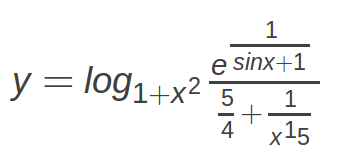
[Запись алгебраических выражений](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html" \l "id6)

Фундаментальная задача программирования -- вычисление математических и, в частности, алгебраических функций. Казалось бы, что проще? Однако, запись выражения на языке математики не принимается напрямую языком программирования. Выражение нужно линеаризовать.

Например, y = x^2, должно быть записано как y = x\*x или y = x\*\*2.

[Упражнение №1](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id7)

Линеаризуйте выражение, заданное формулой



И найдите его значения в точках 1, 10, 1000.

[Построение графиков](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id8)

matplotlib - набор дополнительных модулей (библиотек) языка Python. Предоставляет средства для построения самых разнообразных 2D графиков и диаграмм данных. Отличается простотой использования - для построения весьма сложных и красочно оформленных диаграмм достаточно нескольких строк кода. При этот качество получаемых изображений более чем достаточно для их публикования. Также позволяет сохранять результаты в различных форматах, например Postscript, и соответственно, вставлять изображения в документы TeX. Предоставляет API для встраивания своих графических объектов в приложения пользователя.

Пример построения графика функции:

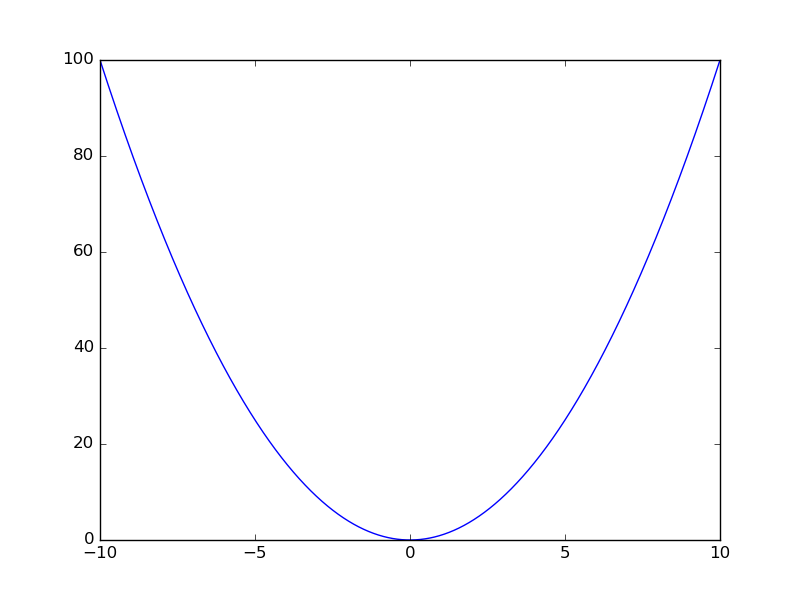
**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

x=np.arange(-10,10.01,0.01)

plt.plot(x,x\*\*2)

plt.show()



На одном рисунке можно построить несколько графиков функций:

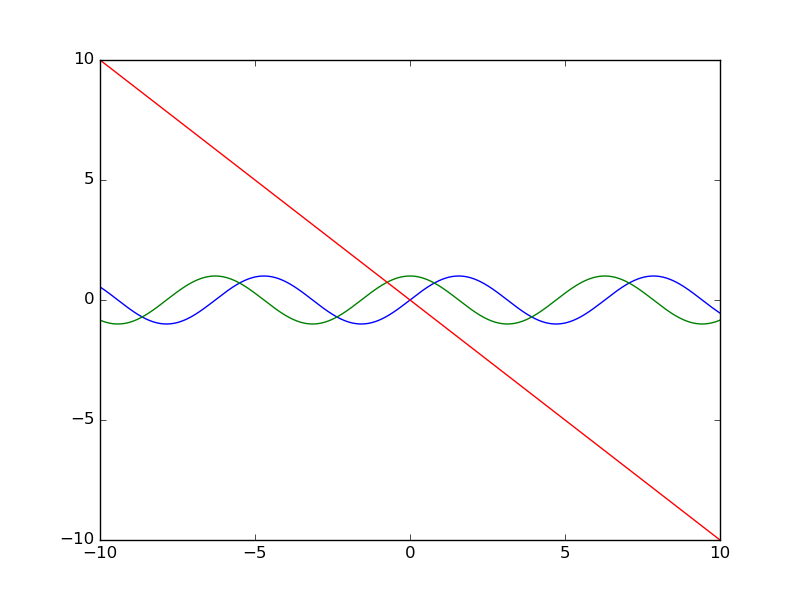
**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

