# Основы Python

## Пакет NumPy

# PIP: **P**IP **I**nstalls **P**ackages

- sudo pip install *packagename*
  sudo pip uninstall *packagename*

  Linux

- cd C:\Python27\Scripts\
  pip install *packagename*
  pip uninstall *packagename*

  Windows

- pip install numpy

# Arrays – Numerical Python (Numpy)

- Списки

```
>>> a = [1,3,5,7,9]
>>> print(a[2:4])
[5, 7]
>>> b = [[1, 3, 5, 7, 9], [2, 4, 6, 8, 10]]
>>> print(b[0])
[1, 3, 5, 7, 9]
>>> print(b[1][2:4])
[6, 8]
```

```
>>> a = [1,3,5,7,9]
>>> b = [3,5,6,7,9]
>>> c = a + b
>>> print c
[1, 3, 5, 7, 9, 3, 5, 6, 7, 9]
```

- Нет арифметических операций (+, -, *, /, …)

- **Numpy**

```
>>> import numpy
```

# Numpy – N-dimensional Array manipulations

NumPy – основная библиотека для научных расчетов в Python:

- поддержка многомерных массивов (арифметика, подмассивы, преобразования)

включает 3 доп. библиотеки с процедурами для:

- линейной алгебры (решение СЛАУ)
- дискретное преобразование Фурье
- работа со случайными числами

# Numpy – Creating vectors

- From lists
  - numpy.array – создание массива из списка значений

```
>>> import numpy
>>> a = numpy.array([1,3,5,7,9])
>>> b = numpy.array([3,5,6,7,9])
>>> c = a + b
>>> print c
[4, 8, 11, 14, 18]

>>> type(c)
(<type 'numpy.ndarray'>)

>>> c.shape
(5,)
```

```
>>> import numpy
>>> M = numpy.array([[1,2], [3, 4], [5,6], [7,8]], dtype=float)
>>> print M
[[ 1.  2.]
 [ 3.  4.]
 [ 5.  6.]
 [ 7.  8.]]

>>> print M.ndim
2
>>> print M.shape
(4, 2)
>>> print M.size
8
>>> print len(M)
4
>>> print numpy.sin(M)
[[ 0.84147098  0.90929743]
 [ 0.14112001 -0.7568025 ]
 [-0.95892427 -0.2794155 ]
 [ 0.6569866   0.98935825]]
```

# Numpy – Creating matrices

```
>>> L = [[1, 2, 3], [3, 6, 9], [2, 4, 6]]  # create a list
>>> a = numpy.array(L)  # convert a list to an array
>>> print a
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> print a.shape
(3, 3)
>>> print a.dtype  # get type of an array
dtype('int64')
```

```
# or directly as matrix
>>> M = numpy.array([[1, 2], [3, 4]])
>>> print M.shape
(2,2)
>>> print M.dtype
dtype('int64')

#only one type
>>> M[0,0] = "hello"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for long() with base 10:

>>> M = numpy.array([[1, 2], [3, 4]], dtype=complex)
>>> print M
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

# Numpy – Matrices use

```
>>> print(a)
[[1 2 3]
 [3 6 9]
 [2 4 6]]
>>> print(a[0])  # this is just like a list of lists
[1 2 3]
>>> print(a[1, 2])  # arrays can be given comma separated indices
9
>>> print(a[1, 1:3])  # and slices
[6 9]
>>> print(a[:,1])
[2 6 4]
>>> a[1, 2] = 7
>>> print(a)
[[1 2 3]
 [3 6 7]
 [2 4 6]]
>>> a[:, 0] = [0, 9, 8]
>>> print(a)
[[0 2 3]
 [9 6 7]
 [8 4 6]]
```

# Numpy – Creating arrays

```
>>> x = numpy.arange(0, 10, 1) # arguments: start, stop, step
>>> print x
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> print numpy.linspace(0, 10, 25)
array([  0.        ,   0.41666667,   0.83333333,   1.25      ,
         1.66666667,   2.08333333,   2.5       ,   2.91666667,
         3.33333333,   3.75      ,   4.16666667,   4.58333333,
         5.        ,   5.41666667,   5.83333333,   6.25      ,
         6.66666667,   7.08333333,   7.5       ,   7.91666667,
         8.33333333,   8.75      ,   9.16666667,   9.58333333,  10.        ])

>>> print numpy.logspace(-2, 3, 6)
array([  1.00000000e-02,   1.00000000e-01,   1.00000000e+00,
         1.00000000e+01,   1.00000000e+02,   1.00000000e+03])

>>> print numpy.logspace(0, 10, 10, base=numpy.e)  # по умолчанию base=10.0
array([  1.00000000e+00,   3.03773178e+00,   9.22781435e+00,
         2.80316249e+01,   8.51525577e+01,   2.58670631e+02,
         7.85771994e+02,   2.38696456e+03,   7.25095809e+03,
         2.20264658e+04])
```

# Numpy – Creating arrays

```
# a diagonal matrix
>>> print numpy.diag([1,2,3])
array([[1, 0, 0],
       [0, 2, 0],
       [0, 0, 3]])

>>> b = numpy.zeros(5)
>>> print(b)
[ 0.  0.  0.  0.  0.]
>>> print b.dtype
float64

>>> c = numpy.ones((3,3))
>>> print c
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])

>>> u = eye(2) # unit 2x2 matrix; "eye" represents "I"
>>> print u
array([[ 1., 0.],
       [ 0., 1.]])
```

# Numpy – array creation and use

```
>>> d = numpy.arange(5)  # just like range()
>>> print(d)
[0 1 2 3 4]

>>> d[1] = 9.7
>>> print(d)  # arrays keep their type even if elements changed
[0 9 2 3 4]

>>> print(d*0.4)  # operations create a new array, with new type
[ 0.   3.6  0.8  1.2  1.6]

>>> d = numpy.arange(5, dtype=float)
>>> print(d)
[ 0.  1.  2.  3.  4.]
```

# Numpy – array creation and use

```
# random data
>>> print numpy.random.rand(5,5)
array([[ 0.51531133,  0.74085206,  0.99570623,  0.97064334,  0.5819413 ],
       [ 0.2105685 ,  0.86289893,  0.13404438,  0.77967281,  0.78480563],
       [ 0.62687607,  0.51112285,  0.18374991,  0.2582663 ,  0.58475672],
       [ 0.72768256,  0.08885194,  0.69519174,  0.16049876,  0.34557215],
       [ 0.93724333,  0.17407127,  0.1237831 ,  0.96840203,  0.52790012]])
```

# Numpy – Creating arrays

- Чтение из файла

```
sample.txt:

"Stn", "Datum", "Tg", "qTg", "Tn", "qTn", "Tx", "qTx"
001, 19010101,   -49, 00,   -68, 00,   -22, 40
001, 19010102,   -21, 00,   -36, 30,   -13, 30
001, 19010103,   -28, 00,   -79, 30,    -5, 20
001, 19010104,   -64, 00,   -91, 20,   -10, 00
001, 19010105,   -59, 00,   -84, 30,   -18, 00
001, 19010106,   -99, 00,  -115, 30,   -78, 30
001, 19010107,   -91, 00,  -122, 00,   -66, 00
001, 19010108,   -49, 00,   -94, 00,    -6, 00
001, 19010109,    11, 00,   -27, 40,    42, 00
...

>>> data = numpy.genfromtxt('sample.txt', delimiter=',', skip_header=1)
>>> print data.shape
(25568, 8)
```

# Numpy – Creating arrays

- Сохранение в файл

```
>>> numpy.savetxt('datasaved.txt', data)

datasaved.txt:

1.0000000000000000e+00 1.9010101000000000e+07 -4.9000000000000000e+01
0.0000000000000000e+00 -6.8000000000000000e+01 0.0000000000000000e+00
-2.2000000000000000e+01 4.0000000000000000e+01
1.0000000000000000e+00 1.9010102000000000e+07 -2.1000000000000000e+01
0.0000000000000000e+00 -3.6000000000000000e+01 3.0000000000000000e+01
-1.3000000000000000e+01 3.0000000000000000e+01
1.0000000000000000e+00 1.9010103000000000e+07 -2.8000000000000000e+01
0.0000000000000000e+00 -7.9000000000000000e+01 3.0000000000000000e+01
-5.0000000000000000e+00 2.0000000000000000e+01
```

# Numpy – Creating arrays

```
>>> M = numpy.random.rand(3,3)
>>> print M
array([[ 0.84188778,  0.70928643,  0.87321035],
       [ 0.81885553,  0.92208501,  0.873464  ],
       [ 0.27111984,  0.82213106,  0.55987325]])
>>>
>>> numpy.save('saved-matrix.npy', M) # сохраняет в бинарном формате
>>>
>>> print numpy.load('saved-matrix.npy')
array([[ 0.84188778,  0.70928643,  0.87321035],
       [ 0.81885553,  0.92208501,  0.873464  ],
       [ 0.27111984,  0.82213106,  0.55987325]])
```

# Numpy – array methods

```
>>> print arr.sum()
145
>>> print arr.mean()
14.5
>>> print arr.std()
2.8722813232690143
>>> print arr.max()
19
>>> print arr.min()
10
```

# Numpy – array methods - sorting

```
>>> arr = numpy.array([4.5, 2.3, 6.7, 1.2, 1.8, 5.5])
>>> arr.sort()  # acts on array itself
>>> print(arr)
[ 1.2  1.8  2.3  4.5  5.5  6.7]

>>> x = numpy.array([4.5, 2.3, 6.7, 1.2, 1.8, 5.5])
>>> y = numpy.sort(x)
>>> print(y)
[ 1.2  1.8  2.3  4.5  5.5  6.7]
>>> print(x)
[ 4.5  2.3  6.7  1.2  1.8  5.5]
```

# Numpy – array functions

```
>>> print arr.sum()
45
>>> print numpy.sum(arr)
45
```

```
>>> x = numpy.array([[1,2],[3,4]])
>>> print x
[[1 2]
 [3 4]]
>>> print numpy.log10(x)
[[ 0.          0.30103   ]
 [ 0.47712125  0.60205999]]
```

# Numpy – array operations

```
>>> a = numpy.array([[1.0, 2.0], [4.0, 3.0]])
>>> print a
[[ 1. 2.]
 [ 3. 4.]]

>>> print a.transpose()
array([[ 1., 3.],
       [ 2., 4.]])

>>> print numpy.inv(a)
array([[-2. , 1. ],
       [ 1.5, -0.5]])


>>> j = numpy.array([[0.0, -1.0], [1.0, 0.0]])
>>> print j
array([[0., -1.],
       [1., 0.]])
>>> print j*j # element product
array([[0., 1.],
       [1., 0.]])
>>> print numpy.dot(j, j) # matrix product
array([[-1., 0.],
       [ 0., -1.]])
```

# Numpy – arrays, matrices

В NumPy есть специальный тип matrix для работы с матрицами. Матрицы могут быть созданы вызовом matrix() или mat() или преобразованы из двумерных массивов методом asmatrix().

