

Python

Библиотеки numpy, scipy, matplotlib, PIL

Пржибельский Андрей,
Соколов Артем.

NumPy

Массивы

- Основым является тип массивов постоянной длины с однотипными элементами
 - Повышается быстродействие
 - Введены поэлементные операции на массивах
- Создание:
 - `>>> x = np.array([2,3,1,0])`
 - `>>> x = np.array([[1,2.0],[0,0]],[1+1j,3.])` # note mix of tuple and lists, and types
 - `>>> x = np.array([[1.+0.j, 2.+0.j], [0.+0.j, 0.+0.j], [1.+1.j, 3.+0.j]])`

Типы данных

Data type	Description
bool	Boolean (True or False) stored as a byte
int	Platform integer (normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float	Shorthand for float64.
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex	Shorthand for complex128.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

Массивы (2)

■ Другие способы задания

- `>>> np.zeros(2, 3)`
`array([[0., 0., 0.], [0., 0., 0.]])`
- `>>> np.arange(10)`
`array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])`
- `>>> np.arange(2, 10, dtype=np.float)`
`array([2., 3., 4., 5., 6., 7., 8., 9.])`
- `>>> np.arange(2, 3, 0.1)`
`array([2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9])`
- `>>> np.linspace(1., 4., 6)`
`array([1. , 1.6, 2.2, 2.8, 3.4, 4.])`
- `>>> np.indices((3,3))`
`array([[[[0, 0, 0], [1, 1, 1], [2, 2, 2]], [[0, 1, 2], [0, 1, 2], [0, 1, 2]]])`

Многомерные массивы

- `>>> x = np.array([[1, 2, 3], [4, 5, 6]], np.int32)`
- `>>> type(x)`
`<type 'numpy.ndarray'>`
- `>>> x.shape`
`(2, 3)`
- `>>> x.dtype`
`dtype('int32')`
- `>>> a = np.array([[1,2,3], [4,5,6]])`
- `>>> np.reshape(a, 6)`
`array([1, 2, 3, 4, 5, 6])`
- `>>> np.reshape(a, (3,-1))` `# the unspecified value is inferred to be 2`
- `array([[1, 2],`
 `[3, 4],`
 `[5, 6]])`

Индексация и слайсинг

- `>>> x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])`

- `>>> x[1:7:2]`
`array([1, 3, 5])`

- `>>> x[-2:10]`
`array([8, 9])`

- `>>> x[-3:3:-1]`
`array([7, 6, 5, 4])`

- `>>> x = np.array([[[1],[2],[3]], [[4],[5],[6]]])`

- `>>> x.shape`
`(2, 3, 1)`

- `>>> x[...,0]`
`array([[1, 2, 3],
 [4, 5, 6]])`

- `>>> x[:,np.newaxis,:,:].shape`
`(2, 1, 3, 1)`

Многочлены

- `>>> p = poly1d([3,4,5])`
- `>>> print p`
$$x^2 + 4x + 3$$
- `>>> print p*p`
$$9x^4 + 24x^3 + 46x^2 + 40x + 25$$
- `>>> print p.integ(k=6)`
$$\frac{1}{6}x^3 + 2x^2 + 5x + 6$$
- `>>> print p.deriv()`
$$2x + 4$$

▪

SciPy

Линейная алгебра (linalg)

- `>>> A = mat('[1 3 5; 2 5 1; 2 3 8]')`

- `>>> A`

```
Matrix([[1, 3, 5],  
        [2, 5, 1],  
        [2, 3, 8]])
```

- `>>> A.I`

```
matrix([[-1.48, 0.36, 0.88],  
        [ 0.56, 0.08, -0.36],  
        [ 0.16, -0.12, 0.04]])
```

- `>>> from scipy import linalg`

- `>>> linalg.inv(A)`

```
array([[-1.48, 0.36, 0.88],  
       [ 0.56, 0.08, -0.36],  
       [ 0.16, -0.12, 0.04]])
```


Линейная алгебра (2)

- `>>> A = mat('[1 3 5; 2 5 1; 2 3 8]')`

- `>>> b = mat('[10;8;3]')`

- `>>> A.l*b`

```
matrix([[ -9.28],  
        [ 5.16],  
        [ 0.76]])
```

- `>>> linalg.solve(A,b)`

```
array([[ -9.28],  
       [ 5.16],  
       [ 0.76]])
```

- `>>> A = mat('[1 3 5; 2 5 1; 2 3 8]')`

- `>>> linalg.det(A)`

```
-25.0000000000000004
```

Собственные значения и вектора

- `>>> A = mat('[1 5 2; 2 4 1; 3 6 2]')`
- `>>> la,v = linalg.eig(A)`
- `>>> l1,l2,l3 = la`
- `>>> print l1, l2, l3`
`(7.95791620491+0j) (-1.25766470568+0j) (0.299748500767+0j)`
- `>>> print v[:,0]`
`[-0.5297175 -0.44941741 -0.71932146]`
- `>>> print v[:,1]`
`[-0.90730751 0.28662547 0.30763439]`
- `>>> print v[:,2]`
`[0.28380519 -0.39012063 0.87593408]`

Разложения матриц

- Сингулярное разложение

```
>>> U,s,Vh = linalg.svd(A)
```

- LU-разложение

```
>>> P,L,U = linalg.lu(A)
```

```
>>> Lu, P = linalg.lu_factor(A)
```

```
>>> x = linalg.lu_solve((Lu, P),b)
```

- Разложение Холецкого

```
>>> U = linalg.cholesky(A)
```

- QR-разложение

```
>>> Q,R = linalg.qr(A)
```

Матричные функции и специальные матрицы

- Встроенные функции
 - `expm`, `logm`
 - `cosm`, `sinm`, `tanm`
 - `coshm`, `sinhm`, `tanhm`
 - `signm`
 - `sqrt`
- Задание произвольной функции
 - `>>> A = random.rand(3,3)`
 - `>>> B = linalg.funm(A,lambda x: special.jv(0,x))`
- Специальные матрицы
 - Блочно-диагональные, циркулянты, ван дер Монда, и т.д.

Интегрирование (integrate)

- Интегрирование явно заданных функций
 - Интегрирование функций, заданных массивами данных
 - Численное решение ОДУ
-
- ```
>>> result = integrate.quad(lambda x: special.jv(2.5,x), 0, 4.5)
```
  - ```
>>> print result
```



```
(1.1178179380783249, 7.8663172481899801e-09)
```

Численное решение ОДУ (integrate)

