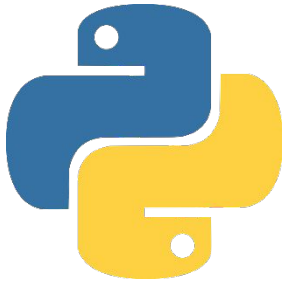


Section XIV: Библиотека pandas

~~10 минут~~ 100 минут до pandas…

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Section XIV: Библиотека pandas

Причем тут панды?

> *[pandas] is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. — Wikipedia*

Section XIV: Библиотека pandas

Устанавливаем и стартуем!

```
conda install pandas
```

```
pip install pandas
```

- Getting Started
  https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html#getting-started
- 10 minutes to pandas
  https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
- Cookbook
  https://pandas.pydata.org/pandas-docs/stable/user_guide/cookbook.html#cookbook
- User Guide
  https://pandas.pydata.org/pandas-docs/stable/user_guide/index.html#user-guide
- Documentation (3071 стр.)
  https://pandas.pydata.org/pandas-docs/stable/pandas.pdf

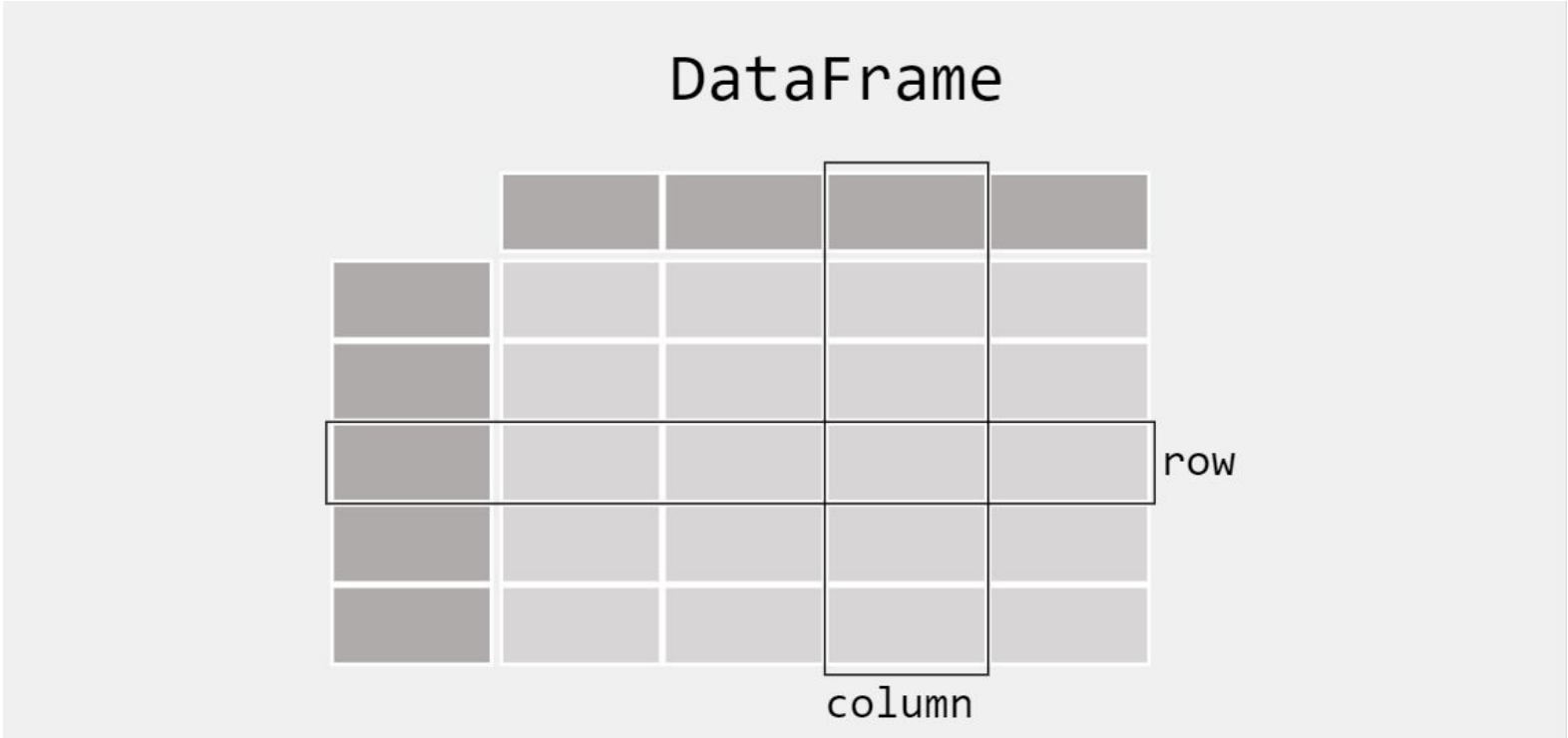
Section XIV: Библиотека pandas

Краткое введение https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html#intro-to-pandas

Какими данными оперирует pandas?

`pandas.DataFrame` – класс – таблица данн

Уже знакомы с SELECT, GROUP BY и JOIN? Большинство SQL операций имеют аналоги в pandas.



Section XIV: Библиотека pandas

Как считывать и сохранять табличные данные?

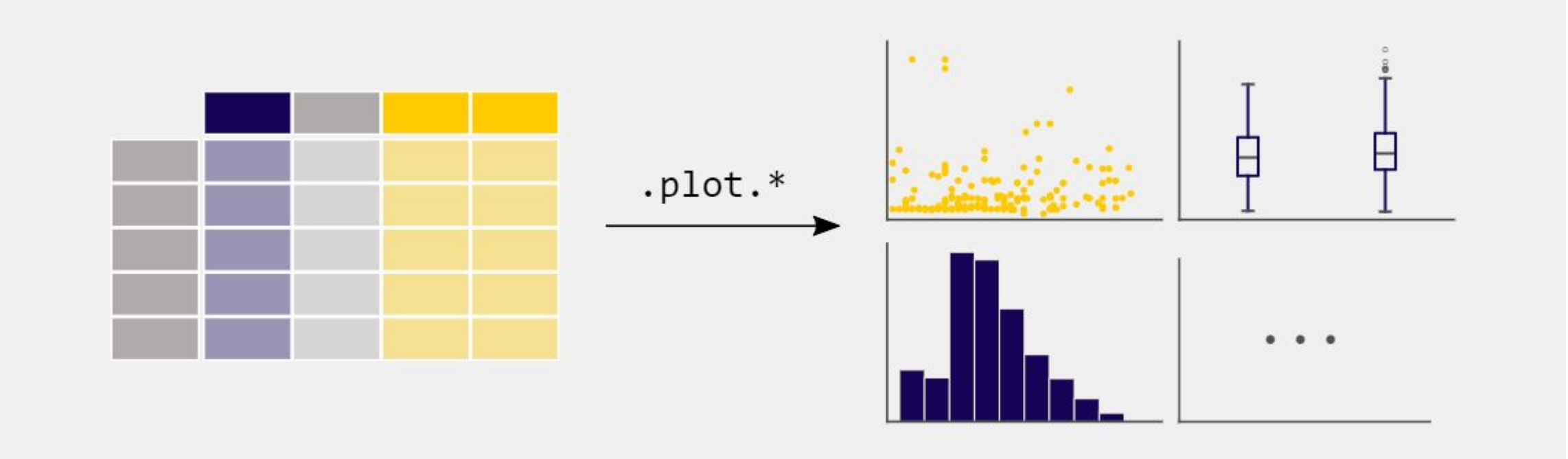
Section XIV: Библиотека pandas

Как выбрать подмножество табличных данных?