```
>>> import numpy
>>> m = numpy.mat([[1,2],[3,4]])

>>> a = numpy.array([[1,2],[3,4]])
>>> m = numpy.mat(a)

>>> a = numpy.array([[1,2],[3,4]])
>>> m = numpy.asmatrix(a)
```

# Numpy – matrices

```
>>> a = numpy.array([[1,2],[3,4]])
>>> m = numpy.mat(a) # convert 2-d array to matrix

>>> print a[0]           # result is 1-dimensional
array([1, 2])
>>> print m[0]           # result is 2-dimensional
matrix([[1, 2]])

>>> print a*a       # element-by-element multiplication
array([[ 1, 4], [ 9, 16]])
>>> print m*m       # (algebraic) matrix multiplication
matrix([[ 7, 10], [15, 22]])

>>> print a**3           # element-wise power
array([[ 1, 8], [27, 64]])
>>> print m**3           # matrix multiplication m*m*m
matrix([[ 37, 54], [ 81, 118]])

>>> print m.T       # transpose of the matrix
matrix([[1, 3], [2, 4]])
>>> print m.H       # conjugate transpose (differs from .T for complex matrices)
matrix([[1, 3], [2, 4]])
>>> print m.I       # inverse matrix
matrix([[-2. , 1. ], [ 1.5, -0.5]])
```

# Numpy – Fourier

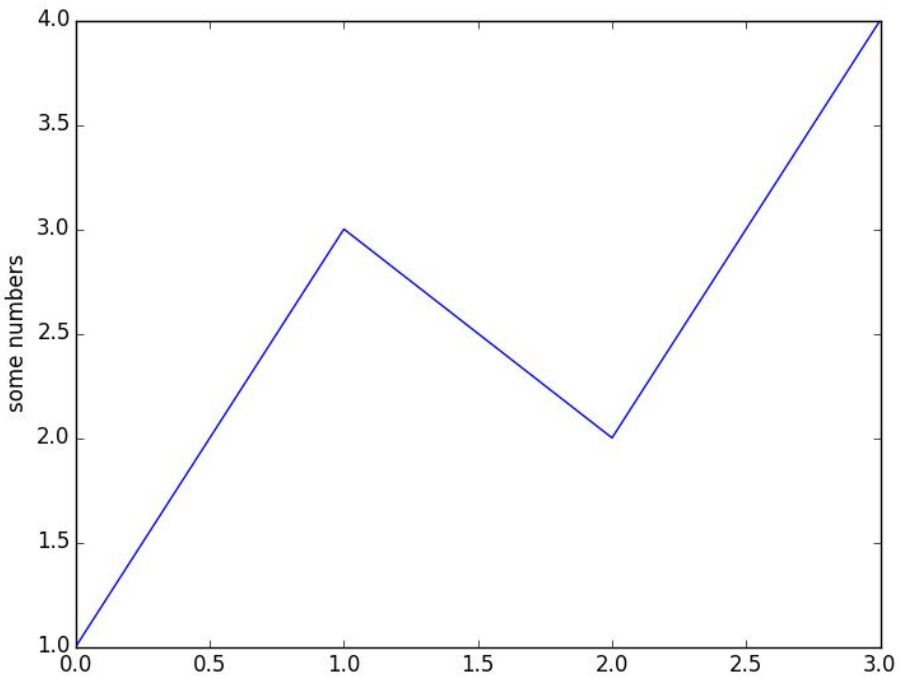
```
>>> a = linspase(0,1,11)
>>> print a
[ 0.   0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1. ]

>>> from numpy.fft import fft, ifft
>>> b = fft(a)
>>> print b
[ 5.50+0.j         -0.55+1.87312798j -0.55+0.85581671j -0.55+0.47657771j
 -0.55+0.25117658j -0.55+0.07907806j -0.55-0.07907806j -0.55-0.25117658j
 -0.55-0.47657771j -0.55-0.85581671j -0.55-1.87312798j]

>>> print(ifft(b))
[  1.61486985e-15+0.j   1.00000000e-01+0.j   2.00000000e-01+0.j
   3.00000000e-01+0.j   4.00000000e-01+0.j   5.00000000e-01+0.j
   6.00000000e-01+0.j   7.00000000e-01+0.j   8.00000000e-01+0.j
   9.00000000e-01+0.j   1.00000000e+00+0.j]
```

# Plotting - matplotlib


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

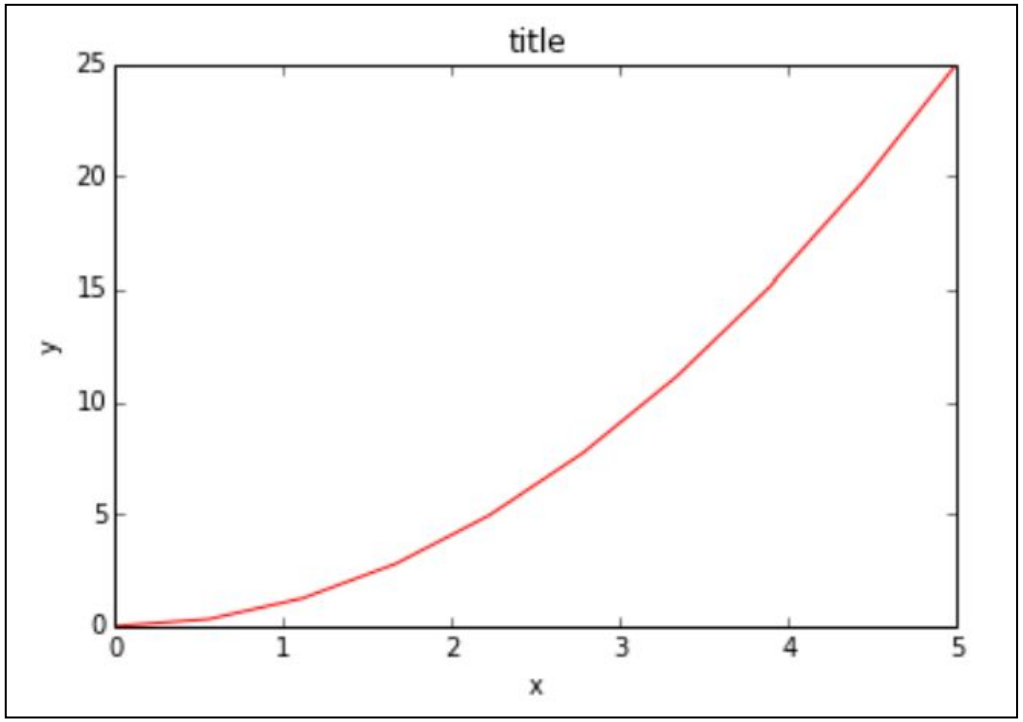
# Matplotlib.pyplot basic example

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



# Matplotlib.pyplot basic example

```
import numpy
import matplotlib.pyplot as plt

x = numpy.linspace(0, 5, 10)
y = x ** 2

plt.plot(x, y, 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.title('title')
plt.show()
```

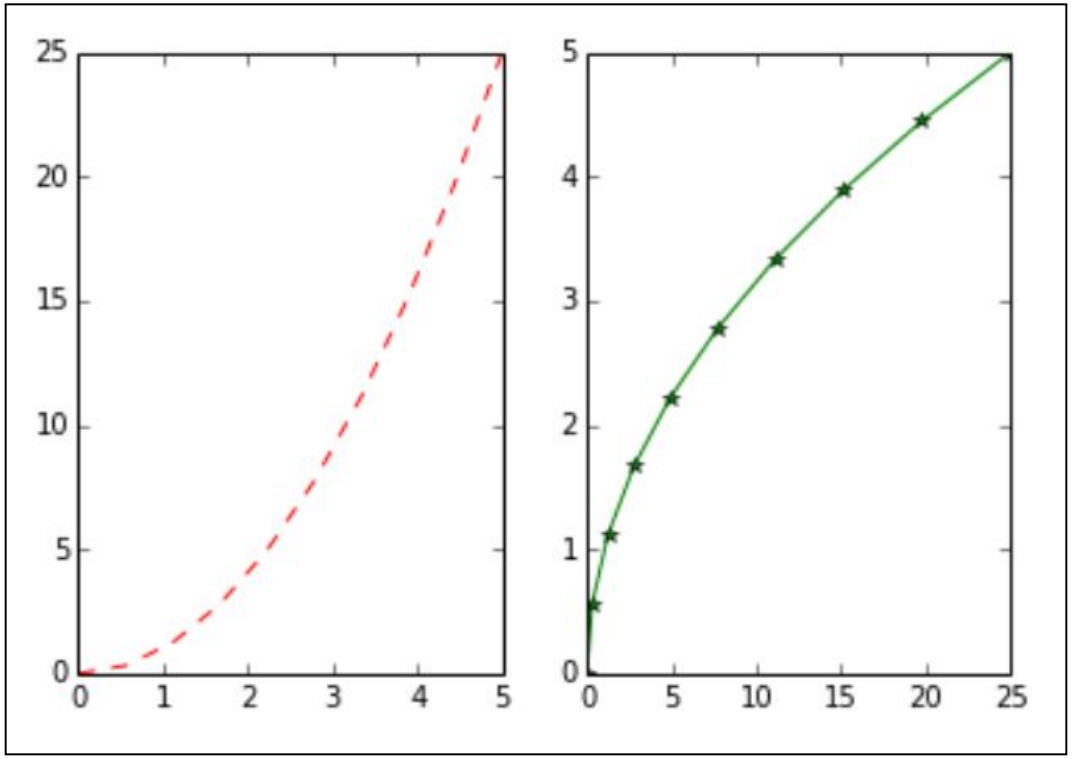


| character | description |
|---|---|
| '-' | solid line style |
| '--' | dashed line style |
| '-.' | dash-dot line style |
| ':' | dotted line style |
| '.' | point marker |
| ',' | pixel marker |
| 'o' | circle marker |
| 'v' | triangle_down marker |
| '^' | triangle_up marker |
| '<' | triangle_left marker |
| '>' | triangle_right marker |
| '1' | tri_down marker |
| '2' | tri_up marker |
| '3' | tri_left marker |
| '4' | tri_right marker |
| 's' | square marker |
| 'p' | pentagon marker |
| '*' | star marker |
| 'h' | hexagon1 marker |
| 'H' | hexagon2 marker |
| '+' | plus marker |
| 'x' | x marker |
| 'D' | diamond marker |
| 'd' | thin_diamond marker |
| '\|' | vline marker |
| '_' | hline marker |

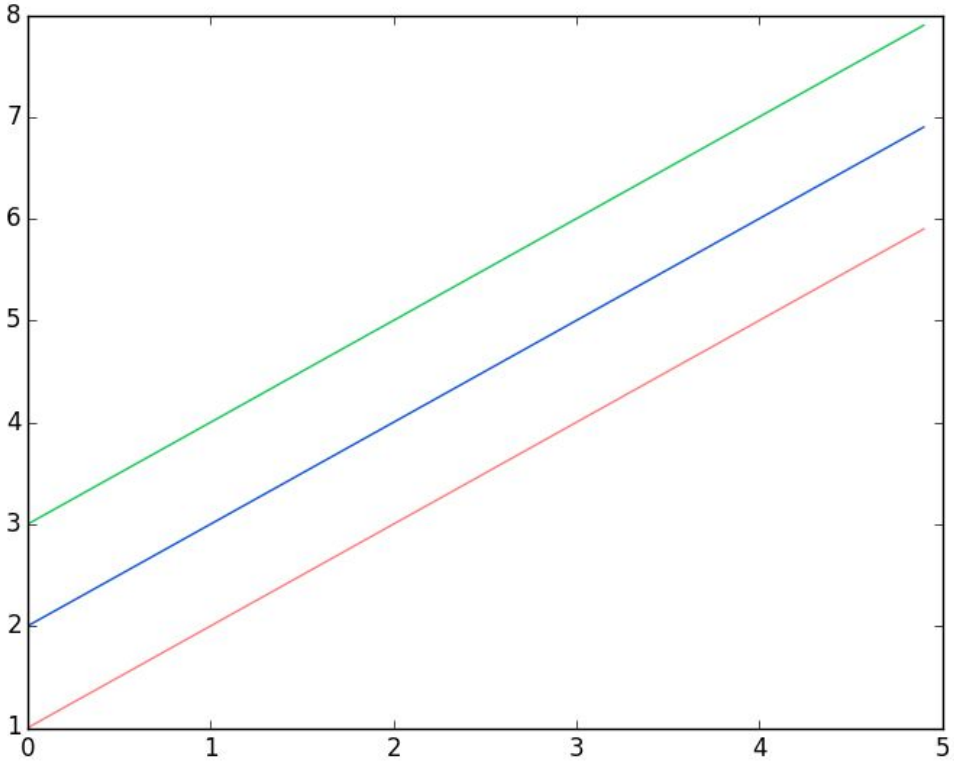
| character | color |
|---|---|
| 'b' | blue |
| 'g' | green |
| 'r' | red |
| 'c' | cyan |
| 'm' | magenta |
| 'y' | yellow |
| 'k' | black |
| 'w' | white |

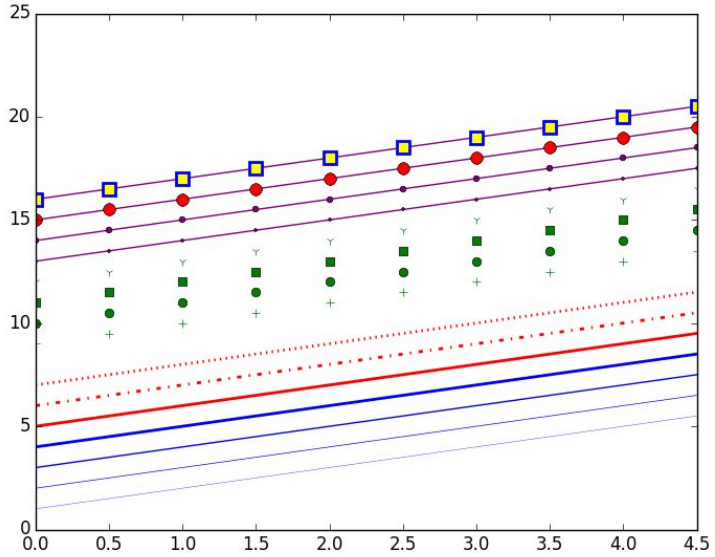
# Matplotlib.pyplot basic example