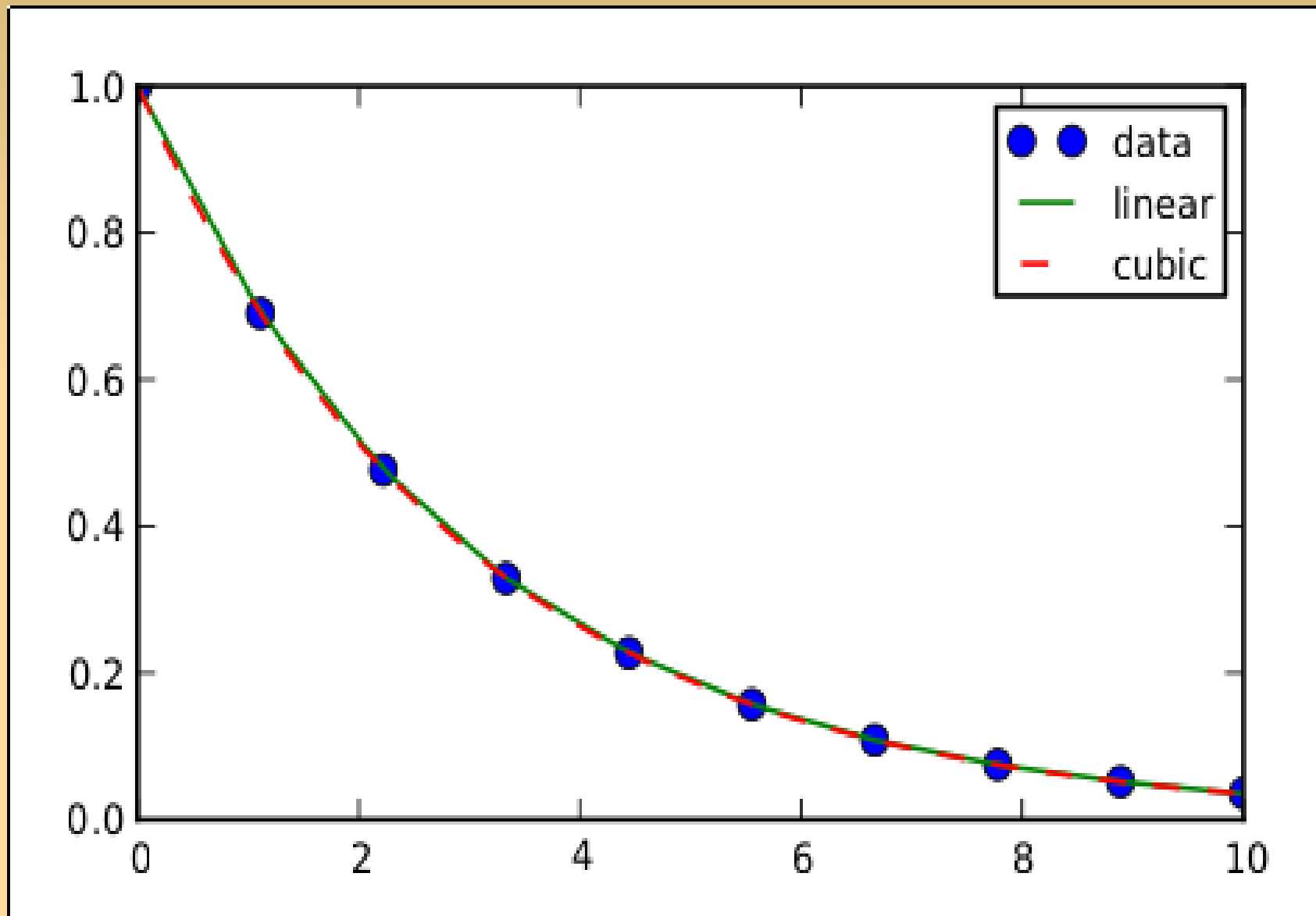
- `>>> y1_0 = 1.0/3**(2.0/3.0)/gamma(2.0/3.0)`
- `>>> y0_0 = -1.0/3**(1.0/3.0)/gamma(1.0/3.0)`
- `>>> y0 = [y0_0, y1_0]`
- `>>> def func(y, t):`
... `return [t*y[1],y[0]]`
- `>>> def gradient(y,t):`
... `return [[0,t],[1,0]]`
- `>>> x = arange(0,4.0, 0.01)`
- `>>> t = x`
- `>>> y = odeint(func, y0, t)`
- `>>> y2 = odeint(func, y0, t, Dfun=gradient)`
- `>>> print y[:36:6,1]`
`[0.355028 0.339511 0.324067 0.308763 0.293658 0.278806]`
- `>>> print y2[:36:6,1]`
`[0.355028 0.339511 0.324067 0.308763 0.293658 0.278806]`

Интерполяция (interpolate)

- `>>> from scipy.interpolate import interp1d`
- `>>> x = np.linspace(0, 10, 10)`
- `>>> y = np.exp(-x/3.0)`
- `>>> f = interp1d(x, y)`
- `>>> f2 = interp1d(x, y, kind='cubic')`

- `>>> xnew = np.linspace(0, 10, 40)`
- `>>> import matplotlib.pyplot as plt`
- `>>> plt.plot(x,y,'o',xnew,f(xnew),'-', xnew, f2(xnew),'--')`
- `>>> plt.legend(['data', 'linear', 'cubic'], loc='best')`
- `>>> plt.show()`

Интерполяция (2)



Преобразование Фурье (fftpack)

- Существуют прямые и обратные дискретные преобразования Фурье:
 - Все функции принимают массив `x` и необязательные параметры `n`, `axis`, `overwrite_x`
 - Для вещественных значений `rfft`, `irfft`
 - Для произвольных типов данных `fft`, `ifft`
 - Двумерное преобразование `fft2d`, `ifft2d`
 - Многомерное `fftn`, `ifftn`

Случайные числа и статистика (stats)

- 84 непрерывных распределения
- 12 дискретных распределений
- Более 70 статистических функций
- Основные методы для распределений
 - Случайная величина (rvs)
 - Плотность вероятности (pdf)
 - Функция распределения (cdf)
 - Статистика (stats)

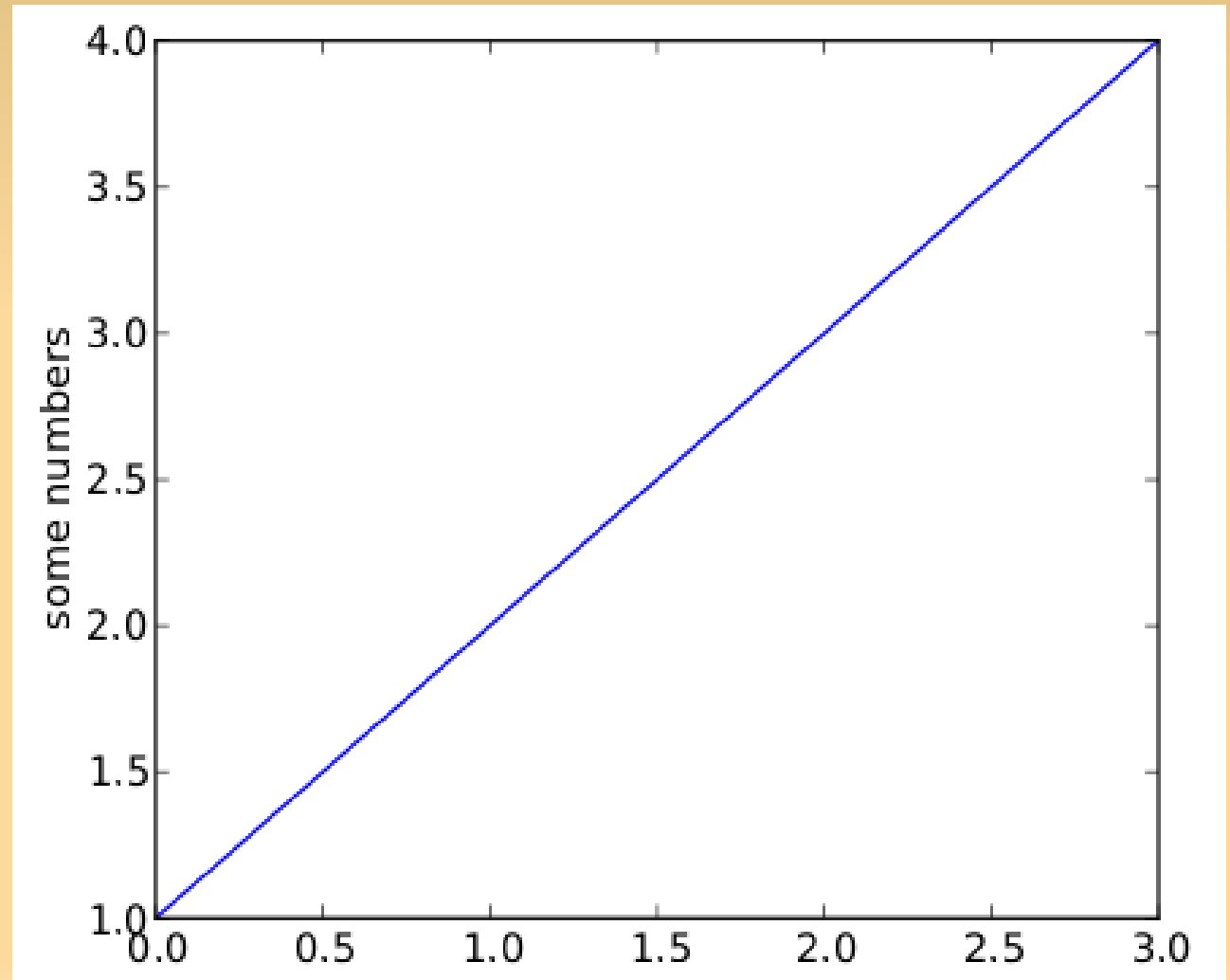
Прочие возможности SciPy

- Константы (`scipy.constants`)
- Специальные функции (`scipy.special`)
- Оптимизация (`scipy.optimize`)
- Чтение и запись различных форматов (`scipy.io`)
- Включение кода на C/C++ (`scipy.weave`)