Выбор или фильтрация определенных строк и / или столбцов? Фильтрация данных по условию?
Методы нарезки (slicing), выбора (selecting) и извлечения (extracting) нужных вам данных доступны в pandas!

Section XIV: Библиотека pandas

Как выбрать подмножество табличных данных?



Вся сила `matplotlib` под капотом! Но о `matplotlib` чуточку позже…

Section XIV: Библиотека pandas

Как создать новые столбцы, основываясь на существующих?

Никаких циклов! Столбцы являются объектами `pandas.Series` – одномерные индексированные массивы для хранения данных. Обработка данных столбцов осуществляется поэлементно.

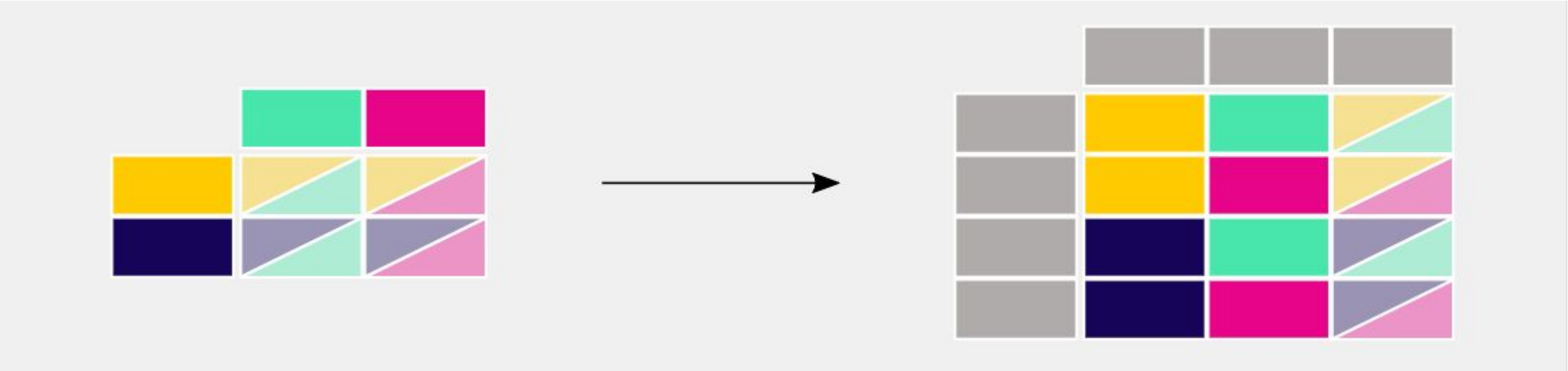
Section XIV: Библиотека pandas

Как вычислить статистики?

Основные статистические данные (mean, median, min, max, counts…) легко поддаются вычислению. Эти или пользовательские агрегации могут быть применены ко всему набору данных, скользящему окну данных или сгруппированы по категориям.

Section XIV: Библиотека pandas

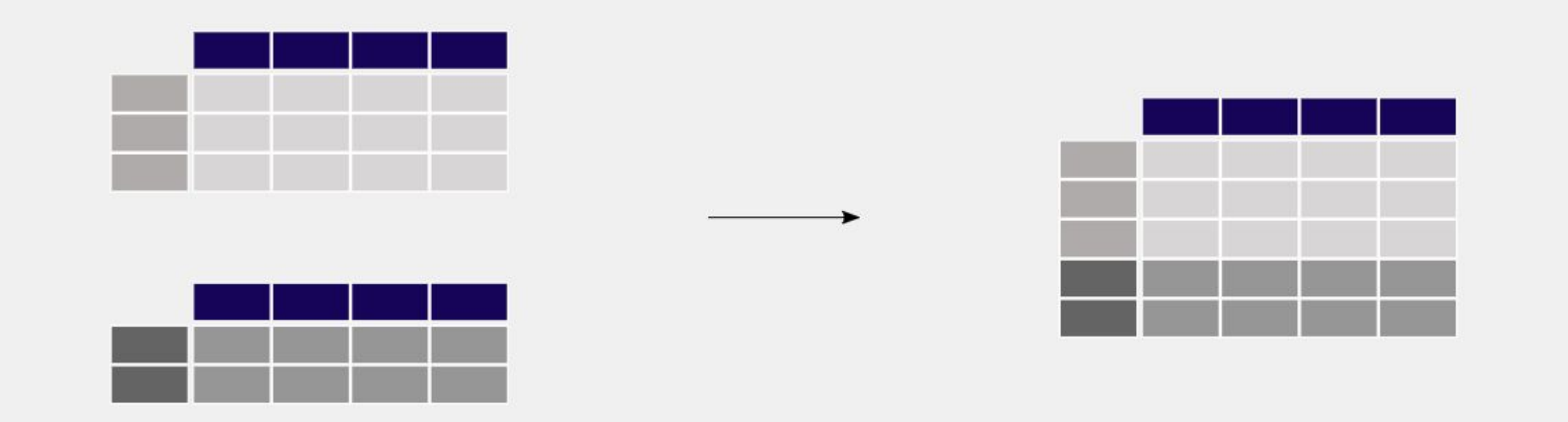
Как изменить структуру таблицы?



Изменить структуру табличных данных можно несколькими способами. А встроенные агрегационные функции позволяют создавать сводные таблицы всего за одну команду.

Section XIV: Библиотека pandas

Как объединить данные из нескольких таблиц?



Несколько таблиц могут быть объединены как по столбцам, так и по строкам, а для объединения нескольких таблиц данных предусмотрены операции соединения и слияния, подобные базам данных.

Section XIV: Библиотека pandas

Как обрабатывать данные временных рядов?

Pandas обладает большой поддержкой временных рядов и имеет обширный набор инструментов для работы с датами, временем и индексированными по времени данными.

