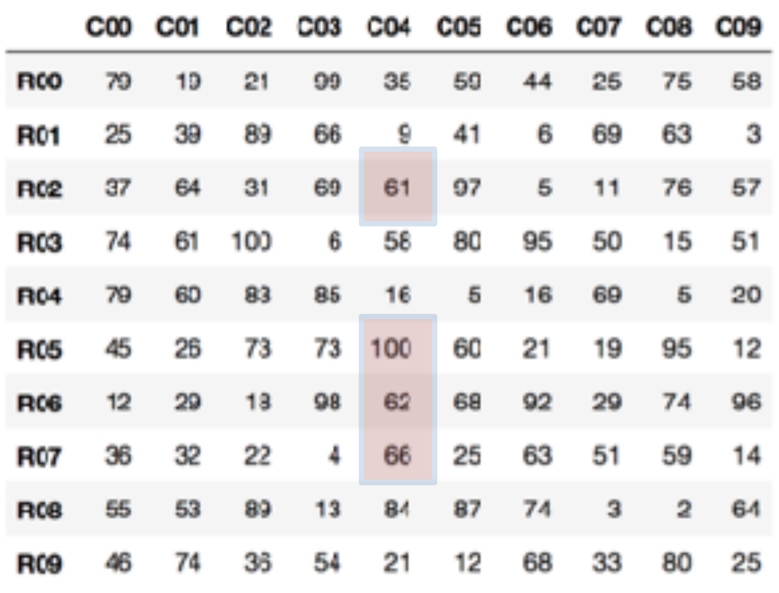
```
In [51]: df['C04']

Out[51]: R00     35
         R01      9
         R02     61
         R03     58
         R04     16
         R05    100
         R06     62
         R07     66
         R08     84
         R09     21
         Name: C04, dtype: int64
```



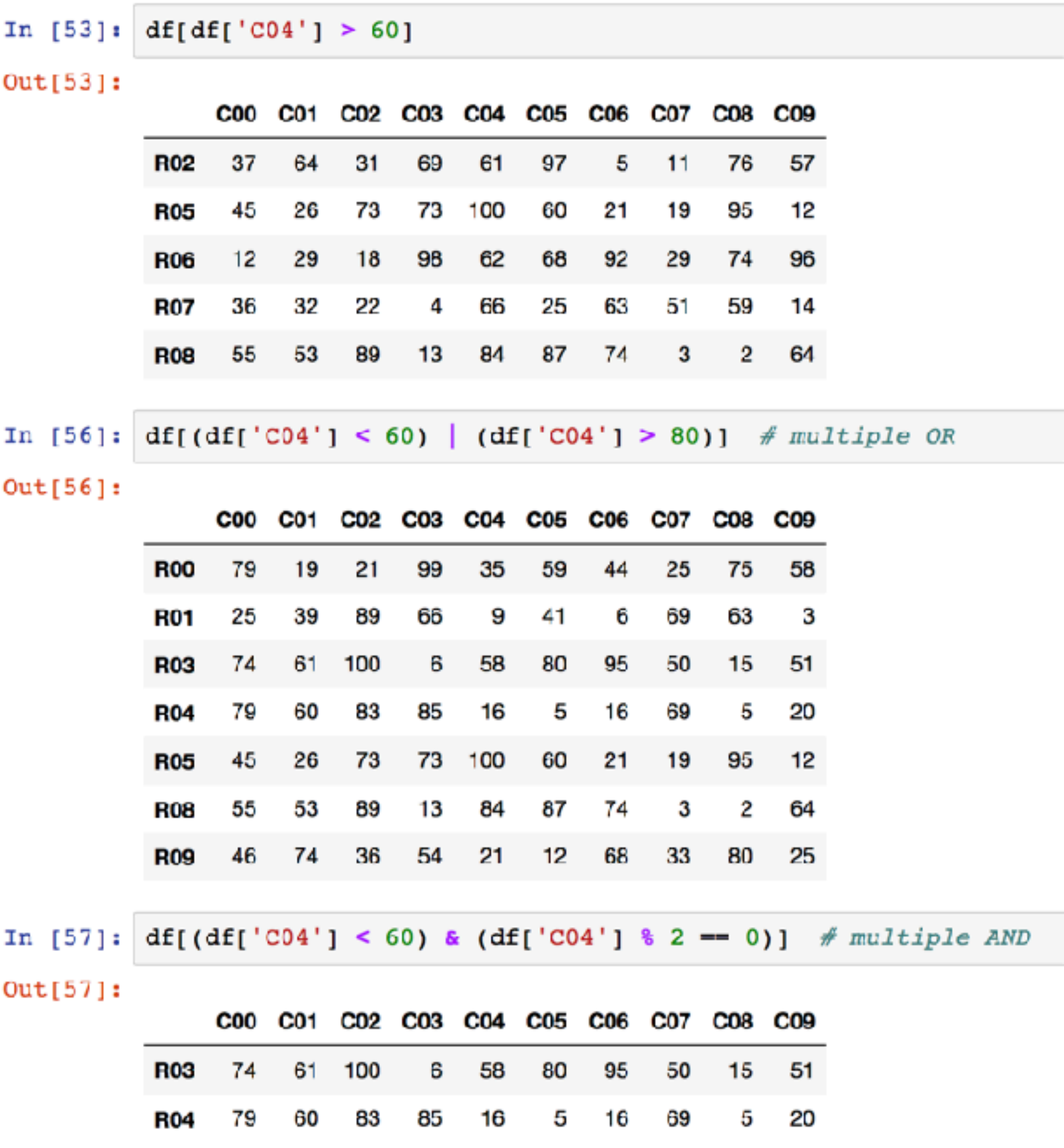
	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	70	10	21	99	35	50	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25



```
In [54]: df['C04'] > 60

Out[54]: R00    False
         R01    False
         R02     True
         R03    False
         R04    False
         R05     True
         R06     True
         R07     True
```