Matplotlib

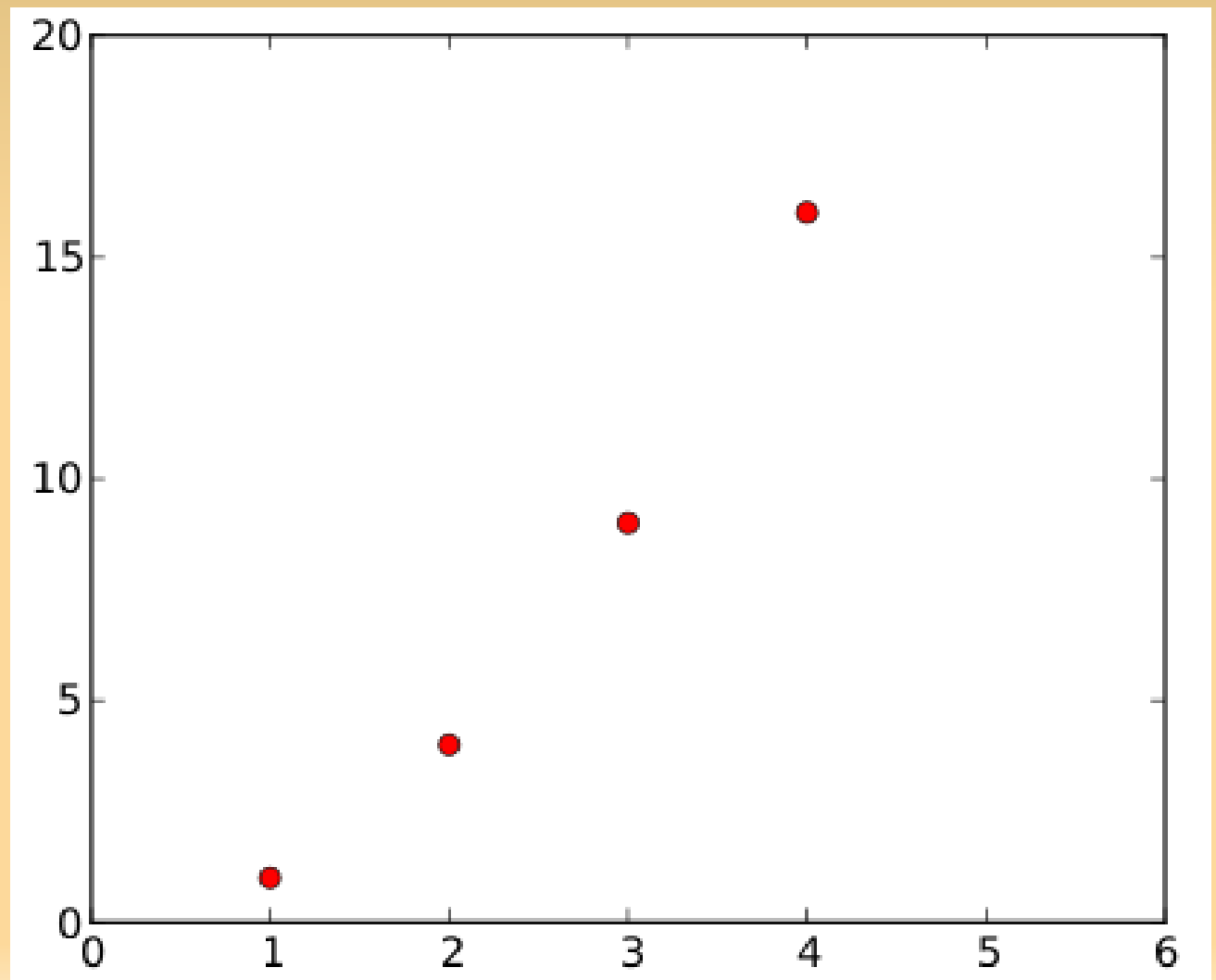
Простейший график

- `import numpy as np`
- `import matplotlib.pyplot as plt`
- `plt.plot([1,2,3,4])`
- `plt.ylabel('some numbers')`
- `plt.show()`



