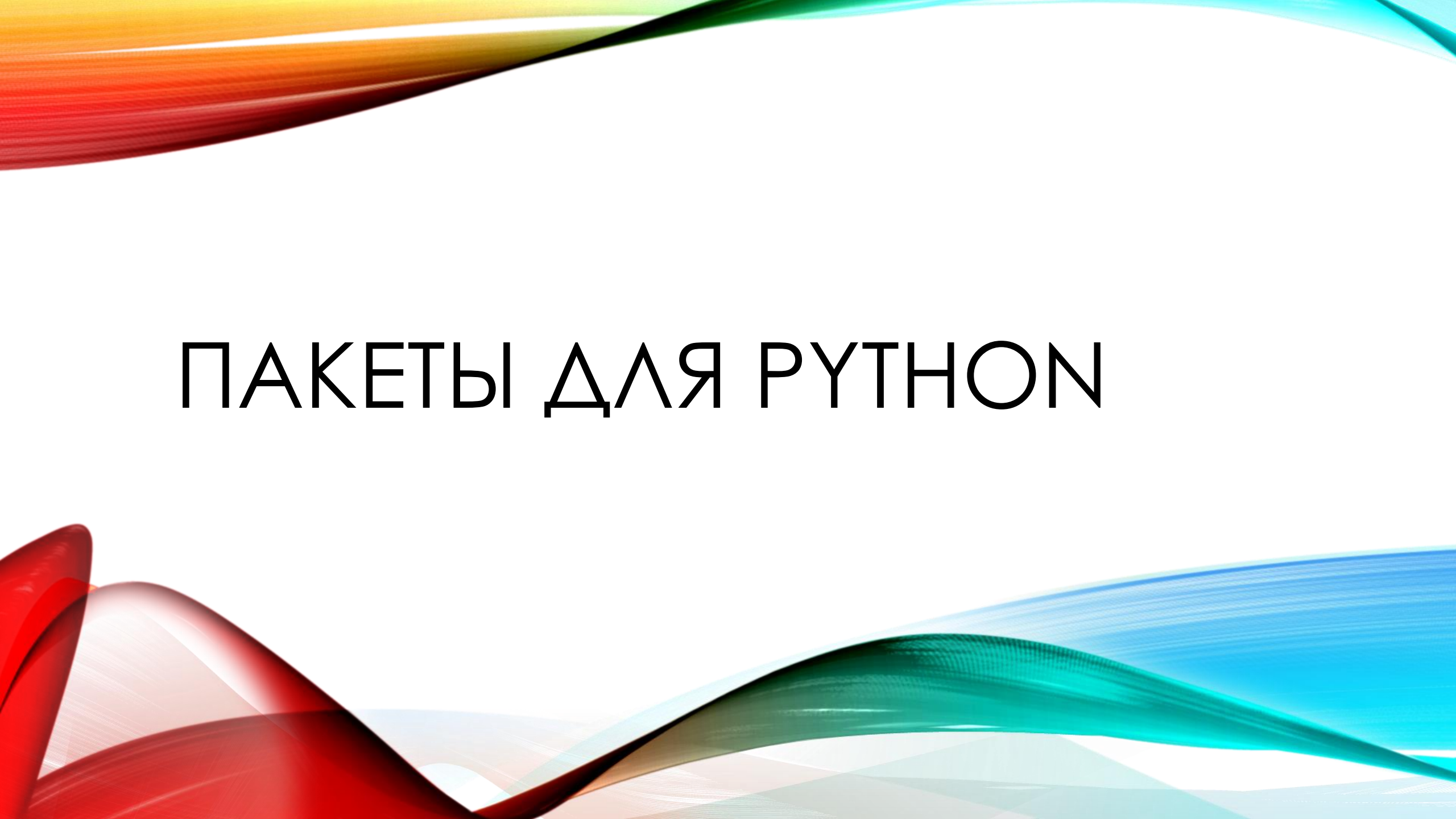




ПАКЕТЫ ДЛЯ PYTHON

ПОНЯТИЕ ПАКЕТА

- **Пакеты Python** — это Py-приложения, дополнения или утилиты, которые можно установить из внешнего репозитория: Github, Bitbucket, Google Code или официального Python Package Index.
- На сервере пакеты хранятся в .zip и .tar архивах, либо в дополнительной упаковке — «яйцах» (.egg, старый формат) или «колесах» (.whl).
- Сценарий установки – setup.py - хранит сведения о зависимостях других пакетах и модулях, без которых пакет работать не будет

АРХИТЕКТУРА ПАКЕТА – ЭТО ПАПКА

Имя	Дата изменения	Тип	Размер
 _internal	03.09.2020 0:19	Папка с файлами	
 _vendor	03.09.2020 0:19	Папка с файлами	
 __init__	11.08.2020 12:42	Python File	1 КБ
 __main__	11.08.2020 12:42	Python File	1 КБ

УСТАНОВКА МЕНЕДЖЕРА ПАКЕТОВ

- **Менеджер пакетов PIP**

Установка

- Скачать скрипт `get_pip.py` в папку с файлом `python.exe`
- Нажать Win+R – cmd
- Набрать команду смены текущей папки

`cd полный_путь\Python37\python.exe get_pip.py`

- Начинается установка менеджера пакетов

- Модуль `easy_install` – установка через `setuptools` (файл `setup.py`)

УСТАНОВКА ПАКЕТА

- Win+R – cmd
- cd полный_путь\Python37\Script\pip3 install <имя пакета>

Обновление пакета pip install имя_пакета -U

Список установленных пакетов pip list

Справка pip help

Удаление пакета pip uninstall имя_пакета

ПОПУЛЯРНЫЕ ПАКЕТЫ (БИБЛИОТЕКИ)

- **NumPy**

вычислительная библиотека (2005-2006)

сложные математические вычисления, многомерные массивы

- **Matplotlib**

двумерные числовые построения

интерактивные графики и схемы

- **SymPy**

символьная математика, алгебраические преобразования



ПОПУЛЯРНЫЕ ПАКЕТЫ

- **Pandas**
обработка и анализ данных
- **Keras**
нейросетевая библиотека, глубокое обучение
- **Pillow**
обработка изображений
- **Pyglet**
разработка игр, работа с мультимедиа



ПОПУЛЯРНЫЕ ПАКЕТЫ

- **Requests**

запросы HTTP

- **TensorFlow**

машинное обучение, работа с графами, нейронные сети

- **SQLAlchemy**

работа с базами данных, sql - запросы

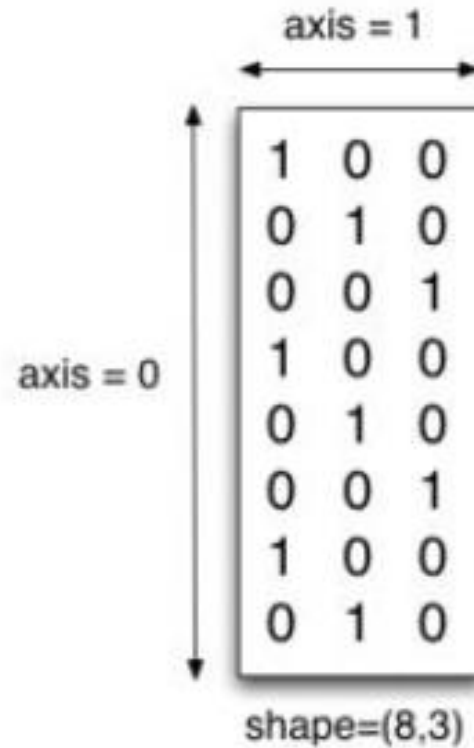


NUMPY

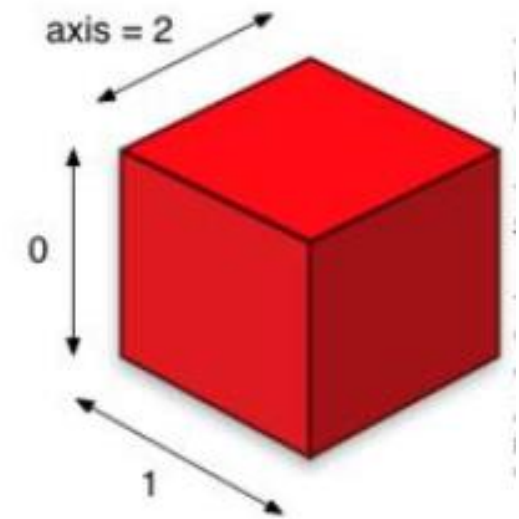
Библиотека для работы с **многомерными массивами**

Все элементы массива принадлежат одному типу данных и последовательно располагаются в ОП

- Выигрыш в скорости, по сравнению со списками
- Код становится компактным и структурированным



ВВЕДЕНИЕ



ТИП ДАННЫХ - ARRAY

- import numpy as np

```
arr = np.array([0, 1, 2, 3, 4, 5,  
6, 7, 8])
```

```
print(arr)
```

```
arr[3:8] = -99
```

```
print(arr)
```

OUTPUT

- [0 1 2 3 4 5 6 7 8]

- [0 1 2 -99 -99 -99 -99 -99 8]

БЫСТРАЯ ПОЭЛЕМЕНТНАЯ ОБРАБОТКА

- `arr = np.arange(5)`
- `print(arr)`
- `print(np.sqrt(arr))`
- `print(np.exp(arr))`

[0 1 2 3 4]

[0. 1. 1.4142 1.7321 2.]

