Лекция 13

Графическое представление данных

Библиотека Matplotlib

Модуль matplotlib

- Модуль matplotlib представляет собой библиотеку для графического представления данных
- Основной субмодуль библиотеки *pyplot* позволяет использовать команды в стиле Matlab. Традиционная инструкция импорта:

```
import matplotlib.pyplot as plt
```

- Библиотека Matplotlib *не входит* в стандартную поставку Питона и должна быть установлена как отдельный пакет или модуль
- В Linux модуль matplotlib оформлен как пакет дистрибутива
- Для MacOS и Windows существуют готовые бинарные сборки Питона с библиотеками для научных расчетов, например Anaconda (лицензия BSD):

```
https://www.anaconda.com/distribution/
```

Формат данных

- Исходные данные для графического отображения организованы в виде массивов библиотеки NumPy.
- Для работы с библиотекой Matplotlib модуль numpy также должен быть импортирован:

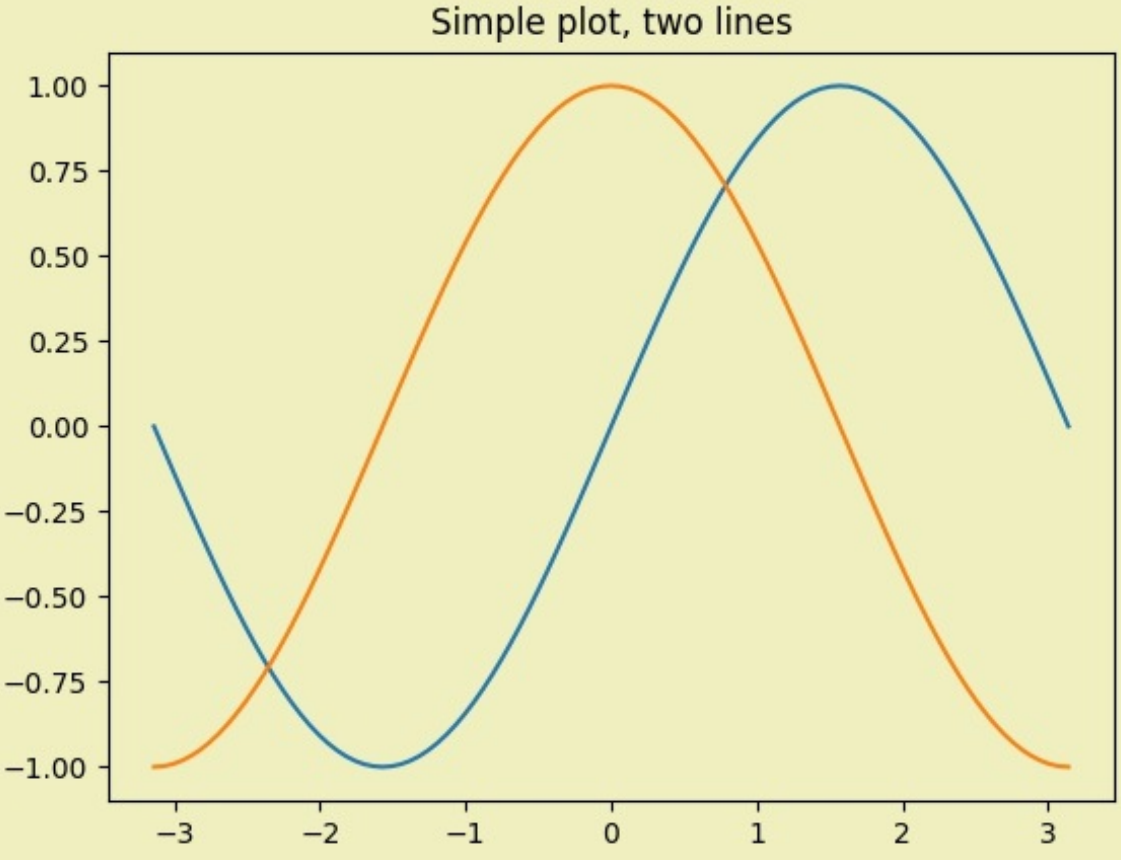
```
import numpy as np
```

- В качестве координат точек графика используются одномерные или двумерные массивы
  - одномерный массив содержит последовательность координат точек графика
  - двумерный массив *в своих столбцах* содержит последовательности координат точек *для нескольких кривых* графика

Простой график

```
# 1. Формирование данных для отображения
xa = np.linspace(-np.pi, np.pi, 101, endpoint=True)
ya1 = np.sin(xa)
ya2 = np.cos(xa)

# 2. Подготовка изображения
plt.title('Simple plot, two lines')
plt.plot(xa, ya1)
plt.plot(xa, ya2)

# 3. Вывод изображения на экран или в файл
if to_file:
  plt.savefig('ex01.jpg')
else:
 plt.show()
```

- Разработчики библиотеки Matplotlib старались следовать логике работы пакета Matlab. Контекст изображения (figure) создается автоматически и последовательно модифицируется вызовом функций имитирующих интерактивные команды.