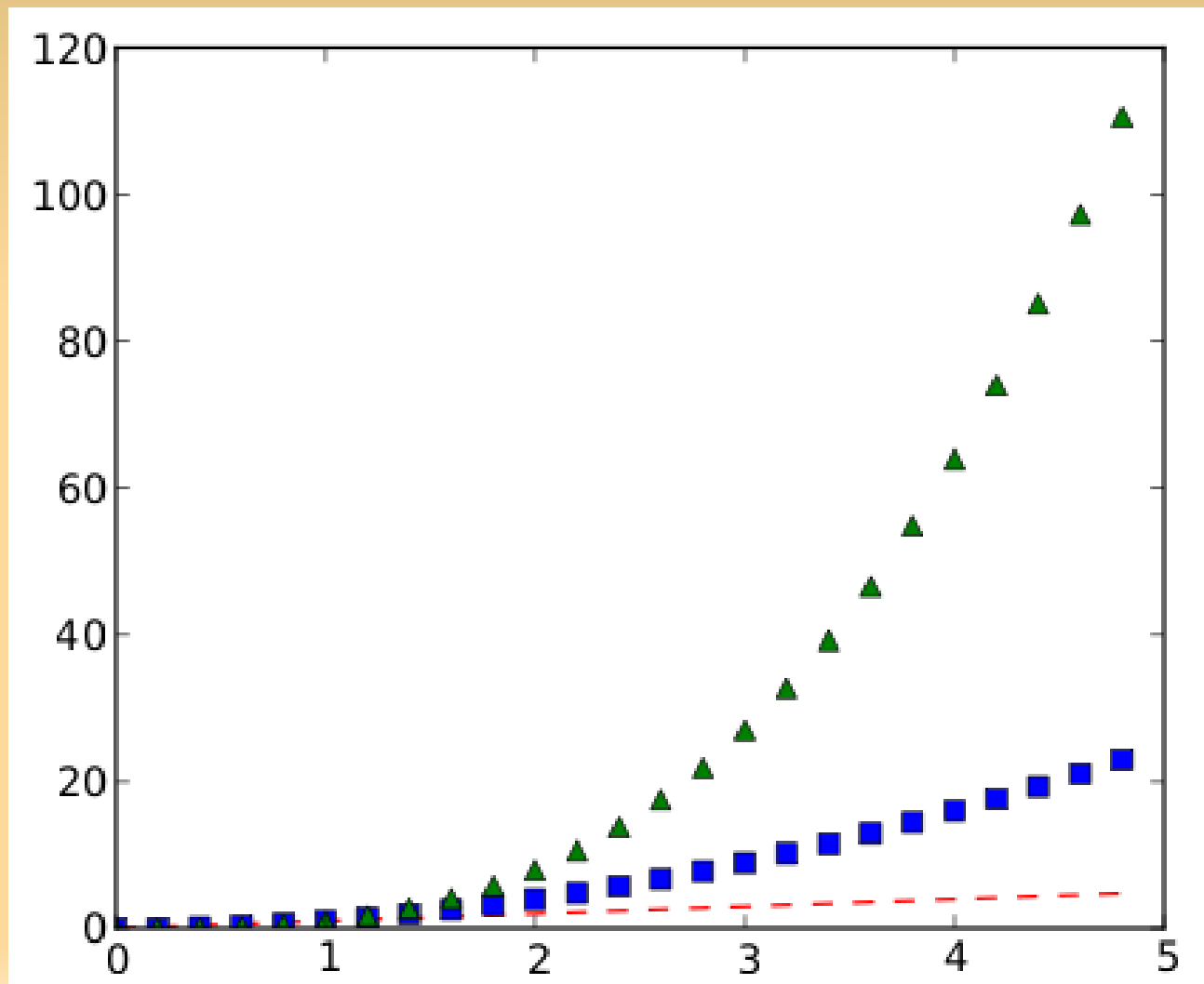
Простой график

- `plt.plot([1,2,3,4], [1,4,9,16], 'ro')`
- `plt.axis([0, 6, 0, 20])`



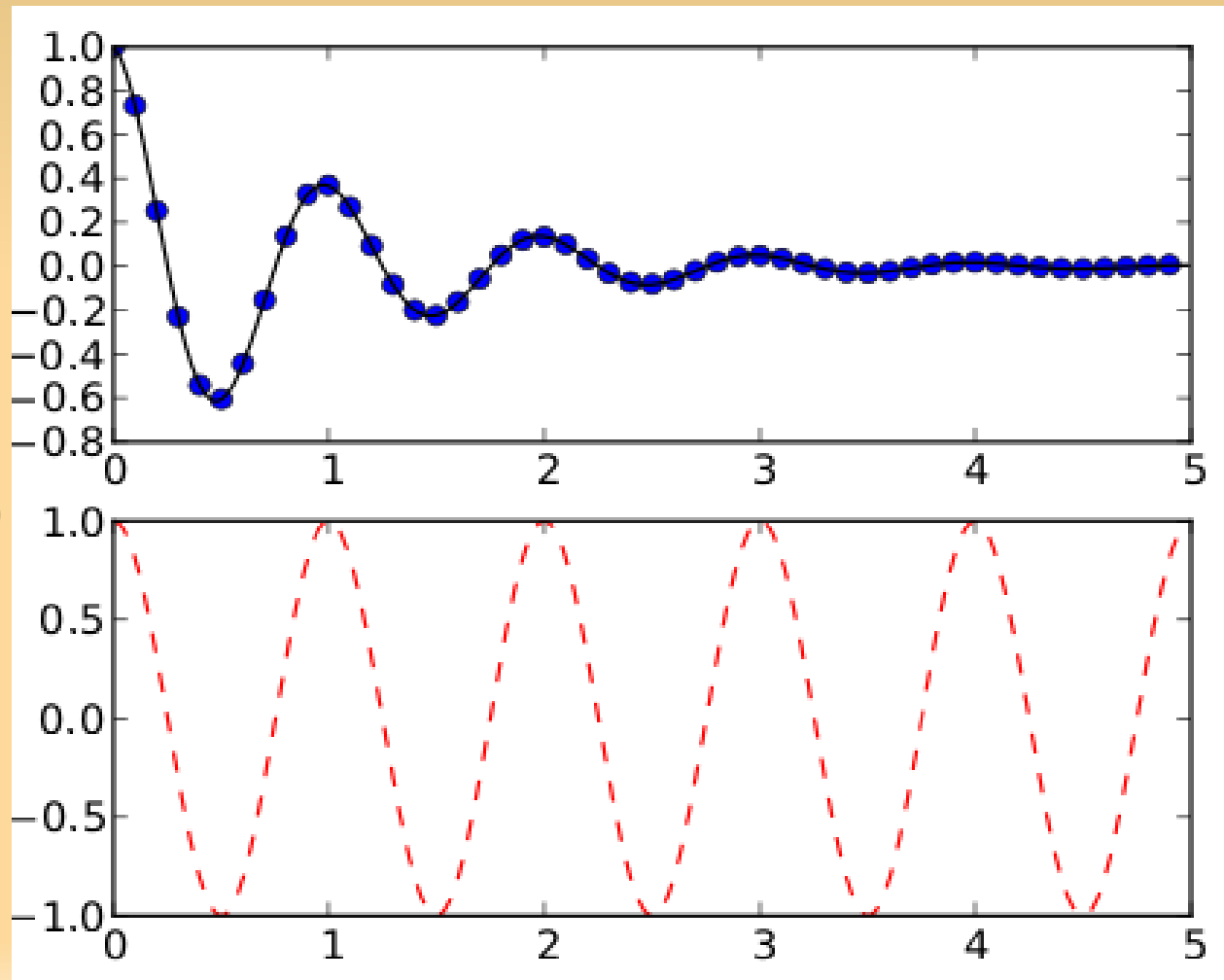
Несколько графиков

- `t = np.arange(0., 5., 0.2)`
- `plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')`



Несколько графиков на различных осях

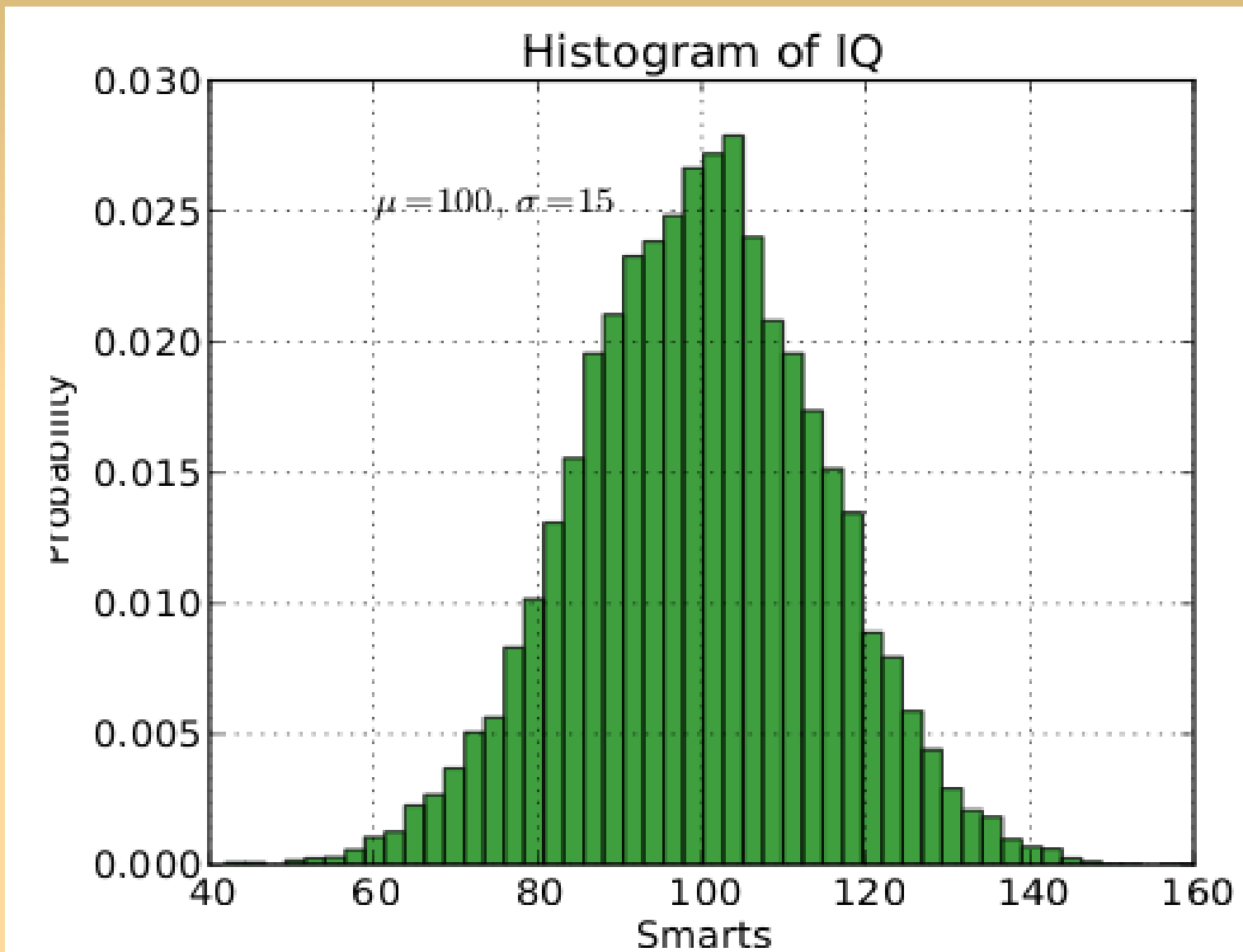
- def f(t):
- return np.exp(-t) * np.cos(2*np.pi*t)
- t1 = np.arange(0.0, 5.0, 0.1)
- t2 = np.arange(0.0, 5.0, 0.02)
- plt.figure(1)
- plt.subplot(211)
- plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
- plt.subplot(212)
- plt.plot(t2, np.cos(2*np.pi*t2), 'r--')