[1. 2.7183 7.3891 20.0855 54.5982]

ПОИСК МАКСИМУМА

- `x = np.random.randn(4)`
- `y = np.random.randn(4)`
- `print(x)`
- `print(y)`
- `print(np.maximum(x,y))`
- `[-0.9691 -1.4411 1.2614 -0.9615]`
- `[-0.0398 -0.0692 -1.6854 -0.3902]`
- `[-0.0398 -0.0692 1.2614 -0.3902]`

СЛОЖЕНИЕ

- `x = np.random.randn(4)`
- `y = np.random.randn(4)`
- `print x`
- `print y`
- `print np.add(x, y)`
- `[ 0.0987 -1.2579 -1.4827 -1.4299]`
- `[-0.2855 -0.7548 -1.0134 0.7546]`
- `[-0.1868 -2.0127 -2.4961 -0.6753]`

$x+y$

WHERE

- `arr = np.random.rand(3,2)`
- `print arr`
- `print np.where(arr > 0.5, 2, -2)`

```
[[ 0.44292516  0.68852318]
 [ 0.52094621  0.98764891]
 [ 0.19472641  0.38844208]]
```

```
[[ -2  2]
 [ 2  2]
 [-2 -2]]
```

МАТЕМАТИЧЕСКИЕ И СТАТИСТИЧЕСКИЕ МЕТОДЫ

- `arr = np.random.randn(2, 2)`
- `print arr`
- `print arr.mean()`
- `print np.mean(arr)`
- `print arr.sum()`

МЕТОДЫ ПО СТРОКАМ И СТОЛБЦАМ

- `arr = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])`

- `print arr` **[[0 1 2]**
- `print arr.mean(axis=0)` **[3 4 5]**
- `print arr.mean(axis=1)` **[6 7 8]**

[3. 4. 5.]

[1. 4. 7.]

МЕТОДЫ ПО СТРОКАМ И СТОЛБЦАМ

- `arr = np.array([[0, 1, 2], [3, 4, 5], [6, 7, 8]])`

- `print arr`

`[[0 1 2]`

- `print arr.sum(0)`

`[3 4 5]`

- `print arr.sum(1)`

`[6 7 8]`

`[ 9 12 15]`

`[ 3 12 21]`

СОПТИРОВАКА

- `arr = randn(4)`
- `print arr`
- `arr.sort()`
- `print arr`
- **OUTPUT**
- `[-0.301 -0.1785 -0.9659 -0.6087]`
- `[-0.9659 -0.6087 -0.301 -0.1785]`

```
arr = randn(2, 3)
```

```
print arr
```

```
arr.sort(0)
```

```
print arr
```

```
arr.sort(1)
```

```
print arr
```

<code>[[1 3 2]</code>	<code>[[1 3 2]</code>	<code>[[1 2 3]</code>
<code>[8 4 9]</code>	<code>[3 4 8]</code>	<code>[3 4 8]</code>
<code>[3 5 8]]</code>	<code>[8 5 9]]</code>	<code>[5 8 9]]</code>

УНИКАЛЬНЫЕ ЗНАЧЕНИЯ

```
names = np.array(['Bob', 'Joe',  
                  'Will', 'Bob', 'Will', 'Joe',  
                  'Joe'])
```

```
print np.unique(names)
```

```
print sorted(set(names))
```

- **OUTPUT**

- ['Bob' 'Joe' 'Will']

- ['Bob', 'Joe', 'Will']

ВКЛЮЧАЕТ

```
values = np.array([6,0,0,3,2,5,6])  
print(np.isin(values, [2,3,6]))
```

```
[ True False False  True  True False  True]
```

РАБОТА С ФАЙЛОМ

- `arr = np.arange(10)`
- `print arr`
- `np.save('some_array', arr)`
- `arr1 = np.load('some_array.npy')`
- `print arr1`

- `arr = np.loadtxt('array_ex.txt', delimiter=',')`

- `print arr`
- `print type(arr)`

array_ex.txt

1,2,3,4

3,4,5,6

OUTPUT

[[1. 2. 3. 4.]

[3. 4. 5. 6.]

<type

'numpy.ndarray'>

СРЕЗЫ

		axis 1		
		0	1	2
axis 0	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2

Expression

`arr[:2, 1:]`

`arr[2]`

`arr[2, :]`

`arr[2:, :]`

`arr[:, :2]`

`arr[1, :2]`

`arr[1:2, :2]`

3d 2x2x2

```
a=np.array([  
    [  
        [3, 1],[4, 3]  
    ],  
    [  
        [2, 4],[3, 3]  
    ]  
])
```

2,4
3,1
3,3
4,3

2,4
3,1
3,3
4,3

a[0][1]

2,4
3,1
3,3
4,3

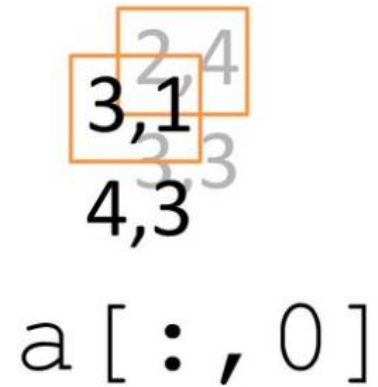
a[1]

2,4
3,1
3,3
4,3

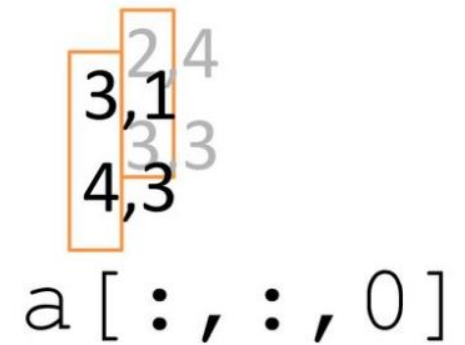
a[0][0][0]

СРЕЗЫ ДЛЯ ТРЕХМЕРНОГО МАССИВА

```
a=np.array([  
    [3, 1], [4, 3]  
],  
    [  
    [2, 4], [3, 3]  
])
```


`a[:, 0]`

```
a=np.array([  
    [3, 1], [4, 3]  
],  
    [  
    [2, 4], [3, 3]  
])
```