Simple plot, two lines
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−3
−2
−1
0
1
2
3

Функции plot() и close()

- Функция plot() используется для отображения традиционных двумерных графиков
- Функция plot() воспринимает любое количество позиционных и/или именованных аргументов

```
plot(*args, **kwargs)
```

- После завершения работы с графиком и его отображения на экране или сохранения в файле следует вызвать функцию close()
- Функция close() "очищает холст" делая его готовым для отображения нового графика, то есть устанавливает значения по умолчанию для контекста изображения figure

Правила интерпретации аргументов функции plot()

- Аргумент может быть одномерным или двумерным массивом
- Одномерный массив это последовательность координат по одной оси графика
- Двумерный массив это *несколько последовательностей* координат по одной оси графика.
  - размерность по оси X это число последовательностей
  - размерность по оси Y это число точек в последовательности, то есть каждому *столбцу* матрицы соответствует своя кривая на графике

Правила интерпретации аргументов функции plot()

- Единственный аргумент-массив это координаты по оси Y. При этом координаты по оси X формируются как последовательность целых чисел, начинающаяся с нуля.
- Если аргументов-массивов два, то первый массив это координаты по оси X, второй массив это координаты по оси Y
- После аргументов-массивов может следовать аргумент-формат, текстовая строка определяющая вид графика
- Последовательность "аргумент(ы)-массив(ы), аргумент-формат" может повторяться несколько раз

```
# Один массив + формат, три раза
plt.plot(a1, 'o', a2, '+', a3, '*')

# Два массива + формат, два раза
plt.plot(range(a1.size), a1, 'o', range(a2.size), a2, '+')
```

Аргумент-формат

- Аргумент-формат это строка, содержащая символ вида линии, символ вида маркера и символ цвета
- Символы вида линии:

```
График в виде линии
'-'  : сплошная линия          '-.' : штрих-пунктирная линия
'--' : штриховая линия         ':'  : пунктирная линия

График в виде точек (маркеров)
'.'  : точка                   's'  : квадрат
','  : один пиксель            'p'  : пятиугольник
'o'  : большая точка           '*'  : звезда
'v'  : треугольник вниз        'h'  : шестиугольник ^
'^'  : треугольник вверх       'H'  : шестиугольник >
'<'  : треугольник влево       '+'  : знак +
'>'  : треугольник вправо      'x'  : знак x
'1'  : три луча вниз           'D'  : ромб обычный
'2'  : три луча вверх          'd'  : ромб тонкий
'3'  : три луча влево          '|'  : знак |
'4'  : три луча вправо         '_'  : знак _
```

Аргумент-формат, цвет

- Символы цвета:

```
'b' : синий (blue)
'g' : зеленый (green)
'r' : красный (red)
'c' : голубой (cyan)
'm' : фиолетовый (magenta)
'y' : желтый (yellow)
'k' : черный (black)
'w' : белый (white)
```

- Примеры формата

```
'Db-' : синяя сплошная линия + маркеры-ромбы
'g--' : зеленая штриховая линия
's'   : маркеры в виде квадратов, цвет выбирается автоматически
```

Именованные аргументы

- Некоторые именованные аргументы функции plot()

```
alpha             : прозрачность графика,
                    от 0.0 (прозрачный) до 1.0 (сплошной)
clip_box          : прямоугольник, ограничивающий изображение
color             : цвет линии или маркера
linestyle или ls  : стиль линии
linewidth или lw  : ширина линии в пикселях
marker            : вид маркера
markersize или ms : размер маркера в пикселях
```

а также аргументы уточняющие стили отображения и аргументы для придания графику интерактивных возможностей

- Именованный аргумент color и аналогичные именованные аргументы других функций воспринимают значение цвета в виде однобуквенного сокращения, полного названия некоторых цветов и RGB-нотацию в виде #rrggbb, например #ffcc66

Масштаб и оси координат

- Масштаб по каждой оси координат выбирается автоматически исходя из минимального и максимального значения отображаемых данных
- Предельные значения координат по каждой оси можно задать явно используя функции xlim() и ylim()
- Метки на координатных осях можно задать явно используя функции xticks() и yticks()

```
plt.xlim(-6, 6)
plt.ylim(-1.5, 1.5)
plt.xticks(np.linspace(-6, 6, 13, endpoint=True))
plt.yticks(np.linspace(-1.5, 1.5, 13, endpoint=True))
```

- После вызова функции close() настройки возвращаются к значениям по умолчанию

Функция figure()

- Функция plot() это агрегатор многих действий, в том числе создания "холста", на котором производится отображение графиков. Холст это объект класса *matplotlib.figure.Figure*
- Виртуальную поверхность для рисования ("холст" или "картинку") можно создать явным образом, вызвав функцию *figure()*
- Функция figure() воспринимает ряд позиционных и именованных параметров, все параметры необязательные
- Первый параметр *num* - номер картинки
  - если не задан - создает новую картинку, номер картинки устанавливается автоматически
  - если задан как число N, создает картинку с номером N; если картинка с номером N уже существует, делает ее активной
  - если задан как строка, создает новую картинку, строка становится заголовком окна картинки