	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	70	10	21	99	35	50	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25



In [53]:
```
df[df['C04'] > 60]
```

Out[53]:

	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R02	37	64	31	69	61	97	5	11	76	57
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64

In [56]:
```
df[(df['C04'] < 60) | (df['C04'] > 80)]  # multiple OR
```

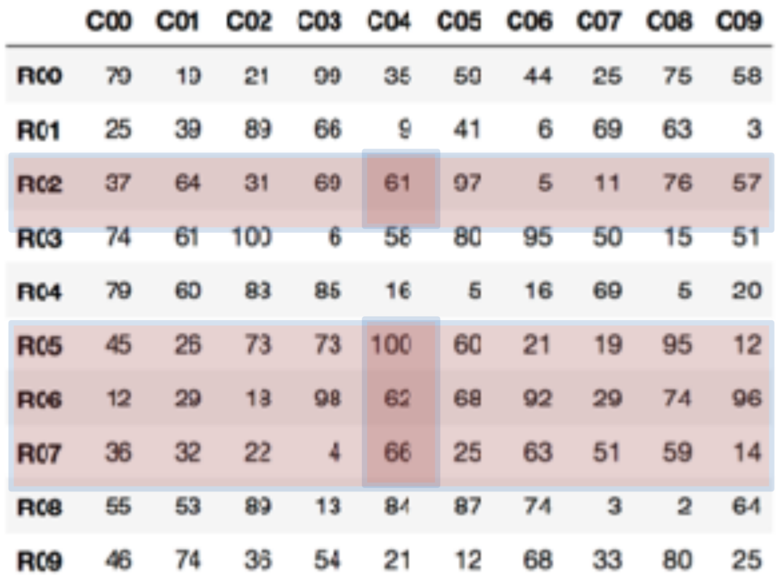
Out[56]:

	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	19	21	99	35	59	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25

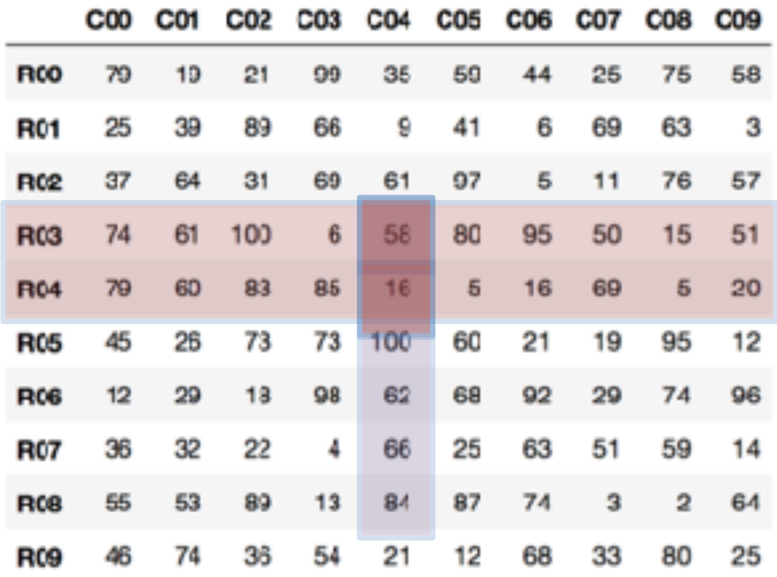
In [57]:
```
df[(df['C04'] < 60) & (df['C04'] % 2 == 0)]  # multiple AND
```