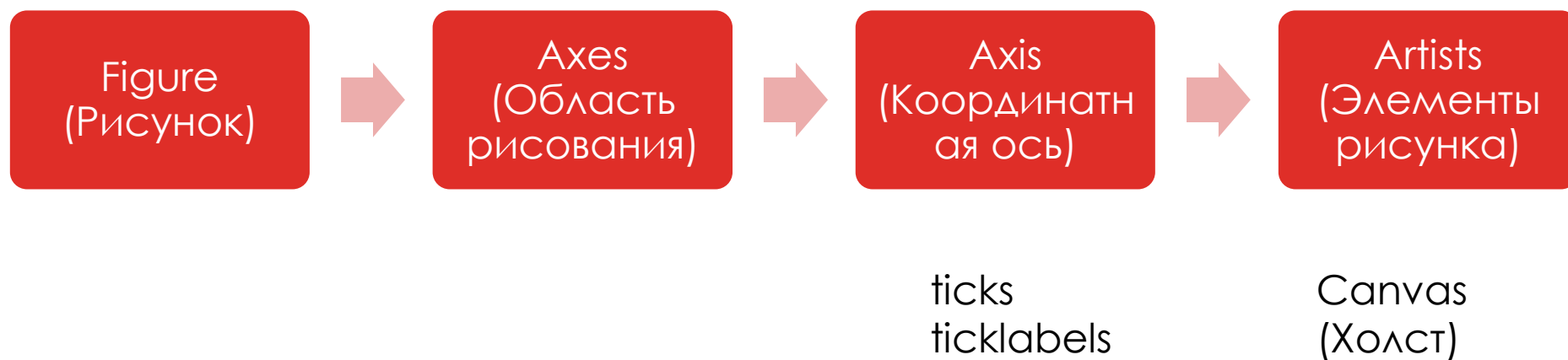

`a[:, :, 0]`

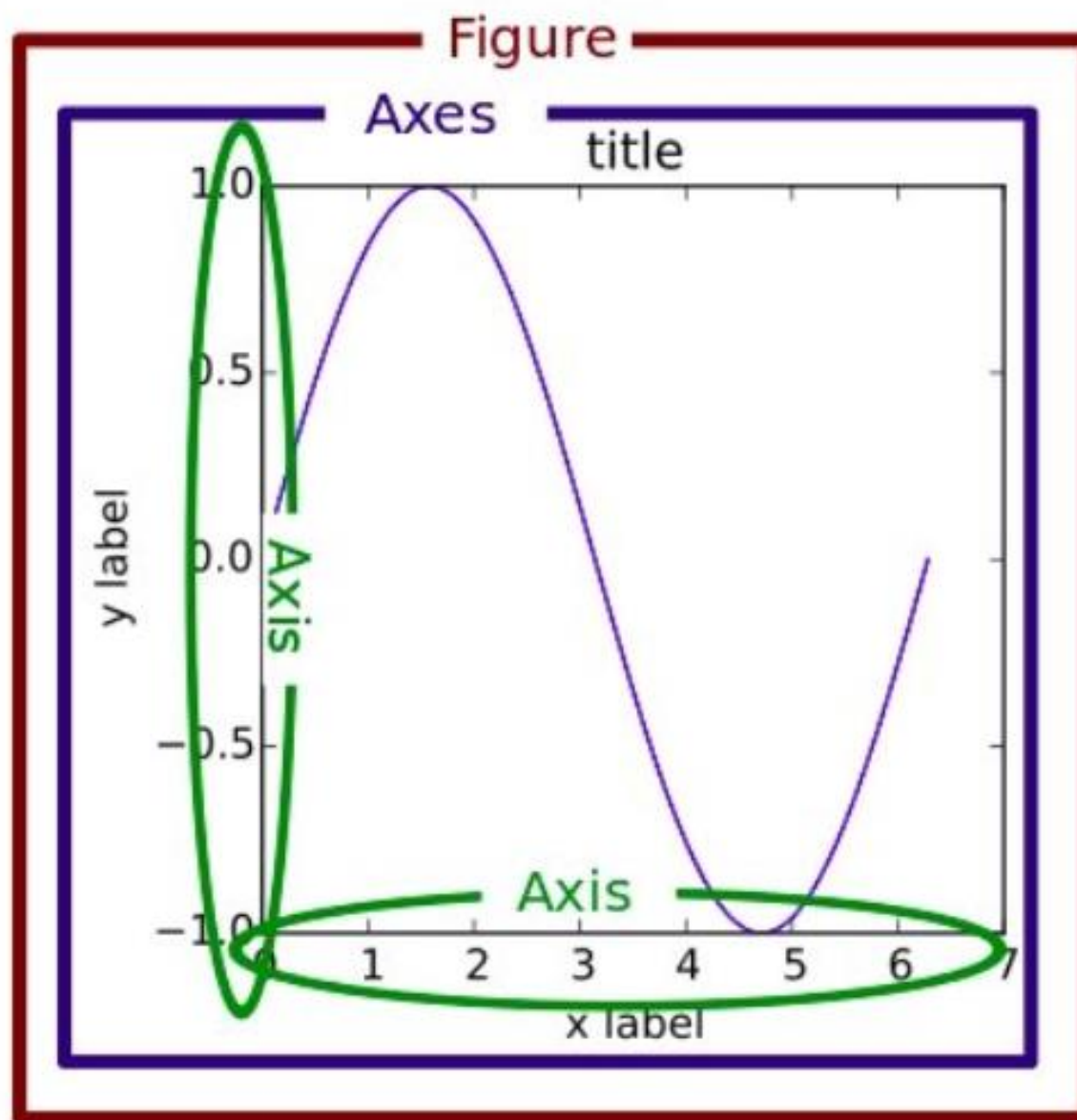


MATPLOTLIB

СТРУКТУРА РИСУНКА

- `import matplotlib as mpl`





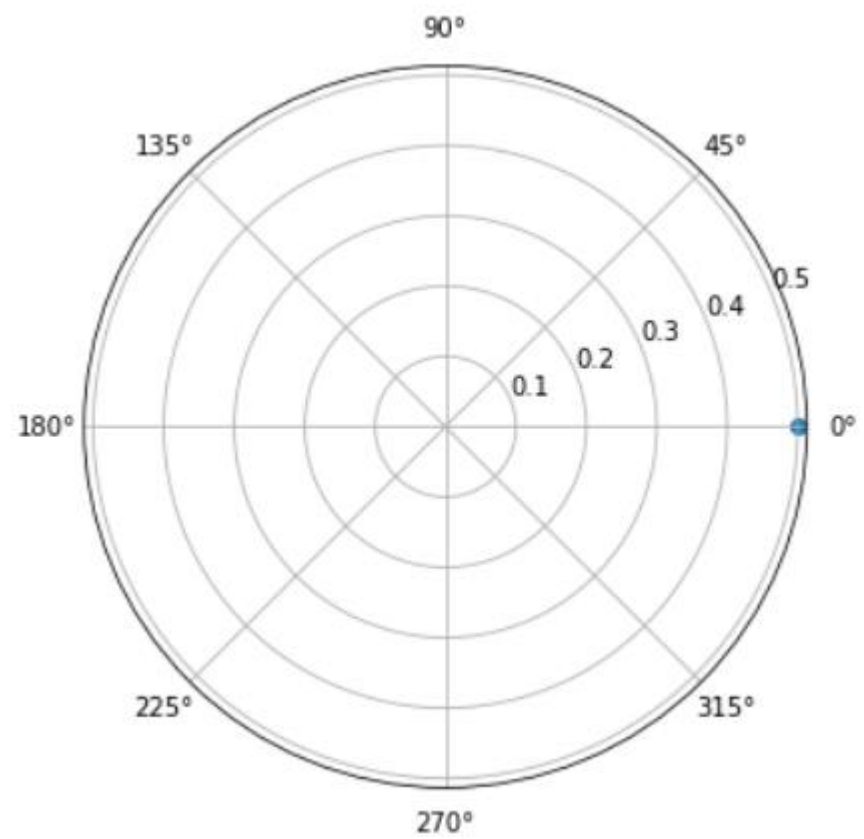
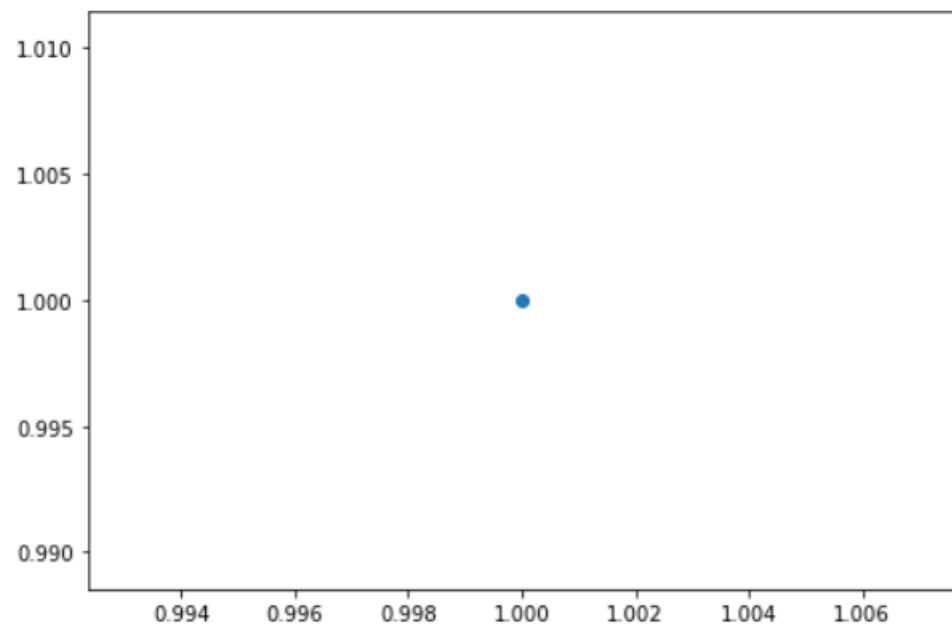


АЛГОРИТМ СОЗДАНИЯ РИСУНКА

- Экземпляр класса Figure
- Области рисования Axes (subplot)
- Возможно настройки координатной сетки, деления и подписей
- Создание графических примитивов

ИНТЕРФЕЙС PYPLOT

- `import matplotlib.pyplot as plt`
- `fig = plt.figure()`
- `ax=fig.add_axes([0,0,1,1])` или `ax = fig.add_axes([0, 0, 1, 1], polar=True)`
- `plt.scatter(1.0, 1.0)`
- `plt.savefig('pic.png', fmt='png')`
- `plt.show()`



1. Область рисования Axes

- Заголовок области рисования -> `plt.title()`;

2. Ось абсцисс Xaxis

- Подпись оси абсцисс OX -> `plt.xlabel()`;

3. Ось абсцисс Yaxis

- Подпись оси абсцисс OY -> `plt.ylabel()`;

4. Легенда -> `plt.legend()`

5. Цветовая шкала -> `plt.colorbar()`

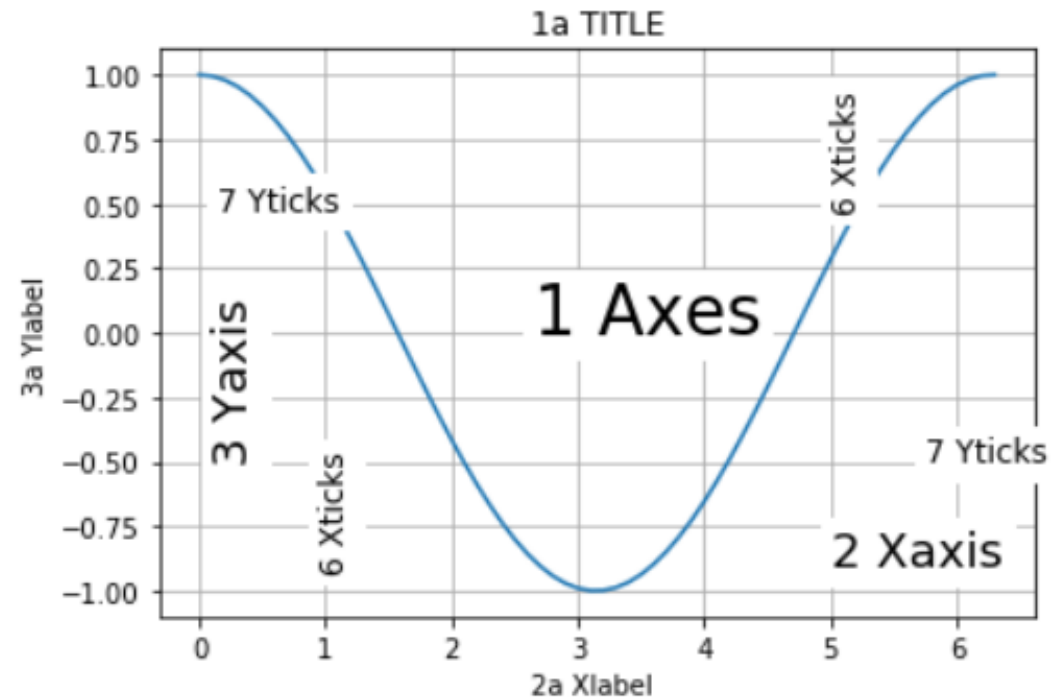
- Подпись горизонтальной оси абсцисс OY -> `cbar.ax.set_xlabel()`;
- Подпись вертикальной оси абсцисс OY -> `cbar.ax.set_ylabel()`;