Подписи на графике

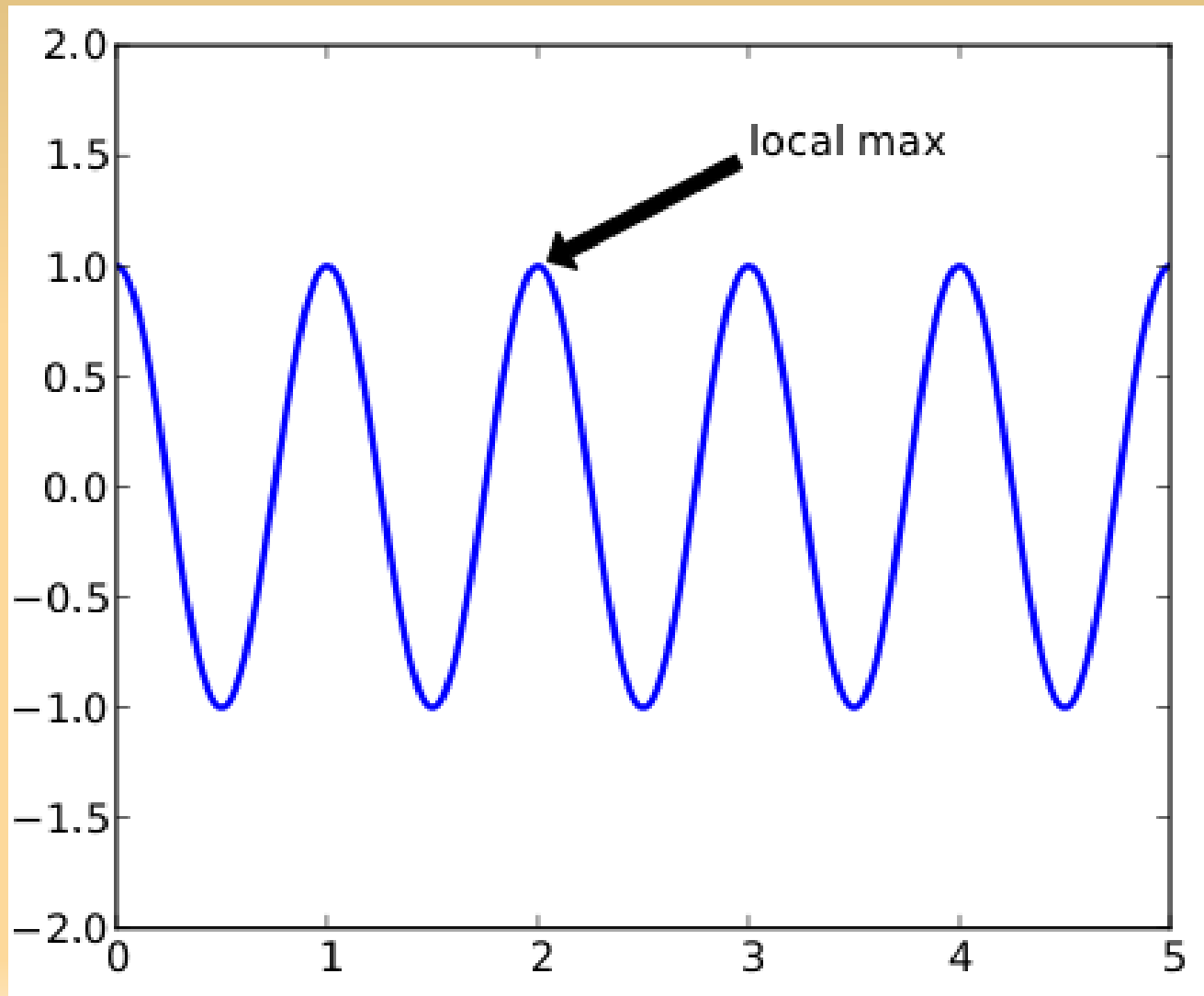
- `mu, sigma = 100, 15`
- `x = mu + sigma * np.random.randn(10000)`
- `n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)`
- `plt.xlabel('Smarts')`
- `plt.ylabel('Probability')`
- `plt.title('Histogram of IQ')`
- `plt.text(60, .025, r'$\mu=100,\ \sigma=15$')`
- `plt.axis([40, 160, 0, 0.03])`
- `plt.grid(True)`

