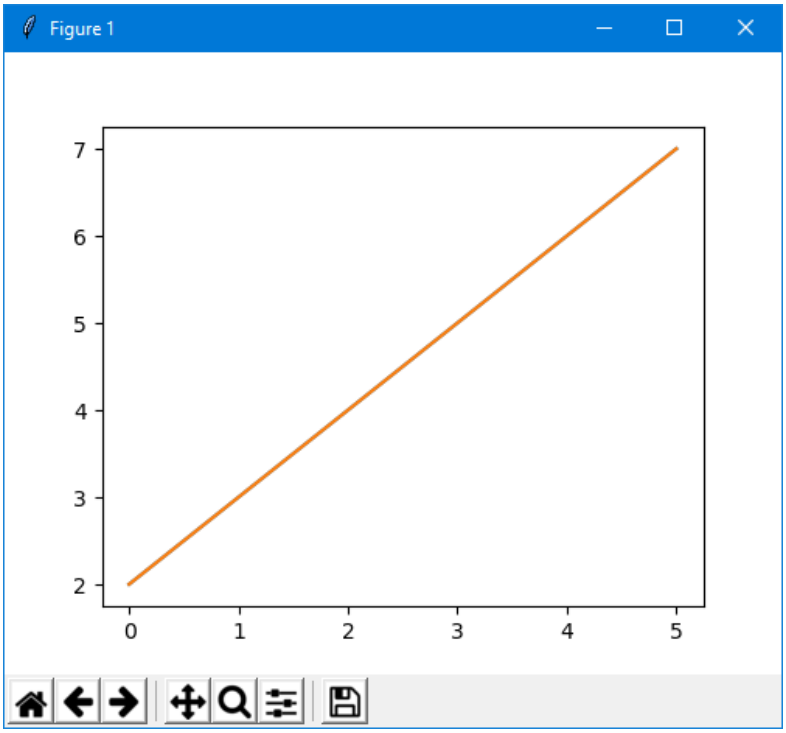
Тема 4. Побудова графіків функцій

П1. Список y координат, x координаты числа 0, 1, 2, 3...

Список *y*[2,3,4,5,6,7] координат, *x* координаты образуют последовательность 0, 1, 2, ...:

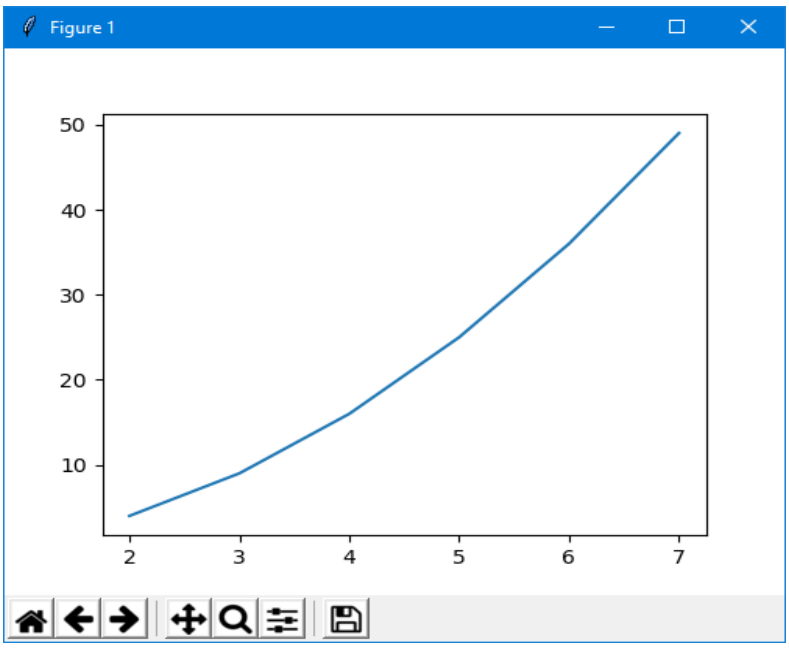
```
plt.plot([2,3,4,5,6,7]);plt.show()
```



П2. Списоки x и y-координат

Список x и y координат - [2,3,4,5,6,7], [4,9,16,25,36,49]);

```
plt.plot([2,3,4,5,6,7],[4,9,16,25,36,49]);plt.show()
```

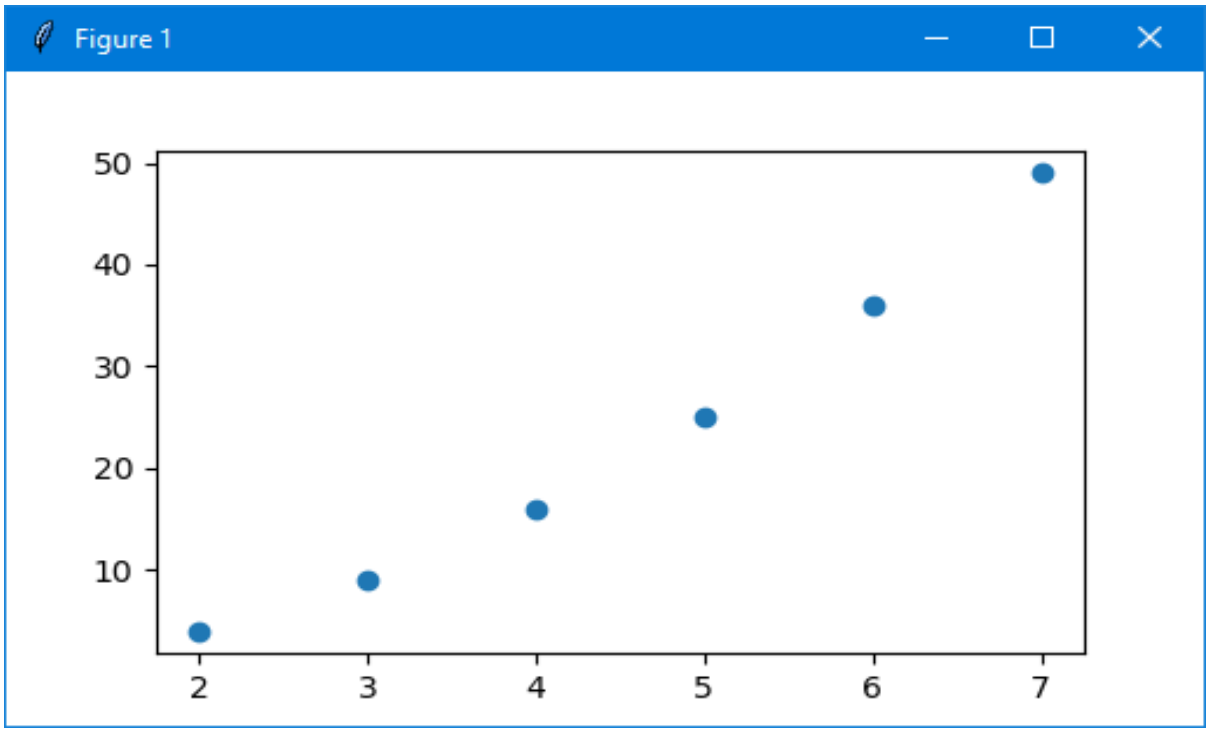


П3. Изображаем точки

Список x и y координат - [2,3,4,5,6,7], [4,9,16,25,36,49]);

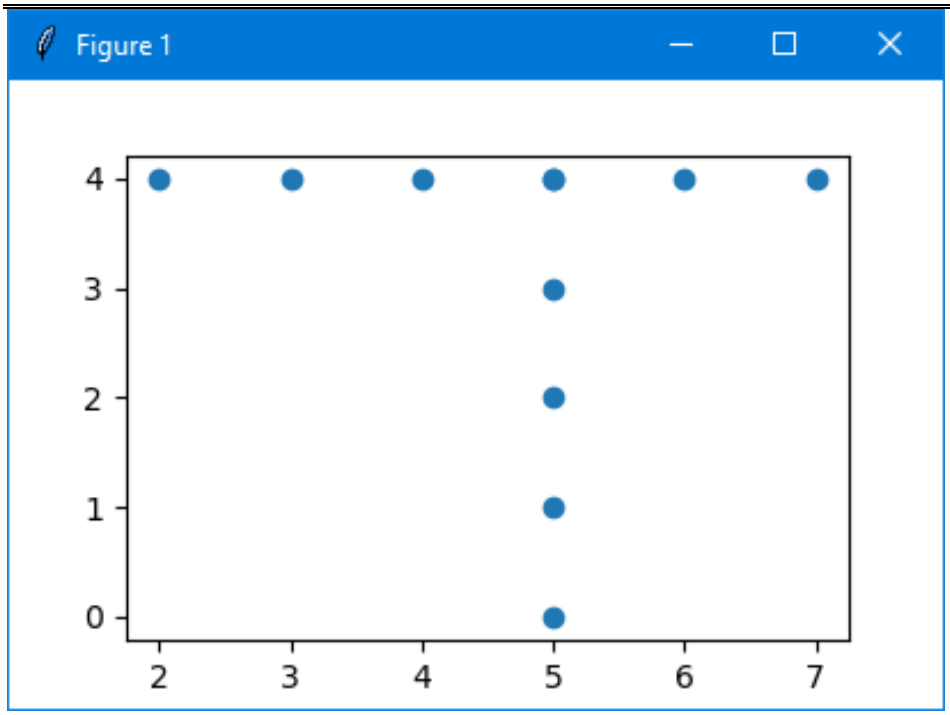
Изображаем только точки

```
plt.scatter([2,3,4,5,6,7],[4,9,16,25,36,49]);plt.show()
```



