**Построение графиков в Python**

**Документация и учебники по Mathplotlib**

* DEA! <http://matplotlib.org/examples/index.html> - Примеры
* На русском языке
  + [Matplotlib: Научная графика в Python](https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html) (на русском языке)
* на английском языке
  + <http://matplotlib.org/> - официальный сайт (устанавливаем оттуда)
  + <http://matplotlib.org/users/index.html> - Полное руководство пользователю (Eng)
  + <http://matplotlib.org/users/beginner.html> - Руководство пользователю для начинающих (Eng)
  + <http://matplotlib.org/users/pyplot_tutorial.html> - Tutorial (1 страница)
  + <https://www.labri.fr/perso/nrougier/teaching/matplotlib/> Matplotlib tutorial

Nicolas P. Rougier

* + [Youtube video](https://www.youtube.com/watch?v=q7Bo_J8x_dw&list=PLQVvvaa0QuDfefDfXb9Yf0la1fPDKluPF) (Eng)

**Простой график.**

Построим график функции y = x3+5x2+10 и ее производных y' и y" при х от -5 до 2.

DEA! xn записывается как x\*\*n

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(-5, 2, 100) # от -5 до 2 сделать 100 точек

y1 = x\*\*3 + 5\*x\*\*2 + 10 # y1 - тоже много точек

y2 = 3\*x\*\*2 + 10\*x

y3 = 6\*x + 10

fig, ax = plt.subplots() # будет 1 график, на нем:

ax.plot(x, y1, color="blue", label="y(x)") # функция y1(x), синий, надпись y(x)

ax.plot(x, y2, color="red", label="y'(x)") # функция y2(x), красный, надпись y'(x)

ax.plot(x, y3, color="green", label="y''(x)") # функция y3(x), зеленый, надпись y''(x)

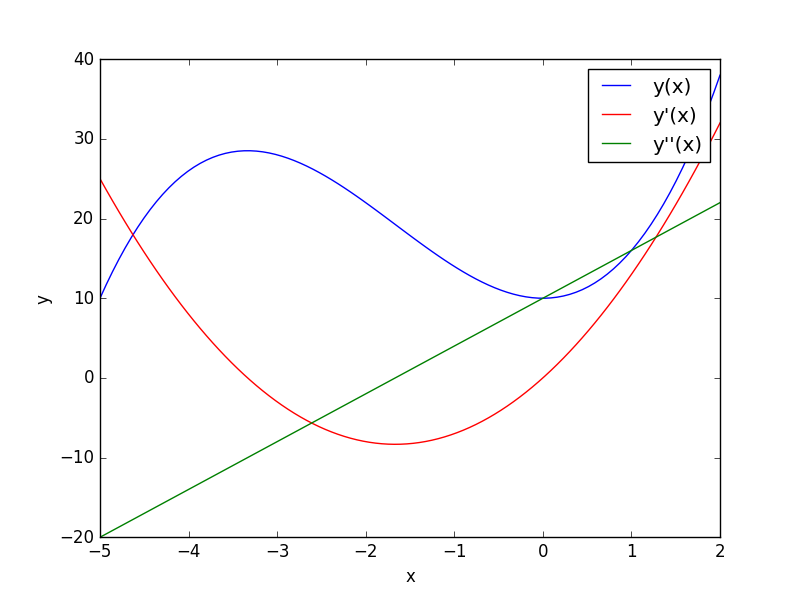
ax.set\_xlabel("x") # подпись у горизонтальной оси х

ax.set\_ylabel("y") # подпись у вертикальной оси y

ax.legend() # показывать условные обозначения

plt.show() # показать рисунок

fig.savefig('1.png') # сохранить в файл 1.png



**Import**

**Короткие имена**

Если для работы нужен пакет (package, library) numpy (работа с векторами), то его нужно подключить к файлу

import numpy

numpy.cos(x) # берем функцию cos именно из библиотеки numpy (а не из math.cos)

Чтобы писать меньше, придумываем **короткое** имя пакета

import numpy as np

np.cos(x)

**Нужные import**

import matplotlib as mpl

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d.axes3d import Axes3D # если рисуем 3D графики

import numpy as np

import sympy

**Интерпретатор или запуск файла**

**Работа в интерпретаторе**

DEA! Для работы в IPython или Qtconsole нужно написать в начале (включить построение графиков в тетрадь).

In [1]: %matplotlib inline

Для построения графиков не в тетради, а в файле в формате png нужно написать

In [1]: %config InlineBackend[?](http://acm.mipt.ru/twiki/bin/edit/Cintro/InlineBackend?topicparent=Cintro.PythonGraphs).figure\_format='png'

Писать plt.show() или plt.draw() не нужно.

**Выполнение программы**

В конце кода обязательно должно стоять или plt.show() (показать интерактивный график) или fig.savefig('1.png')(сохранить в файле с именем 1.png).

**Как получить график в repl.it**

* Добавить в начало

import matplotlib as mpl

mpl.use('Agg')

* Обязательно сохранять рисунок в файл (например, 1.png)

fig.savefig('1.png')

* в repl.it
  + (1) Перейти в режим проекта, программа находится в файле main.py
  + (2) Run
  + (3) Появится (обновится) панель 1.py, на ней можно увидеть график.



Полный код программы для repl.it

import matplotlib as mpl

mpl.use('Agg') # Не рисовать на экране

import matplotlib.pyplot as plt

import numpy as np

x = np.linspace(-5, 2, 100) # от -5 до 2 сделать 100 точек

y1 = x\*\*3 + 5\*x\*\*2 + 10 # y1 - тоже много точек

y2 = 3\*x\*\*2 + 10\*x

y3 = 6\*x + 10

fig, ax = plt.subplots() # будет 1 график, на нем:

ax.plot(x, y1, color="blue", label="y(x)") # функция y1(x), синий, надпись y(x)

ax.plot(x, y2, color="red", label="y'(x)") # функция y2(x), красный, надпись y'(x)

ax.plot(x, y3, color="green", label="y''(x)") # функция y3(x), зеленый, надпись y''(x)

ax.set\_xlabel("x") # подпись у горизонтальной оси х

ax.set\_ylabel("y") # подпись у вертикальной оси y

ax.legend() # показывать условные обозначения

#plt.show() # показать рисунок - не нужно

fig.savefig('1.png') # сохранить в файл 1.png

**Введение в Numpy**

Numpy (numeric python) - библиотека работы с числами.