x=np.arange(-10,10.01,0.01)

plt.plot(x,np.sin(x),x,np.cos(x),x,-x)

plt.show()



Также довольно просто на график добавить служебную информацию и отобразить сетку:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

x=np.arange(-10,10.01,0.01)

plt.plot(x,np.sin(x),x,np.cos(x),x,-x)

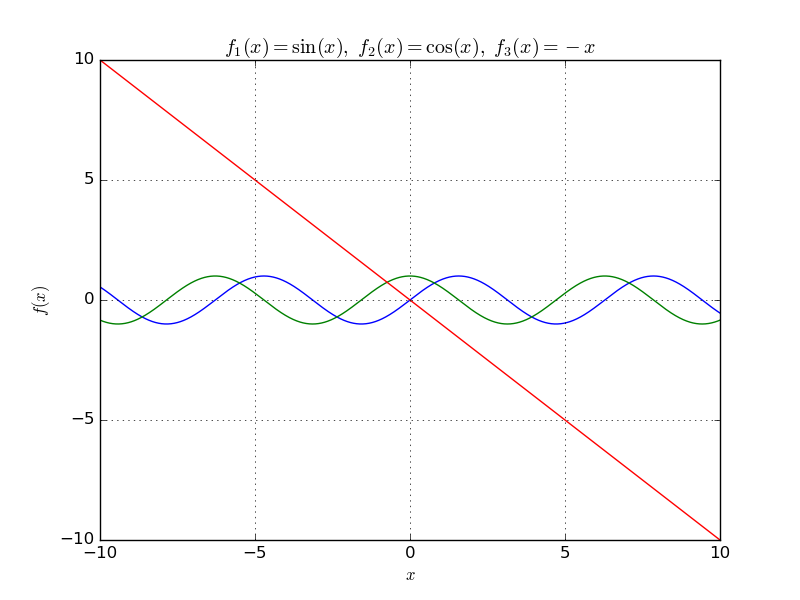
plt.xlabel(r'$x$')

plt.ylabel(r'$f(x)$')

plt.title(r'$f\_1(x)=\sin(x),\ f\_2(x)=\cos(x),\ f\_3(x)=-x$')

plt.grid(True)

plt.show()



Работа с matplotlib основана на использовании графических окон и осей (оси позволяют задать некоторую графическую область). Все построения применяются к текущим осям. Это позволяет изображать несколько графиков в одном графическом окне. По умолчанию создается одно графическое окно figure(1) и одна графическая область subplot(111) в этом окне. Команда subplot позволяет разбить графическое окно на несколько областей. Она имеет три параметра: nr; nc; np. Параметры nr и nc определяют количество строк и столбцов на которые разбивается графическая область, параметр np определяет номер текущей области (np принимает значения от 1 до nr\*nc). Если nr\*nc<10, то передавать параметры nr,nc,np можно без использовния запятой. Например, допустимы формы subplot(2,2,1) и subplot(221).