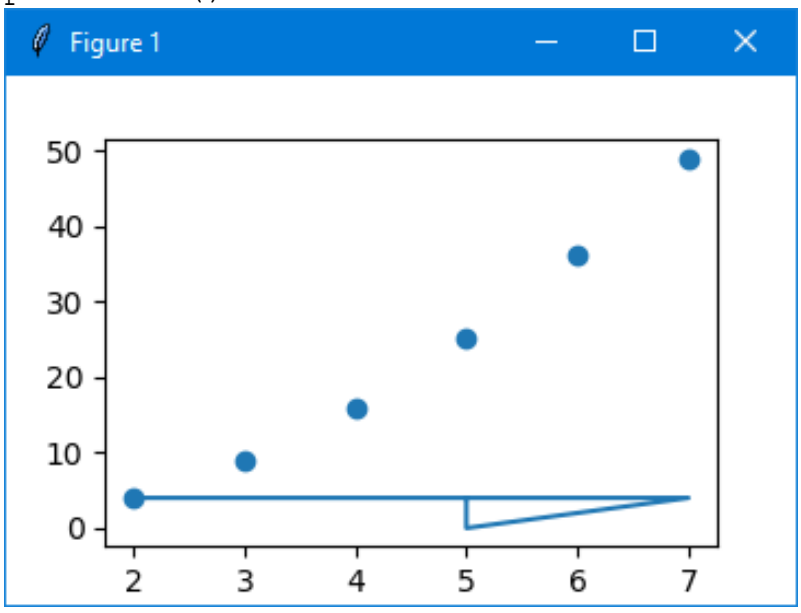
П4. «Произвольные» координаты

plt.scatter([2,3,4,5,6,7,5,5,5,5,5],[4,4,4,4,4,4,0,1,2,3,4,]);plt.show()



П5. Несколько графиков на одном листе

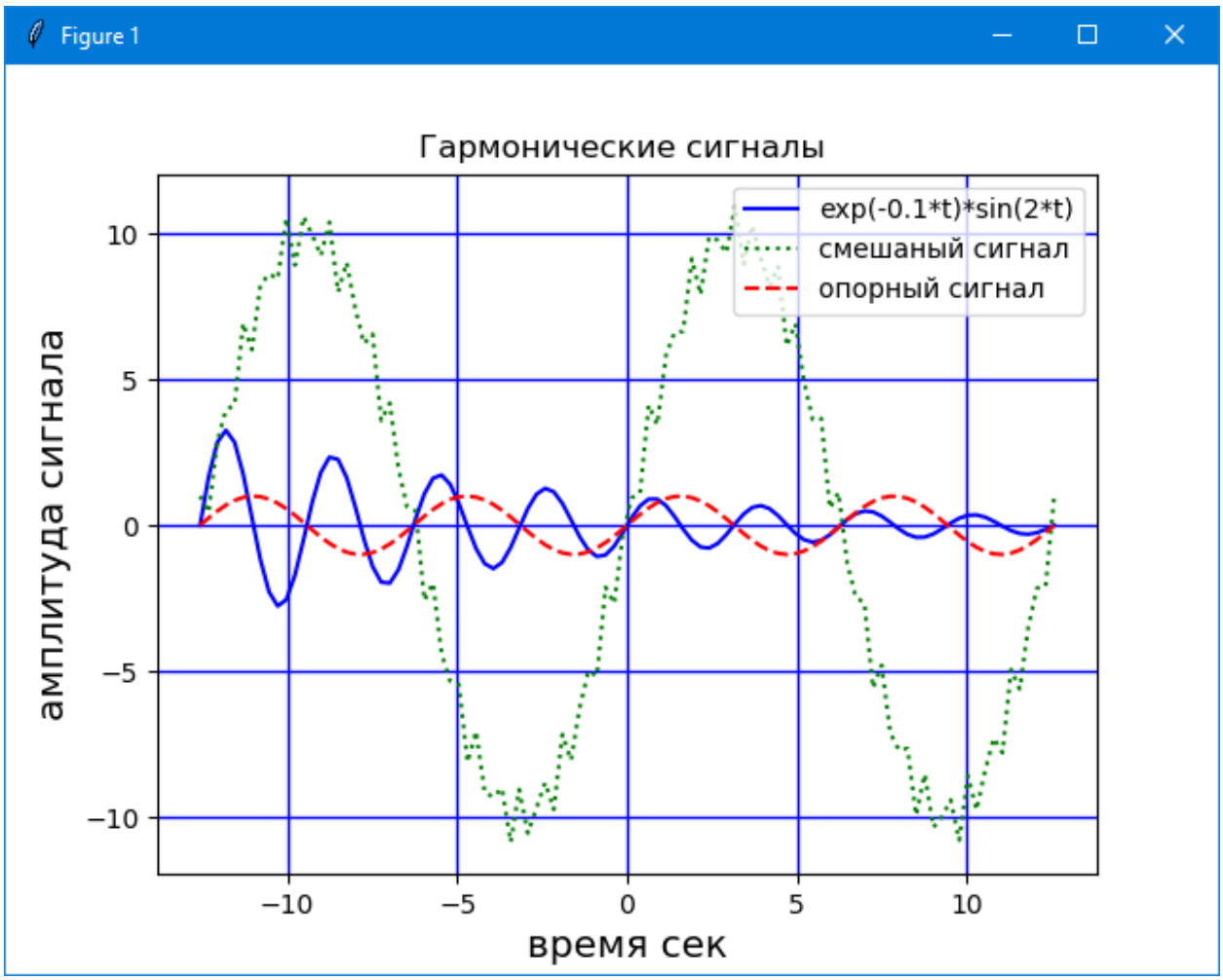
```
plt.scatter([2,3,4,5,6,7],[4,9,16,25,36,49])
plt.plot([2,3,4,5,6,7,5,5,5,5,5],
                                [4,4,4,4,4,4,0,1,2,3,4,])
plt.show()
```



П5. «Украшательство» и много графиков на диаграмме

Рисуем и сохраняем в файле три графика на одной диаграмме в одном масшабе. Смотрите комментарии скрипта.

```
import matplotlib.pyplot as plt
import numpy as np
# независимая переменная
x = np.linspace(-4*np.pi, 4*np.pi, 100)
# название осей
xlab='время сек'
ylab='амплитуда сигнала'
# первый график
y1= np.sin(2*x)*np.exp(-0.1*x)
# легенда
leg1='exp(-0.1*t)*sin(2*t)'
# второй график
y2=10*np.sin(0.5*x)+np.cos(10*x)
leg2='смешаны сигнал'
# третий графие
y3=np.sin(x)
leg3='опорный сигнал'
# Включаем сетку по оси X и оси Y.
# Задаем цвет толщину сетки
plt.grid(color = 'b',linewidth = 1)
# Задаем подписи к осям X и оси Y и размер шрифта
plt.xlabel(xlab, fontsize = 'x-large')
plt.ylabel(ylab, fontsize = 'x-large')
# Задаем заголовок диаграммы
plt.title('Гармонические сигналы ')
# строим графики
plt.plot(x,y1,'b-',label=leg1)
plt.plot(x,y2,'g:',label=leg2)
plt.plot(x,y3,'r--',label=leg3)

# задаем вывод легенды и ее расположение
plt.legend(loc='best')
# Включаем сетку
plt.grid(True)
# Сохраняем построенную диаграмму в файл
# Задаем имя файла и его тип
plt.savefig('signal3.png', format = 'png')
# визуализируем графики
plt.show()
```



Как задаются стили линий, маркеры и цвета линий смотрите в документации или воспользуйтесть «демо скриптом»:

```
import sys
import math
import os
import matplotlib.pyplot as plt
import numpy as np
##sys.exit(0)
import matplotlib as mpl
# Вывод на экран текущей версии библиотеки matplotlib
print ('Current version on matplotlib library is',
                                        mpl.__version__)

x = np.linspace(-2, 2, 10)
y1=1*x
y2=2*x
y3=4*x
y4=8*x

s1= ['-',   'solid line ',  'непрерывная линия']
s2= ['--',  'dashed line ',  'линия из штрихов']
```

```
s3= ['-.',  'dash-dot line ',  'чередование штрихов и
точек']
s4= [':',   'dotted line ',  'линия из точек']

leg1=s1[0]+' '+s1[1]+' '+s1[2]
leg2=s2[0]+' '+s2[1]+' '+s2[2]
leg3=s3[0]+' '+s3[1]+' '+s3[2]
leg4=s4[0]+' '+s4[1]+' '+s4[2]

ax=plt.subplot(111)
#можно поменть размеры окна графика в граф. окне
box=ax.get_position()
ax.set_position([box.x0, box.y0,
                    box.width*1.0 , box.height*1.0])

p1=plt.plot(x,y1,linestyle=s1[0], label=leg1)
p2=plt.plot(x,y2,linestyle=s2[0], label=leg2)
p3=plt.plot(x,y3,linestyle=s3[0], label=leg3)
p4=plt.plot(x,y4,linestyle=s4[0], label=leg4)

plt.title('Стили линий (linestyle=)')
#ax.legend(loc=(1.0,0.5), mode='expand' )
#ax.legend(mode='expand', bbox_to_anchor=(1, 0.05),loc
#                    ='upper center' ) #left best
ax.legend(loc='lower right')
# Включаем сетку
plt.grid()
plt.show()

#sys.exit(0)
# marker
plt.close()
mar=[
'.', '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'
]
leg=[]
for i in range(0,len(mar),2):
 leg.append(mar[i]+'  '+mar[i+1])
mpl.rcParams['figure.figsize'] = (8.0, 6.0)
kk=int( len(mar)/2 )
for i in range(kk):
    plt.plot(x, x+i, marker=leg[i][0], label=leg[i])
plt.legend(loc='lower right')
plt.title('Маркеры (marker=)')
plt.show()

#color
col=[
'b', 'blue    синий',
'g', 'green   зеленый',
'r', 'red     красный',
'c', 'cyan    бирюзовый',
'm', 'magenta пурпурный',
'y', 'yellow  желтый',
'k', 'black   черный',
'w', 'white   белый'
]

leg=[]
for i in range(0,len(col),2):
 leg.append(col[i]+'  '+col[i+1])
mpl.rcParams['figure.figsize'] = (5.0, 4.0)
kk=int( len(col)/2 )
for i in range(kk):
    plt.plot(x, x+i, color=leg[i][0], label=leg[i])
plt.legend(loc='lower right')
plt.title('Цвета (color=)')
plt.show()
```