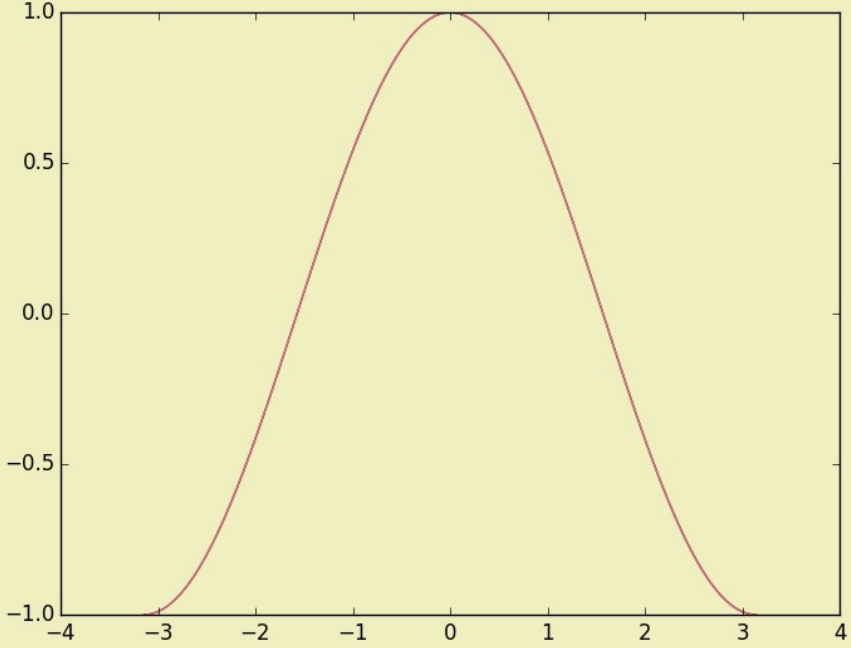
Размер изображения

- Размер изображения можно задать при вызове функции figure() параметром figsize
  - figsize=(x,y) - размер картинки в дюймах
- Параметр dpi - разрешение изображения в точках на дюйм, позволяет пересчитать дюймы в пиксели на экране или точки в файле, содержащем изображение
- Другие именованные параметры
  - facecolor - цвет фона графика
  - edgecolor - цвет фона полей графика
  - frameon - отображать рамку вокруг графика, по умолчанию True
  - clear - очистить картинку при выборе, по умолчанию False
- Атрибут *number* объекта класса Figure содержит номер картинки.

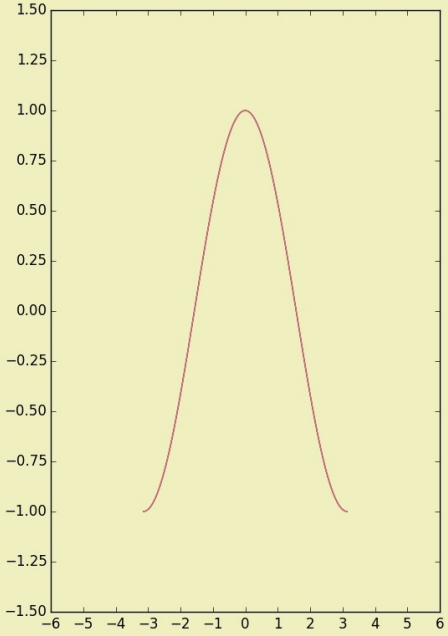
Пример

```
# Оригинальный график
xa = np.linspace(-np.pi, np.pi, 101, endpoint=True)
ya = np.cos(xa)
plt.plot(xa, ya, color='#ef7050')
plt.show()

# Модифицированный график
plt.figure(figsize=(6, 9), dpi=75)
plt.xlim(-6, 6)
plt.ylim(-1.5, 1.5)
plt.xticks(np.linspace(-6, 6, 13, endpoint=True))
plt.yticks(np.linspace(-1.5, 1.5, 13, endpoint=True))
xa = np.linspace(-np.pi, np.pi, 101, endpoint=True)
ya = np.cos(xa)
plt.plot(xa, ya, color='#ef7050')
plt.show()
```