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

x=np.arange(-10,10.01,0.01); t=np.arange(-10,11,1)

*#subplot 1*

plt.subplot(221)

plt.plot(x,np.sin(x))

plt.title(r'$\sin(x)$')

plt.grid(True)

*#subplot 2*

plt.subplot(222)

plt.plot(x,np.cos(x),'g')

plt.axis('equal')

plt.grid(True)

plt.title(r'$\cos(x)$')

*#subplot 3*

plt.subplot(223)

plt.plot(x,x\*\*2,t,t\*\*2,'ro')

plt.title(r'$x^2$')

*#subplot 4*

plt.subplot(224)

plt.plot(x,x)

plt.subplot(224).spines['left'].set\_position('center')

plt.subplot(224).spines['bottom'].set\_position('center')

plt.title(r'$x$')

plt.show()

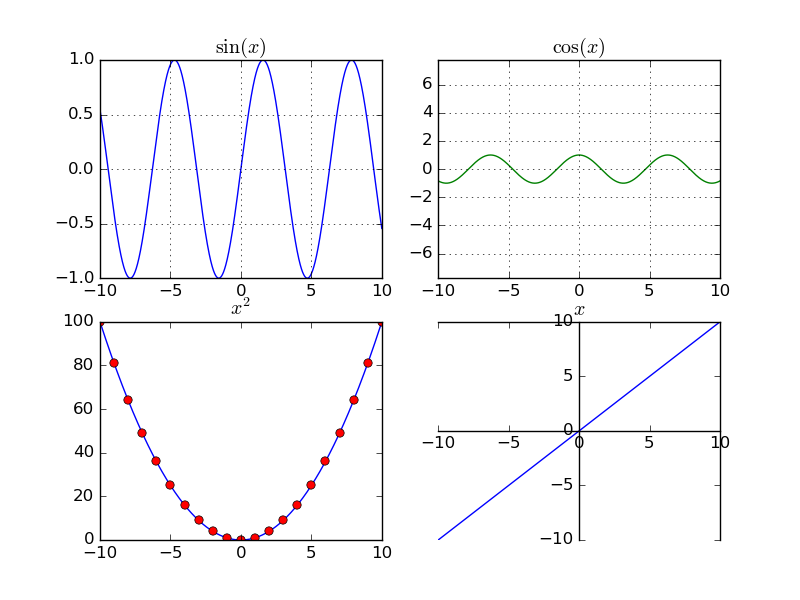


График может быть построен в полярной системе координат:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

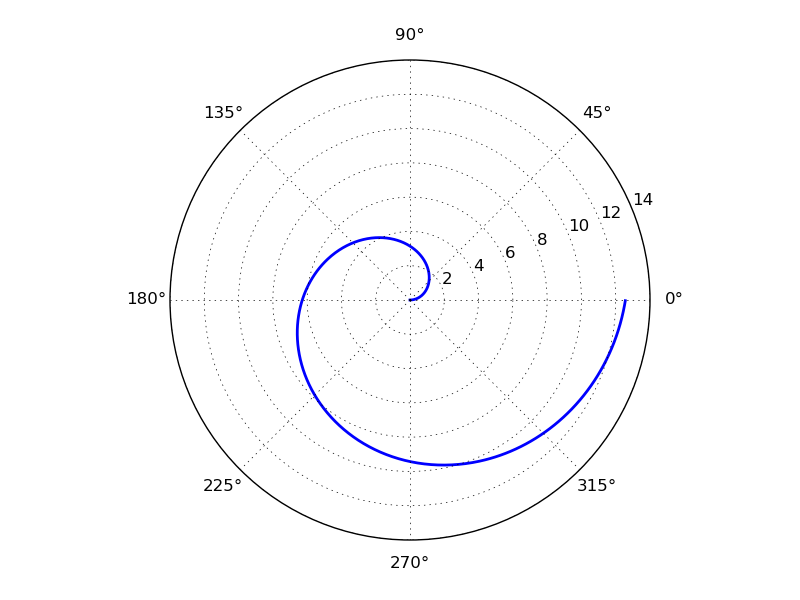
plt.subplot(111, polar=True)

phi = np.arange(0,2\*np.pi,0.01)

rho = 2\*phi

plt.plot(phi, rho, lw=2)

plt.show()