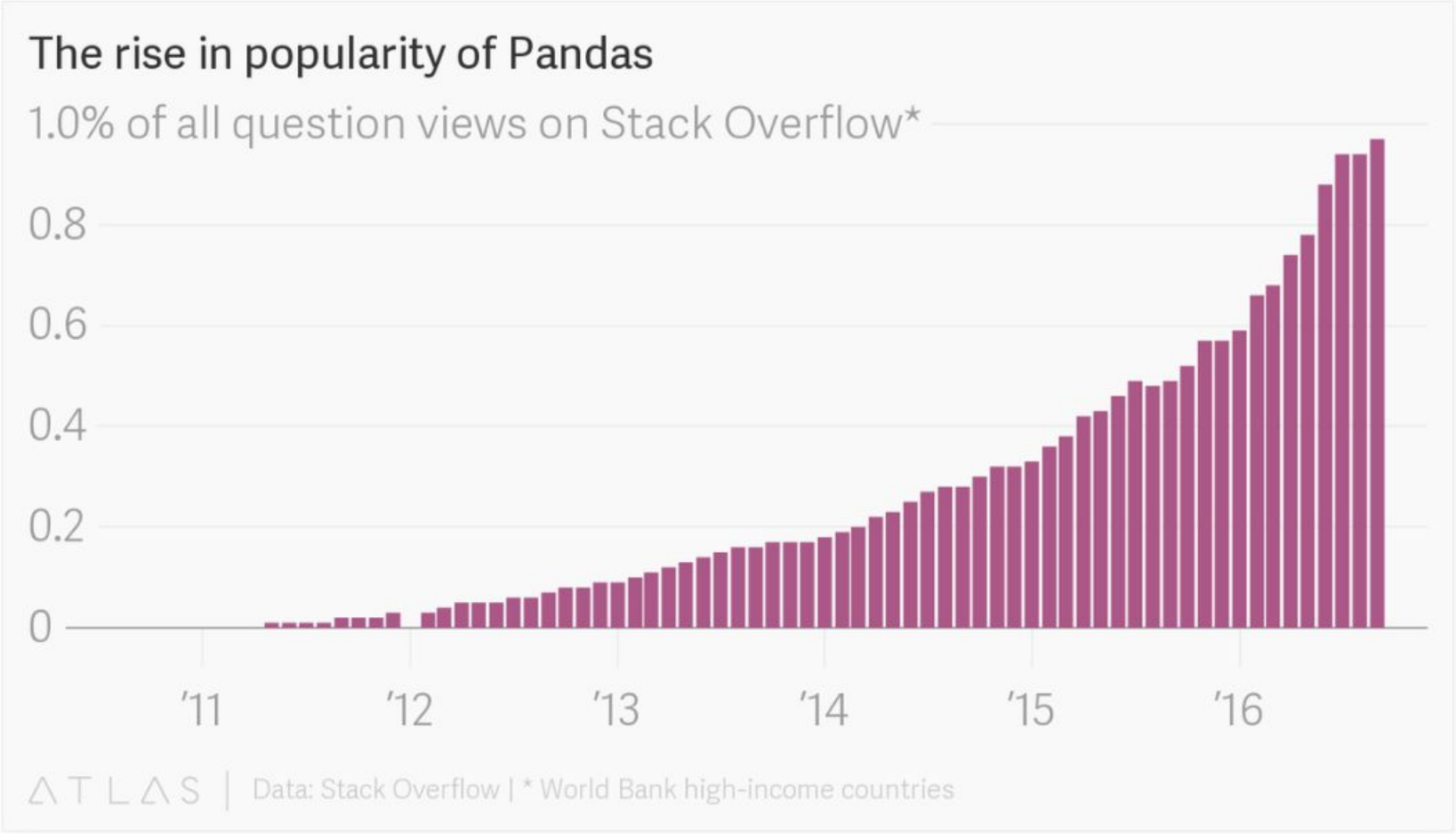
Как обрабатывать текстовые данные?

Наборы данных содержат не только числовые данные. Pandas предоставляет широкий спектр функций для очистки текстовых данных и извлечения из них полезной информации.

И другое… https://pandas.pydata.org/pandas-docs/stable/getting_started/index.html

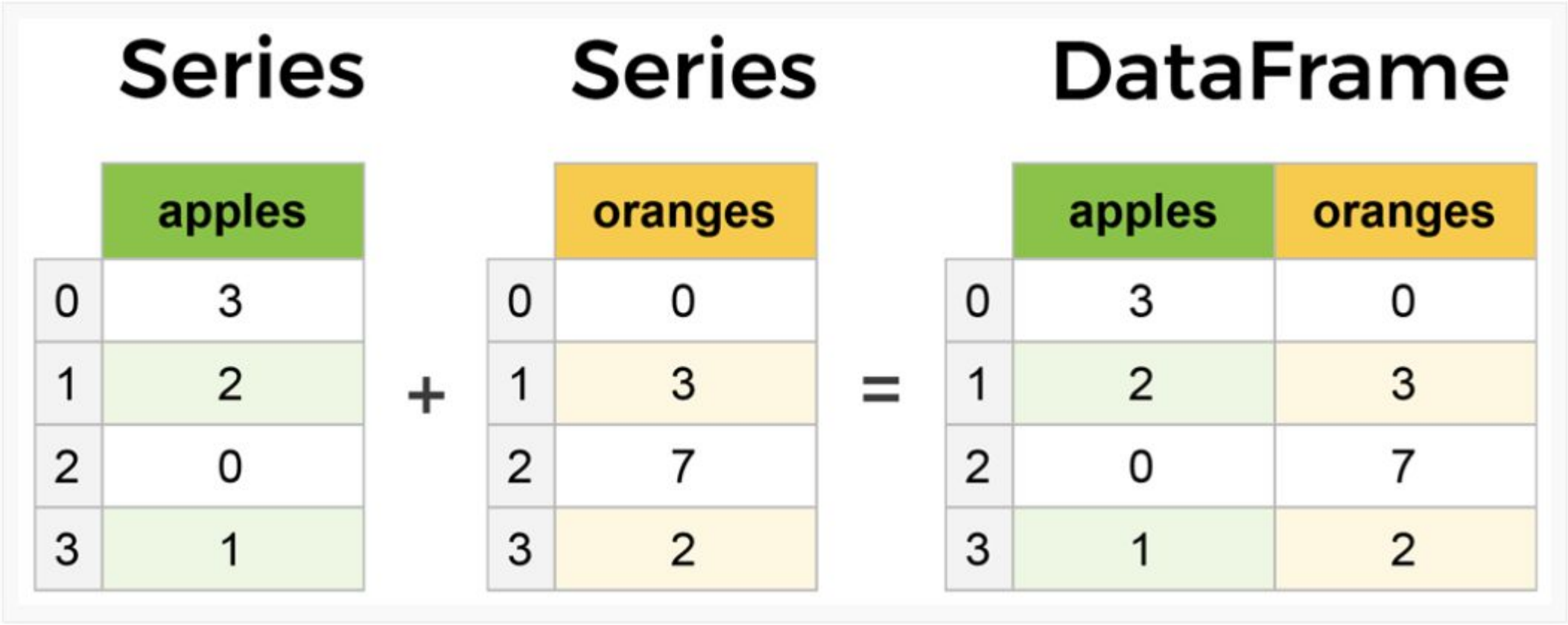
Section XIV: Библиотека pandas

Популярность



Section XIV: Библиотека pandas

Основные компоненты



Section XIV: Библиотека pandas

Создание объектов: `Series`

```
In [1]: import numpy as np

In [2]: import pandas as pd
```

```
>>> s = pd.Series(data, index=index)
```

```
In [6]: pd.Series(np.random.randn(5))
Out[6]:
0   -0.173215
1    0.119209
2   -1.044236
3   -0.861849
4   -2.104569
dtype: float64
```

data может быть:

- Python dict
- ndarray
- scalar value

Section XIV: Библиотека pandas

Создание объектов: `Series`

```
In [3]: s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])

In [4]: s
Out[4]:
a    0.469112
b   -0.282863
c   -1.509059
d   -1.135632
e    1.212112
dtype: float64

In [5]: s.index
Out[5]: Index(['a', 'b', 'c', 'd', 'e'], dtype='object')
```

Индекс может иметь повторяющиеся значения, но с последствиями.

Section XIV: Библиотека pandas

Создание объектов: `Series`

```
In [7]: d = {'b': 1, 'a': 0, 'c': 2}

In [8]: pd.Series(d)
Out[8]:
b    1
a    0
c    2
dtype: int64
```

```
In [12]: pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])
Out[12]:
a    5.0
b    5.0
c    5.0
d    5.0
e    5.0
dtype: float64
```

```
In [9]: d = {'a': 0., 'b': 1., 'c': 2.}

In [10]: pd.Series(d)
Out[10]:
a    0.0
b    1.0
c    2.0
dtype: float64

In [11]: pd.Series(d, index=['b', 'c', 'd', 'a'])
Out[11]:
b    1.0
c    2.0
d    NaN
a    0.0
dtype: float64
```

NaN – стандартный маркер отсутствующих данных в pandas.

Section XIV: Библиотека pandas

`Series` – похожи на массивы и словари

```
In [13]: s[0]
Out[13]: 0.4691122999071863

In [14]: s[:3]
Out[14]:
a    0.469112
b   -0.282863
c   -1.509059
dtype: float64

In [15]: s[s > s.median()]
Out[15]:
a    0.469112
e    1.212112
dtype: float64
```

```
In [16]: s[[4, 3, 1]]
Out[16]:
e    1.212112
d   -1.135632
b   -0.282863
dtype: float64

In [17]: np.exp(s)
Out[17]:
a    1.598575
b    0.753623
c    0.221118
d    0.321219
e    3.360575
dtype: float64
```

```
In [21]: s['a']
Out[21]: 0.4691122999071863

In [22]: s['e'] = 12.

In [23]: s
Out[23]:
a     0.469112
b    -0.282863
c    -1.509059
d    -1.135632
e    12.000000
dtype: float64

In [24]: 'e' in s
Out[24]: True

In [25]: 'f' in s
Out[25]: False
```

```
In [28]: s + s
Out[28]:
a     0.938225
b    -0.565727
c    -3.018117
d    -2.271265
e    24.000000
dtype: float64

In [29]: s * 2
Out[29]:
a     0.938225
b    -0.565727
c    -3.018117
d    -2.271265
e    24.000000
dtype: float64

In [30]: np.exp(s)
Out[30]:
a         1.598575
b         0.753623
c         0.221118
d         0.321219
e    162754.791419
dtype: float64
```

Section XIV: Библиотека pandas

Создание объектов: `DataFrame`

`DataFrame` – 2-dim размеченная структура данных (таблица, SQL таблица или словарь Series объектов), столбцы которой могут иметь разные типы.

Входные данные:

- словарь 1-dim массивов ndarray, листов, словарей или объектов Series
- 2-dim numpy.ndarray массив
- Объект Series
- Другой DataFrame

Вместе с данными можно дополнительно передать аргументы index (метки строк) и columns (метки столбцов). Если вы передаете индекс и / или столбцы, вы явно определяете индекс и / или столбцы результирующего фрейма данных. Таким образом, словарь объектов Series + определенный индекс отбрасывает все данные, не соответствующие переданному индексу.

Section XIV: Библиотека pandas

Создание объектов: `DataFrame`

```
In [44]: d = {'one': [1., 2., 3., 4.],
   ....:      'two': [4., 3., 2., 1.]}
   ....:

In [45]: pd.DataFrame(d)
Out[45]:
   one  two
0  1.0  4.0
1  2.0  3.0
2  3.0  2.0
3  4.0  1.0

In [46]: pd.DataFrame(d, index=['a', 'b', 'c', 'd'])
Out[46]:
   one  two
a  1.0  4.0
b  2.0  3.0
c  3.0  2.0
d  4.0  1.0
```

```
In [37]: d = {'one': pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
   ....:      'two': pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
   ....:

In [38]: df = pd.DataFrame(d)

In [39]: df
Out[39]:
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0

In [40]: pd.DataFrame(d, index=['d', 'b', 'a'])
Out[40]:
   one  two
d  NaN  4.0
b  2.0  2.0
a  1.0  1.0

In [41]: pd.DataFrame(d, index=['d', 'b', 'a'], columns=['two', 'three'])
Out[41]:
   two three
d  4.0   NaN
b  2.0   NaN
a  1.0   NaN
```

Section XIV: Библиотека pandas

Создание объектов: `DataFrame`

```
In [5]: dates = pd.date_range('20130101', periods=6)

In [6]: dates
Out[6]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

In [8]: df
Out[8]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

Section XIV: Библиотека pandas

Создание объектов: `DataFrame`

```
In [9]: df2 = pd.DataFrame({'A': 1.,
   ...:                     'B': pd.Timestamp('20130102'),
   ...:                     'C': pd.Series(1, index=list(range(4)), dtype='float32'),
   ...:                     'D': np.array([3] * 4, dtype='int32'),
   ...:                     'E': pd.Categorical(["test", "train", "test", "train"]),
   ...:                     'F': 'foo'})
   ...:

In [10]: df2
Out[10]:
     A          B    C  D      E    F
0  1.0 2013-01-02  1.0  3   test  foo
1  1.0 2013-01-02  1.0  3  train  foo
2  1.0 2013-01-02  1.0  3   test  foo
3  1.0 2013-01-02  1.0  3  train  foo
```

```
In [11]: df2.dtypes
Out[11]:
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

И множество других опций, детали по ссылке:

https://pandas.pydata.org/pandas-docs/stable/getting_started/dsintro.html#dsintro

Section XIV: Библиотека pandas

Просмотр данных

```
In [13]: df.head()
Out[13]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401

In [14]: df.tail(3)
Out[14]:
                   A         B         C         D
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
```

```
In [15]: df.index
Out[15]:
DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
               '2013-01-05', '2013-01-06'],
              dtype='datetime64[ns]', freq='D')

In [16]: df.columns
Out[16]: Index(['A', 'B', 'C', 'D'], dtype='object')
```

Зачем работать с pandas если есть `to_numpy()`?

```
In [17]: df.to_numpy()
Out[17]:
array([[ 0.4691, -0.2829, -1.5091, -1.1356],
       [ 1.2121, -0.1732,  0.1192, -1.0442],
       [-0.8618, -2.1046, -0.4949,  1.0718],
       [ 0.7216, -0.7068, -1.0396,  0.2719],
       [-0.425 ,  0.567 ,  0.2762, -1.0874],
       [-0.6737,  0.1136, -1.4784,  0.525 ]])
```

```
In [18]: df2.to_numpy()
Out[18]:
array([[1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), 1.0, 3, 'train', 'foo']],
      dtype=object)
```

← Дорогая конвертация!