В ней задаются наборы точек для построения графиков.

**Задаем набор точек**

x = np.arange([1, 2, 3, 5, 7, 10, 15])

x = np.arange(-10, 10.01, 0.01) # от -10 до 10.01 с шагом 0.01 (>1000 точек)

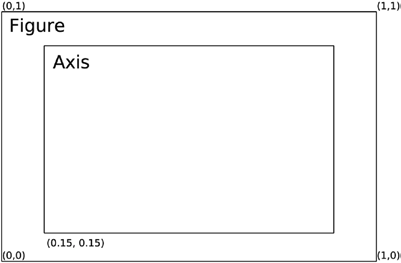
x = np.linspace(-5, 2, 100) # от -5 до 2 сделать 100 точек

**Константы и функции numpy**

|  |  |
| --- | --- |
| [**numpy**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonGraphs?sortcol=0&table=1&up=0#sorted_table) | [**Математика**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonGraphs?sortcol=1&table=1&up=0#sorted_table) |
| np.pi | Число pi |
| np.e | Число e |
| np.[cos](http://docs.scipy.org/doc/numpy/reference/generated/numpy.cos.html) | Косинус |
| np.[sin](http://docs.scipy.org/doc/numpy/reference/generated/numpy.sin.html) | Синус |
| np.[tan](http://docs.scipy.org/doc/numpy/reference/generated/numpy.tan.html) | Тангенс |
| np.[acos](http://docs.scipy.org/doc/numpy/reference/generated/numpy.acos.html) | Арккосинус |
| np.[asin](http://docs.scipy.org/doc/numpy/reference/generated/numpy.asin.html) | Арксинус |
| np.[atan](http://docs.scipy.org/doc/numpy/reference/generated/numpy.atan.html) | Арктангенс |
| np.[exp](http://docs.scipy.org/doc/numpy/reference/generated/numpy.exp.html) | Экспонента |
| np.[log](http://docs.scipy.org/doc/numpy/reference/generated/numpy.log.html) | Логарифм натуральный |
| np.[log2](http://docs.scipy.org/doc/numpy/reference/generated/numpy.log2.html) | Логарифм по основанию 2 |
| np.[log10](http://docs.scipy.org/doc/numpy/reference/generated/numpy.log10.html) | Логарифм десятичный |
| np.[sqrt](http://docs.scipy.org/doc/numpy/reference/generated/numpy.sqrt.html) | x1/2 |

**Термины**

**Figure (рисунок), axes (оси)**



fig, ax - это описание рисунка (figure) и осей (axes).

axis - ось (1), axes - оси (много).

Axes определяет область, где будем рисовать.

Figure имеет координаты (0,0) в левом нижнем углу и (1,1) в правом верхнем углу. Оси можно сделать не на весь рисунок, а на часть и добавить подписи.

figsize=(8, 2.5) определяет размер и масштаб рисунка.

fig.savefig("1a.png", dpi=100, facecolor="#f1f1f1") - сохранить в файл 1a.png с разрешением 100 dpi (значит, при figsize=(8, 2.5) рисунок будет 8\*100=800 на 2.5\*100=250 pixel).

# -\*- coding: utf-8 -\*-

import matplotlib.pyplot as plt

import numpy as np

fig = plt.figure(figsize=(8, 2.5), facecolor="#f1f1f1")

# axes coordinates as fractions of the canvas width and height

left, bottom, width, height = 0.1, 0.1, 0.8, 0.8 # задаем значения переменных

ax = fig.add\_axes((left, bottom, width, height), axisbg="#e1e1e1")

x = np.linspace(-2, 2, 1000)

y1 = np.cos(40 \* x)

y2 = np.exp(-x\*\*2)

ax.plot(x, y1 \* y2)

ax.plot(x, y2, 'g')

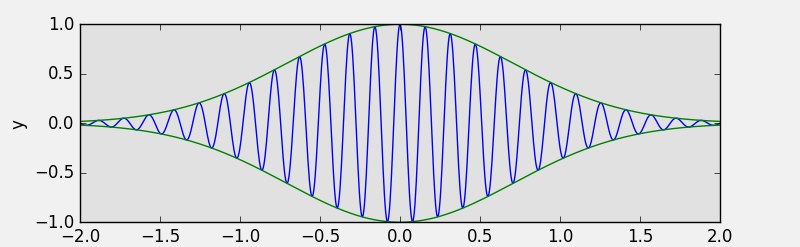
ax.plot(x, -y2, 'g')

ax.set\_xlabel("x")

ax.set\_ylabel("y")

plt.show()

fig.savefig("1a.png", dpi=100, facecolor="#f1f1f1")



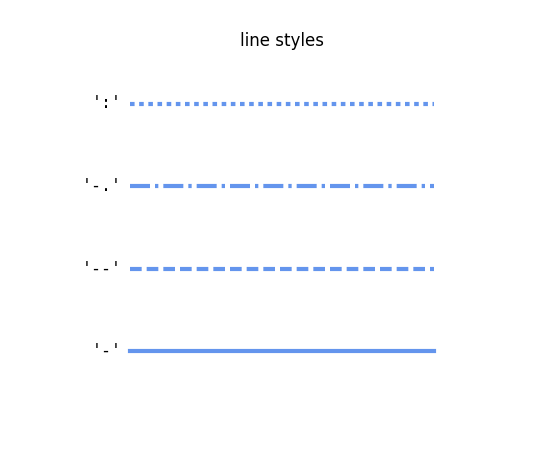
**Термины**

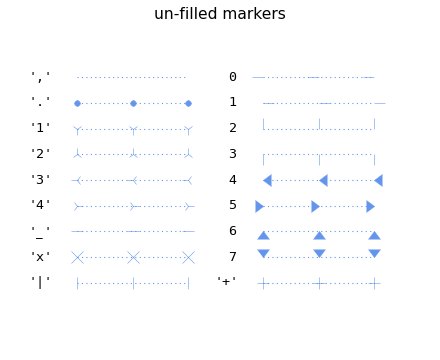
**Описание линий и маркеров**

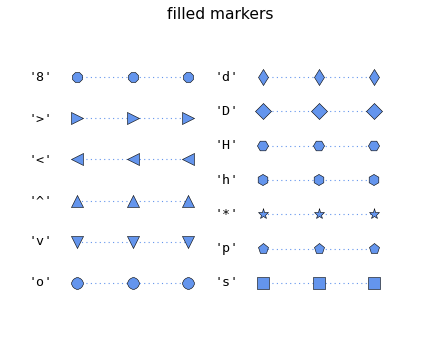
Синяя линия шириной 3 пикселя, линия как ---- и точки отмечены треугольниками.