Демо графики:

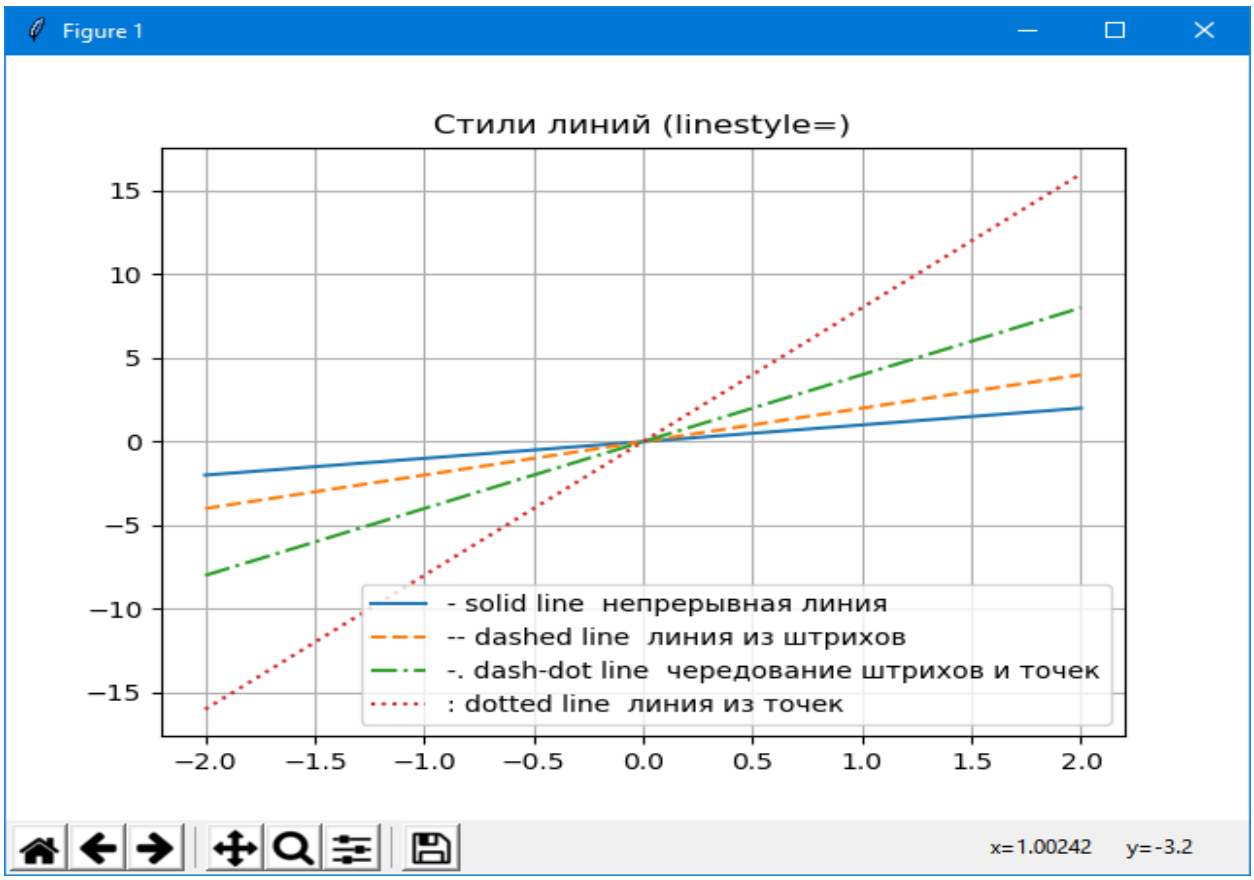


Figure 1
Стили линий (linestyle=)
- solid line непрерывная линия
-- dashed line линия из штрихов
-. dash-dot line чередование штрихов и точек
: dotted line линия из точек
x=1.00242 y=-3.2

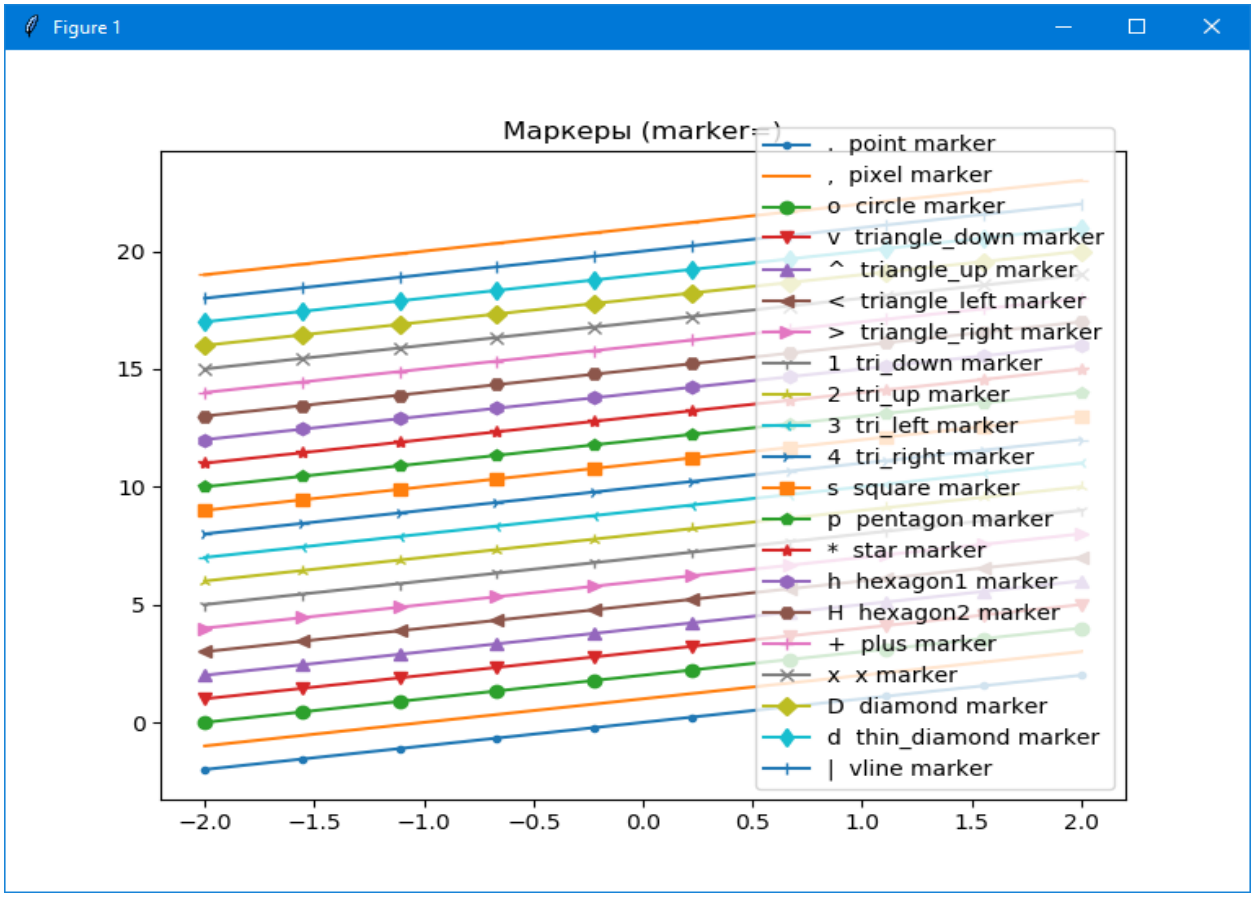
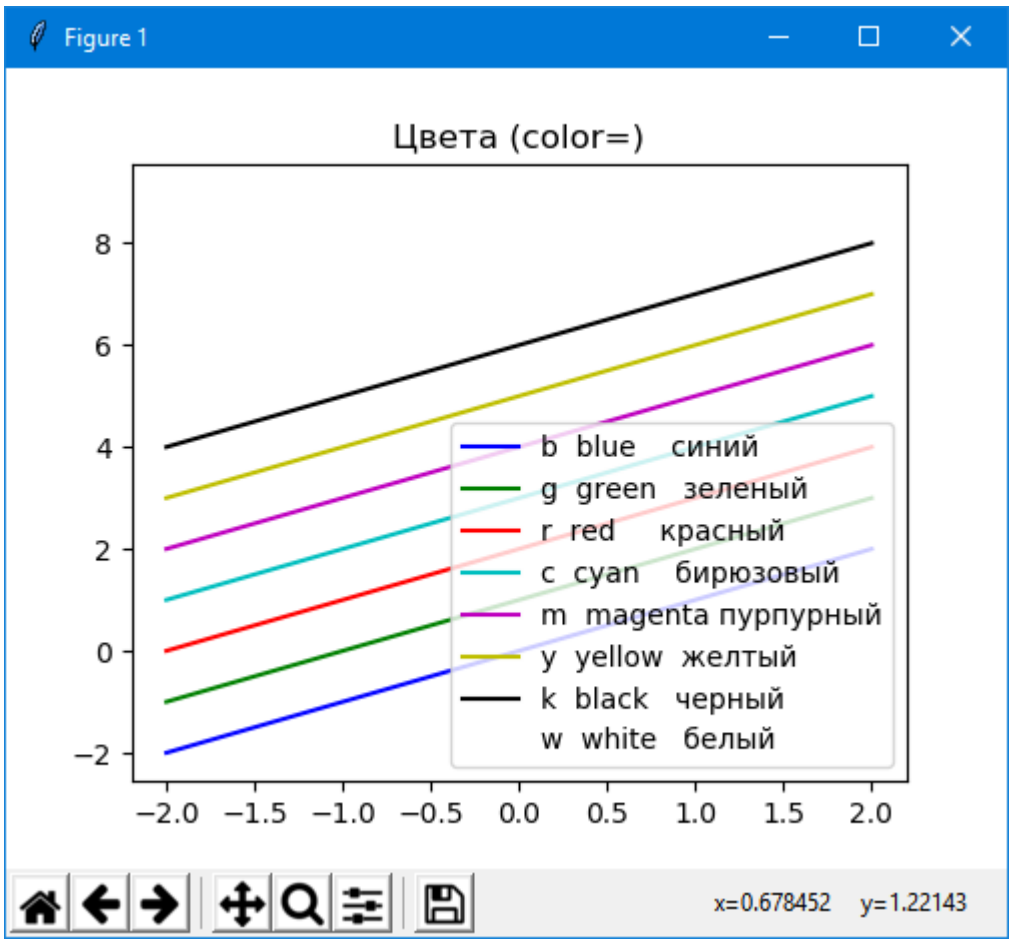