Section XIV: Библиотека pandas

Просмотр данных – статистики

```
In [19]: df.describe()
Out[19]:
              A         B         C         D
count  6.000000  6.000000  6.000000  6.000000
mean   0.073711 -0.431125 -0.687758 -0.233103
std    0.843157  0.922818  0.779887  0.973118
min   -0.861849 -2.104569 -1.509059 -1.135632
25%   -0.611510 -0.600794 -1.368714 -1.076610
50%    0.022070 -0.228039 -0.767252 -0.386188
75%    0.658444  0.041933 -0.034326  0.461706
max    1.212112  0.567020  0.276232  1.071804
```

```
In [21]: df.sort_index(axis=1, ascending=False)
Out[21]:
                   D         C         B         A
2013-01-01 -1.135632 -1.509059 -0.282863  0.469112
2013-01-02 -1.044236  0.119209 -0.173215  1.212112
2013-01-03  1.071804 -0.494929 -2.104569 -0.861849
2013-01-04  0.271860 -1.039575 -0.706771  0.721555
2013-01-05 -1.087401  0.276232  0.567020 -0.424972
2013-01-06  0.524988 -1.478427  0.113648 -0.673690
```

```
In [20]: df.T
Out[20]:
   2013-01-01  2013-01-02  2013-01-03  2013-01-04  2013-01-05  2013-01-06
A    0.469112    1.212112   -0.861849    0.721555   -0.424972   -0.673690
B   -0.282863   -0.173215   -2.104569   -0.706771    0.567020    0.113648
C   -1.509059    0.119209   -0.494929   -1.039575    0.276232   -1.478427
D   -1.135632   -1.044236    1.071804    0.271860   -1.087401    0.524988
```

```
In [22]: df.sort_values(by='B')
Out[22]:
                   A         B         C         D
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-06 -0.673690  0.113648 -1.478427  0.524988
2013-01-05 -0.424972  0.567020  0.276232 -1.087401
```

Section XIV: Библиотека pandas

Data Selection: нативный доступ

NOTE: В то время как стандартные выражения Python / Numpy для отбора данных интуитивно понятны и удобны для интерактивной работы, для продакшн кода рекомендуется использовать оптимизированные методы доступа к данным pandas: `.at`, `.iat`, `.loc` и `.iloc`.

```
In [23]: df['A']
Out[23]:
2013-01-01    0.469112
2013-01-02    1.212112
2013-01-03   -0.861849
2013-01-04    0.721555
2013-01-05   -0.424972
2013-01-06   -0.673690
Freq: D, Name: A, dtype: float64
```

Выбор одного столбца

```
In [24]: df[0:3]
Out[24]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804

In [25]: df['20130102':'20130104']
Out[25]:
                   A         B         C         D
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
```

Выбор слайса рядов

Section XIV: Библиотека pandas

Data Selection: выбор по лейблу

```
In [26]: df.loc[dates[0]]
Out[26]:
A    0.469112
B   -0.282863
C   -1.509059
D   -1.135632
Name: 2013-01-01 00:00:00, dtype: float64
```

```
In [29]: df.loc['20130102', ['A', 'B']]
Out[29]:
A    1.212112
B   -0.173215
Name: 2013-01-02 00:00:00, dtype: float64
```

```
In [28]: df.loc['20130102':'20130104', ['A', 'B']]
Out[28]:
                   A         B
2013-01-02  1.212112 -0.173215
2013-01-03 -0.861849 -2.104569
2013-01-04  0.721555 -0.706771
```

```
In [27]: df.loc[:, ['A', 'B']]
Out[27]:
                   A         B
2013-01-01  0.469112 -0.282863
2013-01-02  1.212112 -0.173215
2013-01-03 -0.861849 -2.104569
2013-01-04  0.721555 -0.706771
2013-01-05 -0.424972  0.567020
2013-01-06 -0.673690  0.113648
```

```
In [30]: df.loc[dates[0], 'A']
Out[30]: 0.4691122999071863
```

```
In [31]: df.at[dates[0], 'A']
Out[31]: 0.4691122999071863
```

Section XIV: Библиотека pandas

Data Selection: выбор по позиции

```
In [32]: df.iloc[3]
Out[32]:
A    0.721555
B   -0.706771
C   -1.039575
D    0.271860
Name: 2013-01-04 00:00:00, dtype: float64
```

```
In [33]: df.iloc[3:5, 0:2]
Out[33]:
                   A         B
2013-01-04  0.721555 -0.706771
2013-01-05 -0.424972  0.567020
```

```
In [34]: df.iloc[[1, 2, 4], [0, 2]]
Out[34]:
                   A         C
2013-01-02  1.212112  0.119209
2013-01-03 -0.861849 -0.494929
2013-01-05 -0.424972  0.276232
```

```
In [35]: df.iloc[1:3, :]
Out[35]:
                   A         B         C         D
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804
```

```
In [36]: df.iloc[:, 1:3]
Out[36]:
                   B         C
2013-01-01 -0.282863 -1.509059
2013-01-02 -0.173215  0.119209
2013-01-03 -2.104569 -0.494929
2013-01-04 -0.706771 -1.039575
2013-01-05  0.567020  0.276232
2013-01-06  0.113648 -1.478427
```

```
In [37]: df.iloc[1, 1]
Out[37]: -0.17321464905330858
```

```
In [38]: df.iat[1, 1]
Out[38]: -0.17321464905330858
```