ax.plot(x, y, color='b', linewidth=3, linestyle='--', markerstyle='v')

* Примеры:
  + <http://matplotlib.org/examples/color/named_colors.html> - цвета
  + <http://matplotlib.org/examples/lines_bars_and_markers/line_styles_reference.html> - стили линий
  + <http://matplotlib.org/1.4.2/examples/lines_bars_and_markers/marker_reference.html> - маркеры

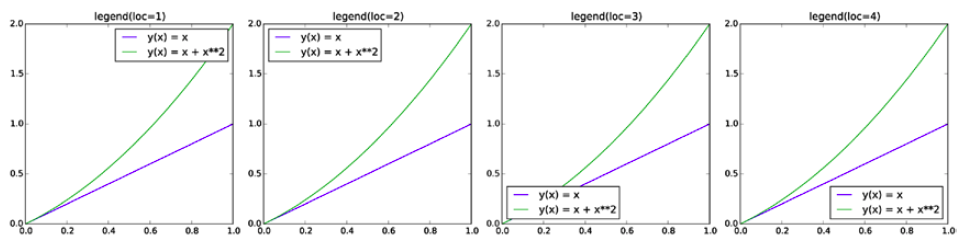




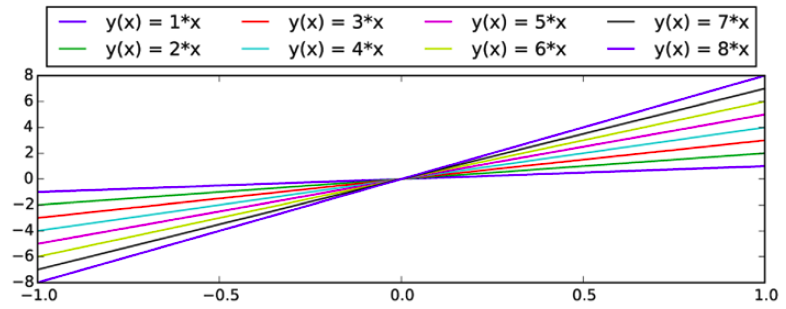




**Legend (описание)**



ax.legend(ncol=4, loc=3, bbox\_to\_anchor=(0, 1))



**Axis labels and titles (надписи у осей и заголовки)**

ax.set\_xlabel("x", labelpad=5, fontsize=18, fontname='serif', color="blue")

ax.set\_ylabel("f(x)", labelpad=15, fontsize=18, fontname='serif', color="blue")

ax.set\_title("axis labels and title example", fontsize=16,

fontname='serif', color="blue")

**Подключить русские буквы (не-латинские буквы, иероглифы)**

Если вы хотите, чтобы надписи были на русском, а вместо них показываются непонятные символы, то, вероятно, у вас выбран фонт без поддержки нужных букв.

Можно в каждом вызове set\_title и set\_xlabel, set\_ylabel указывать фонт явно, а можно указать нужный фонт в начале программы.

**Один раз** указываем фонт:

from matplotlib import rc

font = {'family': 'verdana', # вот это название шрифта, если у вас другой шрифт для нужных букв, пишите правильное имя шрифта

'weight': 'normal',

# 'size': 14 можно указать еще размер символов в points

}

rc('font', \*\*font)

Потом его используем **много раз** в разных графиках (в функциях оставлены только обязательные параметры).

ax.set\_xlabel("x")

ax.set\_ylabel("f(x)")

ax.set\_title("axis labels and title example")

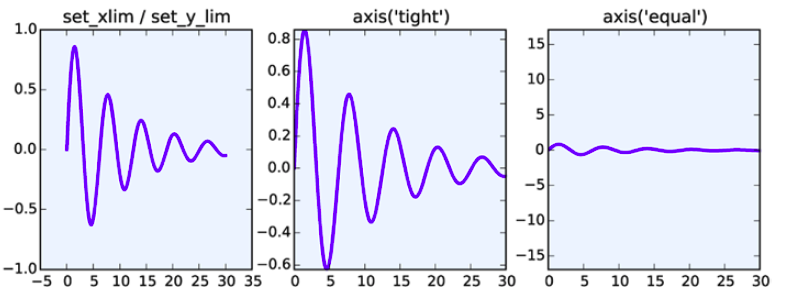
**Axis range (диапазоны осей)**

ax.set\_xlim(-5, 35)

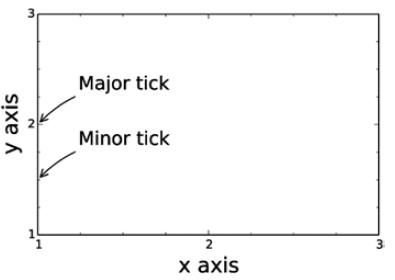
ax.set\_ylim(-1, 1)

ax.axis('tight')

ax.axis('equal')



**Axis ticks, tick labels**



ax1.set\_xticks([-5, 0, 5])

ax1.set\_yticks([-1, 0, 1])

ax2.xaxis.set\_major\_locator(mpl.ticker.MaxNLocator(4))

ax2.yaxis.set\_major\_locator(mpl.ticker.FixedLocator([-1, 0, 1]))

ax2.xaxis.set\_minor\_locator(mpl.ticker.MaxNLocator(8))

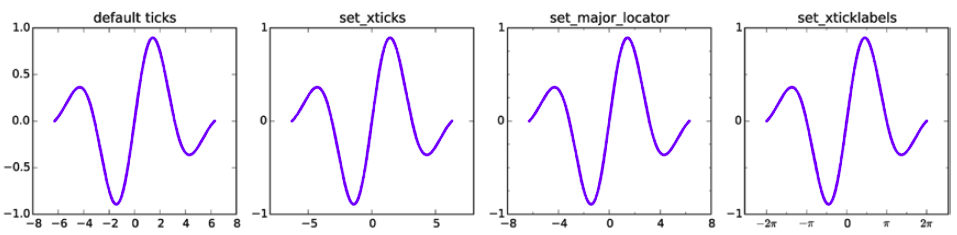
ax2.yaxis.set\_minor\_locator(mpl.ticker.MaxNLocator(8))

ax3.set\_xticks([-2 \* np.pi, -np.pi, 0, np.pi, 2 \* np.pi])

ax3.set\_xticklabels(['$-2\pi$', '$-\pi$', 0, r'$\pi$', r'$2\pi$'])