Подписи на графике

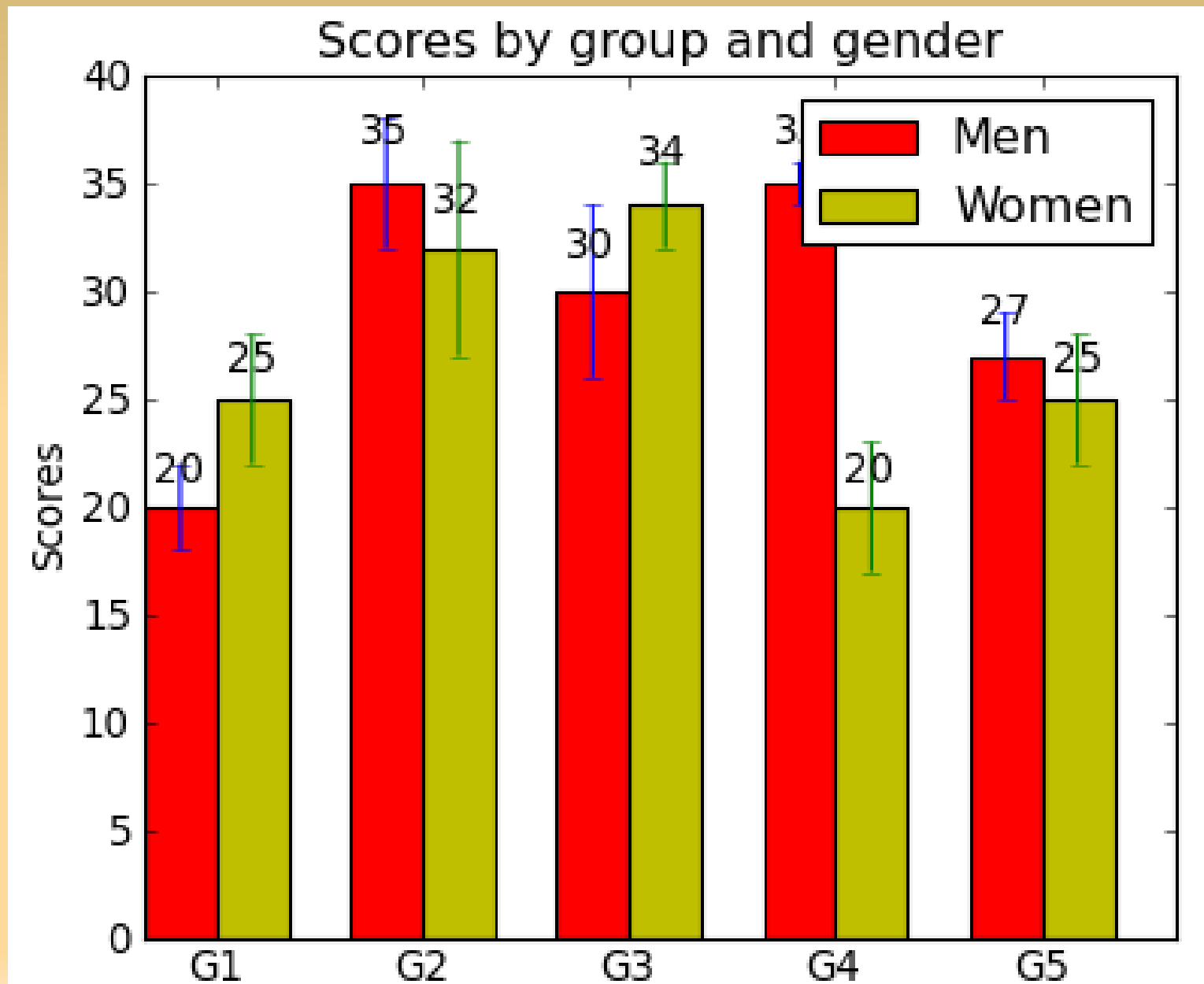


Подписи на графике (2)

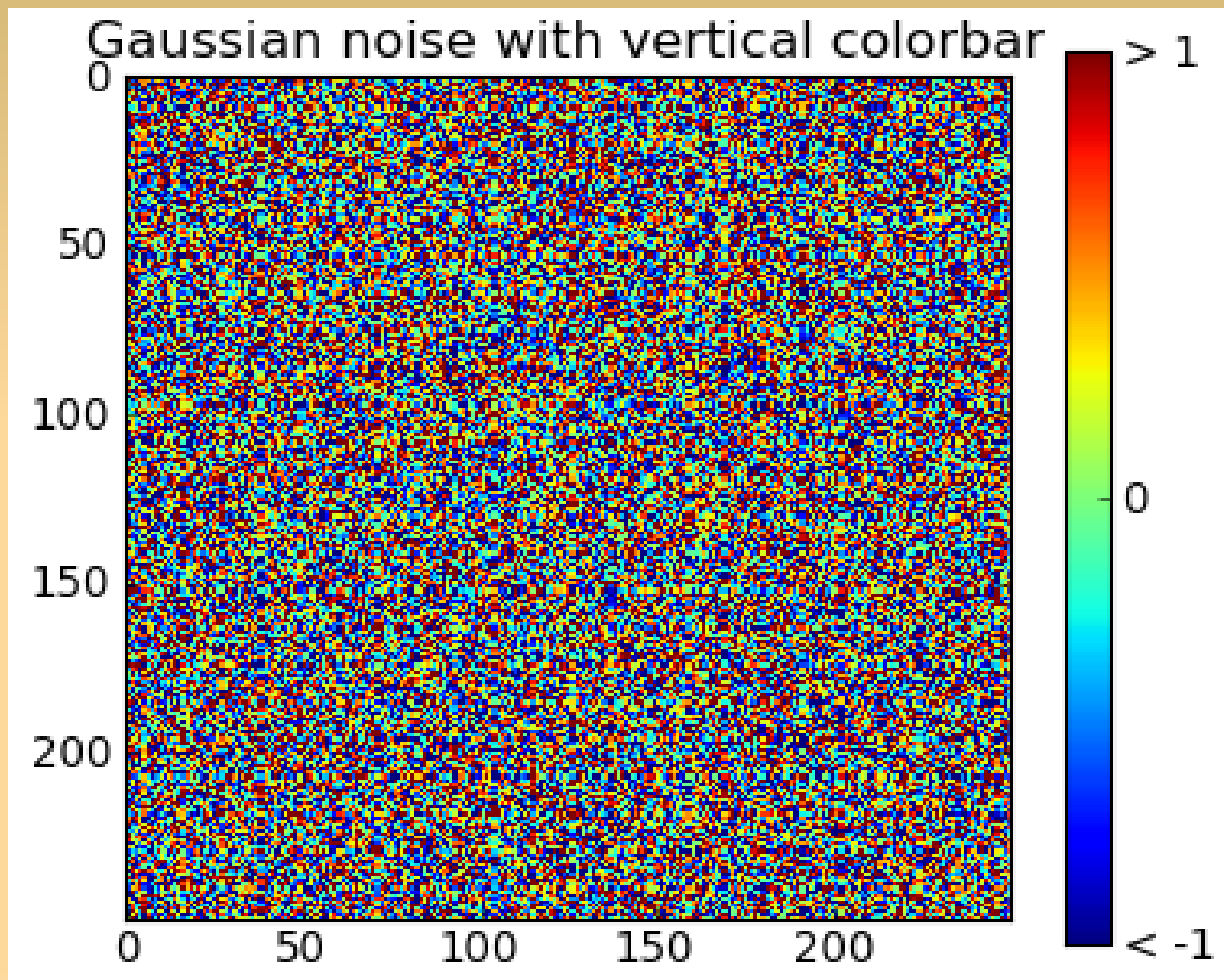
```
plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5),  
            arrowprops=dict(facecolor='black', shrink=0.05),)
```