6. Деления на оси абсцисс OX -> `plt.xticks()`

7. Деления на оси ординат OY -> `plt.yticks()`

```
import matplotlib.pyplot as plt
import numpy as np
lag = 0.1
x = np.arange(0.0, 2*np.pi+lag, lag)
y = np.cos(x)
fig = plt.figure()
plt.plot(x, y)
```

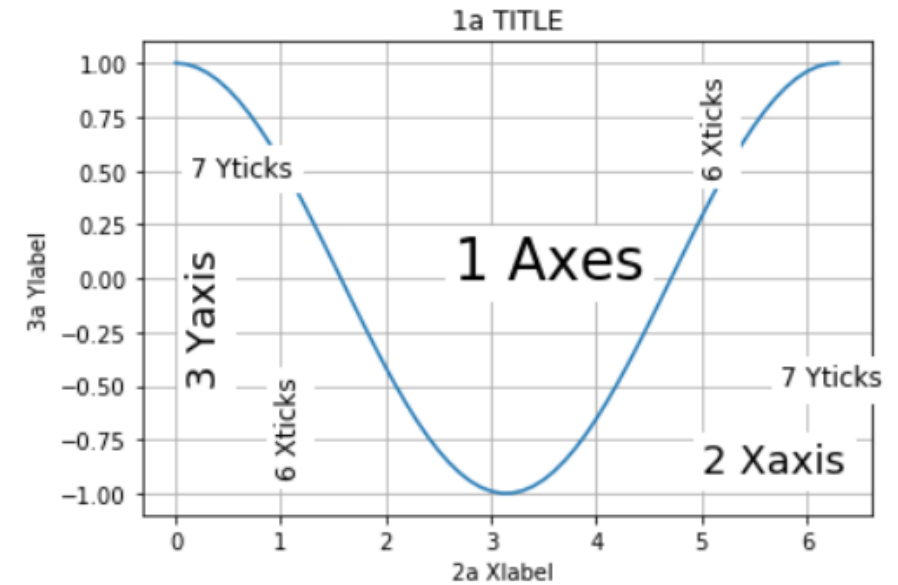
```
plt.text(np.pi-0.5, 0, '1 Axes', fontsize=26, bbox=dict(edgecolor='w', color='w'))
plt.text(0.1, 0, '3 Yaxis', fontsize=18, bbox=dict(edgecolor='w', color='w'), rotation=90)
plt.text(5, -0.9, '2 Xaxis', fontsize=18, bbox=dict(edgecolor='w', color='w'))
plt.title('1a TITLE')
```



```

plt.ylabel('3a Ylabel')
plt.xlabel('2a Xlabel ')
plt.text(5, 0.85, '6 Xticks', fontsize=12, bbox=dict(edgecolor='w', color='w'), rotation=90)
plt.text(0.95, -0.55, '6 Xticks', fontsize=12, bbox=dict(edgecolor='w', color='w'), rotation=90)
plt.text(5.75, -0.5, '7 Yticks', fontsize=12, bbox=dict(edgecolor='w', color='w'))
plt.text(0.15, 0.475, '7 Yticks', fontsize=12, bbox=dict(edgecolor='w', color='w'))
plt.grid(True)
plt.savefig('example.png', fmt='png')
plt.show()

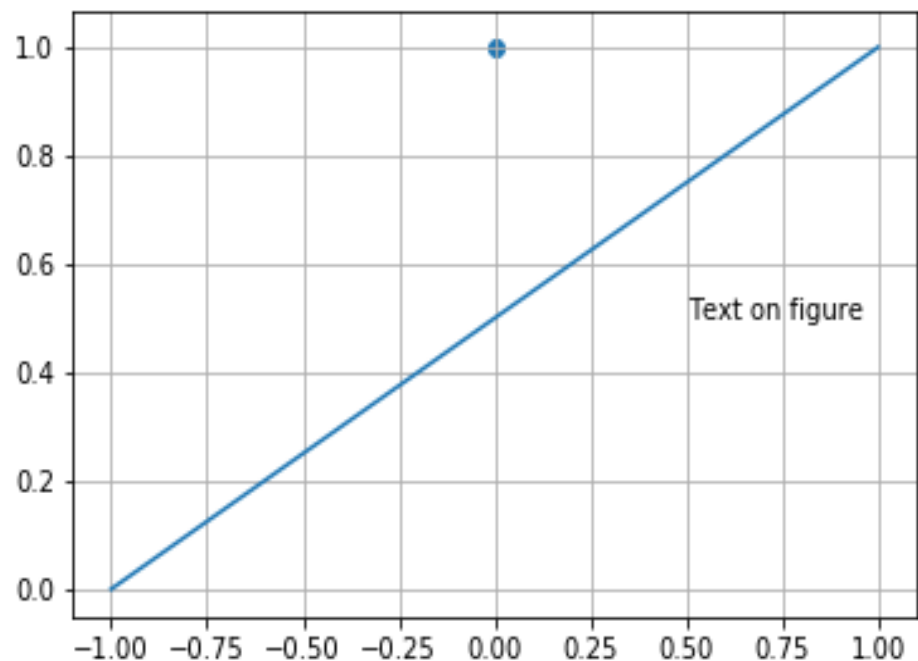
```



ГРАФИЧЕСКИЕ КОМАНДЫ

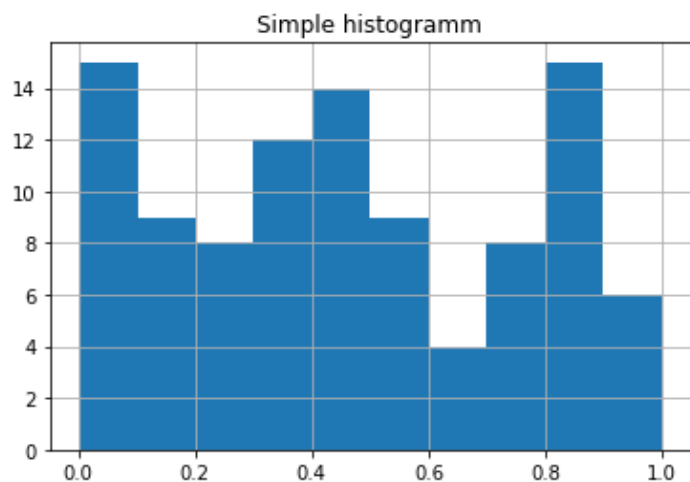
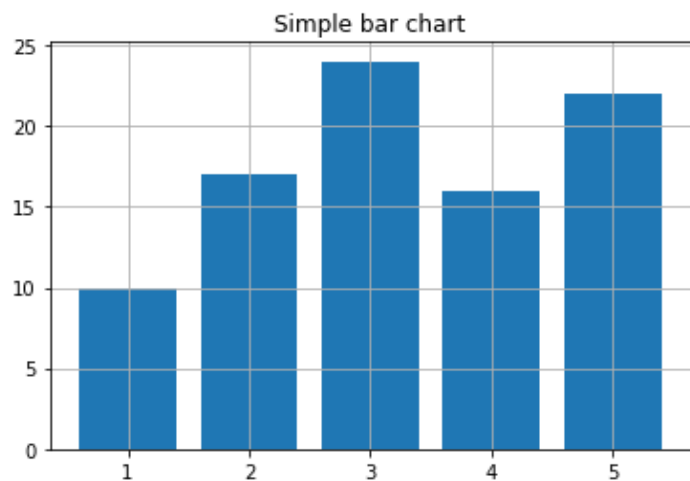
- `plt.scatter()` – маркер, точечное рисование
- `plt.plot()` – ломанная линия
- `plt.text()` – нанесение текста
- `plt.bar()` – столбчатая диаграмма
- `plt.hist()` – гистограмма
- `plt.pie()` – круговая диаграмма

ПРИМЕР



- `import matplotlib.pyplot as plt`
- `fig = plt.figure()`
- `scatter1 = plt.scatter(0.0, 1.0)`
- `graph1 = plt.plot([-1.0, 1.0], [0.0, 1.0])`
- `text1 = plt.text(0.5, 0.5, 'Text on figure')`
- `grid1 = plt.grid(True)`
- `plt.show()`