Поставить ticks, подписать около них labels, вариант через plt (а не оси).

plt.xticks([-2 \* np.pi, -np.pi, 0, np.pi, 2 \* np.pi], ['$-2\pi$', '$-\pi$', 0, r'$\pi$', r'$2\pi$'])



**Grid (решетка)**

axes[0].set\_title("default grid")

axes[0].grid()

axes[1].set\_title("major/minor grid")

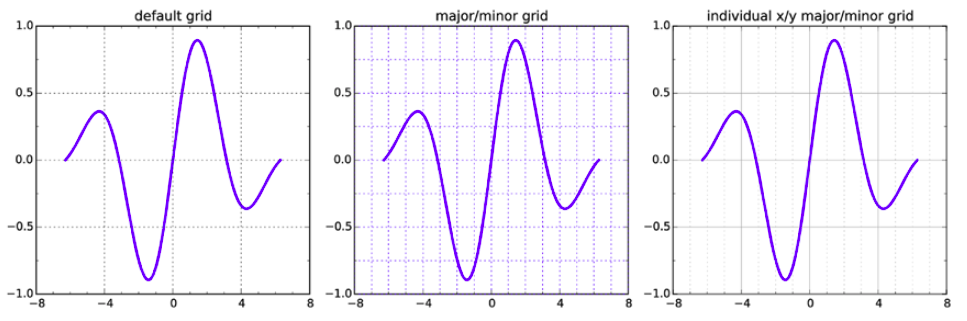
axes[1].grid(color="blue", which="both", linestyle=':', linewidth=0.5)

axes[2].set\_title("individual x/y major/minor grid")

axes[2].grid(color="grey", which="major", axis='x', linestyle='-', linewidth=0.5)

axes[2].grid(color="grey", which="minor", axis='x', linestyle=':', linewidth=0.25)

axes[2].grid(color="grey", which="major", axis='y', linestyle='-', linewidth=0.5)



**Логарифмическая шкала**

Для всего рисунка

plt.xscale('log')

plt.yscale('log')

Для одного графика (другие графики на рисунке могут иметь обычную шкалу)

ax.set\_xscale('log')

ax.set\_yscale('log')

**Разные шкалы оси Y для разных графиков (twinx)**

# -\*- coding: utf-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

fig, ax1 = plt.subplots(figsize=(8, 4))

r = np.linspace(0, 5, 100)

a = 4 \* np.pi \* r \*\* 2 # area

v = (4 \* np.pi / 3) \* r \*\* 3 # volume

ax1.set\_title("surface area and volume of a sphere", fontsize=16)

ax1.set\_xlabel("radius [m]", fontsize=16)

ax1.plot(r, a, lw=2, color="blue")

ax1.set\_ylabel(r"surface area ($m^2$)", fontsize=16, color="blue")

for label in ax1.get\_yticklabels():

label.set\_color("blue")

ax2 = ax1.twinx()

ax2.plot(r, v, lw=2, color="red")

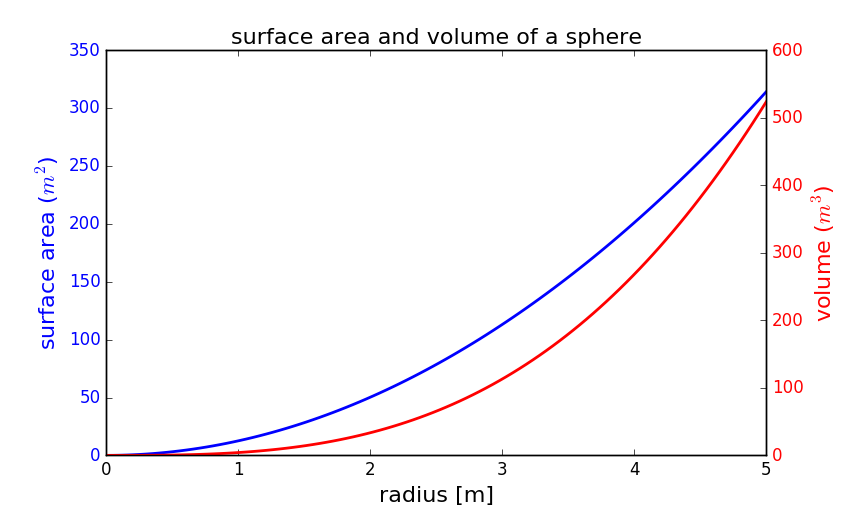
ax2.set\_ylabel(r"volume ($m^3$)", fontsize=16, color="red")

for label in ax2.get\_yticklabels():

label.set\_color("red")

plt.show()

fig.savefig('twinx.png')



**Несколько областей графиков**

По умолчанию создаётся 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: nrows, ncols, nplot.  
nrows - количество строк;  
ncols - количество столбцов;   
nplot - номер области (от 1 до nrows\*ncols).

Если nrows\*ncols<10, то можно писать без запятой nrows, ncols, nplot.  
Можно subplot(2,2,1), а можно subplot(221).

import numpy as np

import matplotlib.pyplot as plt

x = np.arange(-10, 10.01, 0.01) # от -10 до 10.01 с шагом 0.01 (>1000 точек)

t = np.arange(-10, 11, 1) # от -10 до 11 с шагом 1 (20 точек)

fig = plt.figure()

#subplot 1

plt.subplot(221) # первая область

plt.plot(x, np.sin(x)) # sin(x) на [-10, 10] цветом по умолчанию

plt.title(r'$\sin(x)$') # заголовок "sin(x)"

plt.grid(True) # рисовать решетку

#subplot 2

plt.subplot(222) # вторая область

plt.plot(x, np.cos(x), 'g') # cos(x) на [-10, 10] зеленая линия

plt.axis('equal') # одинаковый масштаб по осям Х и Y

plt.grid(True) # рисовать решетку

plt.title(r'$\cos(x)$') # заголовок "cos(x)"

#subplot 3

plt.subplot(223) # третья область

plt.plot(x, x\*\*2, t, t\*\*2, 'ro') # 2 графика, первый линией, второй красными кругами

plt.title(r'$x^2$') # заголовок "x2"