Section XIV: Библиотека pandas

Data Selection: Boolean индексация

```
In [39]: df[df['A'] > 0]
Out[39]:
                   A         B         C         D
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632
2013-01-02  1.212112 -0.173215  0.119209 -1.044236
2013-01-04  0.721555 -0.706771 -1.039575  0.271860
```

```
In [40]: df[df > 0]
Out[40]:
                   A         B         C         D
2013-01-01  0.469112       NaN       NaN       NaN
2013-01-02  1.212112       NaN  0.119209       NaN
2013-01-03       NaN       NaN       NaN  1.071804
2013-01-04  0.721555       NaN       NaN  0.271860
2013-01-05       NaN  0.567020  0.276232       NaN
2013-01-06       NaN  0.113648       NaN  0.524988
```

```
In [41]: df2 = df.copy()

In [42]: df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']

In [43]: df2
Out[43]:
                   A         B         C         D      E
2013-01-01  0.469112 -0.282863 -1.509059 -1.135632    one
2013-01-02  1.212112 -0.173215  0.119209 -1.044236    one
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804    two
2013-01-04  0.721555 -0.706771 -1.039575  0.271860  three
2013-01-05 -0.424972  0.567020  0.276232 -1.087401   four
2013-01-06 -0.673690  0.113648 -1.478427  0.524988  three

In [44]: df2[df2['E'].isin(['two', 'four'])]
Out[44]:
                   A         B         C         D     E
2013-01-03 -0.861849 -2.104569 -0.494929  1.071804   two
2013-01-05 -0.424972  0.567020  0.276232 -1.087401  four
```

Section XIV: Библиотека pandas

Присваивание значений

Добавление нового столбца автоматически сопоставляет данные по индексам:

```
In [45]: s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date_range('20130102', periods=6))

In [46]: s1
Out[46]:
2013-01-02    1
2013-01-03    2
2013-01-04    3
2013-01-05    4
2013-01-06    5
2013-01-07    6
Freq: D, dtype: int64

In [47]: df['F'] = s1
```

```
In [51]: df
Out[51]:
                   A         B         C  D    F
2013-01-01  0.000000  0.000000 -1.509059  5  NaN
2013-01-02  1.212112 -0.173215  0.119209  5  1.0
2013-01-03 -0.861849 -2.104569 -0.494929  5  2.0
2013-01-04  0.721555 -0.706771 -1.039575  5  3.0
2013-01-05 -0.424972  0.567020  0.276232  5  4.0
2013-01-06 -0.673690  0.113648 -1.478427  5  5.0
```

```
In [48]: df.at[dates[0], 'A'] = 0
```

```
In [49]: df.iat[0, 1] = 0
```

```
In [50]: df.loc[:, 'D'] = np.array([5] * len(df))
```

Section XIV: Библиотека pandas

Присваивание значений

```
In [52]: df2 = df.copy()

In [53]: df2[df2 > 0] = -df2

In [54]: df2
Out[54]:
                   A         B         C  D    F
2013-01-01  0.000000  0.000000 -1.509059 -5  NaN
2013-01-02 -1.212112 -0.173215 -0.119209 -5 -1.0
2013-01-03 -0.861849 -2.104569 -0.494929 -5 -2.0
2013-01-04 -0.721555 -0.706771 -1.039575 -5 -3.0
2013-01-05 -0.424972 -0.567020 -0.276232 -5 -4.0
2013-01-06 -0.673690 -0.113648 -1.478427 -5 -5.0
```

Больше деталей можно найти в документах:

- Indexing and Selecting Data
- MultiIndex / Advanced Indexing

Section XIV: Библиотека pandas

Обработка отсутствующих данных: переиндексация

```
>>> index = ['Firefox', 'Chrome', 'Safari', 'IE10', 'Konqueror']
>>> df = pd.DataFrame({'http_status': [200, 200, 404, 404, 301],
...                    'response_time': [0.04, 0.02, 0.07, 0.08, 1.0]},
...                   index=index)
>>> df
           http_status  response_time
Firefox            200           0.04
Chrome             200           0.02
Safari             404           0.07
IE10               404           0.08
Konqueror          301           1.00
```

```
>>> new_index = ['Safari', 'Iceweasel', 'Comodo Dragon', 'IE10',
...              'Chrome']
>>> df.reindex(new_index)
              http_status  response_time
Safari              404.0           0.07
Iceweasel             NaN            NaN
Comodo Dragon         NaN            NaN
IE10                404.0           0.08
Chrome              200.0           0.02
```

```
>>> df.reindex(new_index, fill_value='missing')
              http_status response_time
Safari                404          0.07
Iceweasel         missing       missing
Comodo Dragon     missing       missing
IE10                  404          0.08
Chrome                200          0.02
```

Section XIV: Библиотека pandas

Обработка отсутствующих данных: переиндексация

```
In [55]: df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ['E'])

In [56]: df1.loc[dates[0]:dates[1], 'E'] = 1

In [57]: df1
Out[57]:
                   A         B         C  D    F    E
2013-01-01  0.000000  0.000000 -1.509059  5  NaN  1.0
2013-01-02  1.212112 -0.173215  0.119209  5  1.0  1.0
2013-01-03 -0.861849 -2.104569 -0.494929  5  2.0  NaN
2013-01-04  0.721555 -0.706771 -1.039575  5  3.0  NaN
```

Section XIV: Библиотека pandas

Обработка отсутствующих данных: заполнение, исключение

```
In [58]: df1.dropna(how='any')
Out[58]:
                   A         B         C  D    F    E
2013-01-02  1.212112 -0.173215  0.119209  5  1.0  1.0
```

```
In [59]: df1.fillna(value=5)
Out[59]:
                   A         B         C  D    F    E
2013-01-01  0.000000  0.000000 -1.509059  5  5.0  1.0
2013-01-02  1.212112 -0.173215  0.119209  5  1.0  1.0
2013-01-03 -0.861849 -2.104569 -0.494929  5  2.0  5.0
2013-01-04  0.721555 -0.706771 -1.039575  5  3.0  5.0
```

```
In [60]: pd.isna(df1)
Out[60]:
                A      B      C      D      F      E
2013-01-01  False  False  False  False   True  False
2013-01-02  False  False  False  False  False  False
2013-01-03  False  False  False  False  False   True
2013-01-04  False  False  False  False  False   True
```

Section XIV: Библиотека pandas

Применение функций, гистограммирование

```
In [66]: df.apply(np.cumsum)
Out[66]:
                   A         B         C   D     F
2013-01-01  0.000000  0.000000 -1.509059   5   NaN
2013-01-02  1.212112 -0.173215 -1.389850  10   1.0
2013-01-03  0.350263 -2.277784 -1.884779  15   3.0
2013-01-04  1.071818 -2.984555 -2.924354  20   6.0
2013-01-05  0.646846 -2.417535 -2.648122  25  10.0
2013-01-06 -0.026844 -2.303886 -4.126549  30  15.0

In [67]: df.apply(lambda x: x.max() - x.min())
Out[67]:
A    2.073961
B    2.671590
C    1.785291
D    0.000000
F    4.000000
dtype: float64
```

```
In [68]: s = pd.Series(np.random.randint(0, 7, size=10))

In [69]: s
Out[69]:
0    4
1    2
2    1
3    2
4    6
5    4
6    4
7    6
8    4
9    4
dtype: int64

In [70]: s.value_counts()
Out[70]:
4    5
6    2
2    2
1    1
dtype: int64
```