ПРИМЕР



```
import matplotlib.pyplot as plt
import numpy as np
```

```
s = ['one', 'two', 'three ', 'four' , 'five']
x = [1, 2, 3, 4, 5]
z = np.random.random(100)
z1 = [10, 17, 24, 16, 22]
z2 = [12, 14, 21, 13, 17]
```

```
# bar()
fig = plt.figure()
plt.bar(x, z1)
plt.title('Simple bar chart')
plt.grid(True)    # ЛИНИИ ВСПОМОГАТЕЛЬНОЙ СЕТКИ
```

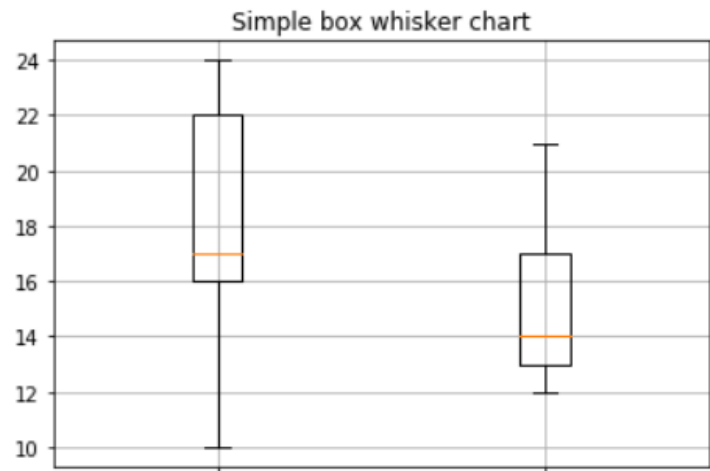
```
# hist()
fig = plt.figure()
plt.hist(z)
plt.title('Simple histogramm')
plt.grid(True)
```

```
..
```

ПРИМЕР



```
# pie()  
fig = plt.figure()  
plt.pie(x, labels=s)  
plt.title('Simple pie chart')
```

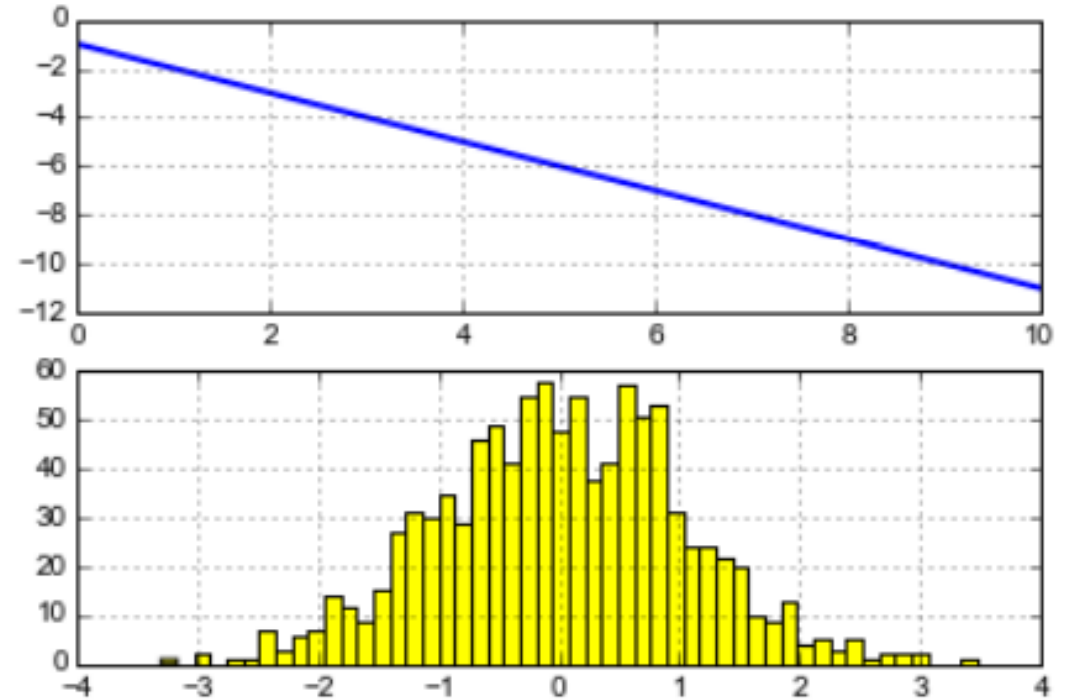


```
fig = plt.figure()  
plt.boxplot([z1, z2])  
plt.title('Simple box whisker chart')  
plt.grid(True)
```

```
plt.show()
```

КОНТЕЙНЕР AXES

- `import matplotlib.pyplot as plt` `import numpy as np`
- `x = [0,1,2,3,4,5,6,7,8,9,10]`
- `y = [-1,-2,-3,-4,-5,-6,-7,-8,-9,-10,-11]`
- `fig = plt.figure()`
- `ax = fig.add_subplot(211)`
- `line = ax.plot(x, y, '-', color='blue', linewidth=2)`
- `ax2 = fig.add_subplot(212)`
- `n, bins, rectangles = ax2.hist(np.random.randn(1000), 50, facecolor='yellow')`
- `for ax in fig.axes:`
 - `ax.grid(True)`
- `plt.show()`



Триплет: число ячеек по вертикали,
число ячеек по горизонтали,
номер ячейки

ПРИМЕР

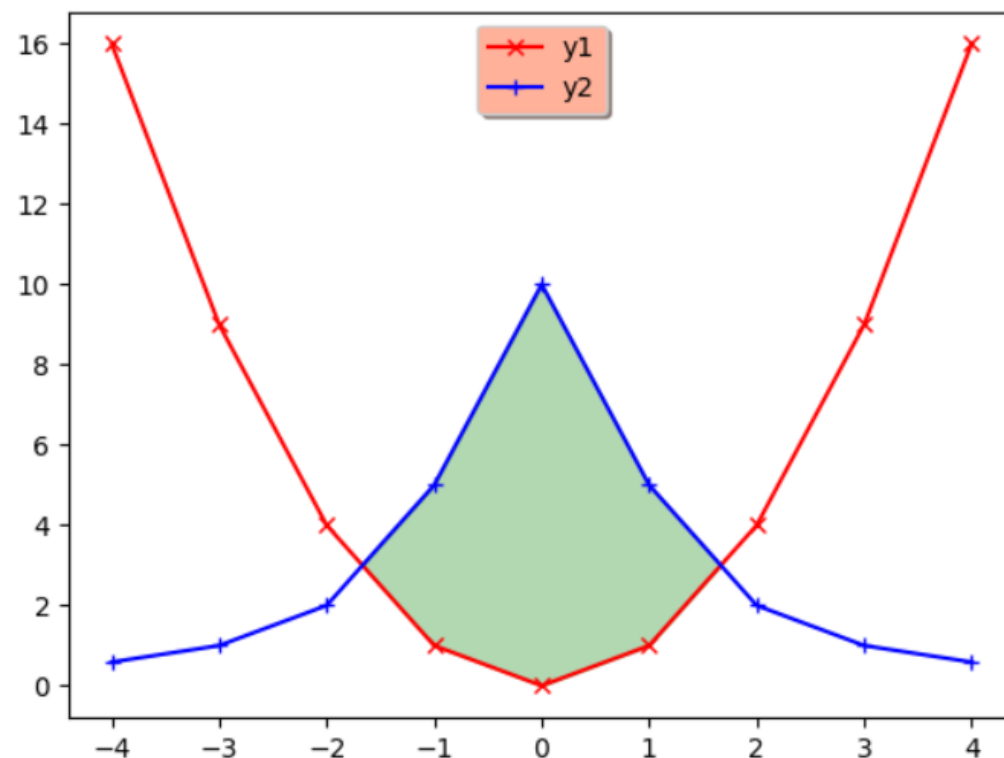
```
import numpy as np
import matplotlib.pyplot as plt

plt.ioff()
x = np.arange(-4, 5)
y1 = x ** 2
y2 = 10 / (x ** 2 + 1)
fig, ax = plt.subplots()

ax.plot(x, y1, 'rx', x, y2, 'b+', linestyle='solid')
ax.fill_between(x, y1, y2, where=y2>y1, interpolate=True,
               color='green', alpha=0.3)

lgnd = ax.legend(['y1', 'y2'], loc='upper center', shadow=True)
lgnd.get_frame().set_facecolor('#ffb19a')

plt.show()
```