Оригинальный график



Модифицированный график



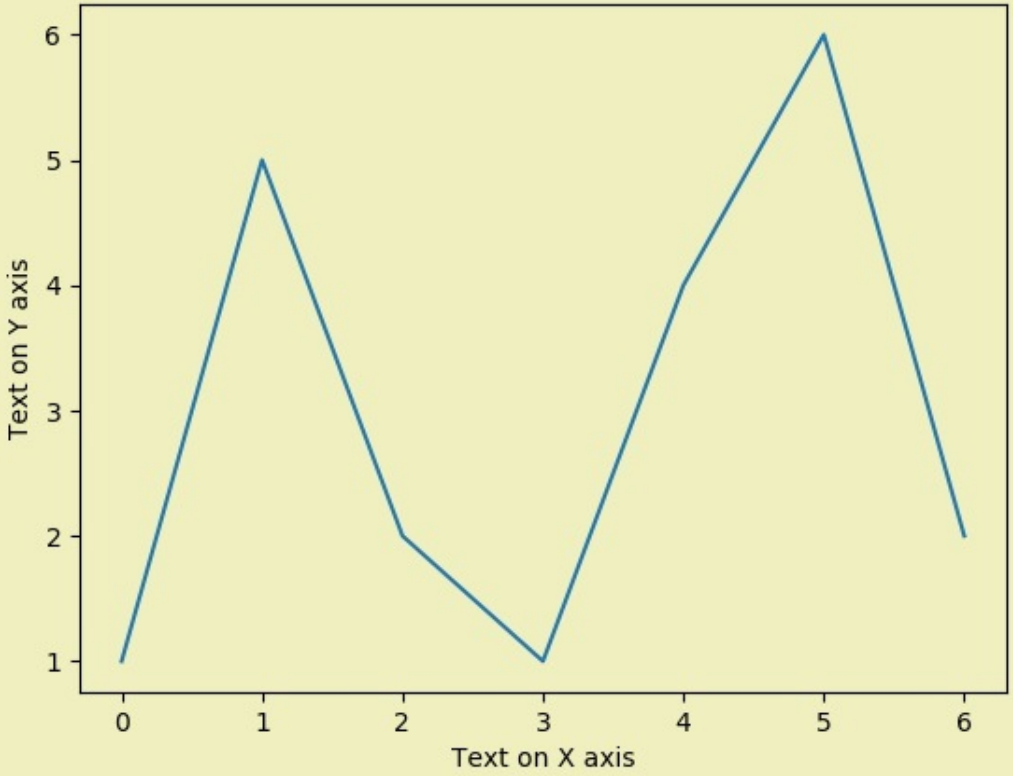
Надписи к осям

- По умолчанию оси графика не имеют надписей
- Для создания надписей к осям используются функции *xlabel()* и *ylabel()*

```
a = 1, 5, 2, 1, 4, 6, 2
plt.plot(a)
plt.xlabel('Text on X axis')
plt.ylabel('Text on Y axis')
plt.show()
```

Функция plot() в качестве массива координат воспринимает не только ndarray. Также можно использовать списки или кортежи. Необходимые преобразования типа данных произойдут автоматически.

Надписи к осям



Пояснения к графику (легенда)

- Пояснения к графику вводятся функцией legend()
- Первым позиционным параметром функции legend() может быть последовательность надписей для кривых графика в порядке их отображения функцией plot()
- Надпись для кривой графика также может быть задана параметром label функции plot()
- Именованные параметры функции legend()
  - loc - задает положение блока пояснений в виде строки с сочетанием слов 'left', 'right', 'upper', 'lower', 'center' и 'best' (умолчание)
  - frameon - разрешает или запрещает рамку вокруг блока пояснений
  - title - строка-заголовок блока пояснений
  - fontsize - задает размер шрифта строкой из ряда: 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large'

Пример

```
xa = np.linspace(-np.pi, np.pi, 101, endpoint=True)

# График 1
plt.plot(xa, np.sin(xa))
plt.plot(xa, np.cos(xa))
plt.legend(('sin(x)', 'cos(x)'), loc='upper left')
plt.show()

# График 2
plt.plot(xa, np.sin(xa), label='sin(x)')
plt.plot(xa, np.cos(xa), label='cos(x)')
plt.legend()
plt.show()
```

- Если легенду одной из кривых отображать не нужно, в первом случае для нее указывается значение '_nolegend_', во втором случае параметр label не задается

График 1

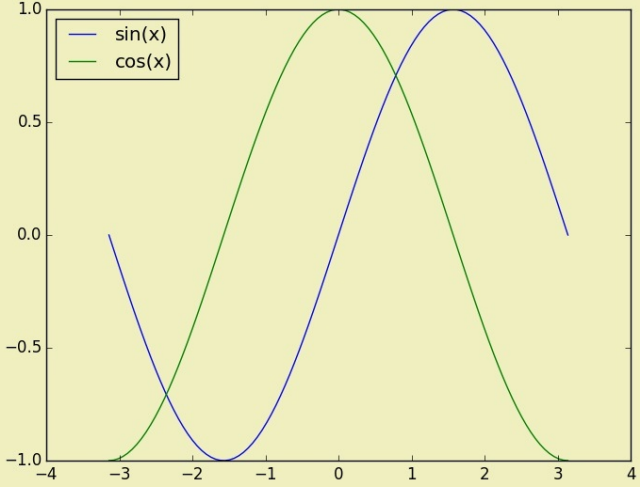
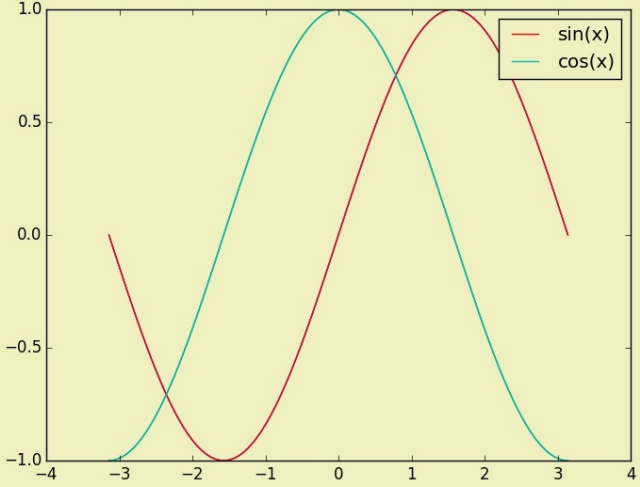