```
x = numpy.linspace(0, 5, 10)
y = x ** 2

plt.subplot(1,2,1)
plt.plot(x, y, 'r--')

plt.subplot(1,2,2)
plt.plot(y, x, 'g*-')

plt.show()
```



```
x = numpy.linspace(0, 5, 2)

plt.plot(x, x+1, color="red", alpha=0.5) # half-transparant red
plt.plot(x, x+2, color="#1155dd") # RGB hex code for a bluish color
plt.plot(x, x+3, color="#15cc55") # RGB hex code for a greenish color

plt.show()
```

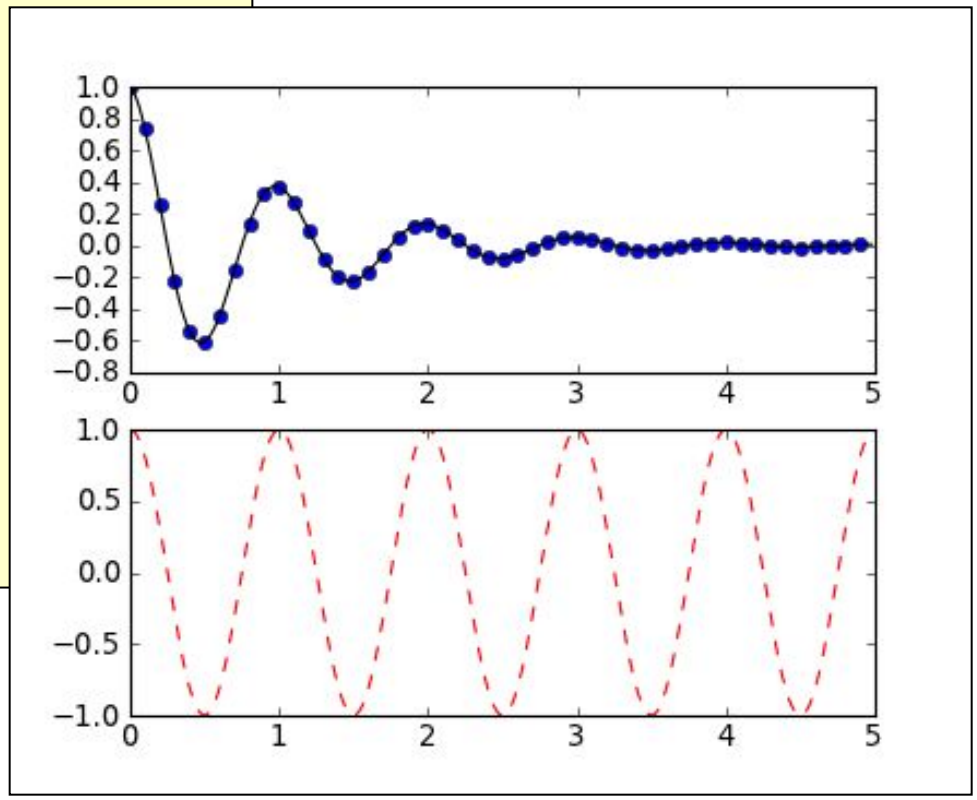





```
plt.plot(x, x+1, color="blue", linewidth=0.25)
plt.plot(x, x+2, color="blue", linewidth=0.50)
plt.plot(x, x+3, color="blue", linewidth=1.00)
plt.plot(x, x+4, color="blue", linewidth=2.00)

# possible linestype options '-', '{', '-.', ':', 'steps'
plt.plot(x, x+5, color="red", lw=2, linestyle='-')
plt.plot(x, x+6, color="red", lw=2, ls='-.')
plt.plot(x, x+7, color="red", lw=2, ls=':')

# possible marker symbols: marker = '+', 'o', '*', 's', ',', '.', '1', '2', '3', '4', ...
plt.plot(x, x+ 9, color="green", lw=2, ls='*', marker='+')
plt.plot(x, x+10, color="green", lw=2, ls='*', marker='o')
plt.plot(x, x+11, color="green", lw=2, ls='*', marker='s')
plt.plot(x, x+12, color="green", lw=2, ls='*', marker='1')

# marker size and color
plt.plot(x, x+13, color="purple", lw=1, ls='-', marker='o', markersize=2)
plt.plot(x, x+14, color="purple", lw=1, ls='-', marker='o', markersize=4)
plt.plot(x, x+15, color="purple", lw=1, ls='-', marker='o', markersize=8,
markerfacecolor="red")
plt.plot(x, x+16, color="purple", lw=1, ls='-', marker='s', markersize=8,
markerfacecolor="yellow", markeredgewidth=2, markeredgecolor="blue")
```

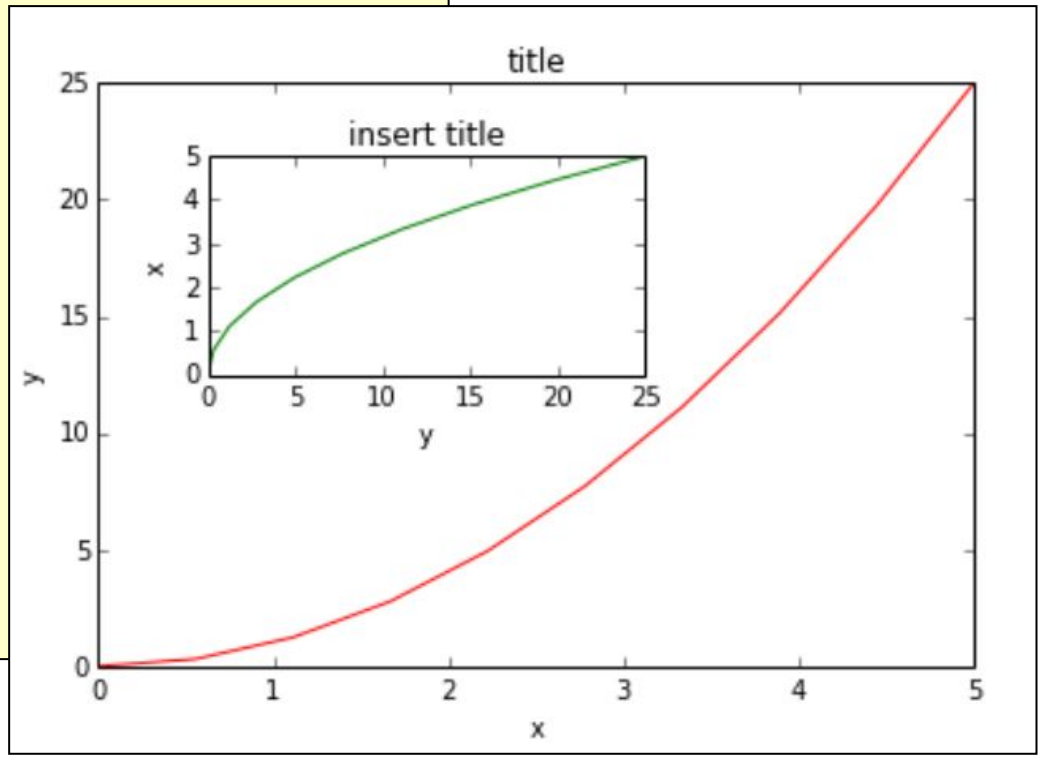
# Matplotlib.pyplot example

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

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.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--')

plt.show()
```