ПРИМЕР

```
import matplotlib.pyplot as plt

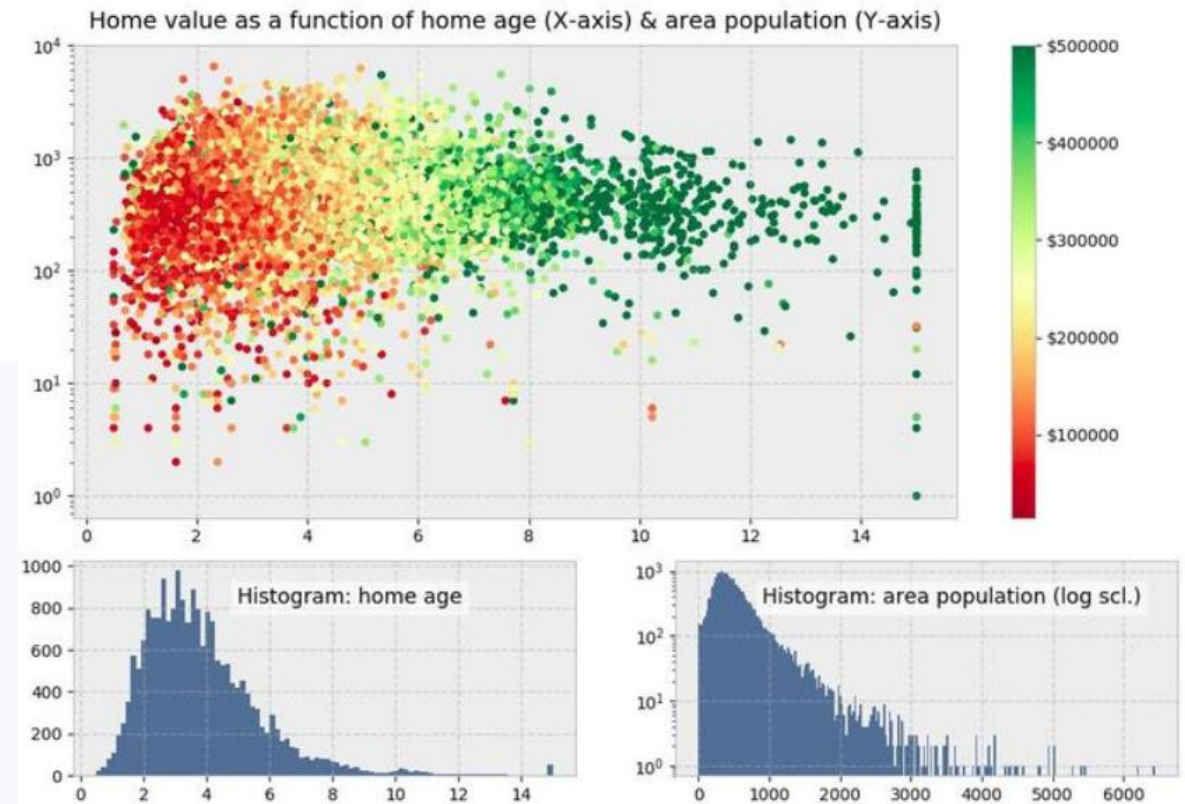
gridsize = (3, 2)
fig = plt.figure(figsize=(12, 8))
ax1 = plt.subplot2grid(gridsize, (0, 0), colspan=2, rowspan=2)
ax2 = plt.subplot2grid(gridsize, (2, 0))
ax3 = plt.subplot2grid(gridsize, (2, 1))

plt.show()

ax1.set_title(
    'Home value as a function of home age & area population',
    fontsize=14
)

sctr = ax1.scatter(x=age, y=pop, c=y, cmap='RdYlGn')
plt.colorbar(sctr, ax=ax1, format='${:d}')
ax1.set_yscale('log')
ax2.hist(age, bins='auto')
ax3.hist(pop, bins='auto', log=True)

add_titlebox(ax2, 'Histogram: home age')
add_titlebox(ax3, 'Histogram: area population (log scl.)')
```





PANDAS

PANDAS

- Главная библиотека в Python для работы с данными.
- Активно используют аналитики данных и дата-сайентисты.
- Библиотека была создана в 2008 году компанией AQR Capital, а в 2009 году она стала проектом с открытым исходным кодом с поддержкой большого комьюнити.
- **`import pandas as pd`**

ОСНОВНЫЕ ОБЪЕКТЫ



SERIES

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

```
s
a    1.321250
b    0.365307
c    0.709577
d    0.542710
e   -0.212721
dtype: float64
```

индекс							
серия							

```
print s['b']
0.365307109524
```

```
print s.get('z', 'error')
error
```

DATAFRAME

```
df = pd.DataFrame(np.random.randn(8, 3),  
index=pd.date_range('1/1/2000', periods=8),  
columns=['A', 'B', 'C'])
```

df

	A	B	C
2000-01-01	0.684918	0.240427	-0.030283
2000-01-02	0.533952	-0.573713	-1.602537
2000-01-03	-1.291314	-0.650594	1.771561
2000-01-04	2.813297	-1.093390	-0.209462
2000-01-05	0.894795	-0.574468	0.765031
2000-01-06	1.513772	0.618505	-1.402341
2000-01-07	-0.435267	-1.199286	0.990490
2000-01-08	-0.541890	0.590653	-0.530153

	КОЛОНКИ			
индекс	серия	серия	серия	серия

ЗАГРУЗКА ДАННЫХ

```
# Excel
data2 = pd.read_excel('D:\\filename.xlsx', sheetname='1')
# csv-файл
data = pd.read_csv('D:\\filename.csv', sep=';', decimal=',')
data.to_csv('foo.csv') # сохранение
```

Файл H5 представляет собой файл данных в формате Hierarchical Data Format [HDF](#). Это разновидность библиотечного файла, используемого для хранения больших объемов числовых, графических и текстовых данных.

```
pd.read_hdf('foo.h5', 'df')
```

ПРИМЕР РАБОТЫ С ДАННЫМИ

```
datatrain = pd.read_csv('D:\\Competitions\\Rossman\\train.csv')  
datatrain[:3]
```

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1

```
datatrain.Date = pd.to_datetime(datatrain.Date)
```

СОЗДАНИЕ ДАТАФРЕЙМА

первый способ

```
data = pd.DataFrame({ 'A' : [1., 4., 2., 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' }, index=pd.period_range('Jan-2000', periods=4,  
freq='M'))  
print data
```

	A	B	C	D	E	F
2000-01	1	2013-01-02	NaN	3	test	foo
2000-02	4	2013-01-02	NaN	3	train	foo
2000-03	2	2013-01-02	NaN	3	test	foo
2000-04	1	2013-01-02	NaN	3	train	foo

ОПЕРАЦИИ С DATAFRAME

```
# простейшие операции
```

```
# столбцы
```

```
print data.columns
```

```
Index([u'A', u'B', u'C', u'D', u'E', u'F'], dtype='object')
```

```
# строки - но тут временная индексация
```

```
print data.index
```

```
<class 'pandas.tseries.period.PeriodIndex'>
```

```
[2000-01, ..., 2000-04]
```

```
# сортировка
```

```
print data.sort(columns='A')
```

	A	B	C	D	E	F
2000-01	1	2013-01-02	NaN	3	test	foo
2000-04	1	2013-01-02	NaN	3	train	foo
2000-03	2	2013-01-02	NaN	3	test	foo
2000-02	4	2013-01-02	NaN	3	train	foo

	A	B	C	D	E	F
2000-01	1	2013-01-02	NaN	3	test	foo
2000-02	4	2013-01-02	NaN	3	train	foo
2000-03	2	2013-01-02	NaN	3	test	foo
2000-04	1	2013-01-02	NaN	3	train	foo

ОПЕРАЦИИ С DATAFRAME

```
# превращение в пр-матрицу
print data.values # без скобок
```

	A	B	C	D	E	F
2000-01	1	2013-01-02	NaN	3	test	foo
2000-02	4	2013-01-02	NaN	3	train	foo
2000-03	2	2013-01-02	NaN	3	test	foo
2000-04	1	2013-01-02	NaN	3	train	foo

```
array([[1.0, Timestamp('2013-01-02 00:00:00'), nan, 3, 'test', 'foo'],
       [4.0, Timestamp('2013-01-02 00:00:00'), nan, 3, 'train', 'foo'],
       [2.0, Timestamp('2013-01-02 00:00:00'), nan, 3, 'test', 'foo'],
       [1.0, Timestamp('2013-01-02 00:00:00'), nan, 3, 'train', 'foo']],
      dtype=object)
```

```
# число уникальных элементов (можно через describe)
for i in data.columns: # можно просто data
    print str(i) + ':' + str(data[i].nunique())
```

B:1
C:0
D:1
E:2
F:1

ОПЕРАЦИИ С DATAFRAME

Переименование колонок

```
df2 = df.rename(columns={'int_col' : 'some_other_name'})

# изменение текущего датафрейма
df2.rename(columns={'some_other_name' : 'int_col'}, inplace = True)

df = pd.DataFrame({'x':[1,3,2], 'y':[2,4,1]})
# удаление строки
df.drop(1, axis=0, inplace=True)
# удаление столбца
del df['x'] # df.drop('x', axis=1)
```

ОПЕРАЦИИ С DATAFRAME

	A	B	C	D	E	F
2000-01	1	2013-01-02	NaN	3	test	foo
2000-02	4	2013-01-02	NaN	3	train	foo
2000-03	2	2013-01-02	NaN	3	test	foo
2000-04	1	2013-01-02	NaN	3	train	foo

```
data.at['2000-01', 'A'] = 10. # по названию
```

```
data.iat[0,1] = pd.Timestamp('19990101') # по номеру
```

```
..
```

```
data.loc['2000-01':'2000-02', ['D', 'B', 'A']] # по названию
```

```
data.iloc[0:2,1:3] # по номеру
```

```
# выбор с проверкой на вхождение
```

```
data[data['E'].isin(['test', 'valid'])] # полезно: isin
```

Изменить порядок записи в датафрейме

```
data.reindex(index=data.index[::-1])
```

```
# или data = data.iloc[::-1]
```