И быть задан в параметрической форме:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

t = np.arange(0,2\*np.pi,0.01)

r=4

plt.plot(r\*np.sin(t),r\*np.cos(t),lw=3)

plt.axis('equal')

plt.show()

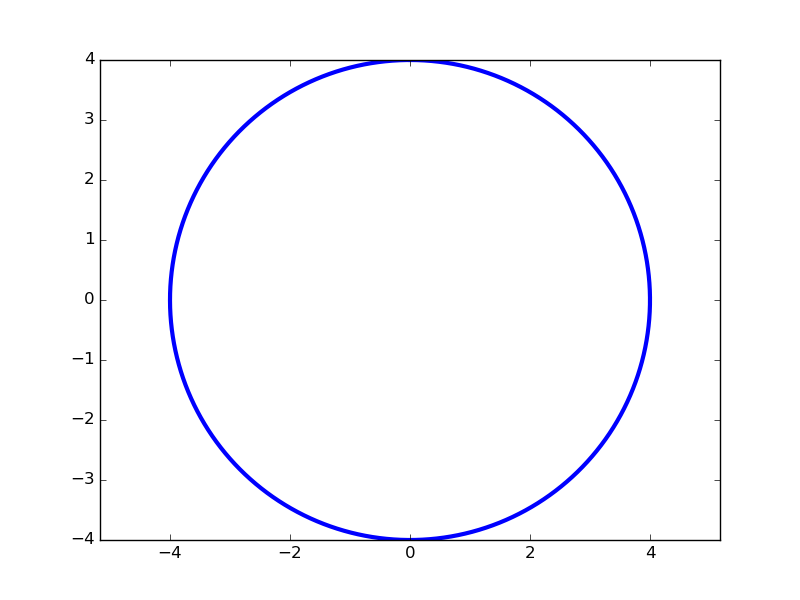


График функции двух переменных может быть построен например так:

**from** **mpl\_toolkits.mplot3d** **import** axes3d

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

ax = axes3d.Axes3D(plt.figure())

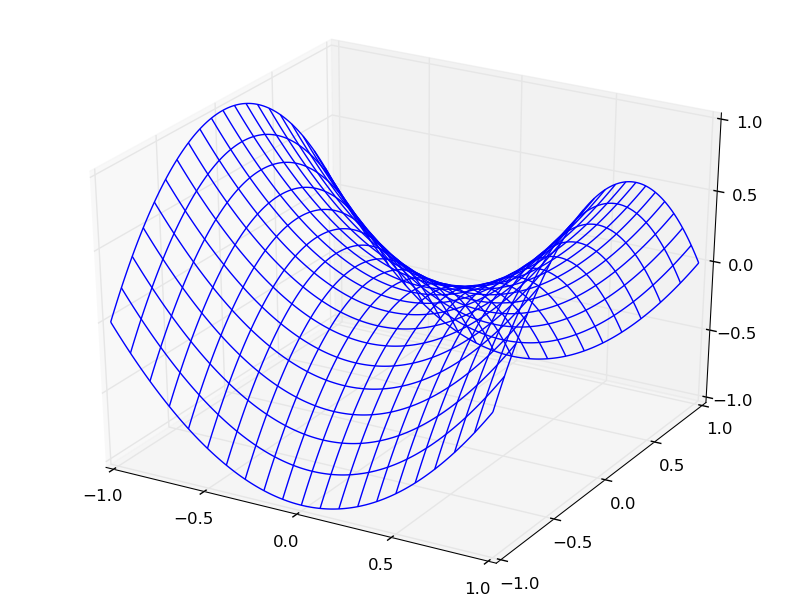
i = np.arange(-1, 1, 0.01)

X, Y = np.meshgrid(i, i)

Z = X\*\*2-Y\*\*2

ax.plot\_wireframe(X, Y, Z, rstride=10, cstride=10)

plt.show()



Добавление текста на график: Команду text() можно использовать для добавления текста в произвольном месте (по умолчанию координаты задаются в координатах активных осей), а команы xlabel(), ylabel() и title() служат соответственно для подписи оси абсцисс, оси ординат и всего графика. Для более полной информации смотрите Text introduction раздел на офсайте.