# решетки нет

#subplot 4

plt.subplot(224) # четвертая область

plt.plot(x, x) # прямая y=x

# оси с подписями нарисовать

# левую в центре

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

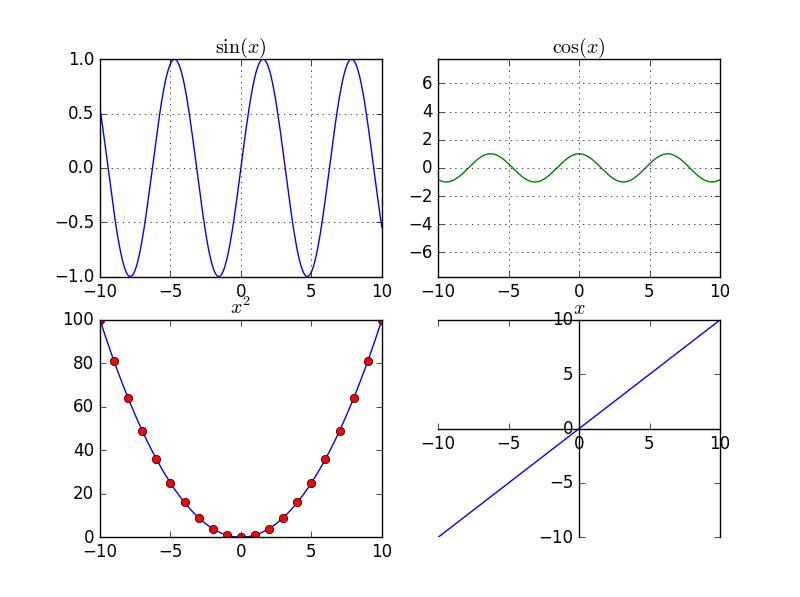
# нижнюю в центре

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

plt.title(r'$x$') # заголовок "x"

plt.show()

fig.savefig('2.png')



**Вариант с squeeze=False**

Можно обращаться ax[0,0], ax[0,1], ax[1,0], ax[1,1], если вызвать subplots с параметром squeeze=False

# -\*- coding: utf-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

x = np.arange(-10, 10.01, 0.01) # от -10 до 10.01 с шагом 0.01 (>1000 точек)

t = np.arange(-10, 11, 1) # от -10 до 11 с шагом 1 (20 точек)

fig, ax = plt.subplots(2, 2, figsize=(8,8), sharex=True, squeeze=False)

# sharex=True общие оси Х для верхних и нижних графиков,

# но оси У для правых и левых графиков - разные

# squeeze=False возвращает массив осей ax,

# обращаемся ax[0,0], ax[0,1], ax[1,0], ax[1,1]

#subplot 1

ax[0,0].plot(x, np.sin(x)) # sin(x) на [-10, 10] цветом по умолчанию

ax[0,0].set\_title(r'$\sin(x)$') # заголовок "sin(x)"

ax[0,0].grid(True) # рисовать решетку

#subplot 2

ax[0,1].plot(x, np.cos(x), 'g') # cos(x) на [-10, 10] зеленая линия

ax[0,1].axis('equal') # одинаковый масштаб по осям Х и Y

ax[0,1].grid(True) # рисовать решетку

ax[0,1].set\_title(r'$\cos(x)$') # заголовок "cos(x)"

#subplot 3

ax[1,0].plot(x, x\*\*2, t, t\*\*2, 'ro') # 2 графика, первый линией, второй красными кругами

ax[1,0].set\_title(r'$x^2$') # заголовок "x2"

# решетки нет

#subplot 4

ax[1,1].plot(x, x) # прямая y=x

# оси с подписями нарисовать

# левую в центре

ax[1,1].spines['left'].set\_position('center')

# нижнюю в центре

ax[1,1].spines['bottom'].set\_position('center')

ax[1,1].set\_title(r'$x$') # заголовок "x"

ax[1, 0].set\_xlabel("x") # подписи у осей (не все графики)

ax[0, 0].set\_ylabel("y")

ax[1, 0].set\_ylabel("y")

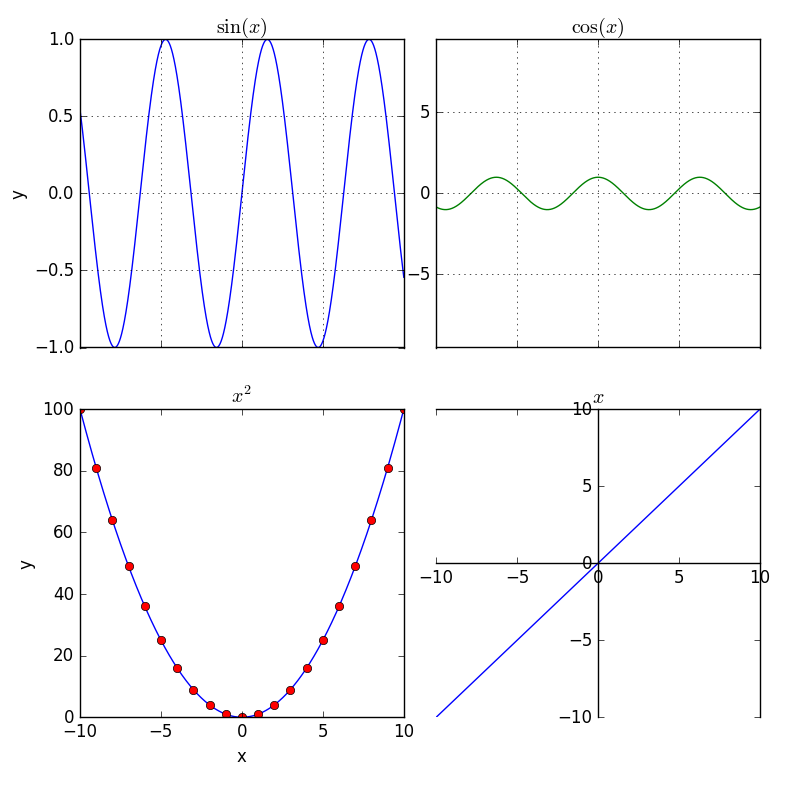
# красиво расположить области с отступами

plt.subplots\_adjust(left=0.1, right=0.95, bottom=0.1, top=0.95, wspace=0.1,

hspace=0.2)

plt.show()

fig.savefig('2a.png')