ОПЕРАЦИИ С DATAFRAME

Для вставки колонок в любое место

`.insert()` # если `df['new'] = ...`, то вставляется в конец

Итерации

```
df = pd.DataFrame({'x': [1, 2, 1, 2], 'y': [1, 2, 3, 3], 'z': [0, 0, 0, 0]},  
index=['a', 'b', 'c', 'd'])
```

	x	y	z
a	1	1	0
b	2	2	0
c	1	3	0
d	2	3	0

```
for col in df: # не обязательно писать df.columns  
    print col
```

x
y
z

ОПЕРАЦИИ: ОБЪЕДИНЕНИЕ

In [7]: `print(pd.concat([left,right]))`

	key	l	r
0	1	1.0	NaN
1	2	2.0	NaN
2	1	3.0	NaN
0	1	NaN	4.0
1	2	NaN	5.0
2	3	NaN	6.0

```
In [1]: import pandas as pd
left=pd.DataFrame({'key':[1,2,1], 'l':[1,2,3]})
right=pd.DataFrame({'key':[1,2,3], 'r':[4,5,6]})
```

```
In [3]: print(left)
```

	key	l
0	1	1
1	2	2
2	1	3

```
In [4]: print(right)
```

	key	r
0	1	4
1	2	5
2	3	6

```
In [5]: print(pd.merge(left, right, on='key'))
```

	key	l	r
0	1	1	4
1	1	3	4
2	2	2	5

ОПЕРАЦИИ С DATAFRAME: ГРУППИРОВКА

Для каждого уникального значения A найти минимальный B

```
d = pd.DataFrame({'A': [1,2,2,1,3,3], 'B': [1,2,3,3,2,1]})  
print d
```

```
print d.loc[d.groupby('A')['B'].idxmin()]
```

	A	B
0	1	1
1	2	2
2	2	3
3	1	3
4	3	2
5	3	1

	A	B
0	1	1
1	2	2
5	3	1

```
# вывод групп
```

```
for x, y in a.groupby(['A', 'B']): # можно for (x1, x2), y in ...
```

```
    print x
```

```
    print y
```

```
(1, 3)
```

```
    A  B  C
```

```
0  1  3  5
```

```
4  1  3  6
```

```
(1, 4)
```

```
    A  B  C
```

```
3  1  4  6
```

```
(2, 3)
```

```
    A  B  C
```

```
2  2  3  5
```

```
5  2  3  6
```

```
(2, 4)
```

```
    A  B  C
```

```
1  2  4  5
```

```
6  2  4  6
```

	A	B	C
0	1	3	5
1	2	4	5
2	2	3	5
3	1	4	6
4	1	3	6
5	2	3	6
6	2	4	6

`.groupby(, sort=True)` – сортировка результата

	A	B	C
0	1	3	5
1	2	4	5
2	2	3	5
3	1	4	6
4	1	3	6
5	2	3	6
6	2	4	6

```
print a.groupby(['A','B']).first() # первые элементы
```

```
      C
```

```
A B
```

```
1 3 5
```

```
   4 6
```

```
2 3 5
```

```
   4 5
```

```
print a.groupby(['A','B'])['C'].mean()
```

```
A B
```

```
1 3 5.5
```

```
   4 6.0
```

```
2 3 5.5
```

```
   4 5.5
```

```
print a.groupby(['A','B']).get_group((1,3)) # выбор конкретной группы
```

```
      A B C
```

```
0 1 3 5
```

```
4 1 3 6
```

```
print a.groupby(['A','B']).sum()
```

```
      C
```

```
A B
```

```
1 3 11
```

```
   4 6
```

```
2 3 11
```

```
   4 11
```

ОПЕРАЦИЯ С DATAFRAME: АГРЕГАЦИЯ

	A	B	C
0	1	3	5
1	2	4	5
2	2	3	5
3	1	4	6
4	1	3	6
5	2	3	6
6	2	4	6

агрегация по одному столбцу

```
print a.groupby(['A', 'B'])['C'].agg({'sum': np.sum,  
    'mean': np.mean})
```

		sum	mean
A	B		
1	3	11	5.5
	4	6	6.0
2	3	11	5.5
	4	11	5.5

агрегация по разным столбцам

```
print a.groupby('A').agg({'B': np.sum, 'C': np.mean})
```

	C	B
A		
1	5.666667	10
2	5.500000	14

Apply

	A	B	C
0	1	3	5
1	2	4	5
2	2	3	5
3	1	4	6
4	1	3	6
5	2	3	6
6	2	4	6

```
def f(x):  
    return pd.DataFrame({'x': x, 'x-mean': x - x.mean()})  
a.groupby('A')['B'].apply(f)
```

	x	x-mean
0	3	-0.333333
1	4	0.500000
2	3	-0.500000
3	4	0.666667
4	3	-0.333333
5	3	-0.500000
6	4	0.500000

Apply

Пример нормировки

```
a.apply(lambda x: x/sum(x))  
# по столбцам
```

	A	B	C
0	0.090909	0.125000	0.128205
1	0.181818	0.166667	0.128205
2	0.181818	0.125000	0.128205
3	0.090909	0.166667	0.153846
4	0.090909	0.125000	0.153846
5	0.181818	0.125000	0.153846
6	0.181818	0.166667	0.153846

```
a.apply(lambda x: x/sum(x), axis=1)  
# по строкам
```

	A	B	C
0	0.111111	0.333333	0.555556
1	0.181818	0.363636	0.454545
2	0.200000	0.300000	0.500000
3	0.090909	0.363636	0.545455
4	0.100000	0.300000	0.600000
5	0.181818	0.272727	0.545455
6	0.166667	0.333333	0.500000

Удаление дубликатов

```
df = pd.DataFrame({'name': ['Al', 'Max', 'Al'],  
                  'surname': [u'Run', u'Crone', u'Run']})  
print df.duplicated()
```

```
df.drop_duplicates(['name'], take_last=True)  
# df.drop_duplicates()
```

```
0    False  
1    False  
2     True
```

	name	surname
1	Max	Crone
2	Al	Run

ВИЗУАЛИЗАЦИЯ ДАННЫХ

```
In [7]: sp500 = pd.read_csv("data/sp500.csv",  
                             index_col='Symbol',  
                             usecols=['Symbol', 'Sector', 'Price',  
                                       'Book Value', 'Market Cap',  
                                       'Dividend Yield'])  
  
sp500
```

Out[7]:

	Sector	Price	Dividend Yield	Book Value	Market Cap
Symbol					
MMM	Industrials	141.14	2.12	26.668	92.345
ABT	Health Care	39.60	1.82	15.573	59.477
ABBV	Health Care	53.95	3.02	2.954	85.784
ACN	Information Technology	79.79	2.34	8.326	50.513
ACE	Financials	102.91	2.21	86.897	34.753

```
print(sp500.loc["MSFT"])
```

```
Sector          Information Technology
Price           40.12
Dividend Yield   2.67
Book Value       10.584
Market Cap       331.4
Name: MSFT, dtype: object
```

считываем исторические данные о котировках акций

```
In [8]: omh = pd.read_csv('data/omh.csv',
                        parse_dates=['Date'])

omh.set_index('Date',
              inplace=True)
```