# Matplotlib.pyplot basic example

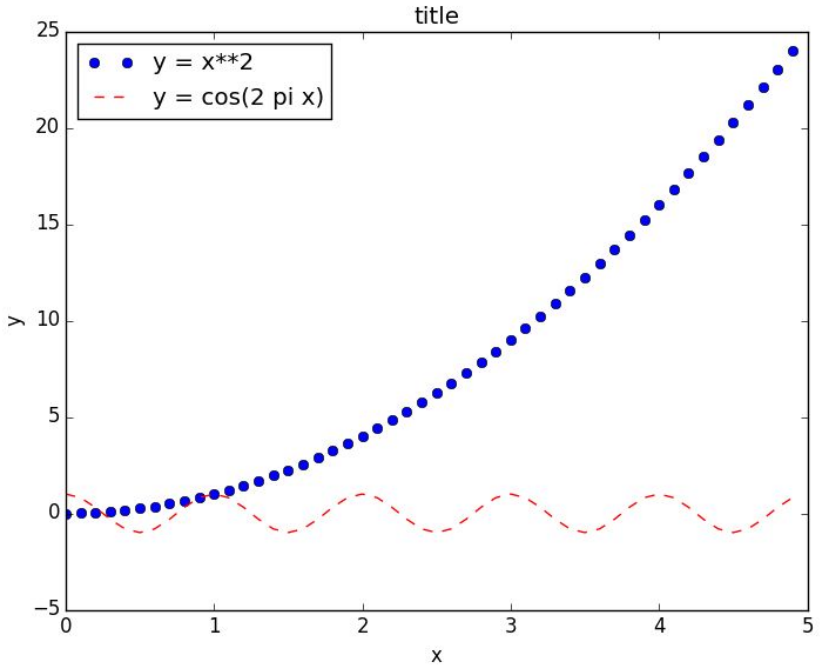
```
fig = plt.figure()

axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes

x = numpy.linspace(0, 5, 10)
y = x ** 2

# main figure
axes1.plot(x, y, 'r')
axes1.set_xlabel('x')
axes1.set_ylabel('y')
axes1.set_title('title')

# insert
axes2.plot(y, x, 'g')
axes2.set_xlabel('y')
axes2.set_ylabel('x')
axes2.set_title('insert title');
```



# Matplotlib.pyplot basic example

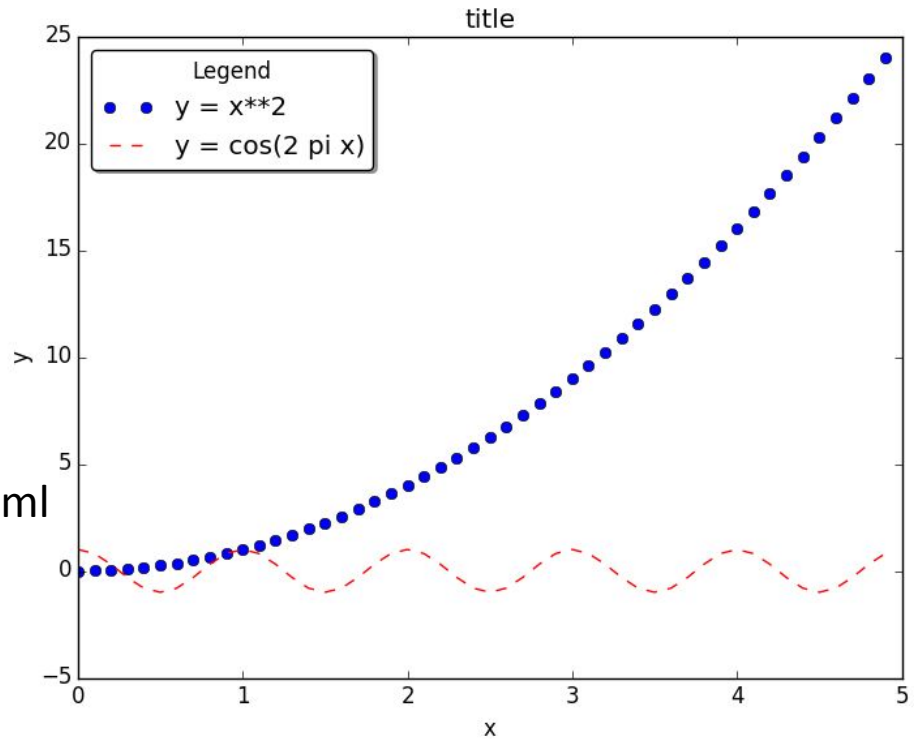
**Labels and legends and titles**

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

x = np.arange(0.0, 5.0, 0.1)

plt.plot(x, x**2, 'bo', label='y = x**2')
plt.plot(x, np.cos(2*np.pi*x), 'r--',
label='y = cos(2 pi x)')
plt.legend(loc=0)
plt.xlabel('x')
plt.ylabel('y')
plt.title('title')

plt.show()
```




```
plt.legend(loc=0) # let matplotlib decide
plt.legend(loc=1) # upper right corner
plt.legend(loc=2) # upper left corner
plt.legend(loc=3) # lower left corner
plt.legend(loc=4) # lower right corner
```

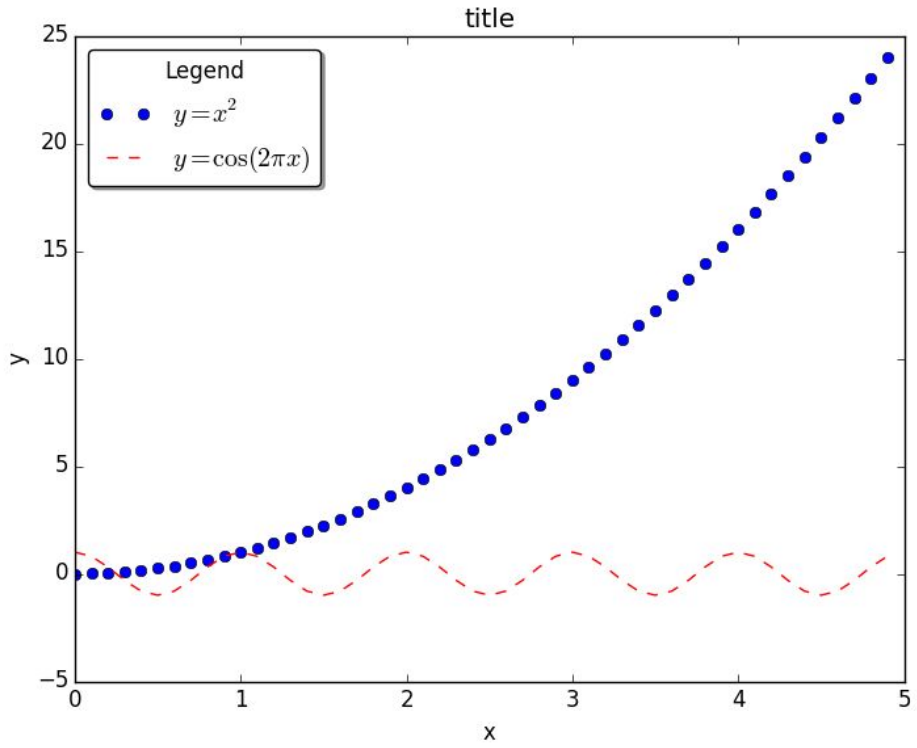
```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0.0, 5.0, 0.1)

plt.plot(x, x**2, 'bo', label='y = x**2')
plt.plot(x, np.cos(2*np.pi*x), 'r--', label='y = cos(2 pi x)')
plt.legend(loc=0,fancybox=True,shadow=True,title='Legend')
plt.xlabel('x')
plt.ylabel('y')
plt.title('title')

plt.show()
```




http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.legend

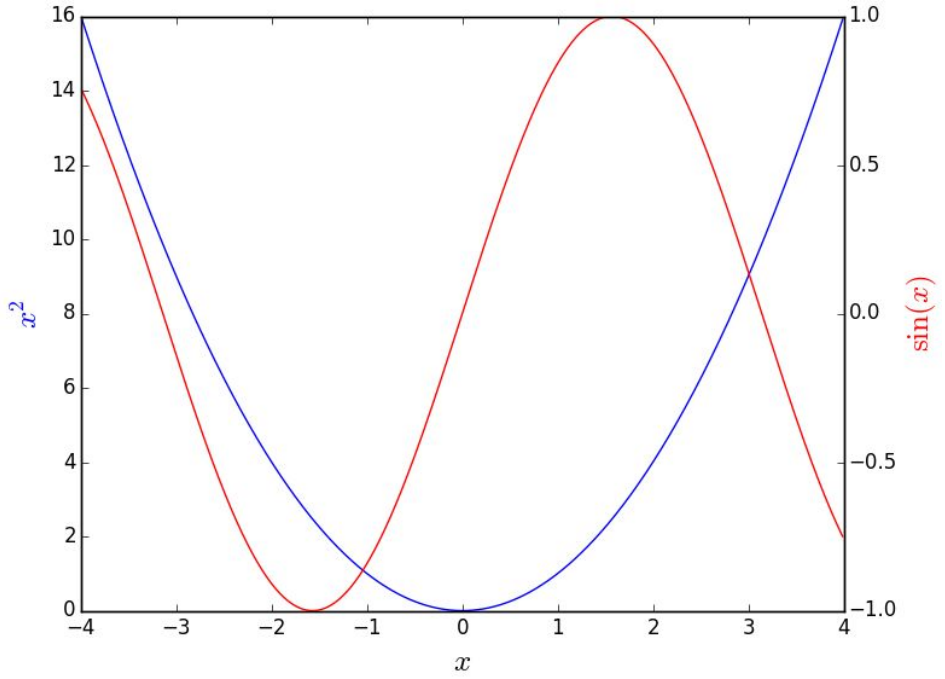
# Matplotlib.pyplot basic example

```
plt.plot(x, x**2, 'bo', label='$y = x^2$')
plt.plot(x, np.cos(2*np.pi*x), 'r--', label='$y = \\cos(2 \\pi x)$')
#plt.plot(x, np.cos(2*np.pi*x), 'r--', label=r'$y = \cos(2 \pi x)$')
```

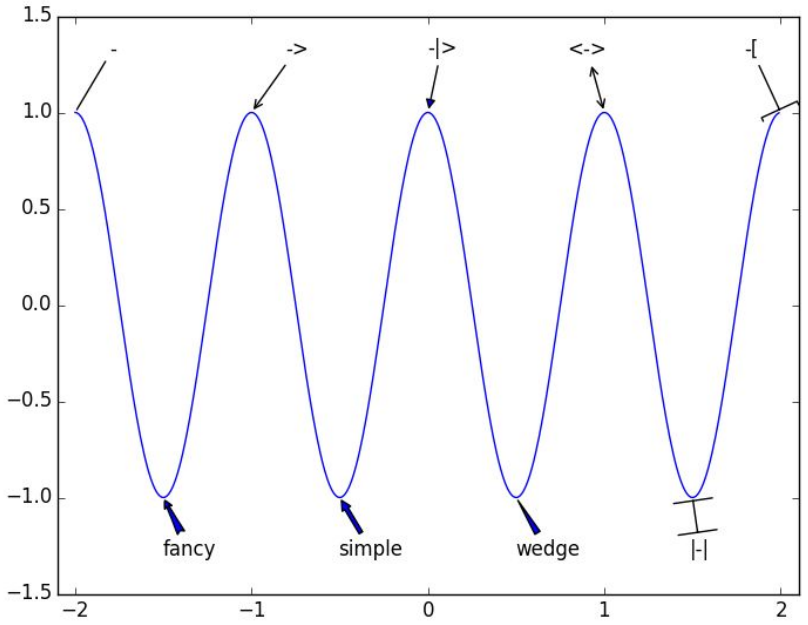


# Matplotlib.pyplot basic example