**Полярная система координат**

import numpy as np

import matplotlib.pyplot as plt

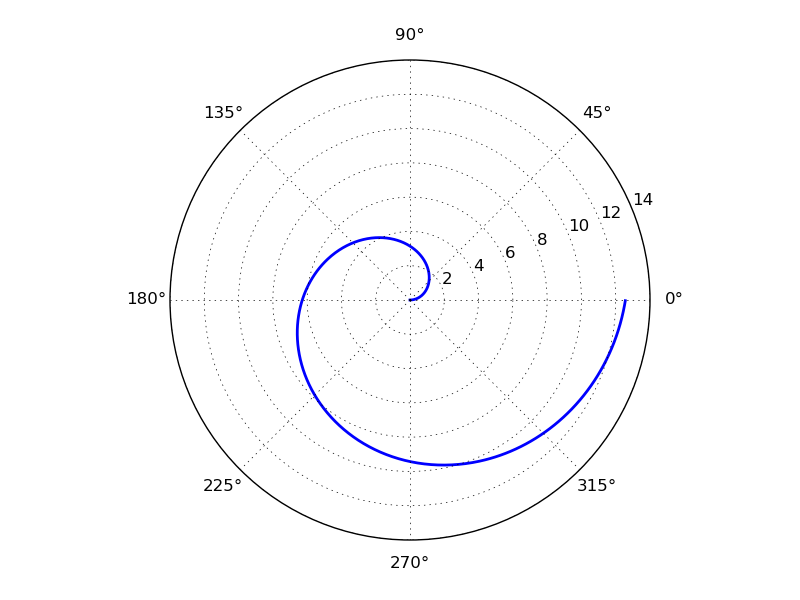
plt.subplot(111, polar=True) # Полярная система координат

phi = np.arange(0, 2\*np.pi, 0.01) # угол phi - массив от от 0 до 2\*pi с шагом 0.01

rho = 2\*phi # расстояние от центра 2\*phi

plt.plot(phi, rho, lw=2) # график rho(phi), толщина линии (line width) 2

plt.show()



**2D график в параметрической форме x(t) и y(t)**

Окружность можно задать как x(t) = r\*sin(t), y(t) = r\*cos(t), где t от 0 до 2π

import numpy as np

import matplotlib.pyplot as plt

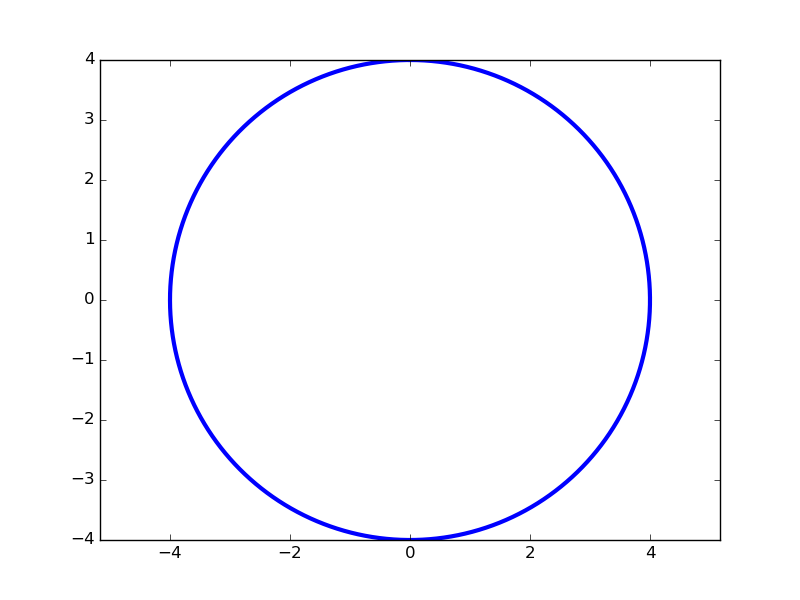
t = np.arange(0, 2\*np.pi, 0.01) # угол t от 0 до 2pi с шагом 0.01

r = 4 # радиус 4

plt.plot(r\*np.sin(t), r\*np.cos(t), lw=3) # x и у задаем как numpy функции от t

plt.axis('equal') # масштаб осей Х и У одинаковый (чтобы круг не был овалом)

plt.show()



**3D графики**

3D axes можно получить разными способами.

from mpl\_toolkits.mplot3d import axes3d

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

Или в add\_subplot задать projection='3d'

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

**axes из Axes3D**

from mpl\_toolkits.mplot3d import axes3d

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

Полный код

import matplotlib.pyplot as plt

import numpy as np

from mpl\_toolkits.mplot3d import axes3d

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

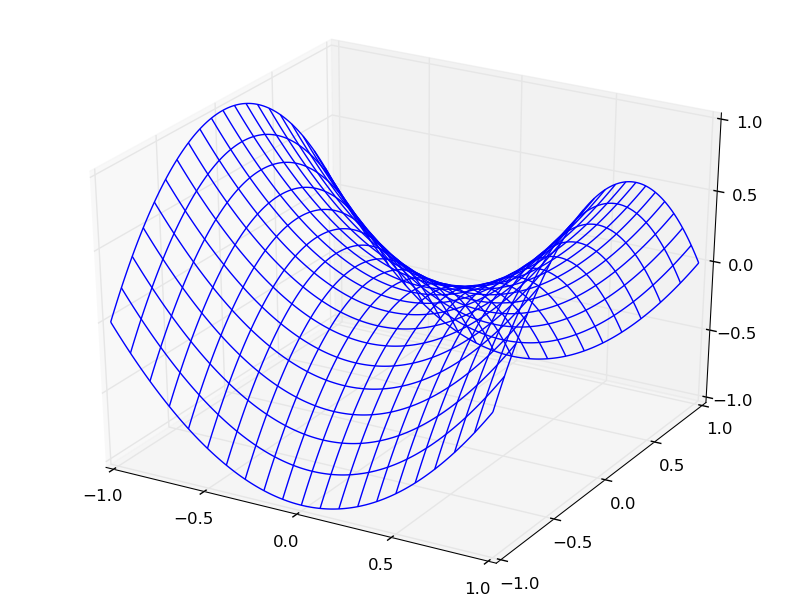
i = np.arange(-1, 1, 0.01)

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

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

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

plt.show()



**projection='3d'**

В примере показаны разные типы 3D графиков: заливка (surface), каркас (wireframe) и проекции (contour)