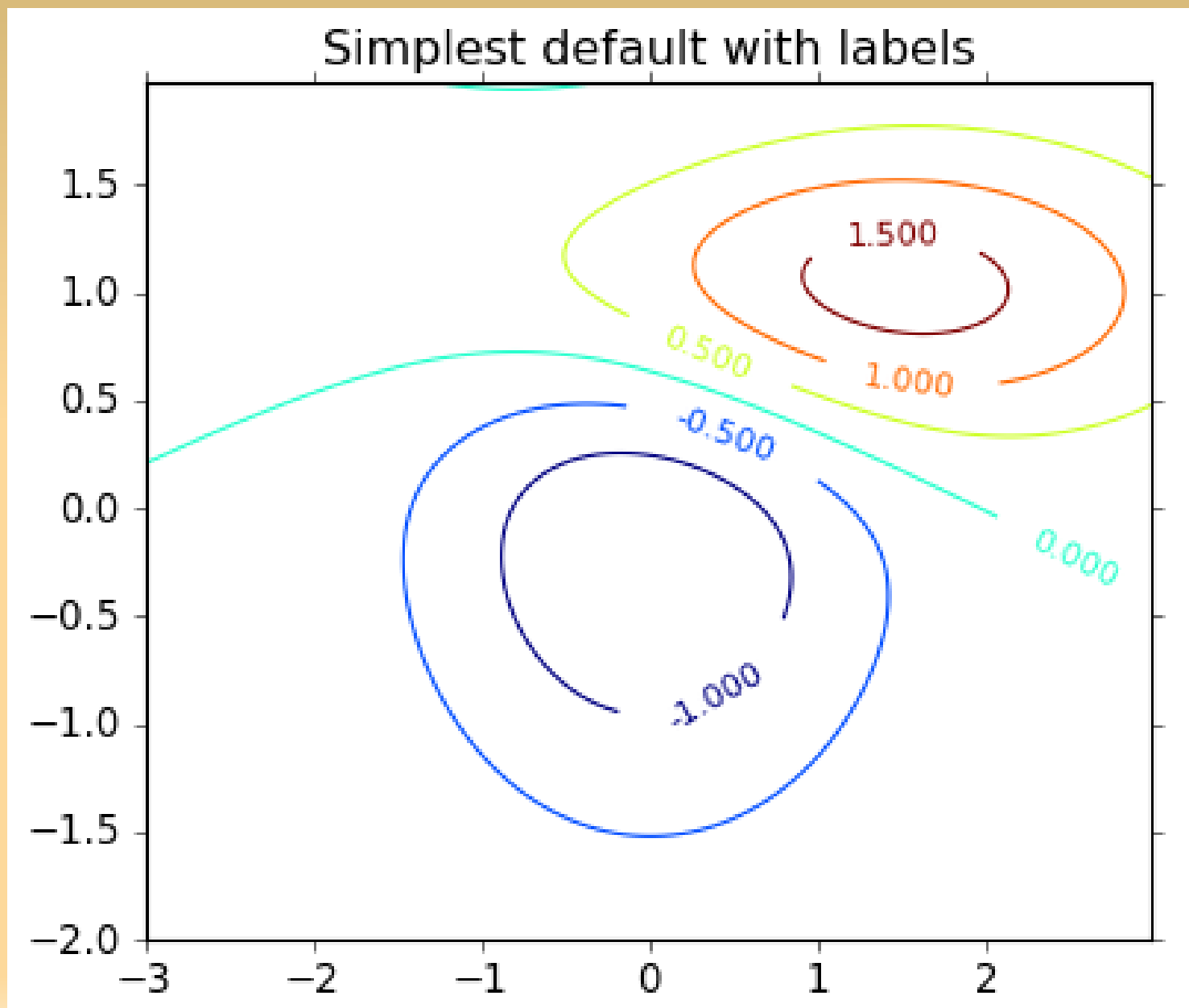
Гистограммы



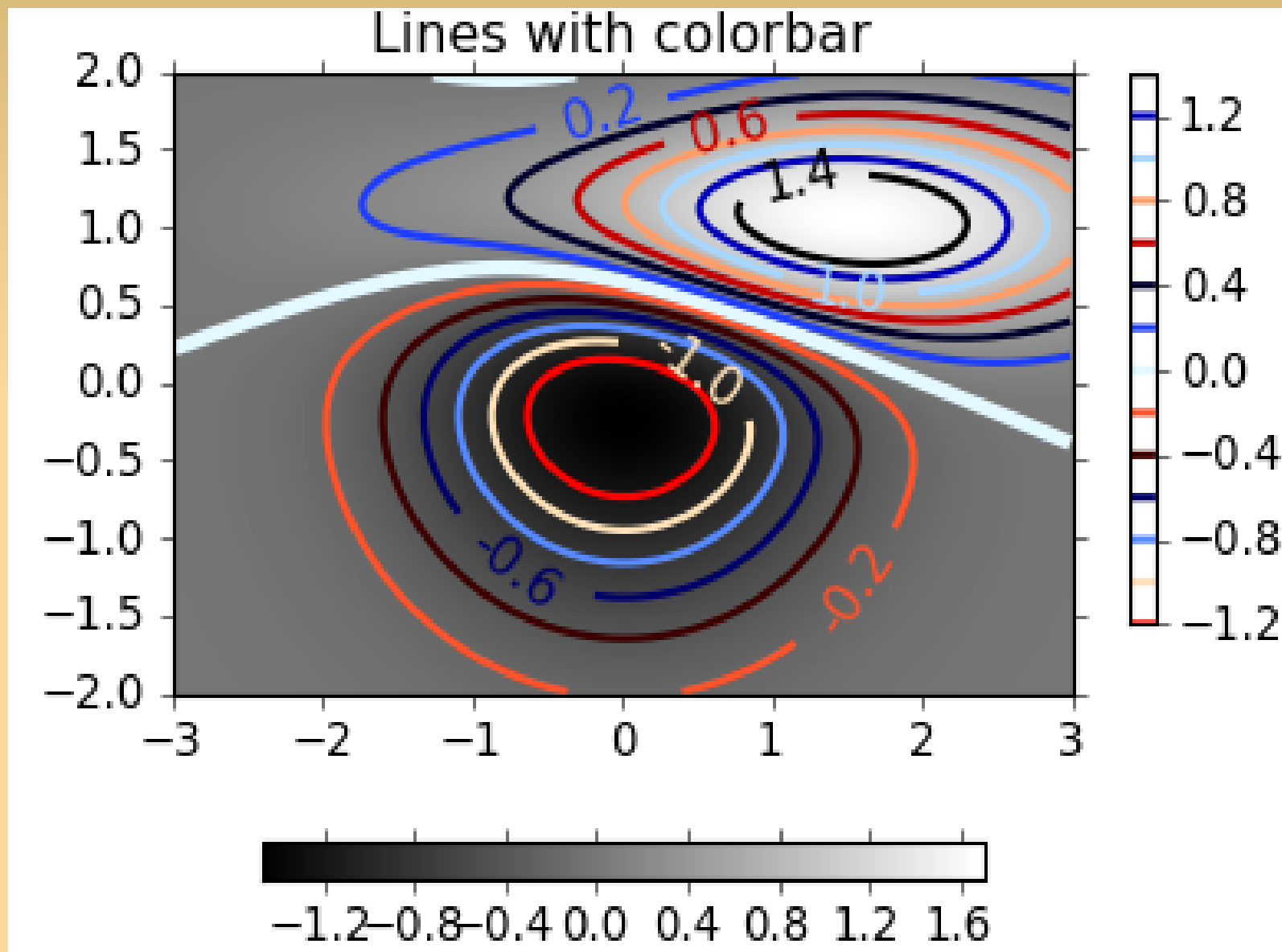
Двумерный гауссовский шум



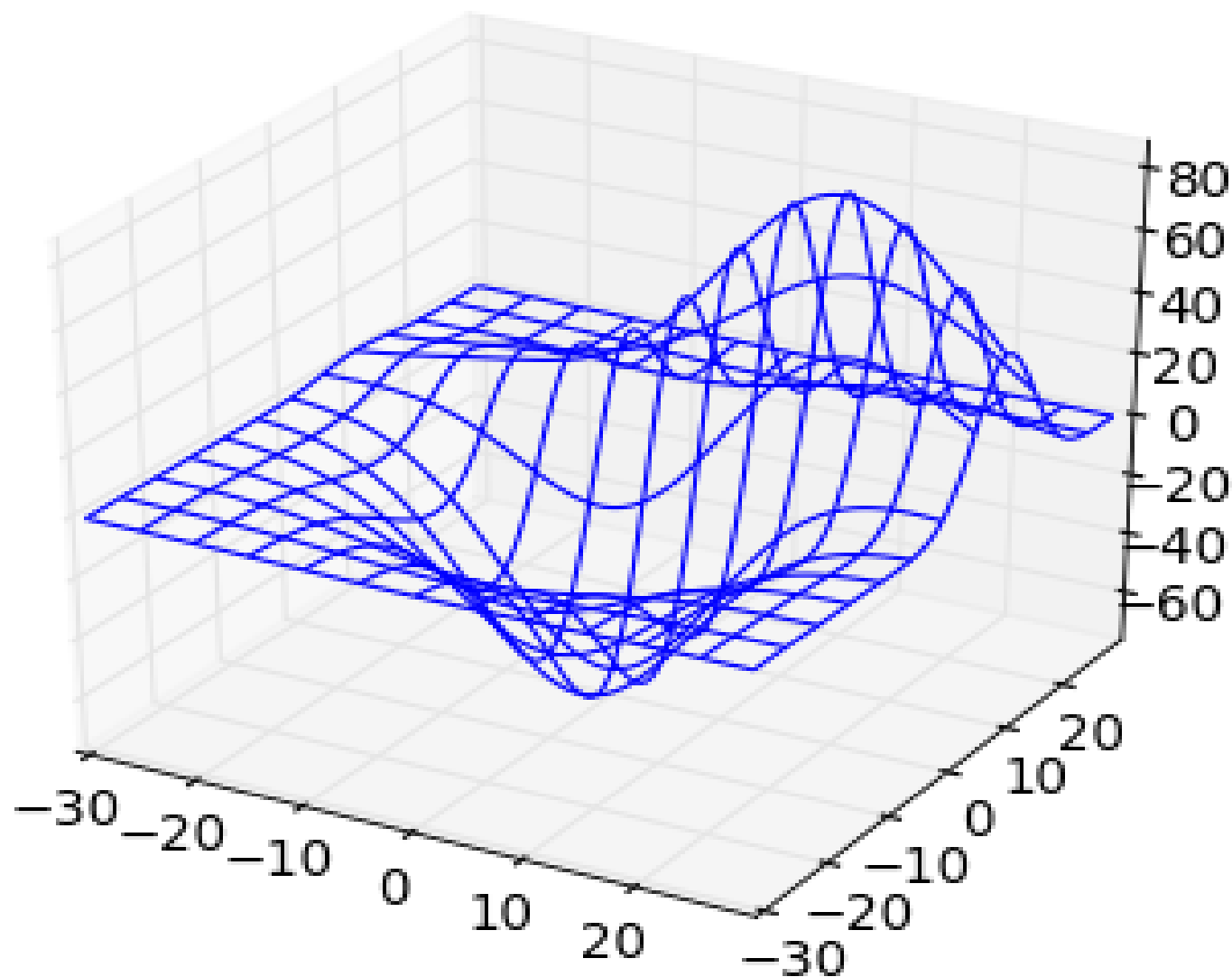
Контуры



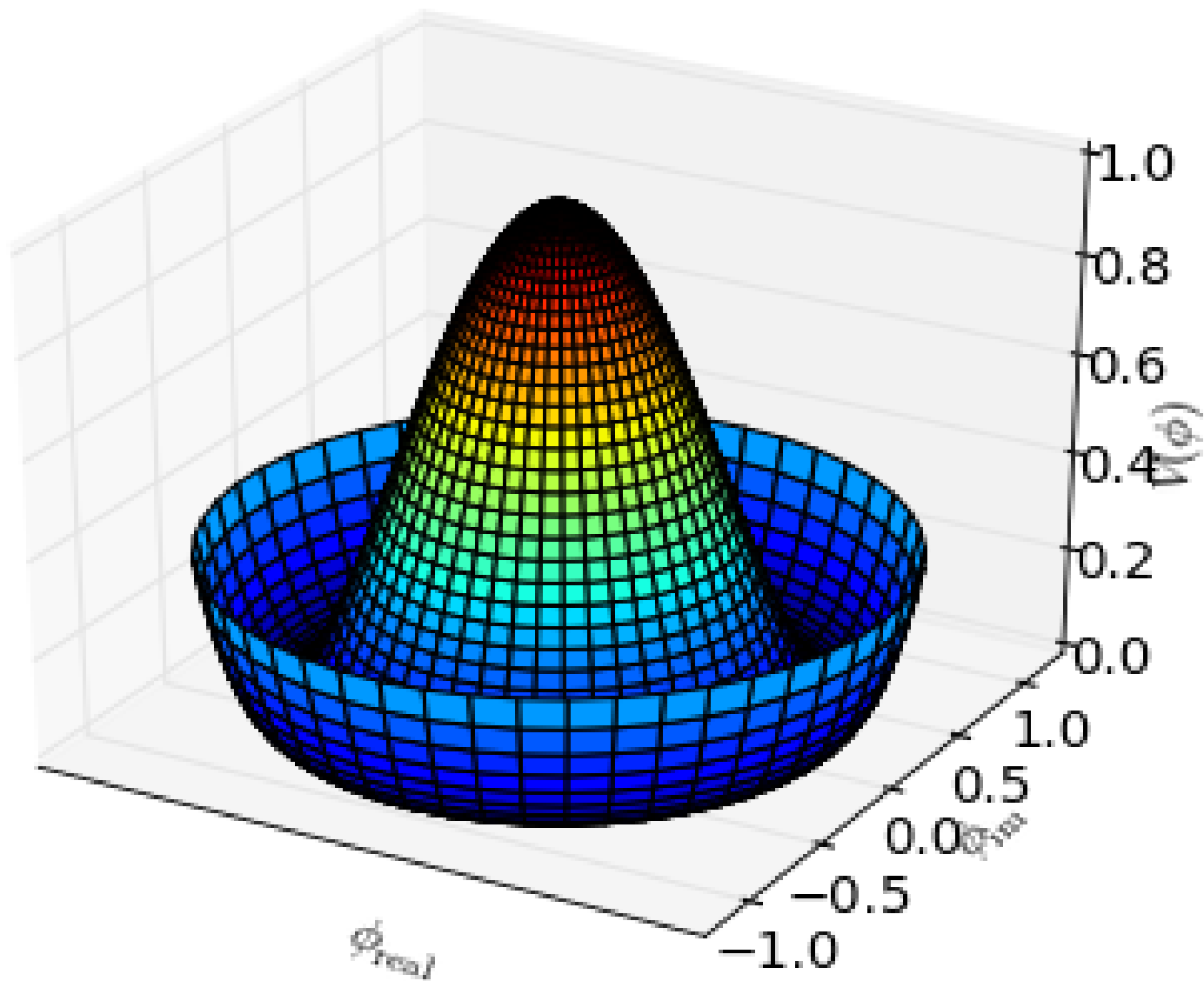
Еще контуры



3D графики



3D графики



Python Imaging Library

- `>>> import Image`
- `>>> im = Image.open("lena.ppm")`
- `>>> print im.format, im.size, im.mode`
PPM (512, 512) RGB
- `>>> im.show()`
- Для вызова этого метода необходима утилита xv

Открытие и сохранение изображений

- `open(infile)` – чтение изображения с диска, формат распознается по содержимому файла
- `save(outfile [,format])` – формат распознается либо отдельным параметром, либо по расширению файла, указанному в `outfile`

```
import os, sys, Image

for infile in sys.argv[1:]:
    f, e = os.path.splitext(infile)
    outfile = f + ".jpg"
    if infile != outfile:
        try:
            Image.open(infile).save(outfile)
        except IOError:
            print "cannot convert", infile
```

Создание превьюшек

```
import os, sys

import Image

size = 128, 128

for infile in sys.argv[1:]:
    outfile = os.path.splitext(infile)[0] + ".thumbnail"
    if infile != outfile:
        try:
            im = Image.open(infile)
            im.thumbnail(size)
            im.save(outfile, "JPEG")
        except IOError:
            print "cannot create thumbnail for", infile
```

Преобразования изображений

- Вырезка

```
box = (100, 100, 400, 400)
```

```
region = im.crop(box)
```

- Вставка