```
x = np.arange(-4.0, 4.0, 0.01)

plt.plot(x, x**2, color='blue')
plt.xlabel('$x$', fontsize=18)
plt.ylabel('$x^2$', color='blue', fontsize=18)

ax2 = plt.twinx()
ax2.plot(x, np.sin(x), color='red')
ax2.set_ylabel('$\\sin(x)$', color='red', fontsize=18)

plt.show()
```



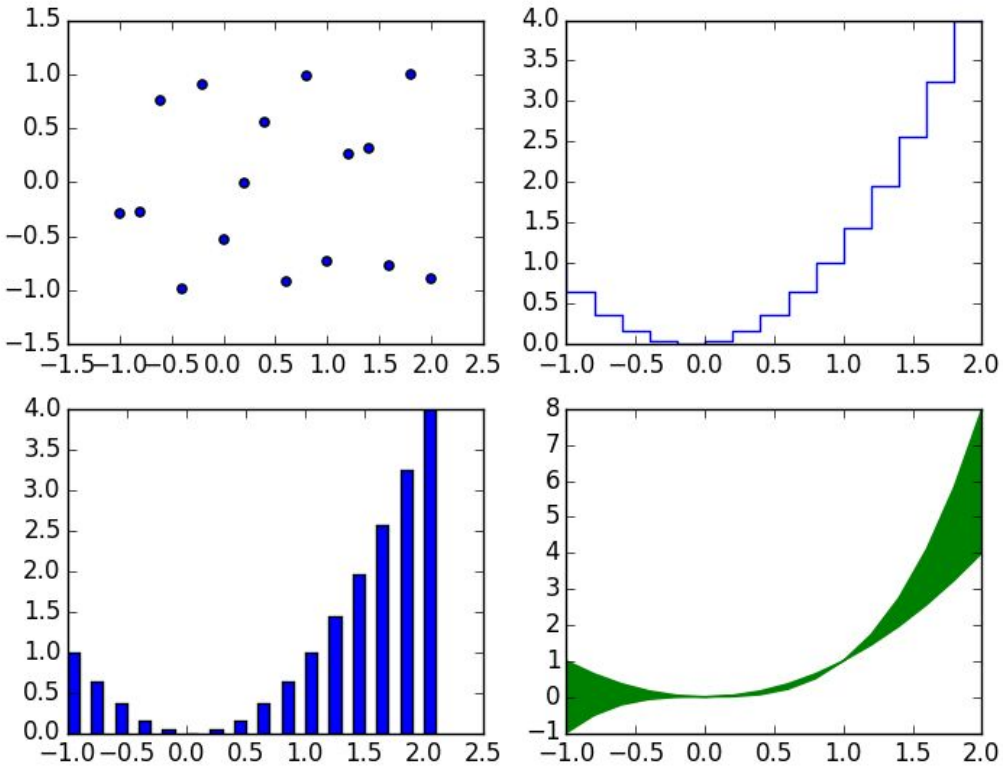
# Overwhelming annotation




```
x = np.arange(-2.0, 2.0, 0.01)

plt.plot(x, np.cos(2*np.pi*x))
plt.xlim(-2.1, 2.1)
plt.ylim(-1.5, 1.5)

plt.annotate('-',   xy=(-2,1), xytext=(-1.8,1.3), arrowprops=dict(arrowstyle='-'))
plt.annotate('->',  xy=(-1,1), xytext=(-0.8,1.3), arrowprops=dict(arrowstyle='->'))
plt.annotate('-|>', xy=(0,1),  xytext=(0,1.3),    arrowprops=dict(arrowstyle='-|>'))
plt.annotate('<->', xy=(1,1),  xytext=(0.8,1.3),  arrowprops=dict(arrowstyle='<->'))
plt.annotate('-[',  xy=(2,1),  xytext=(1.8,1.3),  arrowprops=dict(arrowstyle='-['))

plt.annotate('fancy',  xy=(-1.5,-1), xytext=(-1.5,-1.3), arrowprops=dict(arrowstyle='fancy'))
plt.annotate('simple', xy=(-0.5,-1), xytext=(-0.5,-1.3), arrowprops=dict(arrowstyle='simple'))
plt.annotate('wedge',  xy=(0.5,-1),  xytext=(0.5,-1.3),  arrowprops=dict(arrowstyle='wedge'))
plt.annotate('|-|',    xy=(1.5,-1),  xytext=(1.5,-1.3),  arrowprops=dict(arrowstyle='|-|'))

plt.show()
```

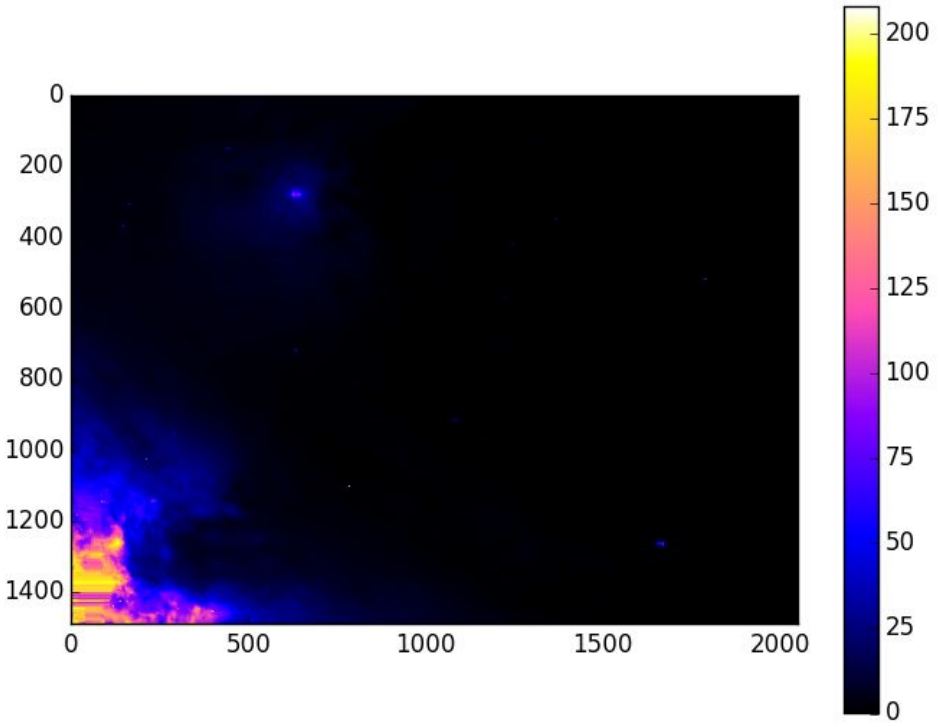
# Matplotlib.pyplot basic example

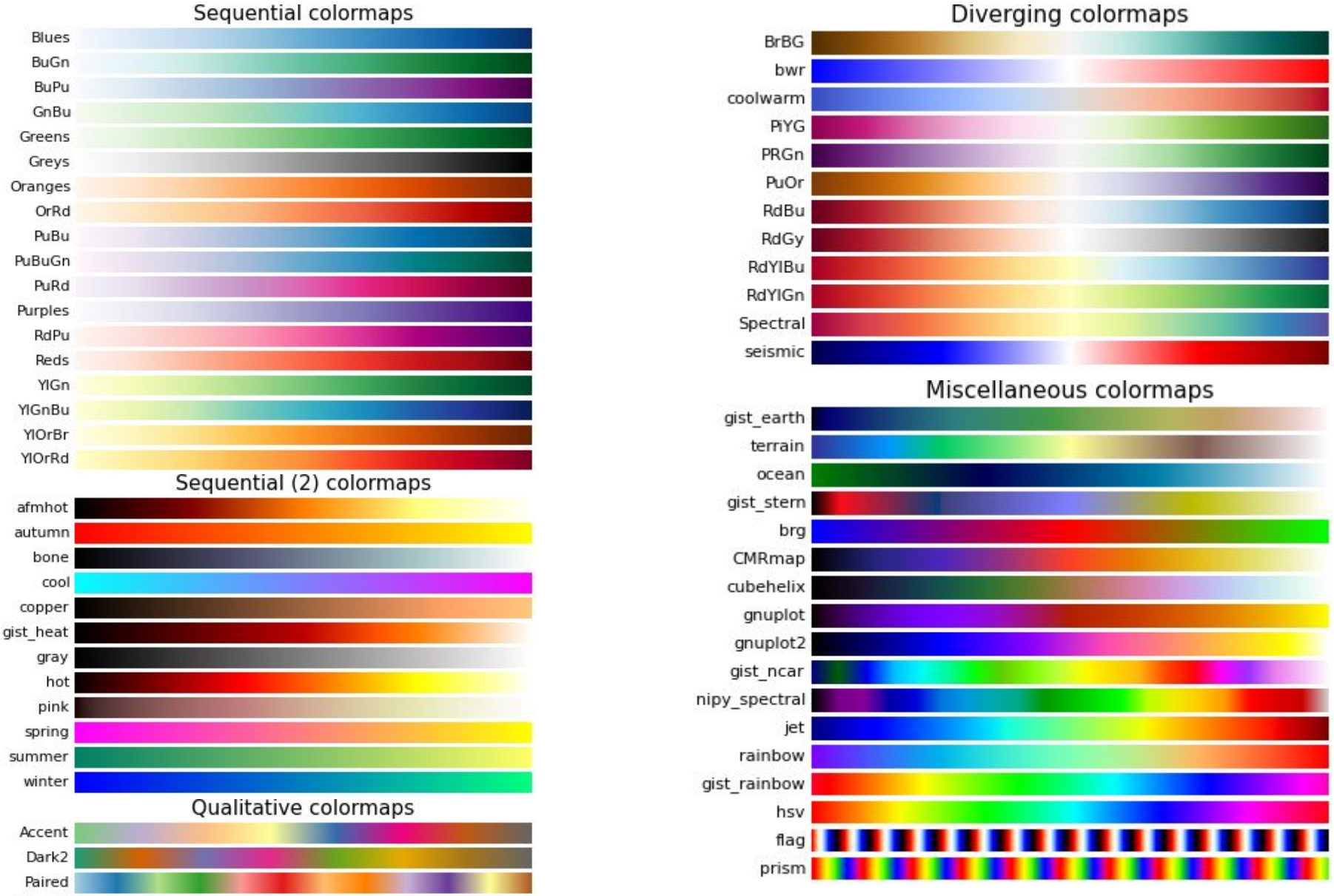
```
x = np.linspace(-1.0, 2.0, 16)

plt.subplot(221)
plt.scatter(x, np.sin(50 * x + 12))

plt.subplot(222)
plt.step(x, x**2)

plt.subplot(223)
plt.bar(x, x**2, width=0.1)

plt.subplot(224)
plt.fill_between(x, x**2, x**3, color='green')

plt.show()
```



# Matplotlib.pyplot basic example

```
import matplotlib.pyplot as plt
import pyfits

data = pyfits.getdata('frame-g-006073-4-0063.fits')

plt.imshow(data, cmap='gnuplot2')
plt.colorbar()

plt.show()
```




Sequential colormaps
Blues
BuGn
BuPu
GnBu
Greens
Greys
Oranges
OrRd
PuBu
PuBuGn
PuRd
Purples
RdPu
Reds
YlGn
YlGnBu
YlOrBr
YlOrRd
Sequential (2) colormaps
afmhot
autumn
bone
cool
copper
gist_heat
gray
hot
pink
spring
summer
winter
Qualitative colormaps
Accent
Dark2
Paired
Pastel1
Pastel2
Set1
Set2
Set3
Diverging colormaps
BrBG
bwr
coolwarm
PiYG
PRGn
PuOr
RdBu
RdGy
RdYlBu
RdYlGn
Spectral
seismic
Miscellaneous colormaps
gist_earth
terrain
ocean
gist_stern
brg
CMRmap
cubehelix
gnuplot
gnuplot2
gist_ncar
nipy_spectral
jet
rainbow
gist_rainbow
hsv
flag
prism