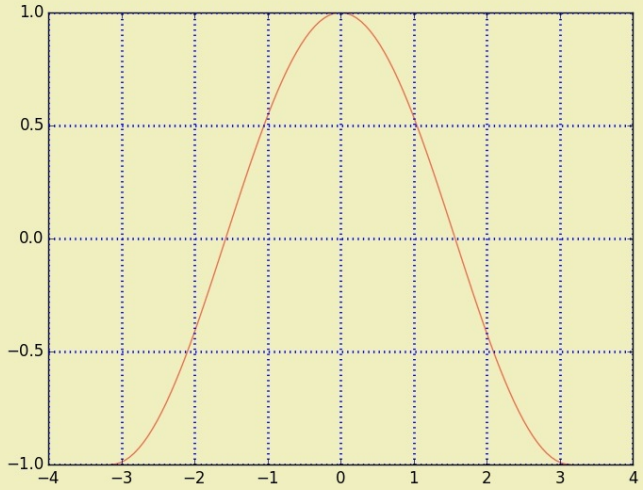
График 2



Координатная сетка

- Координатная сетка включается функцией grid(). Именованные аргументы позволяют устанавливать свойства линий координатной сетки

```
plt.grid(True, color='b', linestyle='-', linewidth=2)
```



Несколько графиков на одном изображении

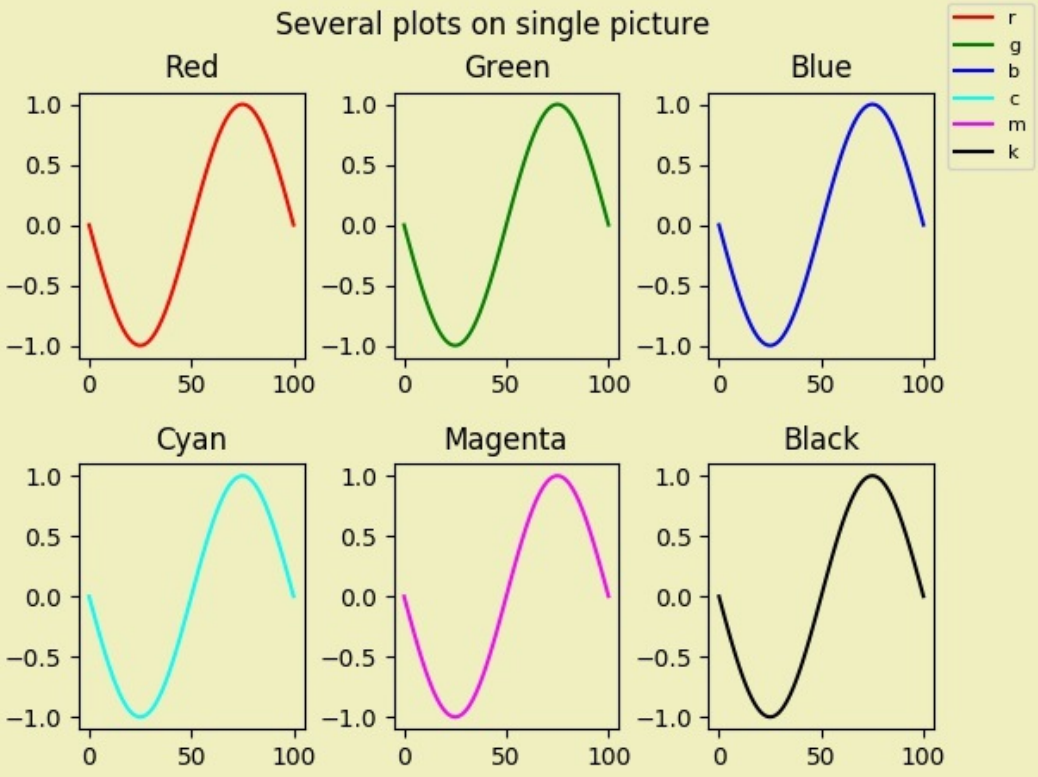
- На одном изображении можно поместить несколько графиков с собственными координатными осями, метками на осях и заголовками
- Пример:

```
fig, axes = plt.subplots(2, 3) # 2 rows, 3 columns
plt.subplots_adjust(wspace=0.4, hspace=0.4) # wspace => width, hspace => height

fig.suptitle('Several plots on single picture')

xa = np.linspace(-np.pi, np.pi, 101, endpoint=True)
ya = np.sin(xa)
axes[0, 0].plot(ya, color='red',     label='r') ; axes[0, 0].set_title('Red')
axes[0, 1].plot(ya, color='green',   label='g') ; axes[0, 1].set_title('Green')
axes[0, 2].plot(ya, color='blue',    label='b') ; axes[0, 2].set_title('Blue')
axes[1, 0].plot(ya, color='cyan',    label='c') ; axes[1, 0].set_title('Cyan')
axes[1, 1].plot(ya, color='magenta', label='m') ; axes[1, 1].set_title('Magenta')
axes[1, 2].plot(ya, color='black',   label='k') ; axes[1, 2].set_title('Black')
fig.legend(prop={'size': 8}) # параметр prop позволяет передать словарь свойств
```