```
region = region.transpose(Image.ROTATE_180)
```

```
im.paste(region, box)
```

- Разделение по цветовым компонентам

```
r, g, b = im.split()
```

```
im = Image.merge("RGB", (b, g, r))
```

Преобразования изображений

- Изменение размера

```
out = im.resize((128, 128))
```

- Поворот и отражение

```
out = im.transpose(Image.FLIP_LEFT_RIGHT)
```

```
out = im.transpose(Image.FLIP_TOP_BOTTOM)
```

```
out = im.transpose(Image.ROTATE_90)
```

```
out = im.transpose(Image.ROTATE_180)
```

```
out = im.transpose(Image.ROTATE_270)
```

- Произвольный поворот

```
out = im.rotate(45) # degrees counter-clockwise
```

Фильтры изображений

- `im1 = im.filter(ImageFilter.BLUR)`
- BLUR
- CONTOUR
- DETAIL
- EDGE_ENHANCE
- FIND_EDGES
- SMOOTH
- SHARPEN
-
- `enh = ImageEnhance.Contrast(im)`
- `enh.enhance(1.3).show("30% more contrast")`

Поточечные операции

- `source = im.split()`
- `R, G, B = 0, 1, 2`
-
- `# select regions where red is less than 100`
- `mask = source[R].point(lambda i: i < 100 and 255)`
- `# process the green band`
- `out = source[G].point(lambda i: i * 0.7)`
- `# paste the processed band back, but only where red was < 100`
- `source[G].paste(out, None, mask)`
- `# build a new multiband image`
- `im = Image.merge(im.mode, source)`

Анимация

- `im = Image.open("animation.gif")`
- `im.seek(1) # skip to the second frame`
-
- `try:`
- `while 1:`
- `im.seek(im.tell()+1)`
- `# do something to im`
- `except EOFError:`
- `pass # end of sequence`

Спасибо за внимание!