```
# plt.show()

plt.savefig('filename', orientation='landscape', format='eps')

# orientation='portrait'
#             'landscape‘

# format='png'(по умолчанию)
#        'pdf'
#        'eps'
#    'ps'
#    'jpeg'
#    'svg'
```

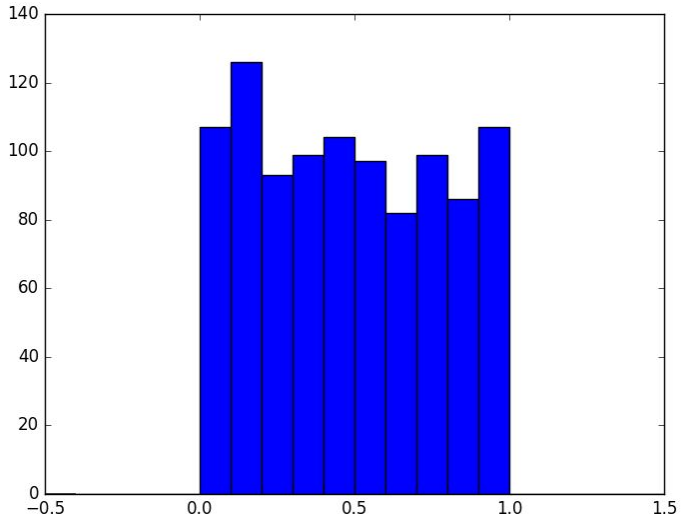
# Пакет SciPy

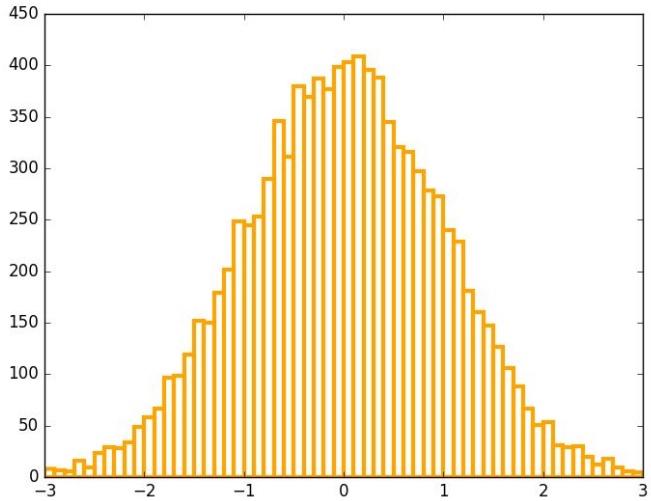
# Генерация и визуализация случайных последовательностей

- Субмодуль **numpy.random** включает векторные версии нескольких различных генераторов случайных чисел.

```
>>> from numpy.random import *
>>> import matplotlib.pyplot as plt
>>> x=rand(1000)
>>> bins=linspace(-0.5,1.5,21)
>>> plt.hist(x,bins)
>>> plt.xlim(-0.5,1.5)
>>> plt.savefig('plot1.png')
>>> plt.show()
```

```
>>> from numpy.random import *
>>> import matplotlib.pyplot as plt
>>> x=randn(10000)
>>> bins=linspace(-3,3,61)
>>> plt.hist(x,bins,fill=0,lw=2,ec='orange')
>>> plt.xlim(-3,3)
>>> plt.savefig('plot2')
>>> plt.show()
```

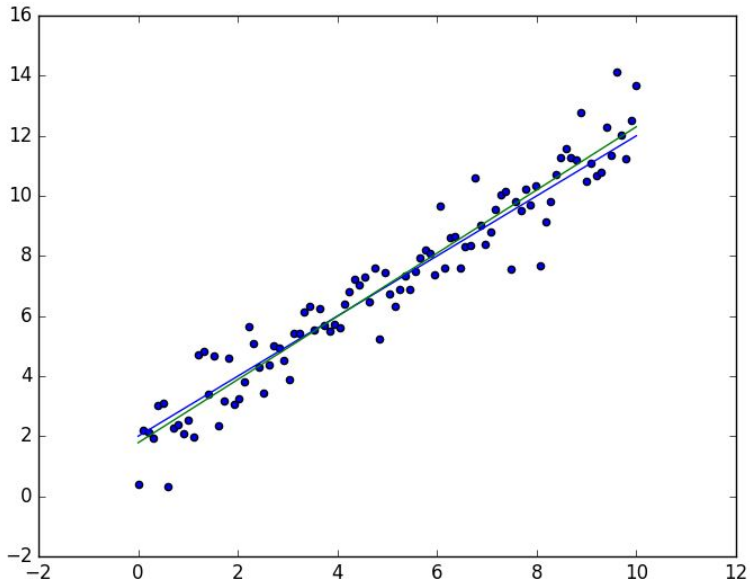


# Data Modeling and Fitting

- curve_fit – метод, позволяющий аппроксимировать набор точек некоторой функциональной зависимостью, основанный на минимизации невязки

```
>>> import numpy as np
>>> from scipy.optimize import curve_fit
>>> import matplotlib.pyplot as plt

>>> def func(x, a, b):
>>>     return a * x + b
>>> x = np.linspace(0, 10, 100)
>>> y = func(x, 1, 2)
>>> yn = y + 0.9 * np.random.normal(size=len(x))
>>> abopt, abcov = curve_fit(func, x, yn)
>>> print abopt
>>> print abcov

[ 1.05170285  1.78315256]
[[ 0.00085998 -0.0042999 ]
 [-0.0042999   0.02881076]]
```

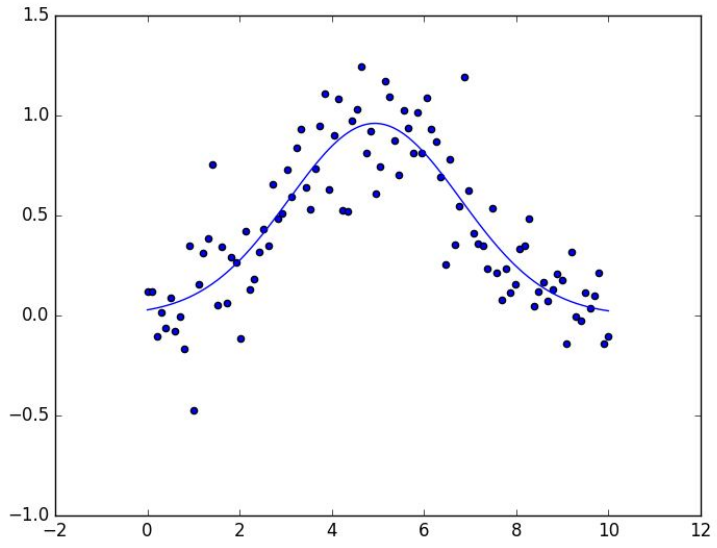


# Data Modeling and Fitting

- curve_fit – метод, позволяющий аппроксимировать набор точек некоторой функциональной зависимомтью, основанный на минимизации невязки

```
>>> import numpy as np
>>> from scipy.optimize import curve_fit
>>> import matplotlib.pyplot as plt

>>> def func(x, a, b, c):
>>>     return a*np.exp(-(x-b)**2/(2*c**2))
>>> x = np.linspace(0, 10, 100)
>>> y = func(x, 1, 5, 2)
>>> yn = y + 0.2 * np.random.normal(size=len(x))
>>> abopt, abcov = curve_fit(func, x, yn)
>>> print abopt
>>> print abcov

[ 0.96080062  4.93407373 -1.85655036]
[[  1.83508136e-03   1.88545100e-06   2.38610451e-03]
 [  1.88545100e-06   9.11757267e-03   8.51863087e-06]
 [  2.38610451e-03   8.51863087e-06   9.23829956e-03]]
```

$$a * \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$$



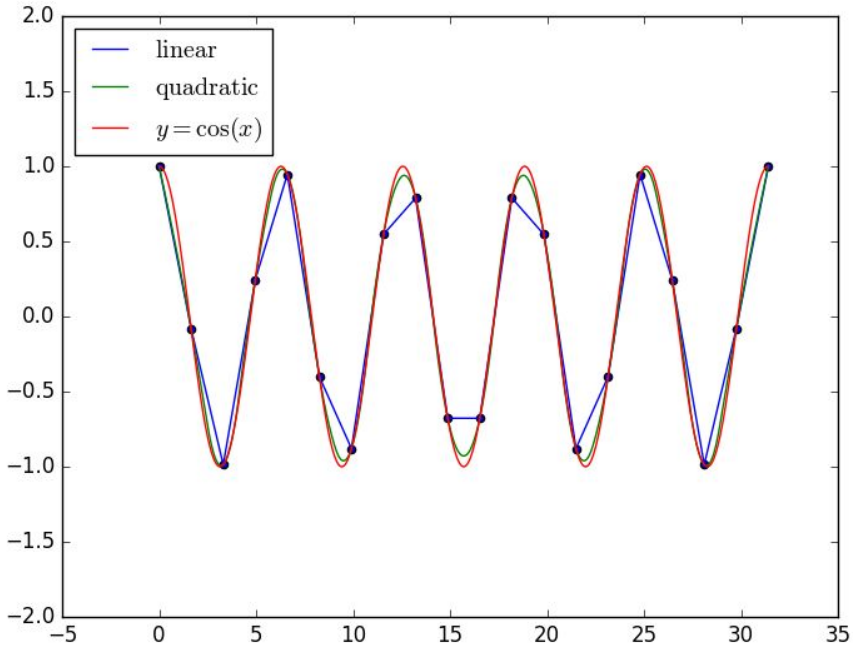
- функция **fsolve** – решение уравнений

```
fsolve(<функция>, <стартовая точка>)
```

```
>>> import numpy as np
>>> from scipy.optimize import fsolve
>>> x=fsolve(lambda x:np.sin(x)-x/10,np.pi/2)
>>> print x
>>> # Проверка
>>> print np.sin(x)-x/10
[ 2.85234189]
[ -1.11022302e-16]
```

# Interpolation

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> from scipy.interpolate import interp1d

>>> x = np.linspace(0, 10 * np.pi, 20)
>>> y = np.cos(x)
>>> fl = interp1d(x, y, kind='linear')
>>> fq = interp1d(x, y, kind='quadratic')

>>> xint = np.linspace(x.min(), x.max(), 1000)
>>> yintl = fl(xint)
>>> yintq = fq(xint)

>>> plt.scatter(x,y)
>>> plt.plot(xint,yintl,label='$\\rm linear$')
>>> plt.plot(xint,yintq, label='$\\rm quadratic$')
>>> plt.plot(xint,np.cos(xint),label='$y = \\cos(x)$')
>>> plt.legend(loc=2)
>>> plt.ylim(-2,2)
>>> plt.savefig('plot5.png')
>>> plt.show()
```