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

mu, sigma = 100, 15

x = mu + sigma \* np.random.randn(10000)

*# the histogram of the data*

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')

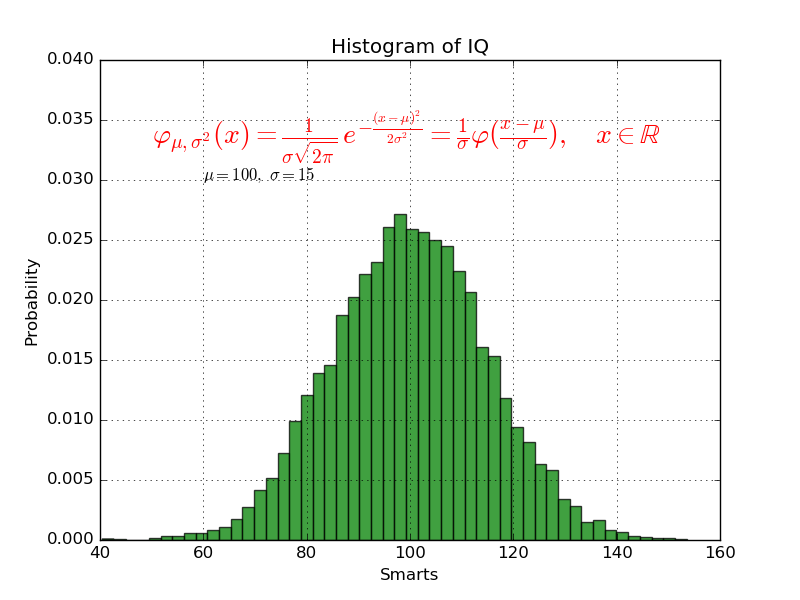
plt.text(60, .030, r'$\mu=100,\ \sigma=15$')

plt.text(50, .033, r'$\varphi\_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} \,e^{ -\frac{(x- \mu)^2}{2\sigma^2}} = \frac{1}{\sigma} \varphi\left(\frac{x - \mu}{\sigma}\right),\quad x\in\mathbb{R}$', fontsize=20, color='red')

plt.axis([40, 160, 0, 0.04])

plt.grid(True)

plt.show()

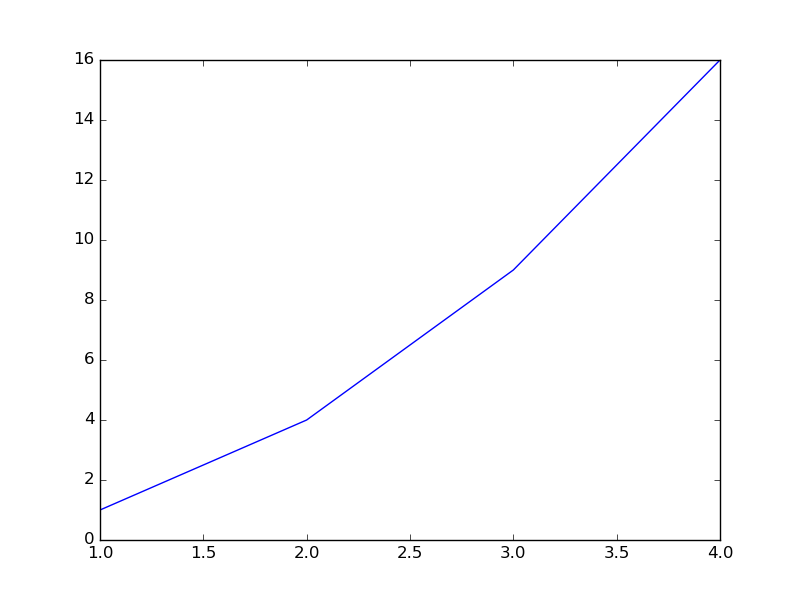


plot() универсальная команда и в неё можно передавать произвольное количество аргументов. Например, для того чтобы отобразить y в зависимости от x, можно выполнить команду:

**import** **matplotlib.pyplot** **as** **plt**

plt.plot([1,2,3,4], [1,4,9,16])

plt.show()



Каждую последовательность можно отобразить своим типом точек:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

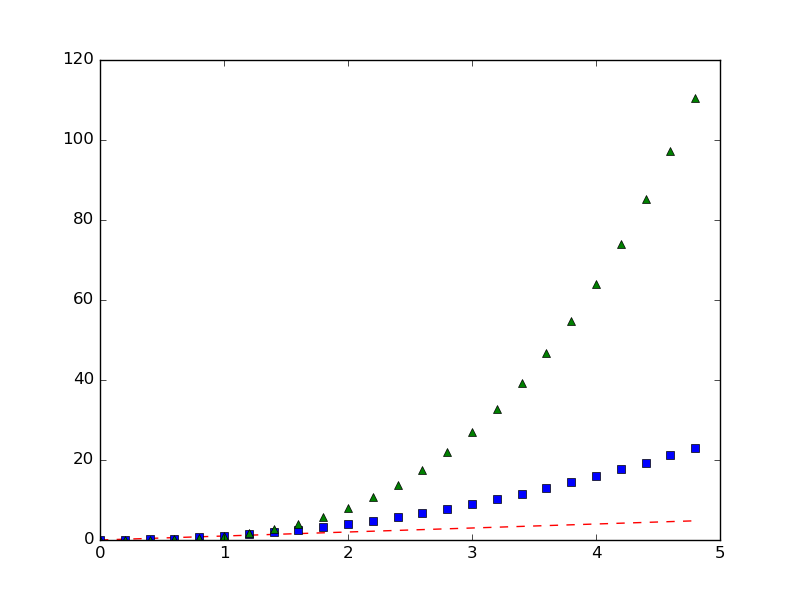
*# равномерно распределённые значения от 0 до 5, с шагом 0.2*

t = np.arange(0., 5., 0.2)

*# красные чёрточки, синие квадраты и зелёные треугольники*

plt.plot(t, t, 'r--', t, t\*\*2, 'bs', t, t\*\*3, 'g^')

plt.show()



Иногда нужно показать график в динамике, например, как меняется со временем какая-то величина. Если мы захотим воспользоваться функцией show(), то анимицию сделать не получится по той причине, что эта функция покажет окно с графиком и будет ждать, пока окно закроют. Нам нужен способ периодически обновлять окно с графиком. Для этого используется так называемый интерактивный режим, который включается с помощью функции ion() пакета pylab, а выключается функцией ioff(). Кроме того, вместо функции show() мы должны использовать функцию draw(), которая отображает график и не задерживает выполнение программы. Следующий пример демонстрирует просто движущуюся синусоиду. Для простоты окно закрывается после показа 50 кадров. (запускается только из под python 2.7)

**import** **math**

**import** **pylab**

**from** **matplotlib** **import** mlab

xmin = -20.0

xmax = 20.0

dx = 0.01

xlist = mlab.frange (xmin, xmax, dx)

pylab.ion()

**for** n **in** range (50):

ylist = [math.sin (x + n / 2.0) **for** x **in** xlist]

pylab.clf()

pylab.plot (xlist, ylist)

pylab.draw()

pylab.close()

Также существует возможность строить круговые диаграммы:

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

data = [33, 25, 20, 12, 10]