- Альтернативный способ получения нескольких графиков - последовательный вызовов метода add_subplot() объекта класса Figure:

```
fig.add_subplot(2, 3, 1) # 2 rows, 3 columns, plot No. 1
fig.add_subplot(232)     # 2 rows, 3 columns, plot No. 2
```

Пример нескольких графиков



Аргументы sharex, sharey и fig_kw

- Функция subplots имеет именованные аргументы *sharex* и *sharey*
- Значения аргументов:
  - True или 'all'
  - False или 'none'
  - 'row'
  - 'column'
- Значения передаваемые через эти аргументы говорят о том, что соответствующая ось (X или Y) *разделяется* всеми графиками, или всеми в одной строке или всеми в одном столбце. Таким образом для графиков гарантируется одинаковый масштаб.
- Аргументы sharex и sharey имеют значения по умолчанию False
- Аргумент *fig_kw* это словарь, который используется как набор именованных аргументов при создании объекта класса Figure

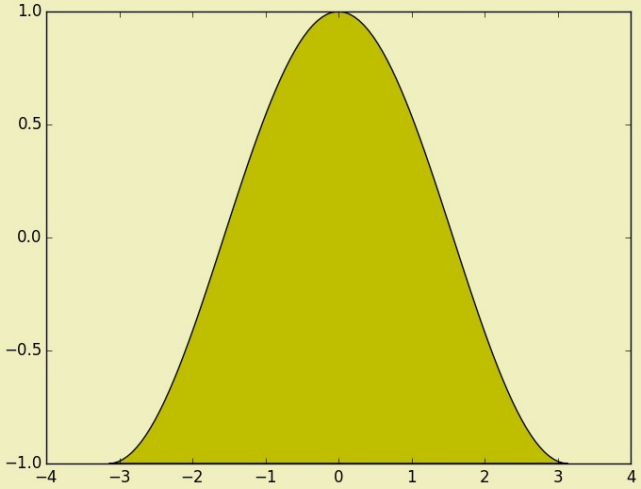
Стили отображения

- В библиотеку включен ряд предопределенных стилей отображения
- Атрибут *available* субмодуля *style* содержит список доступных стилей
- Функция *use* субмодуля *style* устанавливает стиль для последующей отрисовки графиков
- Примеры:

```
for style in plt.style.available:
  plt.style.use(style)
  plt.plot(x, y)
  plt.show()
  plt.close()

my_preferred_style = 'seaborn-bright'
if my_preferred_style in plt.style.available:
  plt.style.use(my_preferred_style)
plt.plot(x, y)
plt.show()
```

Заливка графика

```
plt.fill(xa, np.cos(xa), 'y')
plt.fill_between(xa, np.cos(xa), np.cos(xa) * 0.7, color='y')
```



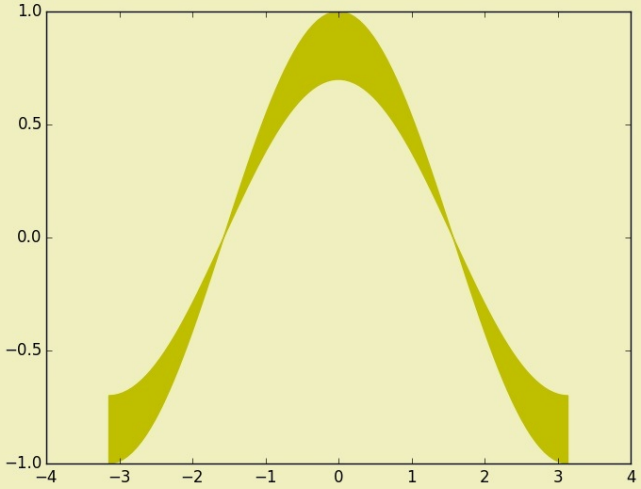
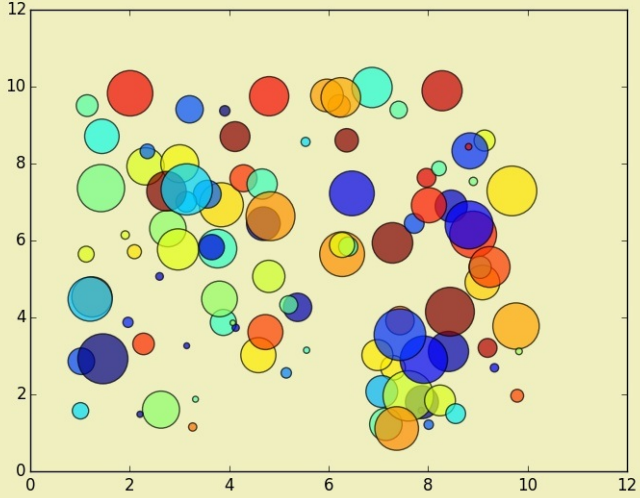


Диаграмма рассеяния, функция scatter()