Figure 1
Маркеры (marker=)
. point marker
, pixel marker
o circle marker
v triangle_down marker
^ triangle_up marker
< triangle_left marker
> triangle_right marker
1 tri_down marker
2 tri_up marker
3 tri_left marker
4 tri_right marker
s square marker
p pentagon marker
* star marker
h hexagon1 marker
H hexagon2 marker
+ plus marker
x x marker
D diamond marker
d thin_diamond marker
| vline marker



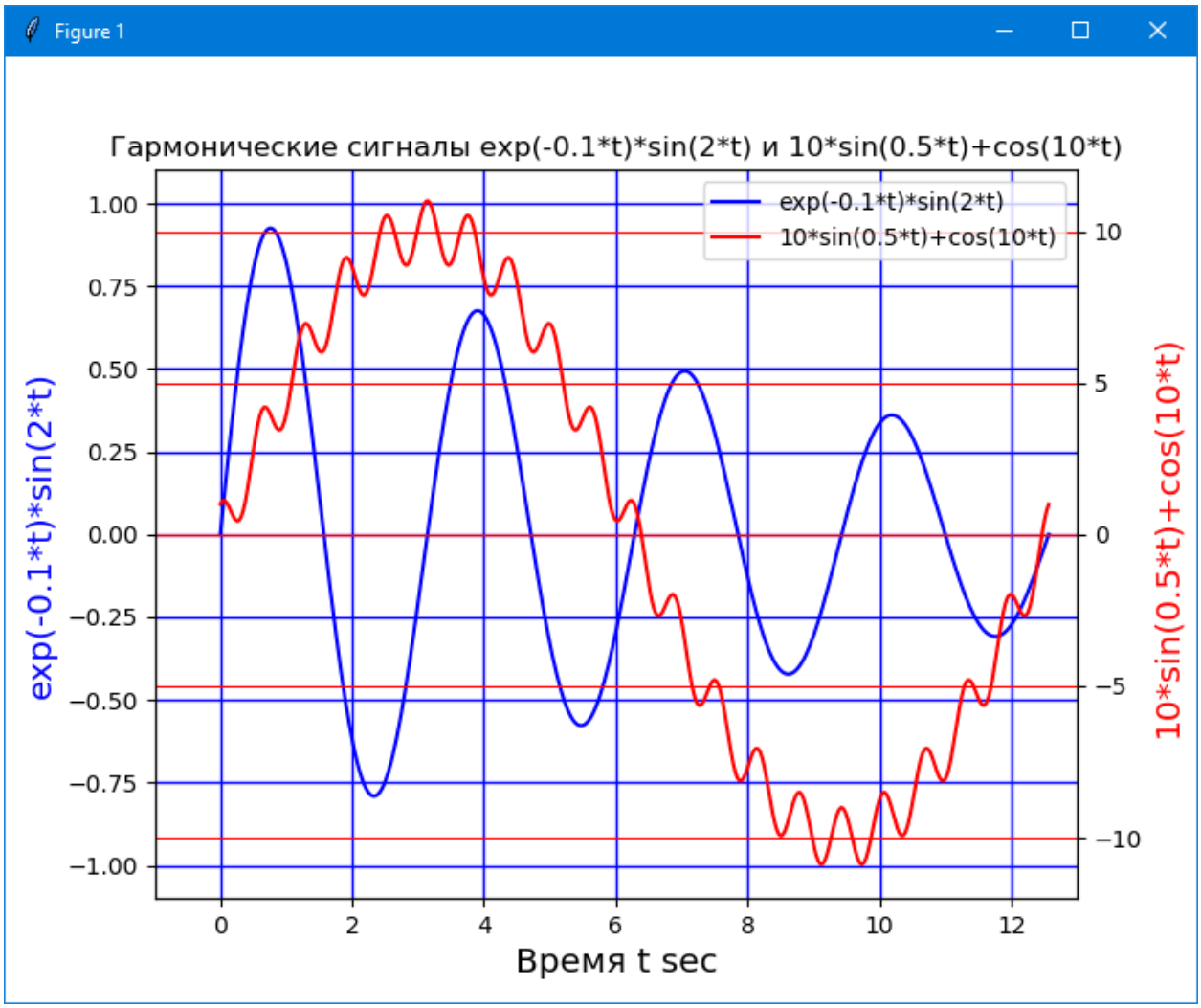
П6. Два графика на диаграмме в индивидуальных масштабах

Обратите внимание на «запятую» после имени переменной **line_01,** = ax_01.plot(X, Y_01, 'b-')

```
# -*- coding: UTF-8 -*-
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
# Значения по оси X время, t
X = np.linspace(0, 4*np.pi, 400)
# Значения по осям Y_01 & Y_02
leg_01='exp(-0.1*t)*sin(2*t)'
Y_01 = np.sin(2*X)*np.exp(-0.1*X)
# "модулированный" сигнал
leg_02="10*sin(0.5*t)+cos(10*t)"
Y_02=10*np.sin(0.5*X)+np.cos(10*X)
# Зададим размеры изображения диаграммы
mpl.rcParams['figure.figsize'] = (8.0, 6.0)
# Строим диаграмму
```

```
# Получаем ссылку на объект типа
matplotlib.axes.AxesSubplot, текущую диаграмму
ax_01 = plt.axes()
#  Задаем исходные данные для первой линии диаграммы
#  (линии степени разложения), внешний вид линии и
#   маркера.
# Функция plot() возвращает ссылку на list,
# первый элемент, которого есть объект класса
#  matplotlib.lines.Line2D
line_01, = ax_01.plot(X, Y_01, 'b-') #, label=leg_01)
# если надо в дальнейшем использовать объект класса
#  matplotlib.lines.Line2D
# то применяют ДВА варианта
# 1) line_01, = ax_01.plot(X, Y_01, 'b-')
#     прием первого элемента списка
# 2) line_01 (без запятой)= … использовать line_01[0]
# Задаем интервалы значений по осям X и
#  основной оси Y
ax_01.axis([-1, 13,   -1.1, 1.1])
# Включаем сетку по оси X и основной оси Y.
#  Задаем цвет сетки
ax_01.grid(color = 'b',linewidth = 1)
# Задаем подписи к осям X и основной оси Y
#  и размер фонта
ax_01.set_xlabel('Время t sec', fontsize = 'x-large')
ax_01.set_ylabel(leg_01, color = 'b',
                                  fontsize = 'x-large')
# Задаем заголовок диаграммы
ax_01.set_title('Гармонические сигналы '+
                                 leg_01+" и "+ leg_02)
#ax_01.legend( loc='upper right')
# Включаем дополнительную ось Y
ax_02 = ax_01.twinx()
# Задаем исходные данные для второй линии диаграммы,
#  внешний вид линии и маркера.
# Функция plot() возвращает ссылку на объект класса
#   matplotlib.lines.Line2D (см выше)
line_02, = ax_02.plot(X, Y_02, 'r-') #, label=leg_02)
# Задаем интервалы значений по осям X
#  и дополнительной оси Y
ax_02.axis([-1, 13, -12, 12])
# Задаем подпись к дополнительной оси Y
ax_02.set_ylabel(leg_02,color='r',
                               fontsize = 'x-large')
```

```
# Задаем исходные данные для легенды и
# место ее размещение
ax_02.legend((line_01, line_02), (leg_01, leg_02),
                                          loc = 'best')
#ax_02.legend(loc='lower left')
# Включаем сетку по дополнительной оси Y.
#  Задаем цвет сетки
ax_02.grid(color = 'r')
# Сохраняем построенную диаграмму в файл
# Задаем имя файла и его тип
#plt.savefig('trigo.png', format = 'png')
# визуализируем
plt.show()
```



П7 Несколько графиков в одном и в разных графических окнах

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.