Section XIV: Библиотека pandas

Слияние (merge): конкатенация

```
In [73]: df = pd.DataFrame(np.random.randn(10, 4))

In [74]: df
Out[74]:
          0         1         2         3
0 -0.548702  1.467327 -1.015962 -0.483075
1  1.637550 -1.217659 -0.291519 -1.745505
2 -0.263952  0.991460 -0.919069  0.266046
3 -0.709661  1.669052  1.037882 -1.705775
4 -0.919854 -0.042379  1.247642 -0.009920
5  0.290213  0.495767  0.362949  1.548106
6 -1.131345 -0.089329  0.337863 -0.945867
7 -0.932132  1.956030  0.017587 -0.016692
8 -0.575247  0.254161 -1.143704  0.215897
9  1.193555 -0.077118 -0.408530 -0.862495

# break it into pieces
In [75]: pieces = [df[:3], df[3:7], df[7:]]
```

```
In [76]: pd.concat(pieces)
Out[76]:
          0         1         2         3
0 -0.548702  1.467327 -1.015962 -0.483075
1  1.637550 -1.217659 -0.291519 -1.745505
2 -0.263952  0.991460 -0.919069  0.266046
3 -0.709661  1.669052  1.037882 -1.705775
4 -0.919854 -0.042379  1.247642 -0.009920
5  0.290213  0.495767  0.362949  1.548106
6 -1.131345 -0.089329  0.337863 -0.945867
7 -0.932132  1.956030  0.017587 -0.016692
8 -0.575247  0.254161 -1.143704  0.215897
9  1.193555 -0.077118 -0.408530 -0.862495
```

Добавление столбца в DataFrame происходит относительно быстро. Однако добавление строки требует копирования и может быть дорогостоящим!

Section XIV: Библиотека pandas

Слияние (merge): объединение

```
In [77]: left = pd.DataFrame({'key': ['foo', 'foo'], 'lval': [1, 2]})

In [78]: right = pd.DataFrame({'key': ['foo', 'foo'], 'rval': [4, 5]})

In [79]: left
Out[79]:
   key  lval
0  foo     1
1  foo     2

In [80]: right
Out[80]:
   key  rval
0  foo     4
1  foo     5

In [81]: pd.merge(left, right, on='key')
Out[81]:
   key  lval  rval
0  foo     1     4
1  foo     1     5
2  foo     2     4
3  foo     2     5
```

SQL стиль слияния

Section XIV: Библиотека pandas

Слияние (merge): объединение

```
In [82]: left = pd.DataFrame({'key': ['foo', 'bar'], 'lval': [1, 2]})

In [83]: right = pd.DataFrame({'key': ['foo', 'bar'], 'rval': [4, 5]})

In [84]: left
Out[84]:
   key  lval
0  foo     1
1  bar     2

In [85]: right
Out[85]:
   key  rval
0  foo     4
1  bar     5

In [86]: pd.merge(left, right, on='key')
Out[86]:
   key  lval  rval
0  foo     1     4
1  bar     2     5
```

Другой пример

Больше информации:

https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html#merging-join

Section XIV: Библиотека pandas

Группировка (grouping)

Процесс группировки подразумевает следующий пайплайн:

- Разделение данных на группы на основе некоторых критериев
- Применение функции к каждой группе независимо
- Объединение результатов в структуру данных

```
In [87]: df = pd.DataFrame({'A': ['foo', 'bar', 'foo', 'bar',
    ....:                          'foo', 'bar', 'foo', 'foo'],
    ....:                    'B': ['one', 'one', 'two', 'three',
    ....:                          'two', 'two', 'one', 'three'],
    ....:                    'C': np.random.randn(8),
    ....:                    'D': np.random.randn(8)})
    ....:
```

```
In [88]: df
Out[88]:
     A      B         C         D
0  foo    one  1.346061 -1.577585
1  bar    one  1.511763  0.396823
2  foo    two  1.627081 -0.105381
3  bar  three -0.990582 -0.532532
4  foo    two -0.441652  1.453749
5  bar    two  1.211526  1.208843
6  foo    one  0.268520 -0.080952
7  foo  three  0.024580 -0.264610
```

Section XIV: Библиотека pandas

Группировка (grouping)

```
In [89]: df.groupby('A').sum()
Out[89]:
            C         D
A
bar  1.732707  1.073134
foo  2.824590 -0.574779
```

```
In [90]: df.groupby(['A', 'B']).sum()
Out[90]:
                C         D
A   B
bar one    1.511763  0.396823
    three -0.990582 -0.532532
    two    1.211526  1.208843
foo one    1.614581 -1.658537
    three  0.024580 -0.264610
    two    1.185429  1.348368
```


Иерархический индекс

Section XIV: Библиотека pandas

Стекинг (Stacking)

```
In [91]: tuples = list(zip(*[['bar', 'bar', 'baz', 'baz',
   ....:                      'foo', 'foo', 'qux', 'qux'],
   ....:                     ['one', 'two', 'one', 'two',
   ....:                      'one', 'two', 'one', 'two']]))
   ....:

In [92]: index = pd.MultiIndex.from_tuples(tuples, names=['first', 'second'])

In [93]: df = pd.DataFrame(np.random.randn(8, 2), index=index, columns=['A', 'B'])

In [94]: df2 = df[:4]

In [95]: df2
Out[95]:
                     A         B
first second
bar   one    -0.727965 -0.589346
      two     0.339969 -0.693205
baz   one    -0.339355  0.593616
      two     0.884345  1.591431
```

Section XIV: Библиотека pandas

Стекинг (Stacking)

```
In [96]: stacked = df2.stack()

In [97]: stacked
Out[97]:
first  second
bar    one     A   -0.727965
               B   -0.589346
       two     A    0.339969
               B   -0.693205
baz    one     A   -0.339355
               B    0.593616
       two     A    0.884345
               B    1.591431
dtype: float64
```

```
In [98]: stacked.unstack()
Out[98]:
                     A         B
first second
bar   one     -0.727965 -0.589346
      two      0.339969 -0.693205
baz   one     -0.339355  0.593616
      two      0.884345  1.591431
```

```
In [99]: stacked.unstack(1)
Out[99]:
second        one       two
first
bar   A -0.727965  0.339969
      B -0.589346 -0.693205
baz   A -0.339355  0.884345
      B  0.593616  1.591431

In [100]: stacked.unstack(0)
Out[100]:
first          bar       baz
second
one    A -0.727965 -0.339355
       B -0.589346  0.593616
two    A  0.339969  0.884345
       B -0.693205  1.591431
```