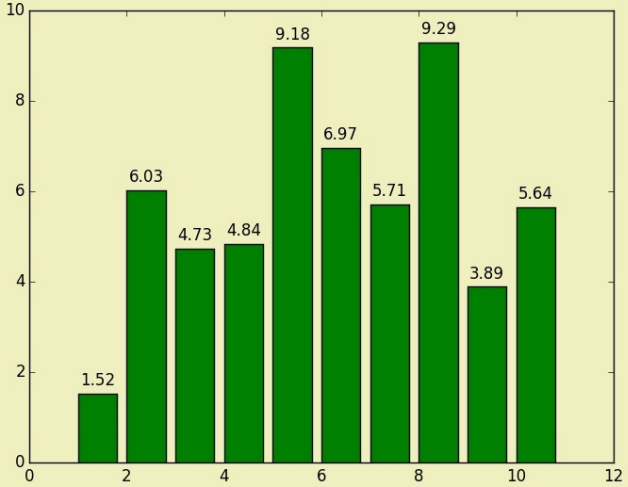
- Диаграмма рассеяния отображает события в виде точек на плоскости. Размер точки и ее цвет могут быть заданы в виде массива, что придает графику еще два измерения.

```
plt.scatter(ax, ay, c=az, s=at, alpha=0.7)
```



Столбчатая диаграмма, функция bar()

```
plt.bar(ax, ay, color='g')
for x, y in zip(ax, ay):
  plt.text(x + 0.4, y + 0.1, '%.2f' % y, ha='center', va='bottom')
```



- Функция text() выводит строку в заданных координатах

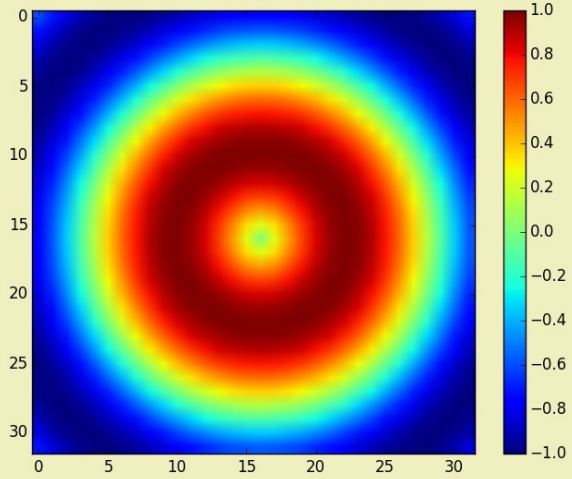
Функция imshow()

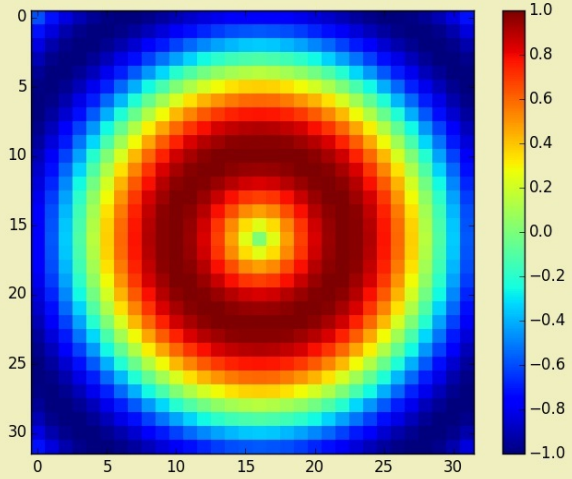
- Функция imshow() интерпретирует массив как растровое изображение
- Первый и единственный обязательный аргумент - отображаемый массив, в форме: (N, M), (N, M, 3) или (N, M, 4)
- Массив (N, M) интерпретируется как монохромное изображение (cmap='gray'), оно может конвертироваться в цветовую гамму при других значениях параметра cmap
- Массивы (N, M, 3) и (N, M, 4) интерпретируются как изображения в формате RGB и RGBA
- Некоторые именованные аргументы функции imshow():

```
- alpha - прозрачность от 0.0 (прозрачный) до 1.0 (сплошной)
- cmap - colormap, имя процедуры преобразования величины в цвет
- interpolation - интерполяция: 'none','nearest','bilinear','bicubic',
 'spline16','spline36','hanning','hamming','hermite','kaiser',
 'quadric','catrom','gaussian','bessel','mitchell','sinc','lanczos'
- shape - кортеж (columns, rows), явное указание размера изображения
```

Массив как цветовая карта

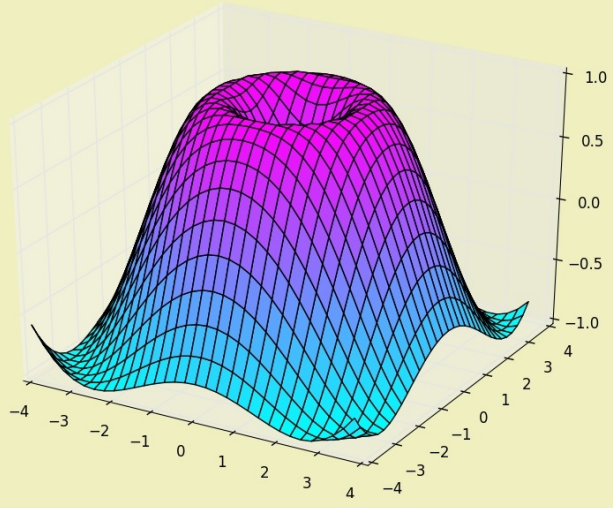
```
r = np.arange(-4, 4, 0.25)
ax, ay = np.meshgrid(r, r)
az = np.sin(np.sqrt(ax**2 + ay**2))
plt.imshow(az)                          # для левого графика
plt.imshow(az, interpolation='none') # для правого графика
plt.colorbar()
```