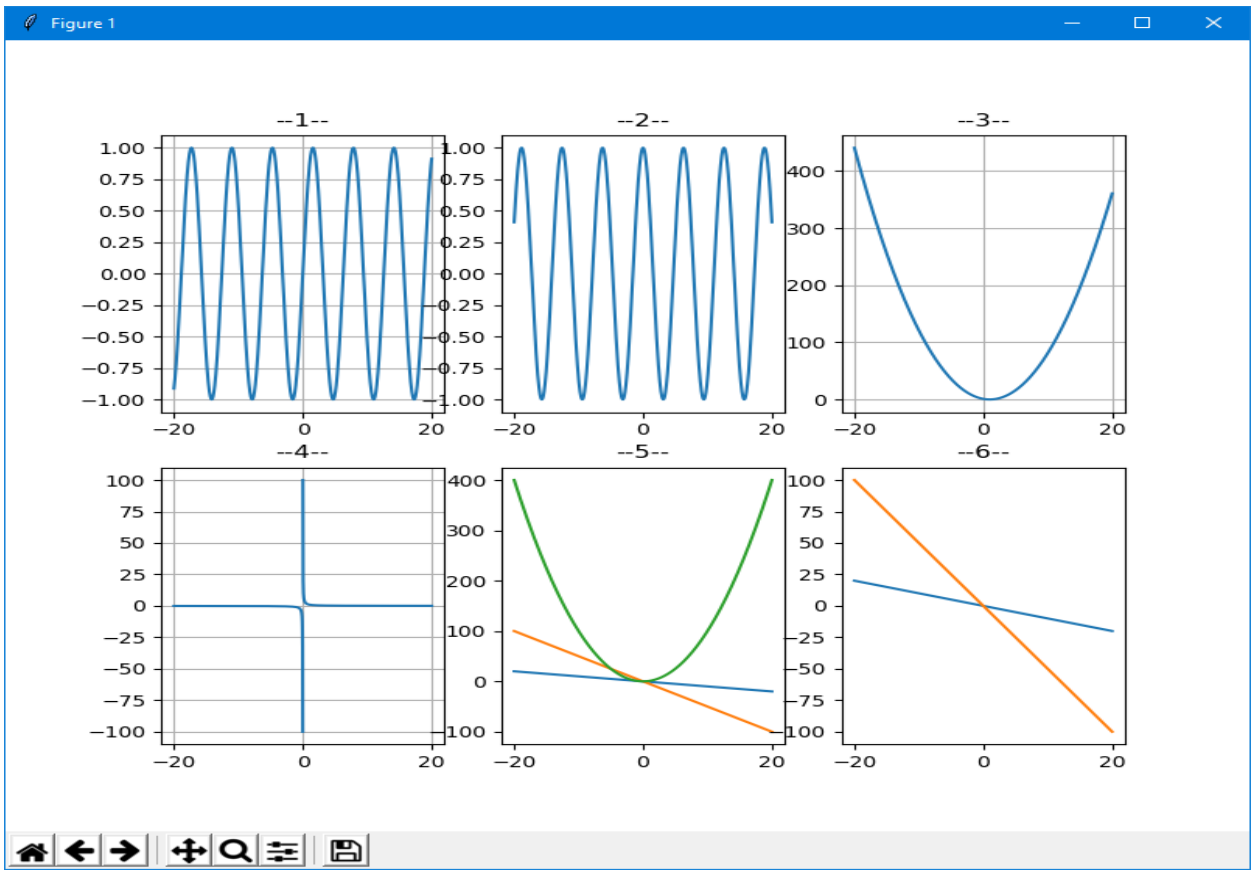
Команда **subplot** позволяет разбить графическое окно на несколько областей. Она имеет три параметра: **nr; nc; np**:

- 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 math
import numpy as np
import matplotlib.pyplot as plt

# Импортируем пакет со вспомогательными функциями
import matplotlib.mlab as ml

# Будем рисовать график этой функции
def func (x):
    """
    sinc (x)
    """
    if x == 0:
        return 1.0
    return math.sin (x) / x

# Интервал изменения переменной по оси X
xmin = -20.0
xmax = 20.0
# Шаг между точками
dx = 0.01
# Создадим список координат по оиси X на отрезке [-
xmin; xmax], включая концы
xlist = ml.frange (xmin, xmax, dx)
# Вычислим значение функции в заданных точках
ylist = [func (x) for x in xlist]

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 1
plt.subplot (2, 3, 1)
plt.plot (xlist, np.sin(xlist) ) #ylist)
```

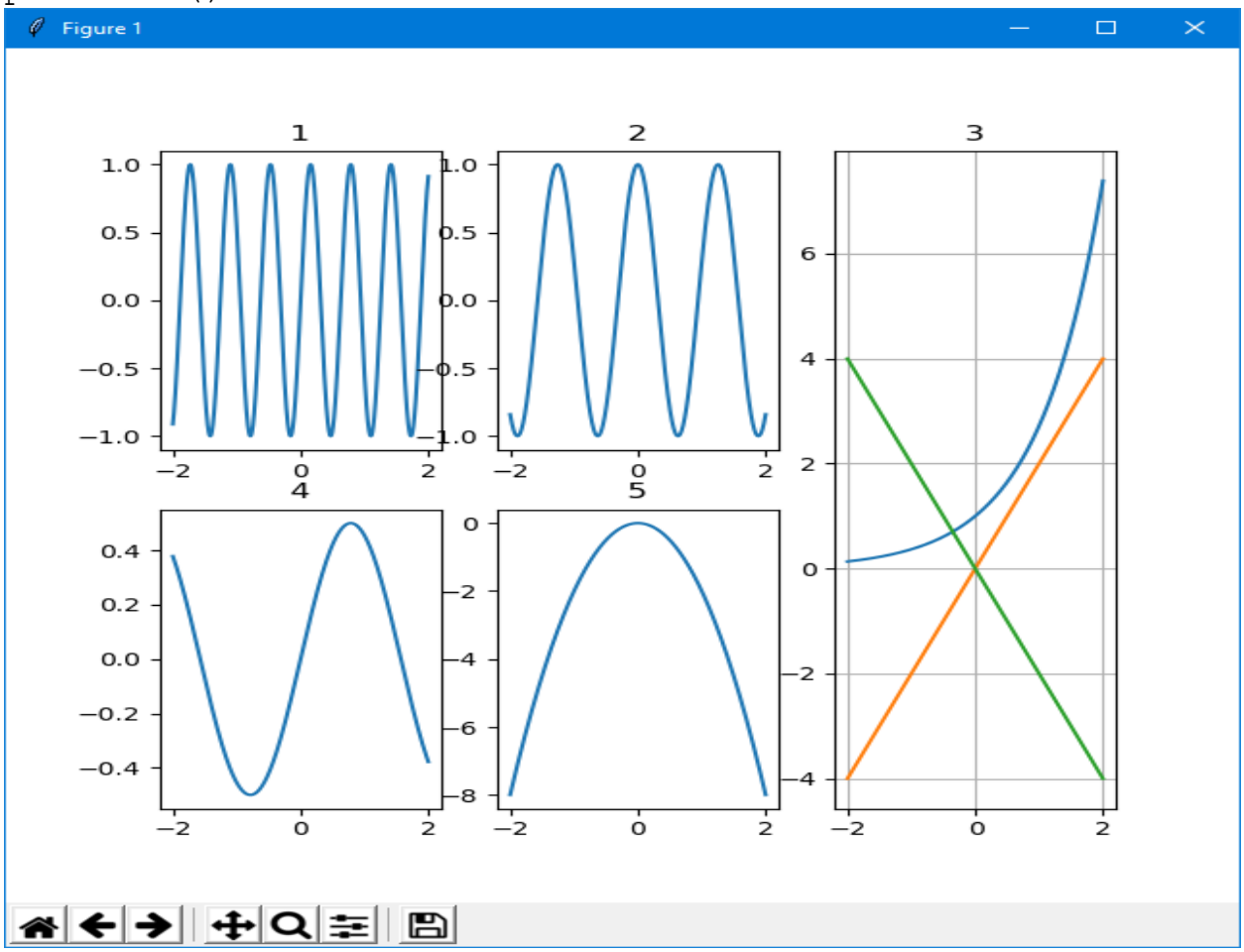
```
plt.grid(True)
plt.title ("--1--")

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 2
plt.subplot (2, 3, 2)
plt.plot (xlist, np.cos(xlist) )
plt.title ("--2--")

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 3
plt.subplot (2, 3, 3)
plt.plot (xlist, (xlist-1)*(xlist-1))
plt.grid(True)
plt.title ("--3--")

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 4
plt.subplot (2, 3, 4)
plt.plot (xlist, 1/xlist)
plt.grid(True)
plt.title ("--4--")