Section XIV: Библиотека pandas

Сводные таблицы (pivot tables)

```
In [101]: df = pd.DataFrame({'A': ['one', 'one', 'two', 'three'] * 3,
   .....:                    'B': ['A', 'B', 'C'] * 4,
   .....:                    'C': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
   .....:                    'D': np.random.randn(12),
   .....:                    'E': np.random.randn(12)})
   .....:

In [102]: df
Out[102]:
        A  B    C         D         E
0     one  A  foo -1.202872  0.047609
1     one  B  foo -1.814470 -0.136473
2     two  C  foo  1.018601 -0.561757
3   three  A  bar -0.595447 -1.623033
4     one  B  bar  1.395433  0.029399
5     one  C  bar -0.392670 -0.542108
6     two  A  foo  0.007207  0.282696
7   three  B  foo  1.928123 -0.087302
8     one  C  foo -0.055224 -1.575170
9     one  A  bar  2.395985  1.771208
10    two  B  bar  1.552825  0.816482
11  three  C  bar  0.166599  1.100230
```

```
In [103]: pd.pivot_table(df, values='D', index=['A', 'B'], columns=['C'])
Out[103]:
C             bar       foo
A     B
one   A  2.395985 -1.202872
      B  1.395433 -1.814470
      C -0.392670 -0.055224
three A -0.595447       NaN
      B       NaN  1.928123
      C  0.166599       NaN
two   A       NaN  0.007207
      B  1.552825       NaN
      C       NaN  1.018601
```

Section XIV: Библиотека pandas

Plotting (построение графиков)

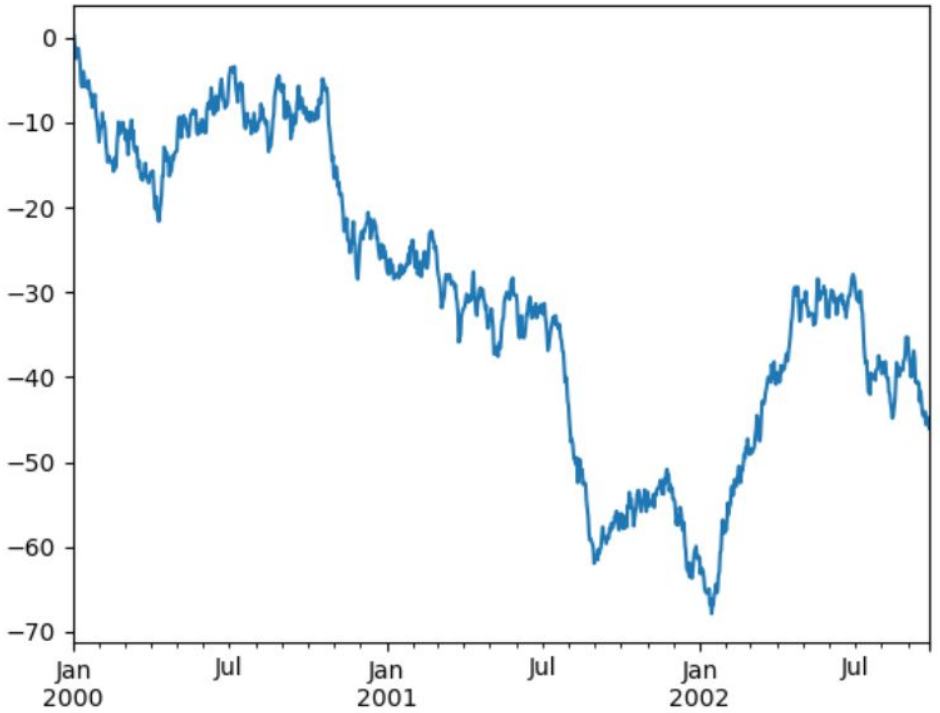
```
In [131]: import matplotlib.pyplot as plt

In [132]: plt.close('all')
```

```
In [133]: ts = pd.Series(np.random.randn(1000),
   .....:                index=pd.date_range('1/1/2000', periods=1000))
   .....:

In [134]: ts = ts.cumsum()

In [135]: ts.plot()
Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3d511cfa10>
```



Section XIV: Библиотека pandas

Plotting (построение графиков)

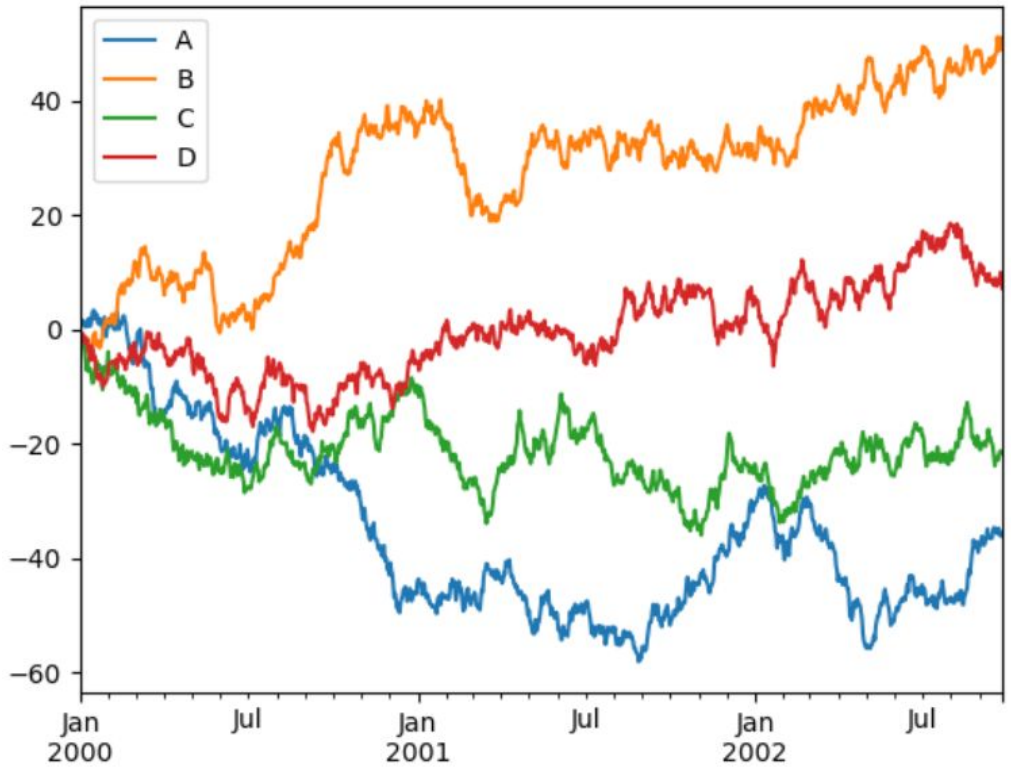
```
In [136]: df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
   .....:                   columns=['A', 'B', 'C', 'D'])
   .....:

In [137]: df = df.cumsum()

In [138]: plt.figure()
Out[138]: <Figure size 640x480 with 0 Axes>

In [139]: df.plot()
Out[139]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3d51028650>

In [140]: plt.legend(loc='best')
Out[140]: <matplotlib.legend.Legend at 0x7f3d50ff5250>
```



Q&A

https://pandas.pydata.org/pandas-docs/stable/index.html