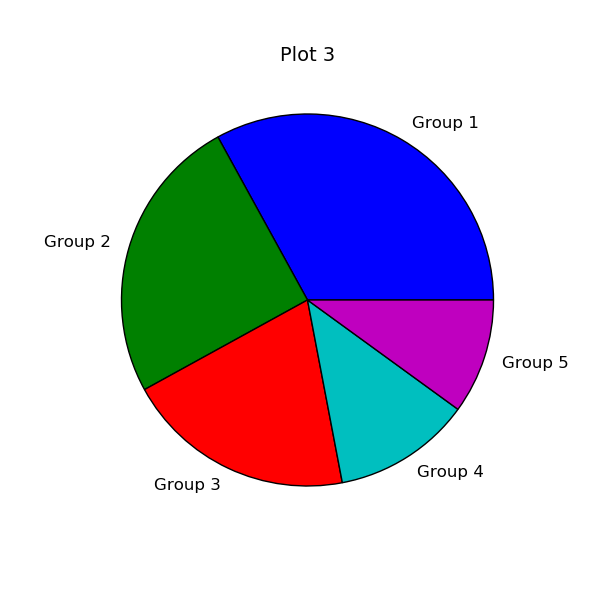
plt.figure(num=1, figsize=(6, 6))

plt.axes(aspect=1)

plt.title('Plot 3', size=14)

plt.pie(data, labels=('Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5'))

plt.show()



И аналогичным образом гистограммы.

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

objects = ('A', 'B', 'C', 'D', 'E', 'F')

y\_pos = np.arange(len(objects))

performance = [10,8,6,4,2,1]

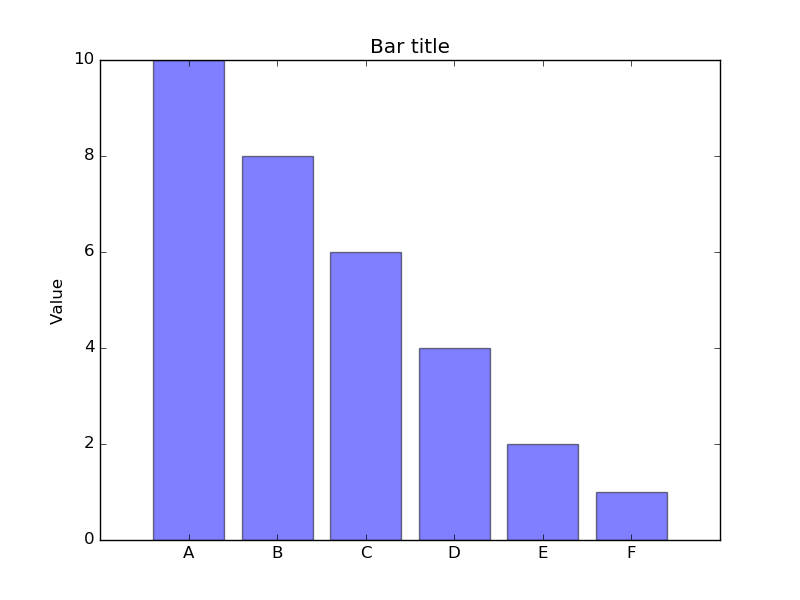
plt.bar(y\_pos, performance, align='center', alpha=0.5)

plt.xticks(y\_pos, objects)

plt.ylabel('Value')

plt.title('Bar title')

plt.show()



Цветовые карты используются, если нужно указать в какие цвета должны окрашиваться участки трехмерной поверхности в зависимости от значения Z в этой области. Цветовую карту можно задать самому, а можно воспользоваться готовой. Рассмотрим использование цветовой карты на примере графика функции z(x,y)=sin(x)\*sin(y)/(x\*y)

**import** **pylab**

**from** **mpl\_toolkits.mplot3d** **import** Axes3D

**from** **matplotlib.colors** **import** LinearSegmentedColormap

**from** **matplotlib** **import** cm

**import** **numpy**

**def** makeData ():

x = numpy.arange (-10, 10, 0.1)

y = numpy.arange (-10, 10, 0.1)

xgrid, ygrid = numpy.meshgrid(x, y)

zgrid = numpy.sin (xgrid) \* numpy.sin (ygrid) / (xgrid \* ygrid)