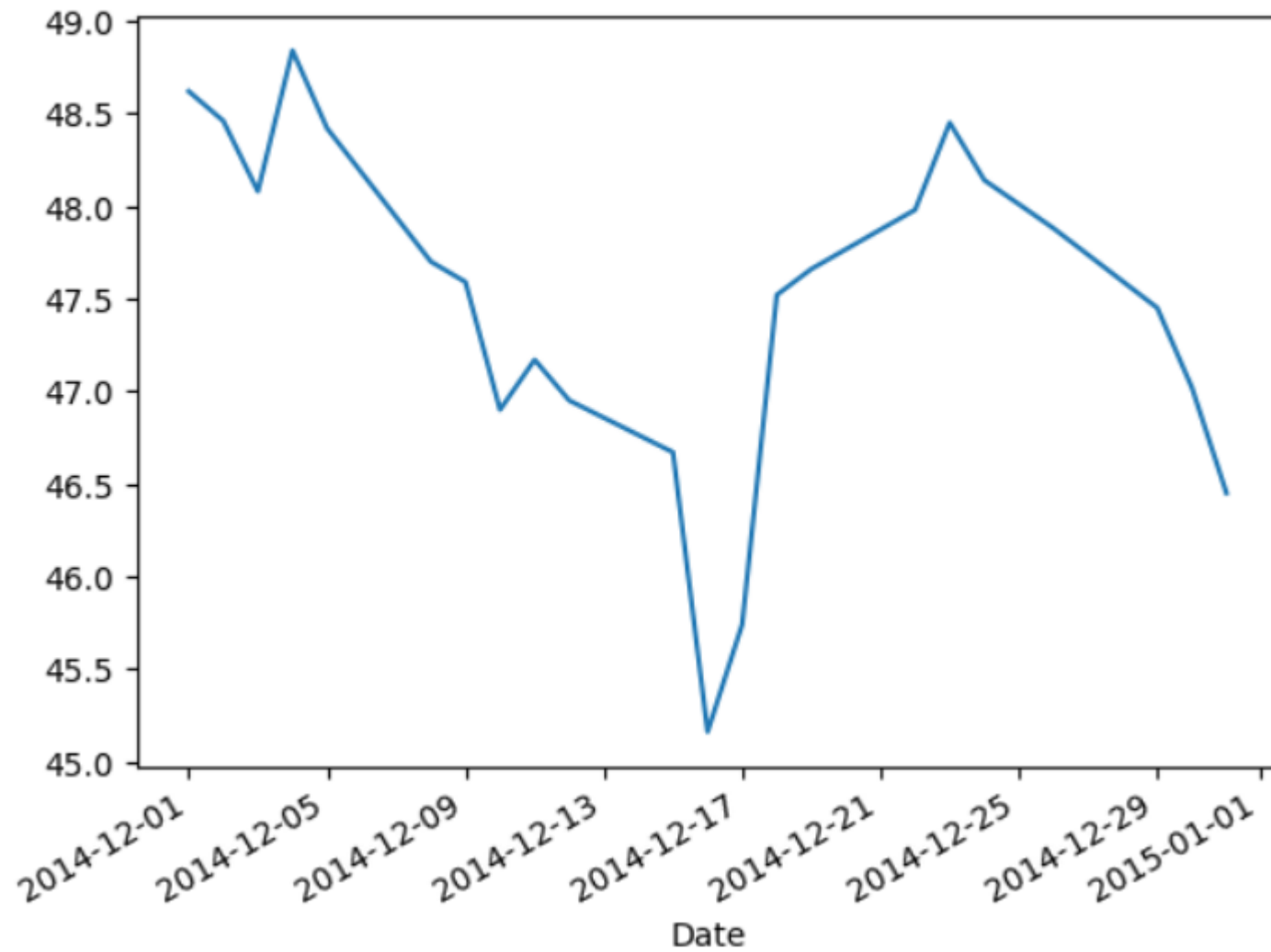
```
In [9]: omh.head()
```

Out[9]:

	MSFT	AAPL
Date		
2014-12-01	48.62	115.07
2014-12-02	48.46	114.63
2014-12-03	48.08	115.93
2014-12-04	48.84	115.49
2014-12-05	48.42	115.00

```
omh.MSFT.plot()
```

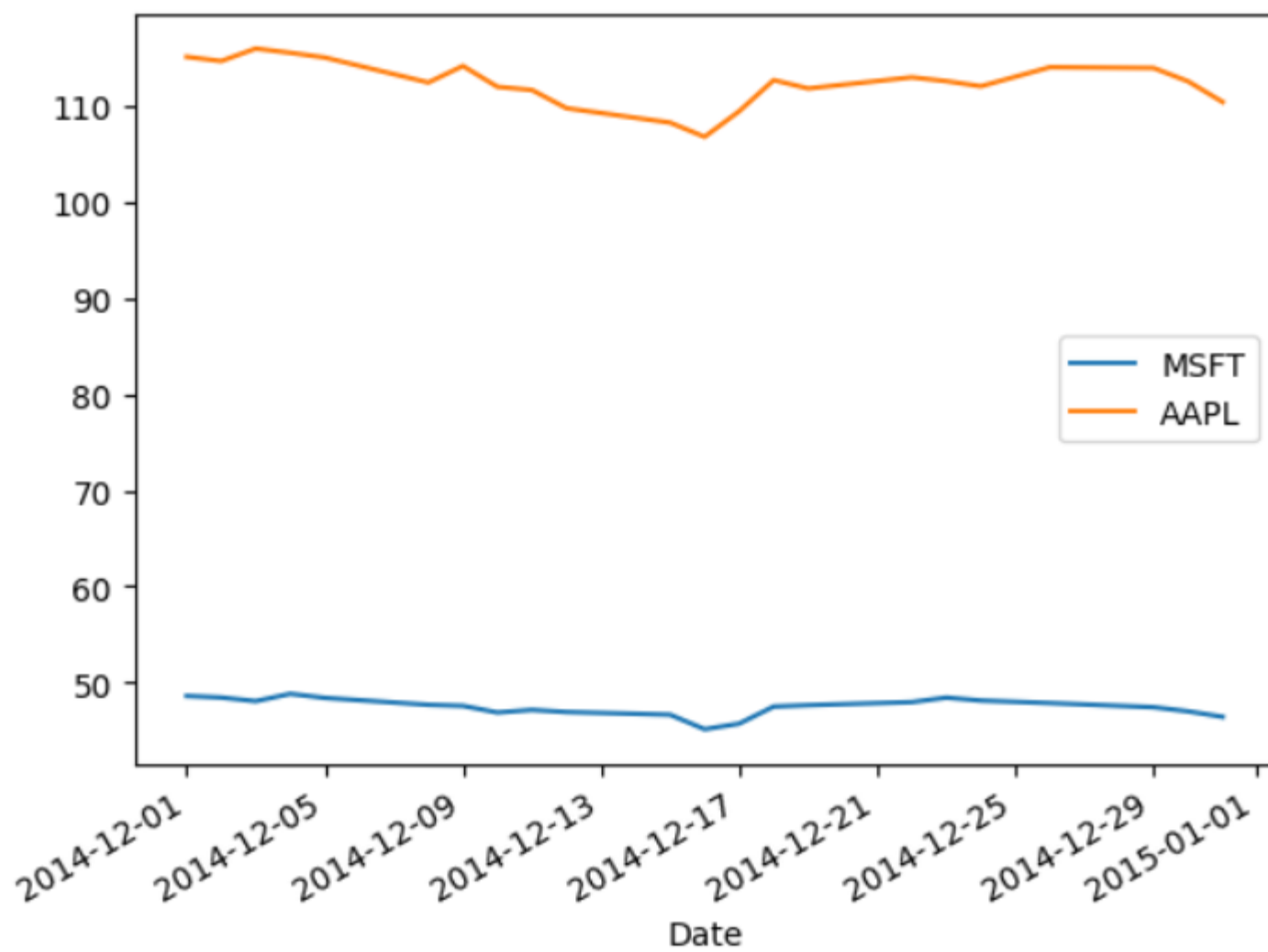
```
<AxesSubplot: xlabel='Date'>
```



	MSFT	AAPL
Date		
2014-12-01	48.62	115.07
2014-12-02	48.46	114.63
2014-12-03	48.08	115.93
2014-12-04	48.84	115.49
2014-12-05	48.42	115.00

```
In [18]: omh.plot()
```

```
Out[18]: <AxesSubplot: xlabel='Date'>
```

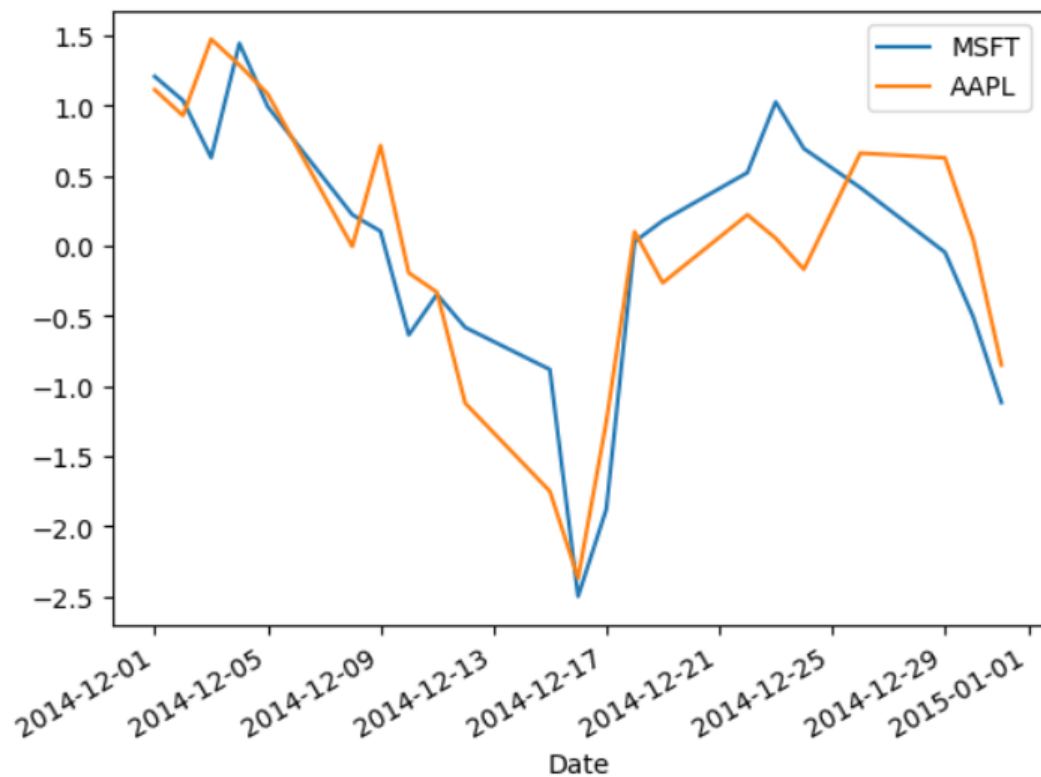


	MSFT	AAPL
Date		
2014-12-01	48.62	115.07
2014-12-02	48.46	114.63
2014-12-03	48.08	115.93
2014-12-04	48.84	115.49
2014-12-05	48.42	115.00

нормализация:

```
omh_copy = (omh - omh.mean())/omh.std()  
omh_copy.plot()
```

<AxesSubplot: xlabel='Date'>



```
omh_copy.plot(style={'MSFT': 'b--^', 'AAPL': 'g:o'})
```

<AxesSubplot: xlabel='Date'>