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib as mpl

import matplotlib.pyplot as plt

import numpy as np

x = y = np.linspace(-3, 3, 74)

X, Y = np.meshgrid(x, y)

R = np.sqrt(X\*\*2 + Y\*\*2)

Z = np.sin(4 \* R) / R

fig, ax = plt.subplots(1, 3, figsize=(14, 4), subplot\_kw=dict(projection='3d'))

# subplot0 - surface

norm = mpl.colors.Normalize(-abs(Z).max(), abs(Z).max())

p = ax[0].plot\_surface(X, Y, Z, rstride=1, cstride=1, linewidth=0,

antialiased=False, norm=norm, cmap=mpl.cm.Blues)

cb = fig.colorbar(p, ax=ax[0], shrink=0.6)

ax[0].set\_xlabel("$x$", fontsize=16)

ax[0].set\_ylabel("$y$", fontsize=16)

ax[0].set\_zlabel("$z$", fontsize=16)

# subplot1 - wireframe

ax[1].plot\_wireframe(X, Y, Z, rstride=2, cstride=2, color="darkgrey")

ax[1].set\_title("plot\_wireframe")

# no ticks

ax[1].set\_xticks([])

ax[1].set\_yticks([])

ax[1].set\_zticks([])

# subplot2

ax[2].contour(X, Y, Z, zdir='z', offset=0, norm=norm, cmap=mpl.cm.Blues)

ax[2].contour(X, Y, Z, zdir='y', offset=3, norm=norm, cmap=mpl.cm.Blues)

ax[2].set\_title("contour")

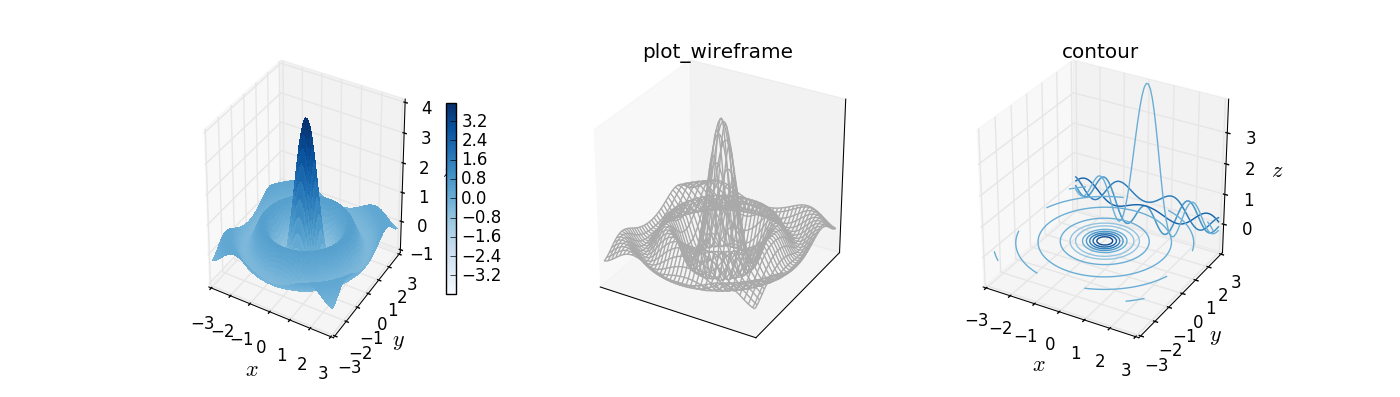
ax[2].set\_xlabel("$x$", fontsize=16)

ax[2].set\_ylabel("$y$", fontsize=16)

ax[2].set\_zlabel("$z$", fontsize=16)

plt.show()

fig.savefig("5b0.png", dpi=100) # results in 4\*160x4\*120 px image



**Задаем функцию Z(X,Y) в отдельной функции, заливка**

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib.colors import LinearSegmentedColormap

from matplotlib import cm

import matplotlib.pyplot as plt

import numpy as np

# отдельная функция, в которой задаются x, y, z

def makeData():

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

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

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

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

return xgrid, ygrid, zgrid

# вызов этой функции и дальше рисуем 3D график

x, y, z = makeData()

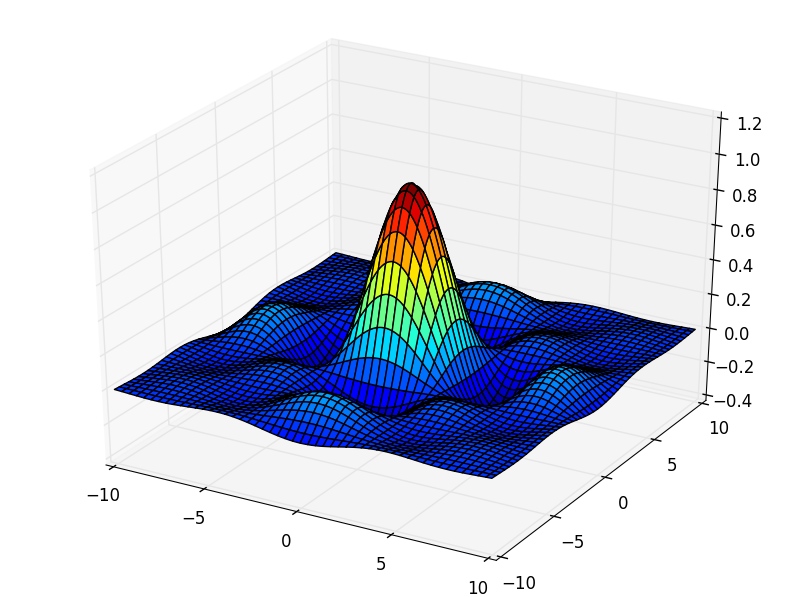
fig = plt.figure()

ax = Axes3D(fig)

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

plt.show()

fig.savefig('5c.png')