# Интегрирование

Субмодуль **scipy.integrate** обеспечивает стандартные методы численного интегрирования.

```
>>> from scipy.integrate import *

[<интеграл>,<ошибка>] = quad(f(x),xmin,xmax)
inf изображает бесконечный предел интегрирования
```

```
>>> import numpy as np
>>> from scipy.integrate import *
>>> f = quad(np.sin,0, np.pi)
>>> print f
(2.0, 2.220446049250313e-14)

>>> def func(x):
       return np.exp(-x**2)
>>> f = quad(func,-np.inf,np.inf)
>>> print f
(1.7724538509055159, 1.4202636756659625e-08)

>>> f = quad(lambda x:np.exp(-x**2),-np.inf,np.inf)
>>> print f
(1.7724538509055159, 1.4202636756659625e-08)
```

# Интегрирование

```
>>> import numpy as np
>>> from scipy.integrate import *

>>> x=np.linspace(0,np.pi,10)
>>> f = trapz(np.sin(x),x)
>>> print f
1.97965081122

>>> f = simps(np.sin(x),x)
>>> print f
1.99954873658
```

# Решение дифференциальных уравнений
# Функция odeint

```
odeint(f(x,t),x0,<вектор t>)
```

В качестве примера рассмотрим систему с сухим трением:

$$\frac{dv}{dt} = -K\,\mathrm{sign}(v) + F(t)\,,$$

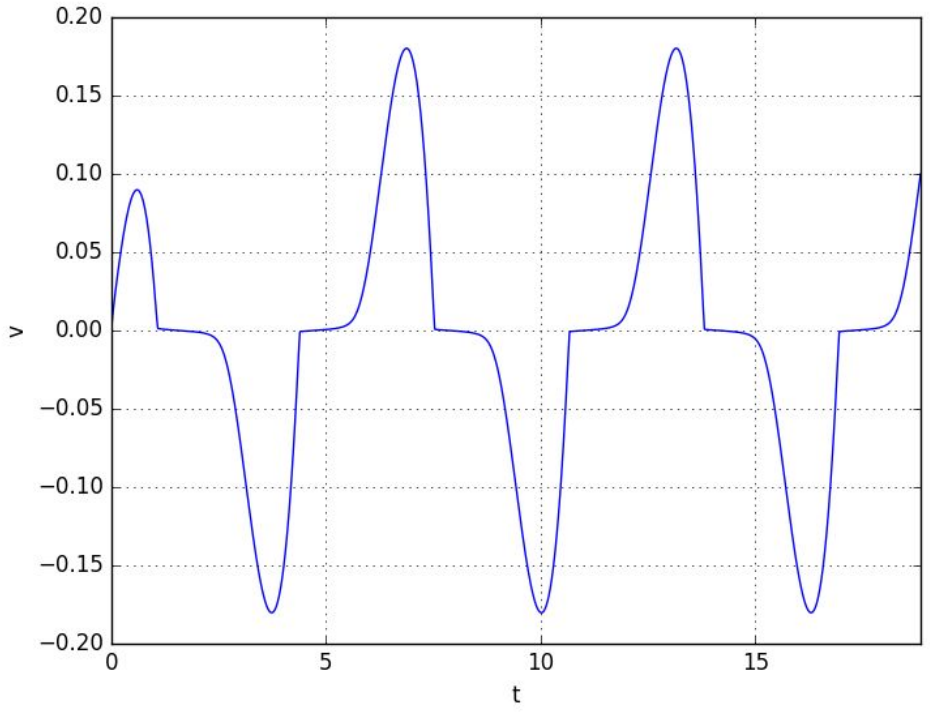
где $v$ – скорость тела, $F(t)$ – внешняя силы и $K$ – коэффициент трения.
Чтобы не использовать разрывную функцию sign, заменим ее арктангенсом:

$$\frac{dv}{dt} = -\frac{2K}{\pi}\arctan(\alpha v) + F(t)\,,$$

где $\alpha$ – большой коэффициент.

# Решение дифференциальных уравнений
# Функция integrate.odeint

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from scipy.integrate import odeint

>>> t = np.linspace(0,6*np.pi,10000)
>>> def f(v,t):
>>>     return -(2/np.pi)*np.arctan(1000*v)+1.2*np.cos(t)

>>> v = odeint(f,0,t)

>>> plt.plot(t,v)
>>> plt.xlabel('t')
>>> plt.ylabel('v')
>>> plt.xlim(0,6*np.pi)
>>> plt.grid()
>>> plt.savefig('plot6‘)
```



# Решение дифференциальных уравнений
## Функция integrate.odeint

В качестве примера уравнения второго порядка, рассмотрим уравнение движения нелинейного осциллятора:

$$\frac{d^2x}{dt^2} + \omega^2(1 - \xi x^2)x = F\cos(\omega t)\,,$$

где $\xi$ – фактор нелинейности. Разделим уравнение на два первого порядка:

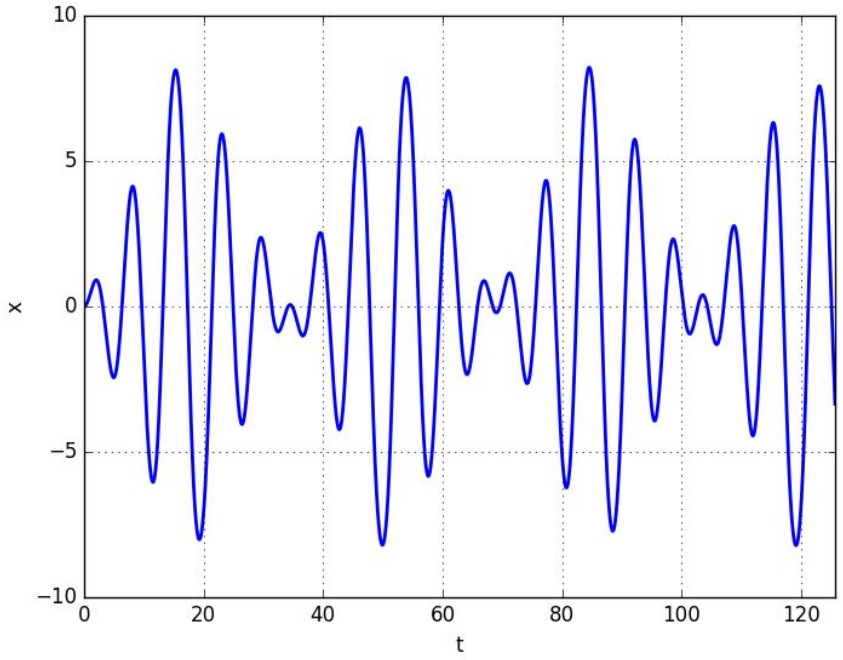
$$\frac{dx}{dt} = v\,,$$
$$\frac{dv}{dt} + \omega^2(1 - \xi x^2)x = F\cos(\omega t)\,,$$

($v$ – скорость) и запишем в стандартном виде:

$$\frac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \begin{pmatrix} v \\ -\omega^2(1 - \xi x^2)x + F\cos(\omega t) \end{pmatrix}\,.$$

# Решение дифференциальных уравнений
# Функция integrate.odeint

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> from scipy.integrate import *

>>> t = np.linspace(0,40*np.pi,10000)
>>> def f(y,t):
>>>     x,v = y
>>>     return [v, -(1-x**2/100)*x+np.cos(t)]

>>> result = odeint(f,[0,0],t)
>>> x = result[:,0]
>>> v = result[:,1]

>>> plt.plot(t,x,lw=2)
>>> plt.xlabel('t')
>>> plt.ylabel('x')
>>> plt.xlim(0,40*np.pi)
>>> plt.grid()
>>> plt.savefig('plot7')
```



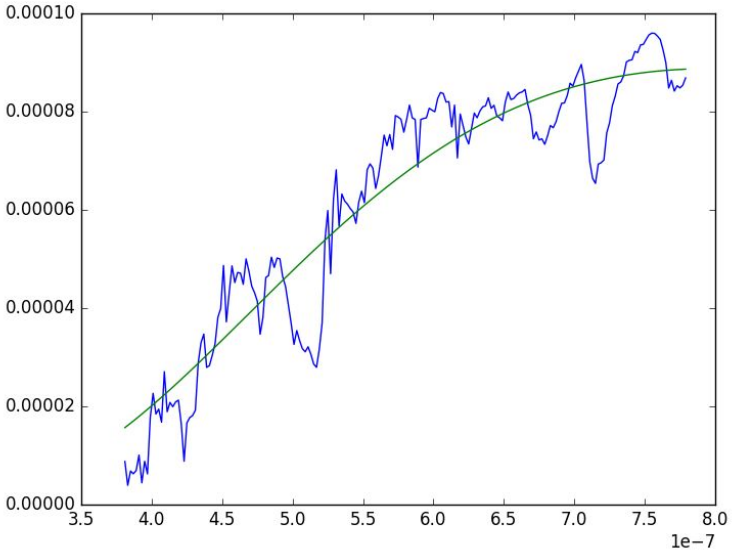
- На языке программирования Python для спектра звезды, полученного при наблюдениях в оптическом диапазоне (spec_star.txt), определить температуру звезды, используя аппроксимацию спектра законом излучения черного тела. Построить исходный спектр и спектр АЧТ для полученных температуры и интенсивности.
- В файле spec_star.txt первая колонка соответствует длине волны [Δ], вторая – интенсивности на данной длине волны.

$$B_\lambda(T) = \frac{2\pi c^2 h}{\lambda^5 (e^{\frac{hc}{\lambda k T}} - 1)}$$

Вт/м³ = Дж/(м³ с)