Трехмерный график

```
from mpl_toolkits.mplot3d import Axes3D
axes = Axes3D(plt.figure())
r = np.arange(-4, 4, 0.25)
ax, ay = np.meshgrid(r, r)
az = np.sin(np.sqrt(ax**2 + ay**2))
axes.plot_surface(ax, ay, az, rstride=1, cstride=1, cmap='cool')
```



Субмодуль image

- Субмодуль image содержит функции для работы с растровыми изображениями
- Традиционная инструкция импорта:

```
import matplotlib.image as mpimg
```

- Пример

```
import matplotlib.image as mpimg
image = mpimg.imread('ex01.jpg')
print(image.shape) # => (480, 640, 3)
print(image.dtype) # => uint8
plt.imshow(image)  # отобразить image в текущей figure
plt.show()
```

- Младшая координата это цвет точки в формате RGB

```
image[100, 200, 2] # => Синяя составляющая точки X=200, Y=100
```

Доступ к точкам растрового изображения

- Пример: обнуление цветовых плоскостей

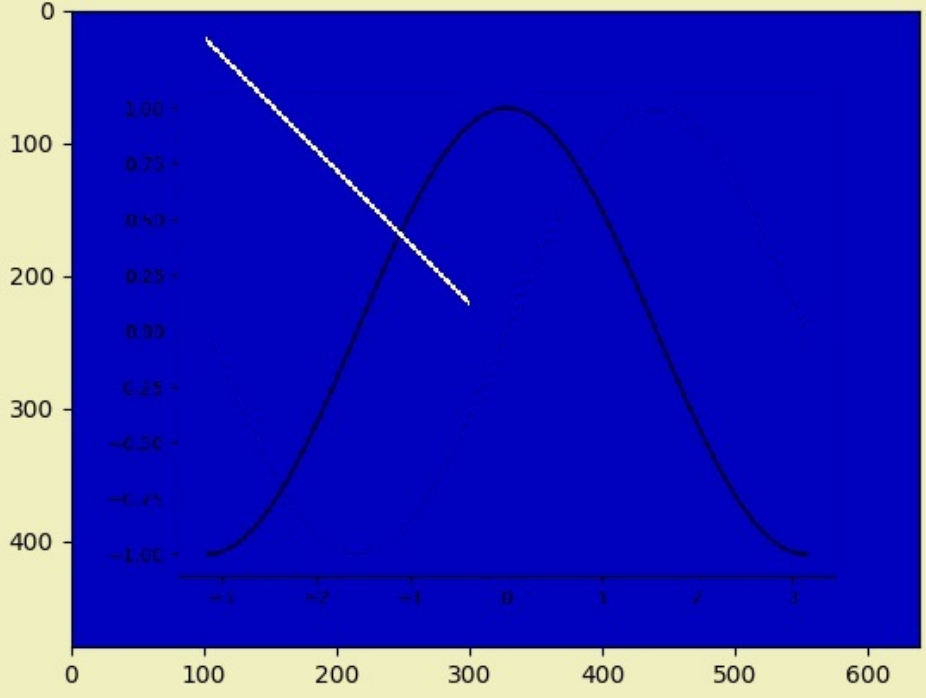
```
image[:,:,1:3]   = 0 # G = 0, B = 0 => Red
image[:,:,[0,2]] = 0 # R = 0, B = 0 => Green
image[:,:,0:2]   = 0 # R = 0, G = 0 => Blue
```

- Пример: белая линия шириной 4 пикселя

```
for i in range(1, 200):
  image[i + 20 : i + 24, i + 100, :] = 255
```

- У растровых изображений начало координат находится в *левом верхнем* углу, а ось Y направлена вниз

Пример: белая линия на синем фоне



Загрузка и сохранение изображений

```
mpimg.imread(fname, format=None)
 - загружает растровое изображение из файла в массив ndarray
   fname  - имя файла
   format - формат файла, если не задан определяется по расширению
            имени файла

mpimg.imsave(fname, arr, vmin=None, vmax=None, cmap=None, format=None,
             origin=None, dpi=100)
 - сохраняет в файле растровое изображение из массива ndarray
   fname  - имя файла
   arr    - массив
   vmin, vmax - ограничение величин цвета
   cmap   - cmap - colormap, имя процедуры преобразования величины в цвет
   format - формат файла, если не задан определяется по расширению
            имени файла
   origin - начало координат: 'upper' или 'lower'
   dpi    - разрешение в точках на дюйм для сохранения
            в метаданных файла
```

Литература к лекции

1. http://matplotlib.org/contents.html

2. Sandro Tosi. "Matplotlib for Python Developers". Packt Publishing, 2009, ISBN 978-1-847197-90-0

3. Alexandre Devert. "Matplotlib Plotting Cookbook". Packt Publishing, 2014, ISBN 978-1-84951-326-5

4. Jake VanderPlas. "Python Data Science Handbook". O'Reilly Media, 2016, ISBN 978-1491912058