**return** xgrid, ygrid, zgrid

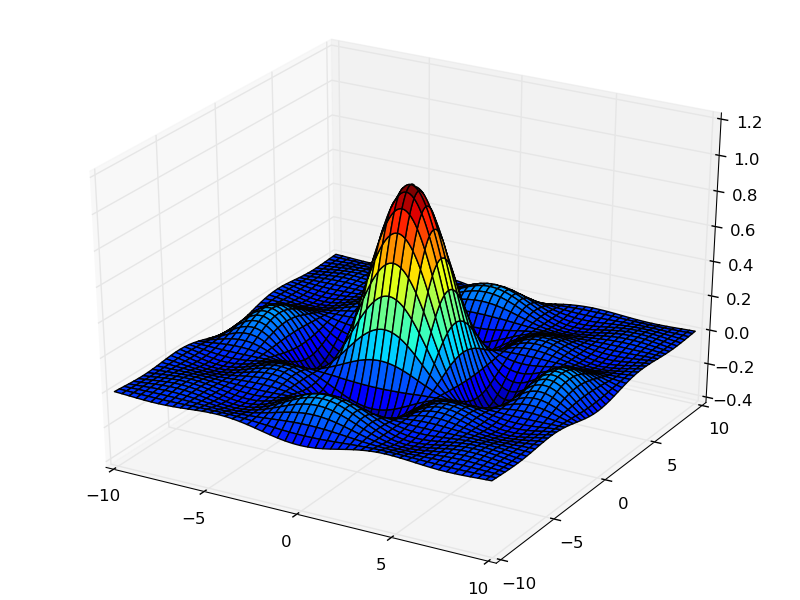
x, y, z = makeData()

fig = pylab.figure()

axes = Axes3D(fig)

axes.plot\_surface(x, y, z, rstride=4, cstride=4, cmap = cm.jet)

pylab.show()



[Функция eval()](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id9)

В Python есть встроенная функция eval(), которая выполняет строку с кодом и возвращает результат выполнения:

>>> eval("2 + 3 \* len('hello')")

17

>>>

Это очень мощная, но в то же время и очень опасная инструкция, особенно если строки, которые вы передаёте в eval, получены не из доверенного источника. Если строкой, которую мы решим скормить eval'у, окажется os.system('rm -rf /'), то интерпретатор честно запустит процесс удаления всех данных с компьютера.

[Упражнение №2](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id10)

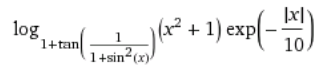
Постройте график функции

y(x) = x\*x-6-x

и численным способом найдите корни уравнения y(x) = 0

[Упражнение №3](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id11)

Постройте график функции



[Упражнение №4](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id12)

Фигура Лиссажу задается выражением: x(t,a) = sin(t+a), y(t) = cos(2\*t)

используя matplotlib анимируйте фигуру Лиссажу меняя в каждом кадре значение параметра a

[Упражнение №5](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id13)

Используя функцию eval() постройте график функции, введенной с клавиатуры. Включите эффект "рисование от руки" посредством вызова plt.xkcd()

[Упражнение №6](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id14)

Напишите программу, которая анализирует текстовый файл input.txt и строит диаграмму, в которой по оси X отложены длины слов, встретившихся во входном файле, а по оси Y - сколько раз слова с указанной длиной встречались.

[Упражнение №7](http://judge.mipt.ru/mipt_cs_on_python3/labs/lab7.html#id15)

Постройте график функции [Вейерштрасса](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%92%D0%B5%D0%B9%D0%B5%D1%80%D1%88%D1%82%D1%80%D0%B0%D1%81%D1%81%D0%B0)