Out[57]:

	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20



	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	19	21	99	35	59	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25



	C00	C01	C02	C03	C04	C05	C06	C07	C08	C09
R00	79	19	21	99	35	59	44	25	75	58
R01	25	39	89	66	9	41	6	69	63	3
R02	37	64	31	69	61	97	5	11	76	57
R03	74	61	100	6	58	80	95	50	15	51
R04	79	60	83	85	16	5	16	69	5	20
R05	45	26	73	73	100	60	21	19	95	12
R06	12	29	18	98	62	68	92	29	74	96
R07	36	32	22	4	66	25	63	51	59	14
R08	55	53	89	13	84	87	74	3	2	64
R09	46	74	36	54	21	12	68	33	80	25

Data selection & Indexing

- **Data-Series / Data-Frames**
- **Boolean Indexing**
- **.iloc** (*integer*loc) **& .loc**
  - .ix: .loc with fallback to .iloc - DEPRECATED
- **axis**
- **Selection returns copy of DataFrame**

Operations

- Adding and removing Series
- Remember: Broadcasting in NumPy
- Adding / subtracting / multiplying & dividing
- .apply()
- .map()
- Changing the data type
- Working with NaN

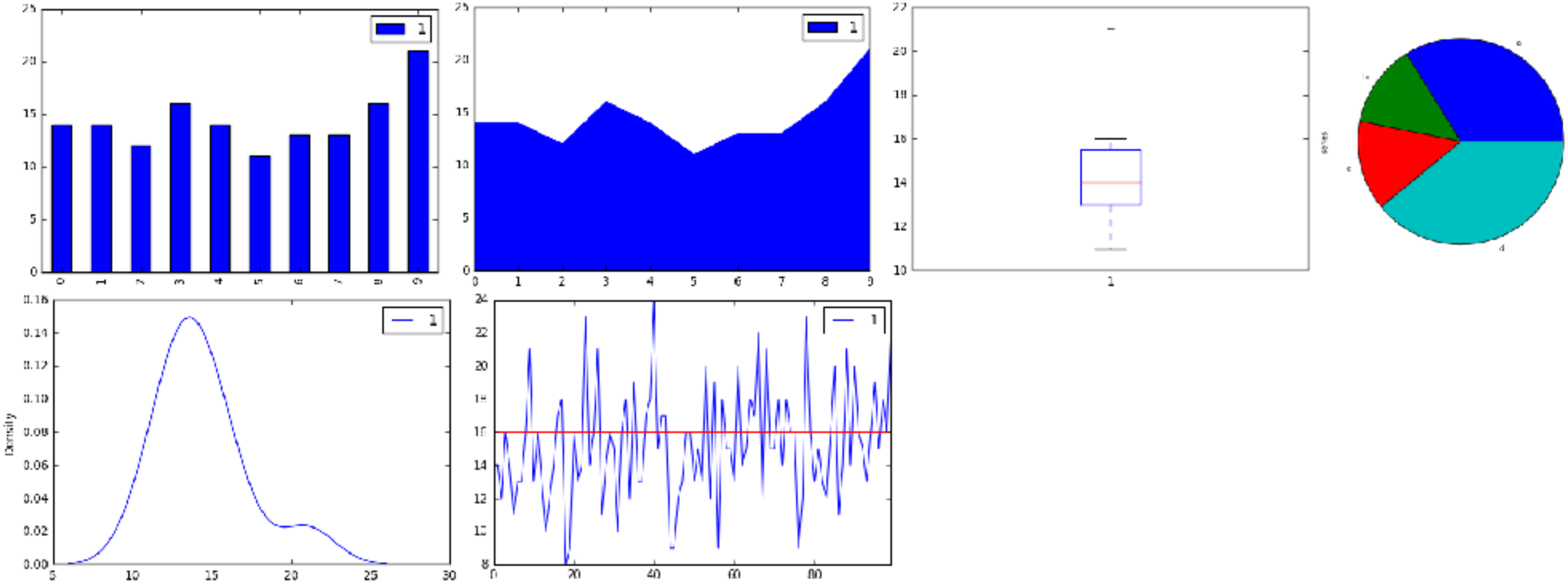
NaN Values & Replacing

- NaN is representation of `null` values