**Диаграммы**

<http://matplotlib.org/users/screenshots.html>

**Круговая диаграмма (pie)**

import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:

labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'

sizes = [15, 30, 45, 10]

explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

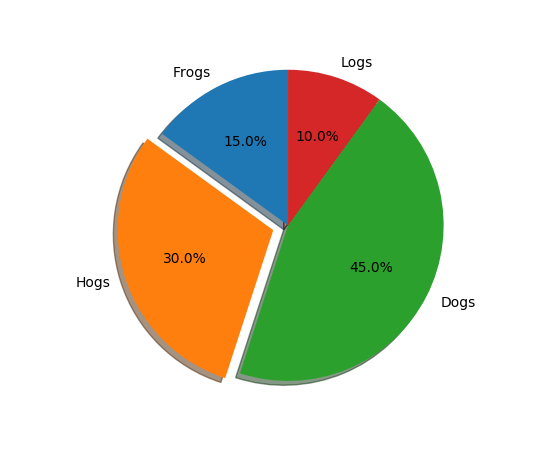
fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',

shadow=True, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()



**Столбцы**

import numpy as np

import matplotlib.pyplot as plt

n\_groups = 5

means\_men = (20, 35, 30, 35, 27)

std\_men = (2, 3, 4, 1, 2)

means\_women = (25, 32, 34, 20, 25)

std\_women = (3, 5, 2, 3, 3)

fig, ax = plt.subplots()

index = np.arange(n\_groups)

bar\_width = 0.35

opacity = 0.4

error\_config = {'ecolor': '0.3'}

rects1 = plt.bar(index, means\_men, bar\_width,

alpha=opacity,

color='b',

yerr=std\_men,

error\_kw=error\_config,

label='Men')

rects2 = plt.bar(index + bar\_width, means\_women, bar\_width,

alpha=opacity,

color='r',

yerr=std\_women,

error\_kw=error\_config,

label='Women')

plt.xlabel('Group')

plt.ylabel('Scores')

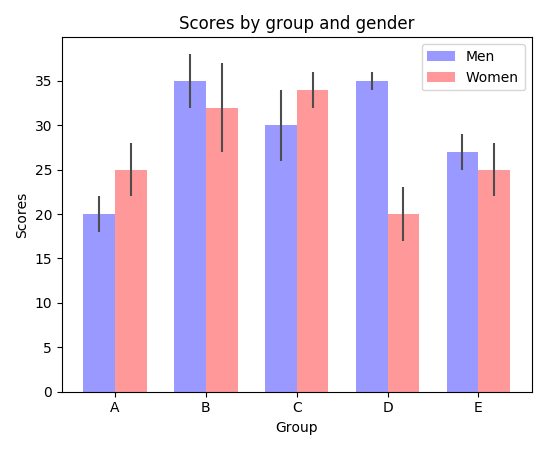
plt.title('Scores by group and gender')

plt.xticks(index + bar\_width / 2, ('A', 'B', 'C', 'D', 'E'))

plt.legend()

plt.tight\_layout()

plt.show()



**Bar stacked**

То же самое, но рисуем столбцы не рядом, а один над другим (оцениваем еще и сумму).

<http://matplotlib.org/examples/pylab_examples/bar_stacked.html>

# a stacked bar plot with errorbars

import numpy as np

import matplotlib.pyplot as plt

N = 5

menMeans = (20, 35, 30, 35, 27)

womenMeans = (25, 32, 34, 20, 25)

menStd = (2, 3, 4, 1, 2)

womenStd = (3, 5, 2, 3, 3)

ind = np.arange(N) # the x locations for the groups

width = 0.35 # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, menMeans, width, color='#d62728', yerr=menStd)

p2 = plt.bar(ind, womenMeans, width,

bottom=menMeans, yerr=womenStd)

plt.ylabel('Scores')

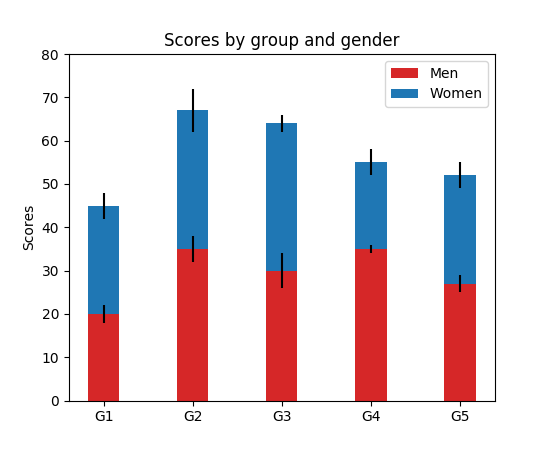
plt.title('Scores by group and gender')

plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))

plt.yticks(np.arange(0, 81, 10))

plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()



**Горизонтальная диаграмма**

<http://matplotlib.org/examples/lines_bars_and_markers/barh_demo.html>

"""

Simple demo of a horizontal bar chart.

"""

import matplotlib.pyplot as plt

plt.rcdefaults()

import numpy as np

import matplotlib.pyplot as plt

plt.rcdefaults()

fig, ax = plt.subplots()

# Example data

people = ('Tom', 'Dick', 'Harry', 'Slim', 'Jim')

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

performance = 3 + 10 \* np.random.rand(len(people))

error = np.random.rand(len(people))

ax.barh(y\_pos, performance, xerr=error, align='center',

color='green', ecolor='black')

ax.set\_yticks(y\_pos)

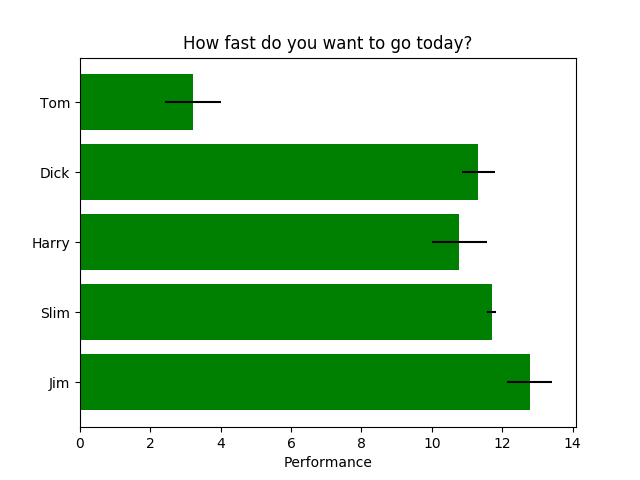
ax.set\_yticklabels(people)

ax.invert\_yaxis() # labels read top-to-bottom

ax.set\_xlabel('Performance')

ax.set\_title('How fast do you want to go today?')

plt.show()



-- [TatyanaDerbysheva](http://acm.mipt.ru/twiki/bin/view/Main/TatyanaDerbysheva) - 11 Feb 2017

* 1.png:
* menwomen\_colors.jpg:   
  