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 5
plt.subplot (2, 3, 5)
plt.plot (xlist, -xlist )
plt.plot (xlist, -5*xlist )
plt.plot (xlist, xlist*xlist)
plt.title ("--5--")

# !!! Две строки, три столбца.
# !!! Текущая ячейка - 6
plt.subplot (2, 3, 6)
plt.plot (xlist, -xlist )
plt.plot (xlist, -5*xlist )
plt.title ("--6--")

# Покажем окно с нарисованным графиком
plt.show()
```



Все эти ячейки чисто условные, поэтому для каждого вызова **subplot()** разбиение может быть свое, что позволяет сделать, например, следующее расположение графиков (обратите внимание на нумерацию ячеек):

```
import math
import numpy as np
import matplotlib.pyplot as plt
# Импортируем пакет со вспомогательными функциями
import matplotlib.mlab as ml
# Интервал изменения переменной по оси X
xmin = -2.0
xmax = 2.0
# Шаг между точками
dx = 0.001
# Создадим список координат по оиси X на отрезке [-
xmin; xmax], включая концы
x = ml.frange (xmin, xmax, dx)
# !!! Две строки, три столбца.
# !!! Текущая ячейка - 1
plt.subplot (2, 3, 1)
plt.plot (x, np.sin(10*x))
plt.title (" 1 ")
# !!! Две строки, три столбца.
# !!! Текущая ячейка - 2
```

```
plt.subplot (2, 3, 2)
plt.plot (x, np.cos(5*x) )
plt.title ("  2  ")
# !!! Две строки, три столбца.
# !!! Текущая ячейка - 4
plt.subplot (2, 3, 4)
plt.plot (x, np.sin(x)*np.cos(x))
plt.title ("  4  ")
# !!! Две строки, три столбца.
# !!! Текущая ячейка - 5
plt.subplot (2, 3, 5)
plt.plot (x, -2*x*x)
plt.title ("  5  ")
# !!! Одна строка, три столбца.
# !!! Текущая ячейка - 3
plt.subplot (1, 3, 3)
plt.plot (x, np.exp(x))
plt.plot (x, 2*x)
plt.plot (x, -2*x)
plt.grid(True)
plt.title ("  3  ")
# Покажем окно с нарисованным графиком
plt.show()
```



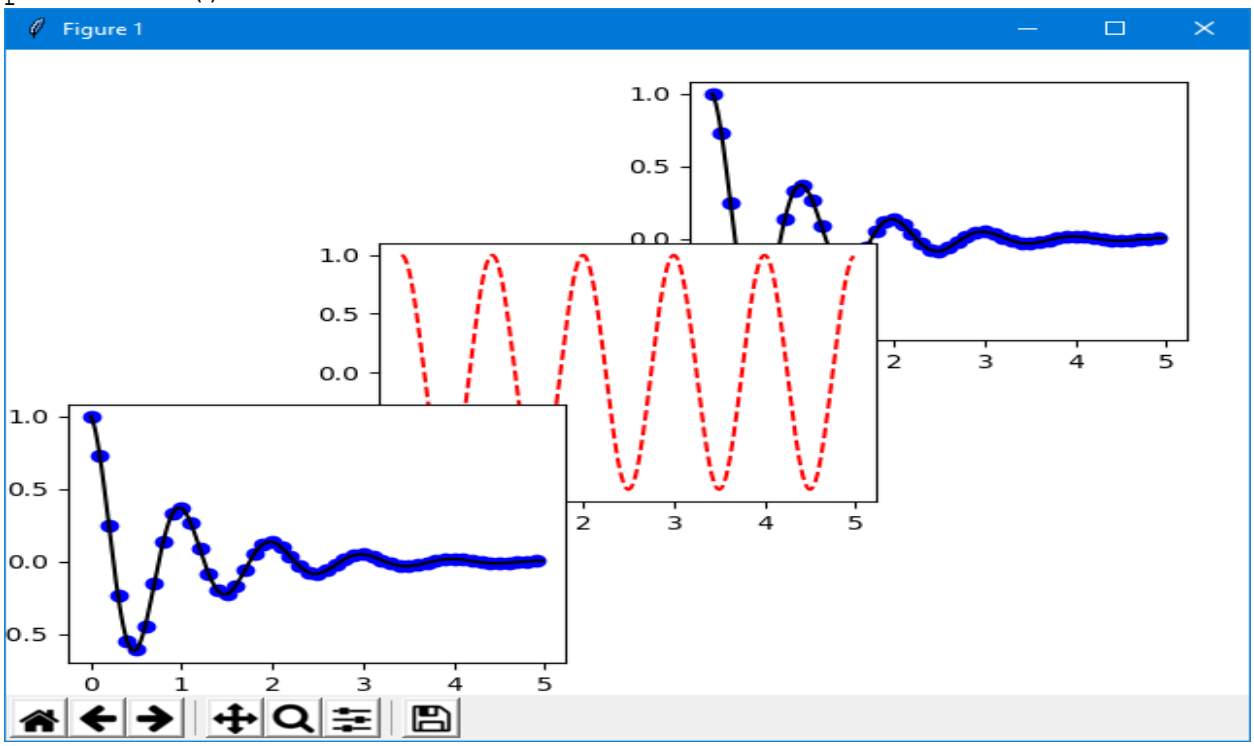
figure() и axes()

Есть возможность расположить оси (подокна) вручную, например не на прямоугольной сетке, а как-нибудь ёлочкой, используйте команду ***axes()***, которая позволяет определить положение осей как

axes([*left, bottom, width, height*]),

где все значения изменяются от 0 до 1. Выглядеть это может так:

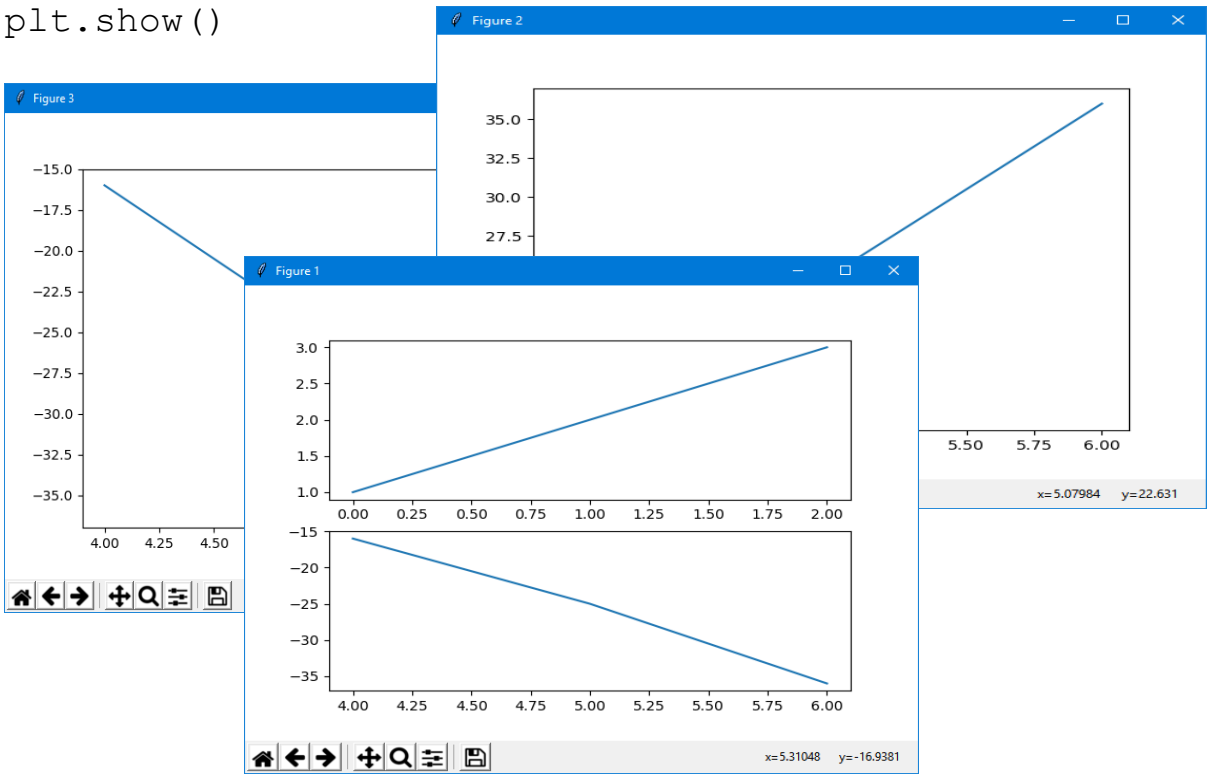
```
import math
import numpy as np
import matplotlib.pyplot as plt
# Импортируем пакет со вспомогательными функциями
import matplotlib.mlab as ml
import numpy as np
import matplotlib.pyplot as plt
def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
plt.figure(1)
plt.axes([0.55, 0.55, 0.4, 0.4])
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
plt.axes([0.3, 0.3, 0.4, 0.4])
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.axes([0.05, 0.05, 0.4, 0.4])
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')
# Покажем окно с нарисованным графиком
plt.show()
```



Имеется возможность создать несколько окон, вызывая ***figure()*** с увеличивающимся номером окна. Конечно, каждое окно может содержать несколько осей и подокон.

Очистить активное окно можно при помощи ***clf()*** , а активные оси при помощи ***cla()***.

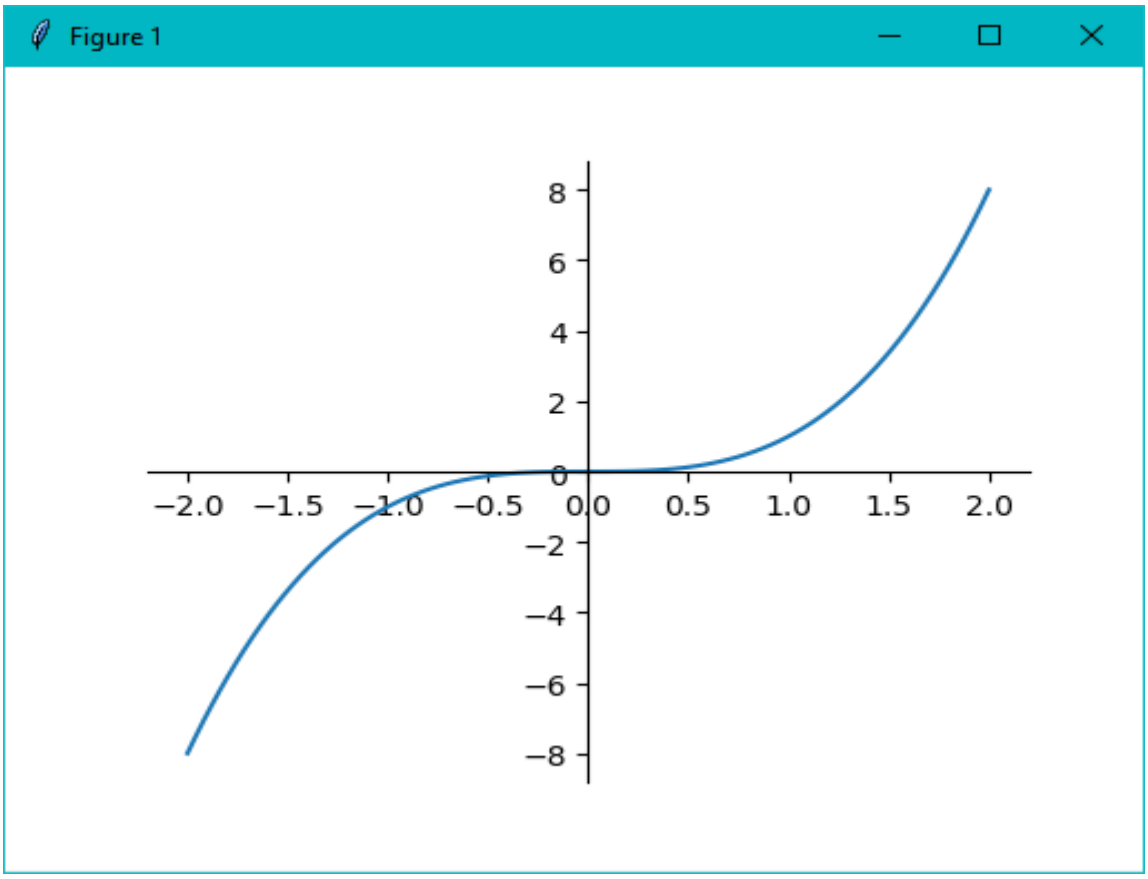
```
import math
import numpy as np
import matplotlib.pyplot as plt
# Импортируем пакет со вспомогательными функциями
import matplotlib.mlab as ml
import numpy as np
import matplotlib.pyplot as plt