- series`.describe()` ignore NaN
- NaNs:
  - remove `drop()`
  - replace with default
  - forward- or backwards-fill, interpolate

Modifying Series/DataFrames

- Methods applied to Series or DataFrames **do not change** them, but **return** the result as Series or DataFrames
- With parameter `inplace` the result can be deployed directly into Series / DataFrames
- Series can be removed from DF with `drop()`

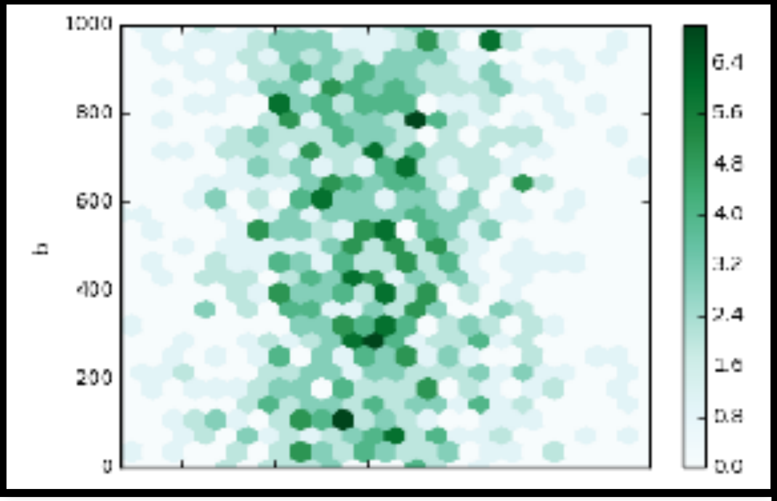
df.plot(kind='…')

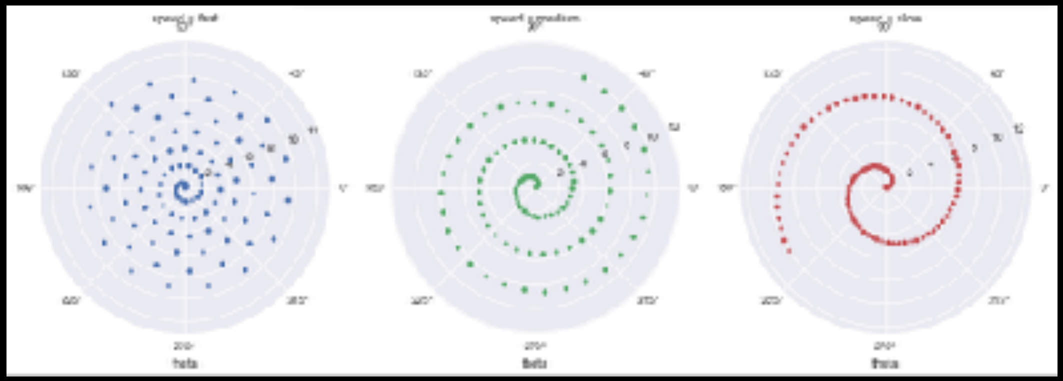


1
0
5
10
15
20
25
0
1
2
3
4
5
6
7
8
9
1
10
12
14
16
18
20
22
1
series
a
b
c
d
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
5
10
15
20
25
30
1
8
10
12
14
16
18
20
22
24
0
20
40
60
80
1

Visualisation

- matplotlib (http://matplotlib.org) integrated, `.plot()`
- custom- and extendable, `plot()` returns `ax`
- Bar-, Area-, Scatter-, Boxplots u.a.





- *Alternatives*:

  Bokeh (http://bokeh.pydata.org/en/latest/)

  Seaborn (https://stanford.edu/~mwaskom/software/seaborn/index.html)

KŌNIGSWEG

Alexander C. S. Hendorf

ah@koenigsweg.com
🐦 @hendorf