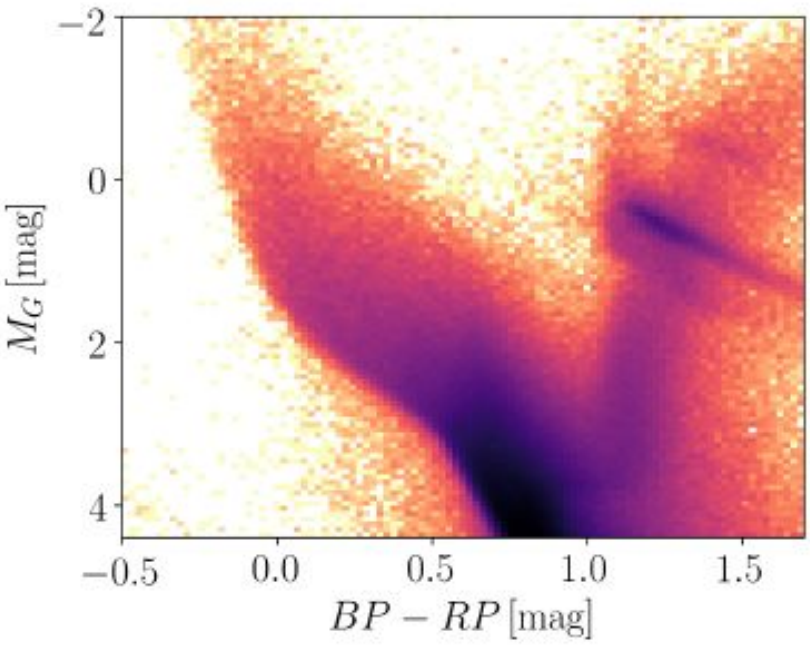
Результат:
lgT=3.6

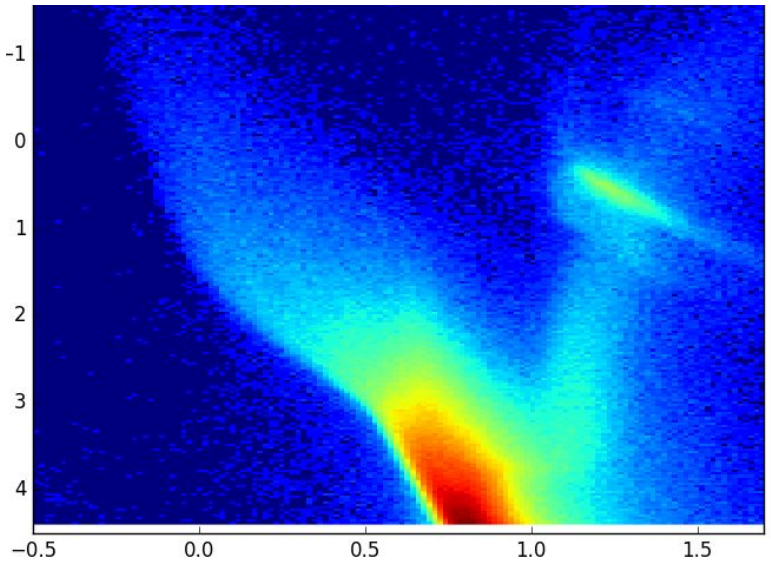
c = 2.997925e8
k = 1.380622e-23
h = 6.626196e-34



# GAIA DR2
# http://gea.esac.esa.int/archive/

Сделать выборку звезд в окрестности Солнца (<= 0.5 пк) из




parallax_over_error >= 5
M_G <= 4.4 (m_G = phot_g_mean_mag)
BP-RP <= 1.7 ((phot_bp_mean_mag - phot_rp_mean_mag))
parallax >= 2.

# Абсолютная звёздная величина

10 пк

и нет поглощения света

*M*

$$m = M + 5\,lg(R/10\text{ пк})$$

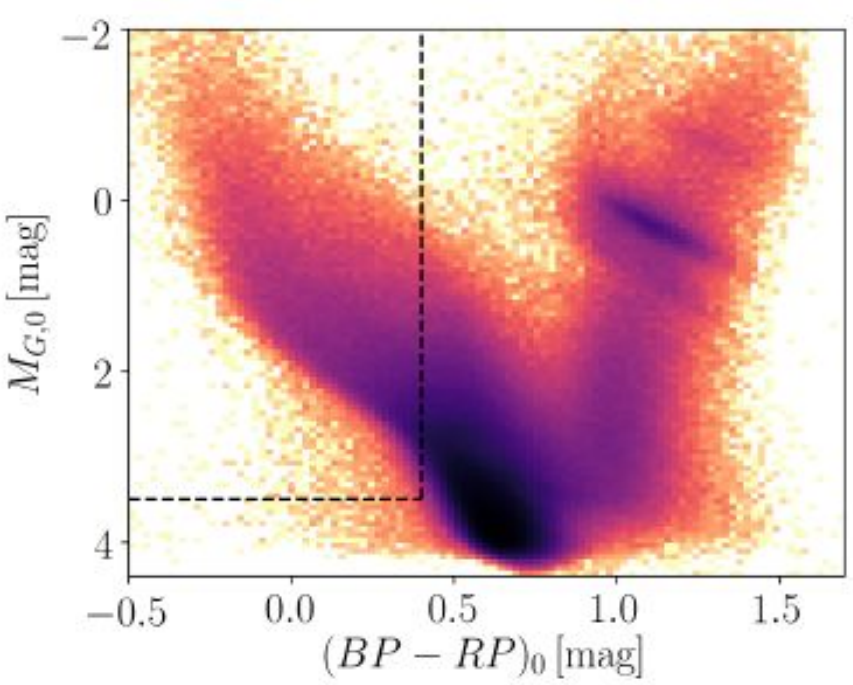
$$m = M - 5 + 5\,lgR\ (\text{пк})$$

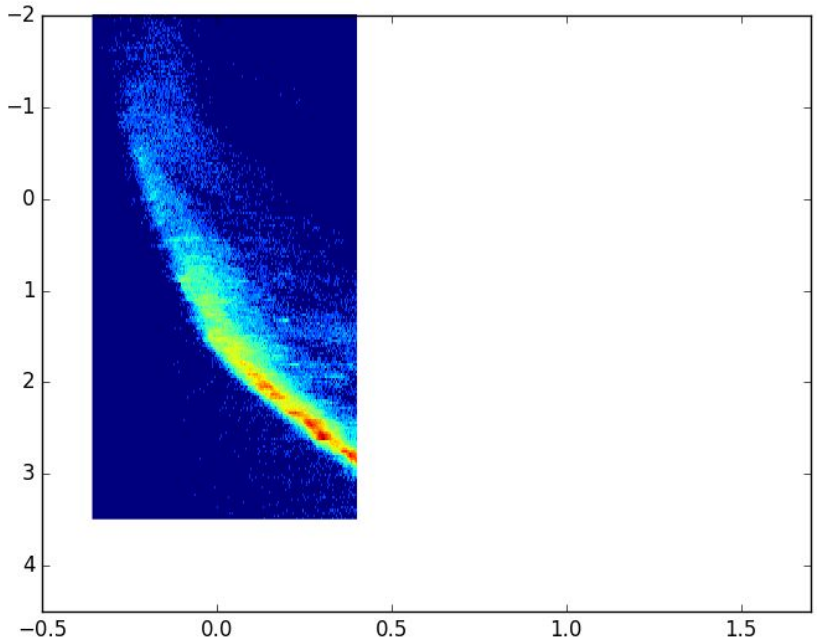
$$M = m + 5 - 5\,lgR = m + 5 + 5\,lg\,p('')$$

$$m - M = -5 + 5\,lgR$$ модуль расстояния

# GAIA DR2

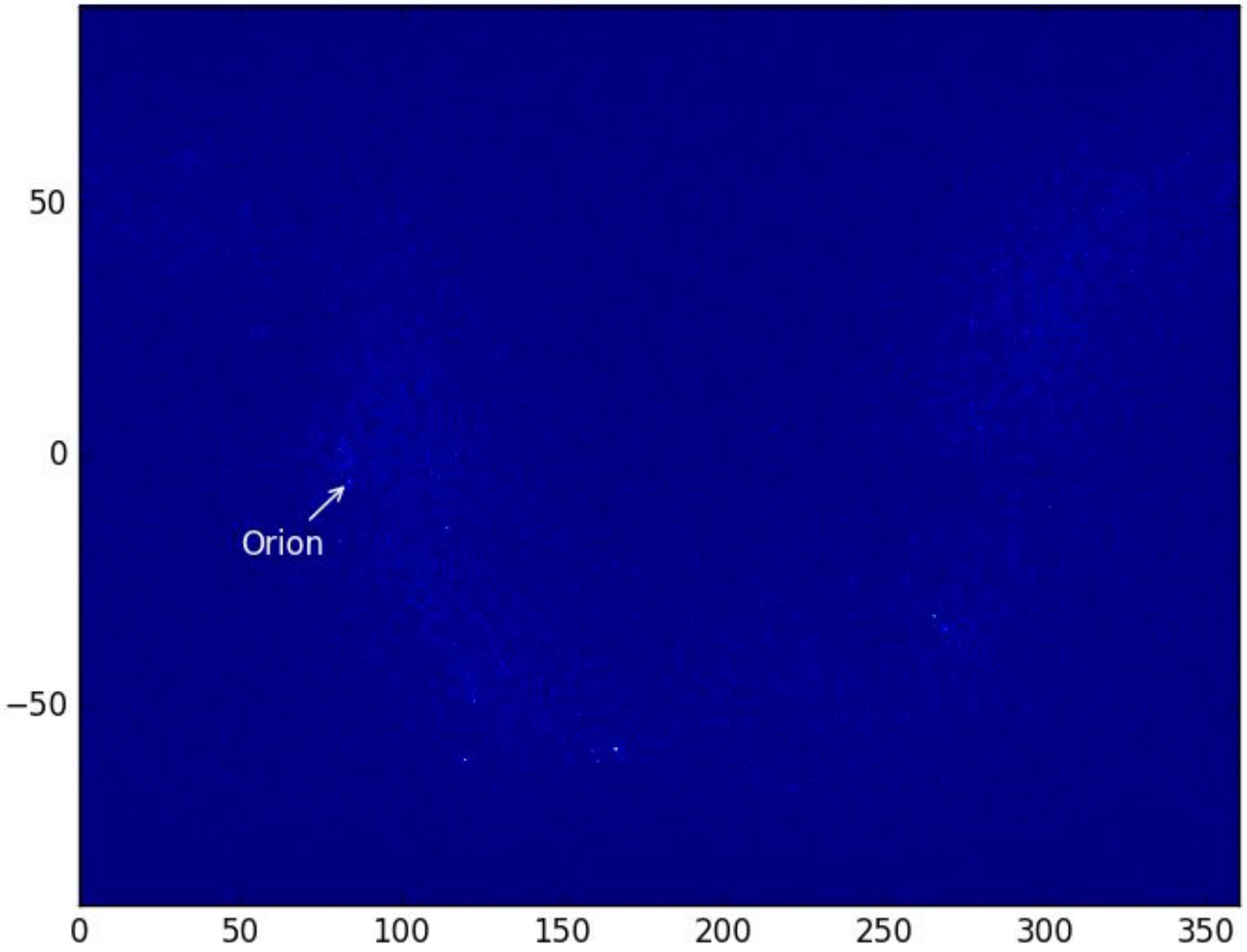
Сделать выборку молодых звезд в окрестности Солнца (<= 0.5 пк) с учетом экстинкции (a_g_val, e_bp_min_rp_val) из gaiadr2.gaia_source.



# GAIA DR2

Для молодых звезд в окрестности Солнца (<= 0.5 пк) с учетом экстинкции из gaiadr2.gaia_source построить распределение в координ



# Numpy, matplotlib

```
data = np.genfromtxt('name.csv', delimiter=',', skip_header=1,
usecols=(1,2))

plt.hist2d(ra, dec, bins=1000, norm=mcolors.PowerNorm(gamma))
```

0<gamma<1