plt.figure(1)                # the first figure
plt.subplot(211)             # the first subplot in the
first figure
plt.plot([1,2,3])
plt.subplot(212)             # the second subplot in
the first figure
plt.plot([4,5,6],[-16,-25,-36])
plt.figure(2)                # a second figure
plt.plot([4,5,6],[16,25,36]) # creates a subplot(111)
                                          by default
plt.figure(3)
plt.plot([4,5,6],[-16,-25,-36])
plt.show()
```



Перенос координатных осей в центр графика

Если необходимо пренести координатные оси в «центр рисунка», можно поступить так:

```
import matplotlib.pyplot as plt
import numpy as np
X = np.linspace(-2,2,100)
Y = X**3
plt.plot(X, Y)
ax = plt.gca()
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('center')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.show()
```

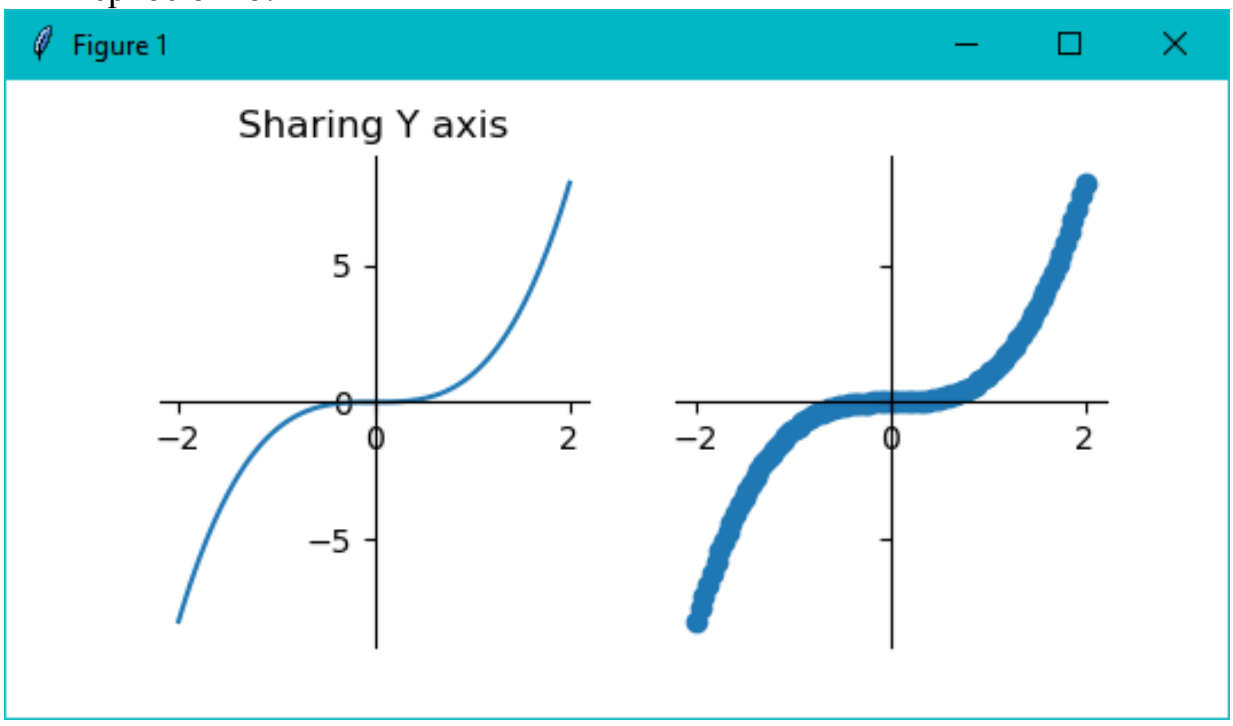


Пример переноса осей для **subplot:**

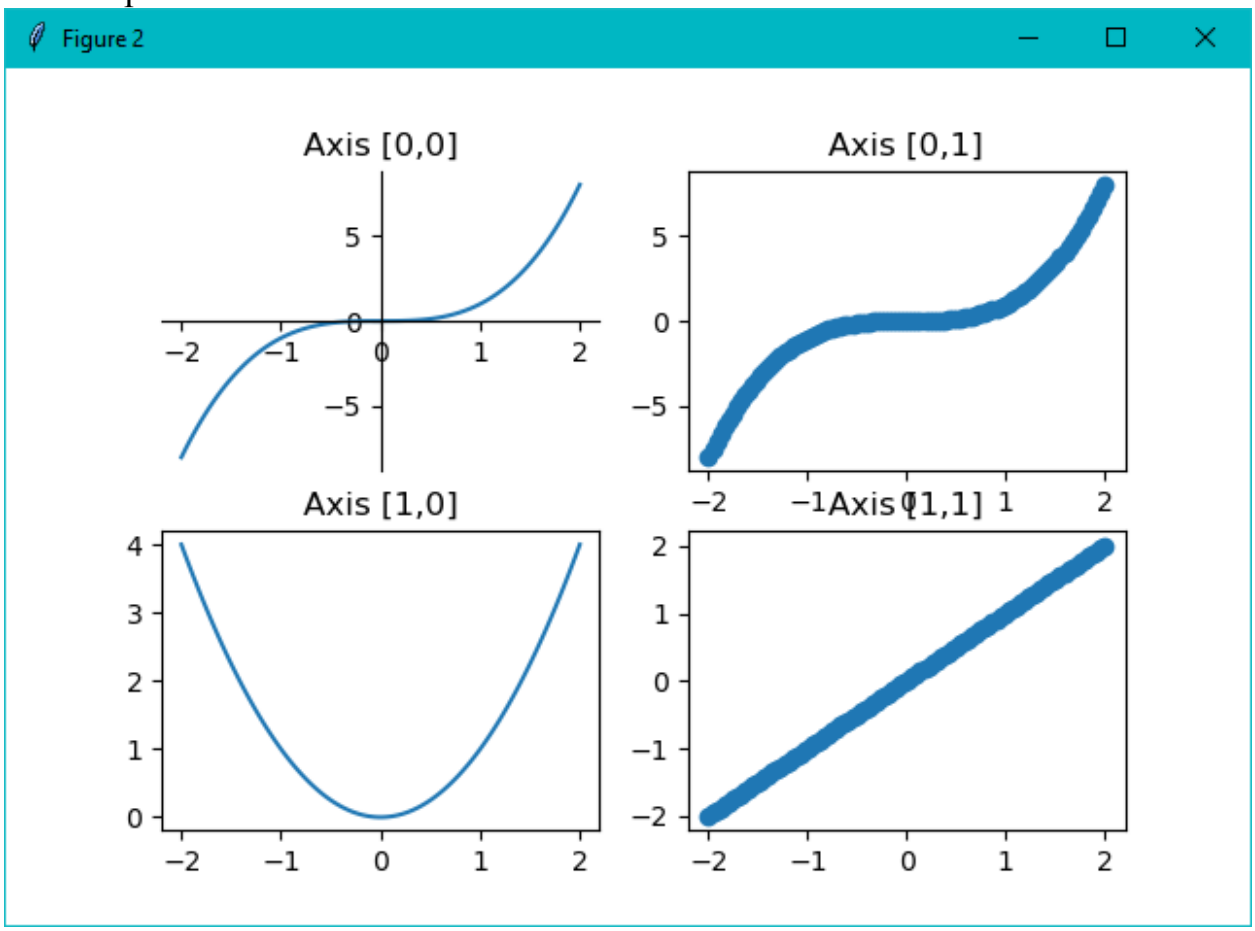
```
#Пример получения осей для subplots:
# Two subplots, unpack the axes array immediately
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
ax1.plot(X, Y)
ax1.set_title('Sharing Y axis')
ax1.spines['left'].set_position('center')
ax1.spines['bottom'].set_position('center')
```

```
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)

ax2.scatter(X, Y)
ax2.spines['left'].set_position('center')
ax2.spines['bottom'].set_position('center')
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)

f, axarr = plt.subplots(2, 2) # Four axes,
                              returned as a 2-d array

axarr[0, 0].plot(X, Y)
axarr[0, 0].spines['left'].set_position('center')
axarr[0, 0].spines['bottom'].set_position('center')
axarr[0, 0].spines['top'].set_visible(False)
axarr[0, 0].spines['right'].set_visible(False)
axarr[0, 0].set_title('Axis [0,0]')

axarr[0, 1].scatter(X, Y)
axarr[0, 1].set_title('Axis [0,1]')
axarr[1, 0].plot(X, X ** 2)
axarr[1, 0].set_title('Axis [1,0]')
axarr[1, 1].scatter(X, X ** 1)
axarr[1, 1].set_title('Axis [1,1]')

plt.show()
```

Первое окно:



Второе окно:

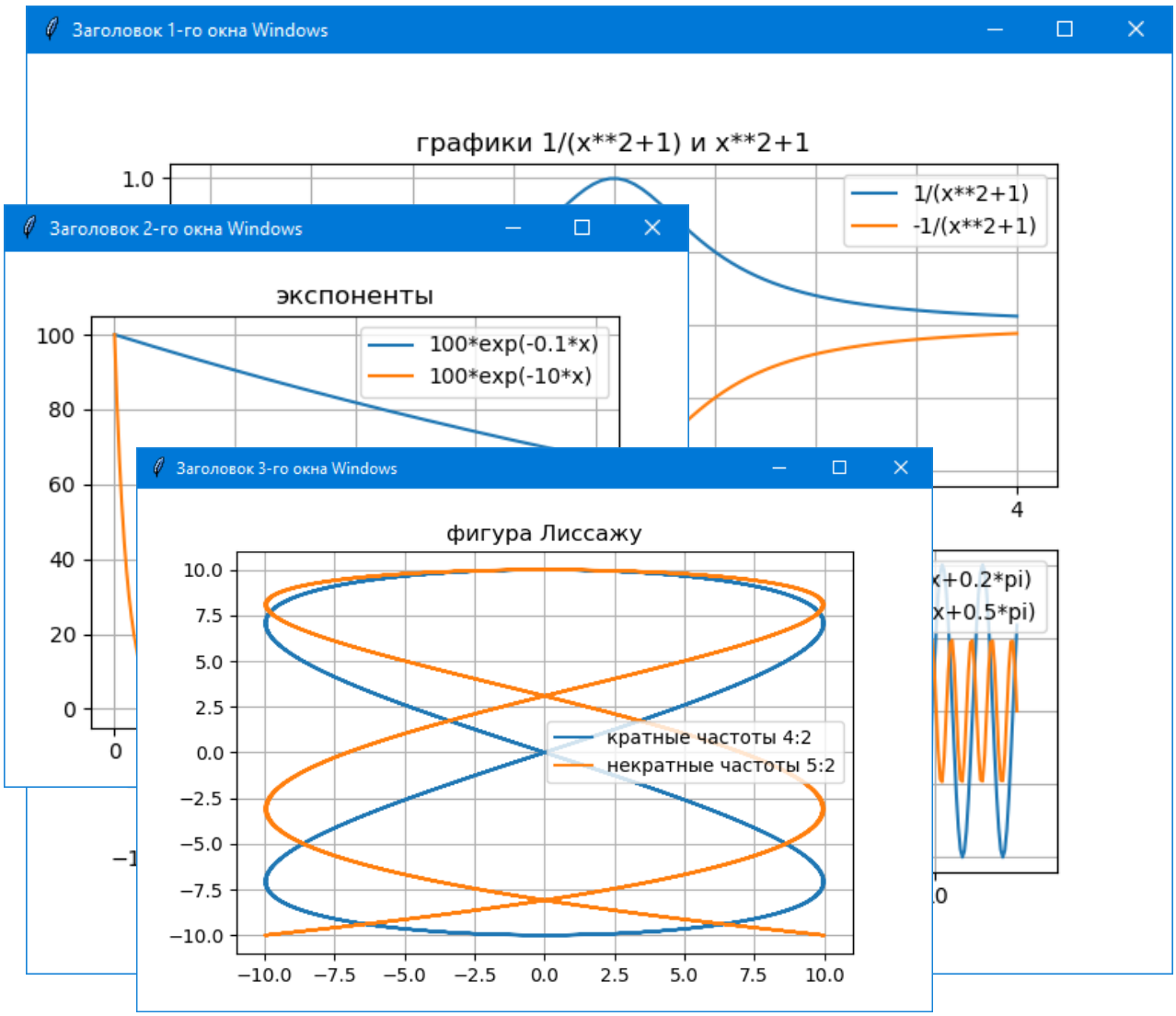


П8 Надписи на окнах диаграмм и окнах Windows, изменение размера окна

Приведем пример, аналогичный рассмотренному выше. Для всех окон и диаграмм заданы титульные (тульбарные) надписи:

```
import math
import numpy as np
import matplotlib.pyplot as plt
# Импортируем пакет со вспомогательными функциями
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt

x1=np.linspace(-4,4,100)
y1=1/(x1**2+1)
y11=-1/(x1**2+1)
# Зададим размеры изображения диаграммы
mpl.rcParams['figure.figsize'] = (8.0, 6.0)
plt.figure("Заголовок 1-го окна Windows")     # первое
                                              # окно windows
plt.subplot(211)        # первые графики в этом окне
plt.plot(x1,y1,label='1/(x**2+1)')
```

```
plt.plot(x1,y11,label='-1/(x**2+1)')
plt.title("графики 1/(x**2+1) и x**2+1" )
plt.grid(True)
plt.legend()
plt.subplot(212)   # вторые графики в этом окне
x2=np.linspace(-4*np.pi,4*np.pi,400)
plt.plot(x2,10*np.sin(5*x2+0.2*np.pi),label='10*sin(5*x
+0.2*pi)')
plt.plot(x2,5*np.cos(10*x2+0.5*np.pi),label="5*cos(10*x
+0.5*pi)")
plt.title("тригонометрические функции" )
plt.grid(True)
plt.legend(loc='best')

plt.figure("Заголовок 2-го окна Windows")  # a second
                                                  figure
# или так
#fig = plt.figure()
#fig.canvas.set_window_title("Заголовок 2-го окна
                                                 Windows")
x3=np.linspace(0,4,400)
plt.plot(x3, 100*np.exp(-0.1*x3), label='100*exp(-
0.1*x)')
plt.plot(x3, 100*np.exp(-10*x3), label='100*exp(-
10*x)')
plt.title("экспоненты" )
plt.grid(True)
plt.legend(loc='best')
fig3=plt.figure()
fig3.canvas.set_window_title("Заголовок 3-го окна
                                                 Windows")
plt.grid(True)
t=np.linspace(0,8*np.pi,500)
plt.plot(10*np.sin(4*t),10*np.cos(2*t),
                         label="кратные частоты 4:2")
plt.plot(10*np.sin(5*t),10*np.cos(2*t),
                        label="некратные частоты 5:2")
plt.title("фигура Лиссажу" )
plt.grid(True)
plt.grid(True)
plt.legend(loc='best')

plt.show()
```


Заголовок 1-го окна Windows
графики 1/(x**2+1) и x**2+1
1.0
1/(x**2+1)
-1/(x**2+1)
4
Заголовок 2-го окна Windows
экспоненты
100
80
60
40
20
0
100*exp(-0.1*x)
100*exp(-10*x)
0
Заголовок 3-го окна Windows
фигура Лиссажу
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
кратные частоты 4:2
некратные частоты 5:2
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5 10.0
x+0.2*pi)
x+0.5*pi)