Компьютерная математика II

Дисциплина для студентов 1-го курса специальности «Компьютерная математика и системный анализ» ММФ БГУ

Тема 11. Пакет matplotlib для визуального представления данных

доц. Лаврова О.А., кафедра дифференциальных уравнений и системного анализа (ауд. 329)

май, 2024

11.1 Структура пакета matplotlib

Пакет `matplotlib` предназначен для построения *двумерных* и *трехмерных* изображений в Python.

Первая версия пакета `matplotlib` создана в 2003 году, создатель Джон Хантер (*John D. Hunter*). Текущая версия пакета `matplotlib` 3.8.4.

Пакет `matplotlib` содержит около 60 модулей.

Модуль `matplotlib.pyplot` предоставляет **основной интерфейс** для построения изображений.

Модули `matplotlib.figure`, `matplotlib.axes`, `matplotlib.axis`, `matplotlib.lines`, `matplotlib.text`, `matplotlib.patches` и др. являются вспомогательными для построения изображений и загружаются автоматически при импортировании модуля `matplotlib.pyplot`.

Модуль `matplotlib.pyplot` предоставляет два подхода для построения изображений: **процедурный подход** и **объектно-ориентированный подход**.

При процедурном подходе изображения строятся с помощью функций модуля `matplotlib.pyplot`.

При объектно-ориентированный подходе изображения строятся на основе создания *графических объектов* и их настройки. Графические объекты формируют объектную архитектуру изображения.

Объектно-ориентированный подход предоставляет больше возможностей для построения изображений, чем процедурный подход.

Пакет `matplotlib` очень похож на систему компьютерной математики для численных вычислений `MATLAB` как в возможностях для построения изображений, так и по синтаксису применения.

В Python создана библиотека `seaborn` на основе `matplotlib` для графическогого представления *статистических данных*.

11.2 Создание графических окон и графических областей

Графическое окно

Графическое окно является основным графическим объектом для объектного представления `matplotlib` -изображения: объектом самого верхнего уровня, *объектом-контейнером*, который будет содержать другие графические объекты.

Графическое окно является экземпляром класса `Figure` . Класс `Figure` расположен в модуле `matplotlib.figure` . Создать графическое окно можно с помощью функции `figure` модуля `matplotlib.pyplot`

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
fig = plt.figure()
type(fig)
```

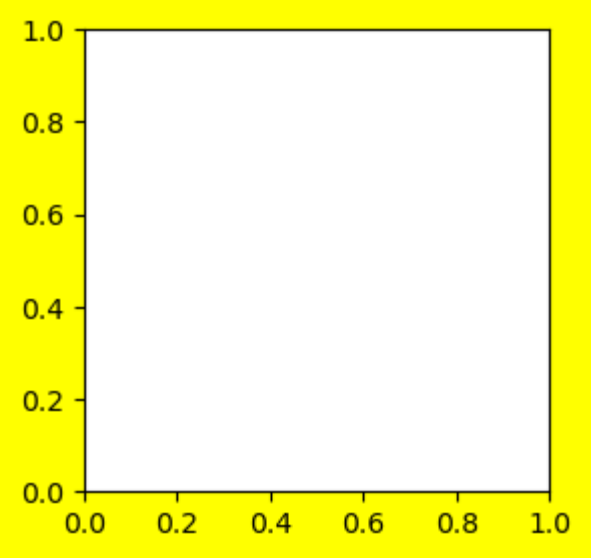
Out[2]:
```
matplotlib.figure.Figure
```

```
<Figure size 640x480 with 0 Axes>
```

При создании графического окна можно указать первым аргументом подпись графического окна. Указанием специальных ключевых аргументов можно задать свойства графического окна: размер в дюймах `figsize` , цвет фона `facecolor` , и т.д.

In [3]:
```
plt.figure("My figure", figsize=(3,3), facecolor='yellow')
plt.axes()
```

Out[3]:
```
<Axes: >
```



Графическая область

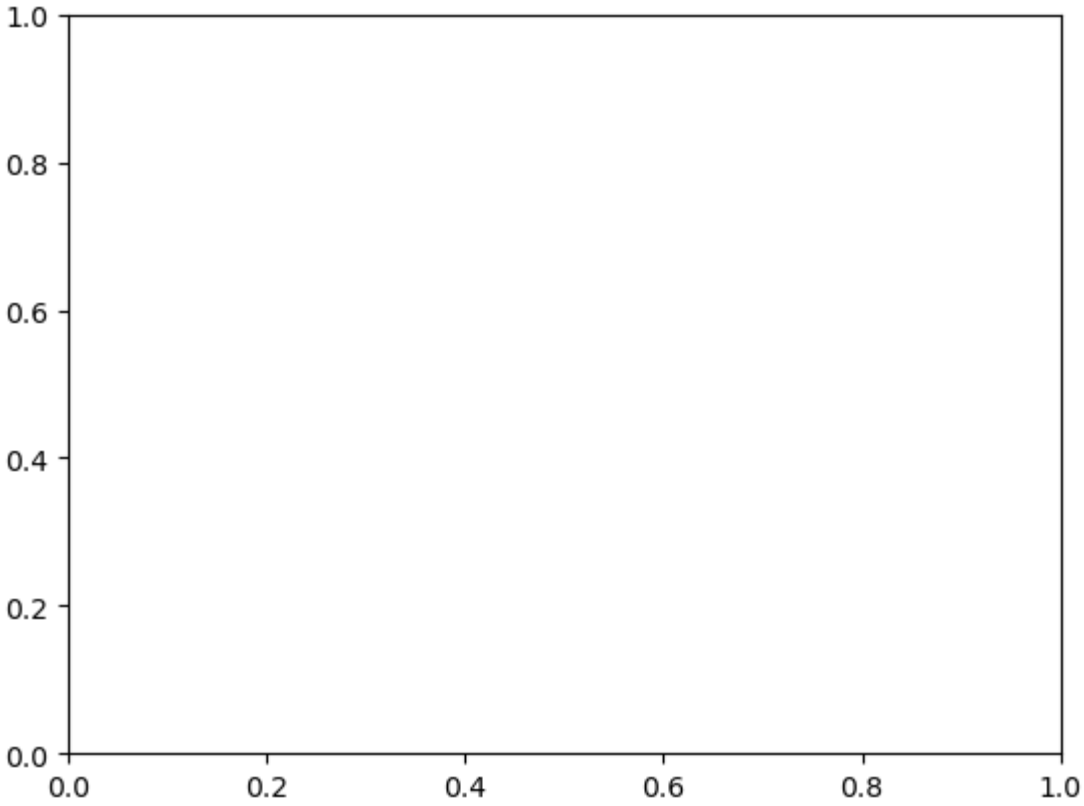
В графическом окне могут содержаться **графические области**. Графические области являются вторыми по важности графическими *объектами-контейнерами* в объектной иерархии представления `matplotlib` -изображения. Именно в графической области происходит формирование изображения.

При построении изображения важным является понятие *текущее* графическое окно и *текущая* графическая область. Действия по построению изображения осуществляются только в *текущем* графическом окне и только в *текущей* графической области.

Графическая область является экземпляром класса `Axes` или `AxesSubplot` . Класс `Axes` расположен в модуле `matplotlib.axes` . Создать графическое окно с одной графической областью можно с помощью функции `axes` модуля `matplotlib.pyplot` .

In [4]:
```
ax = plt.axes()
type(ax)
```

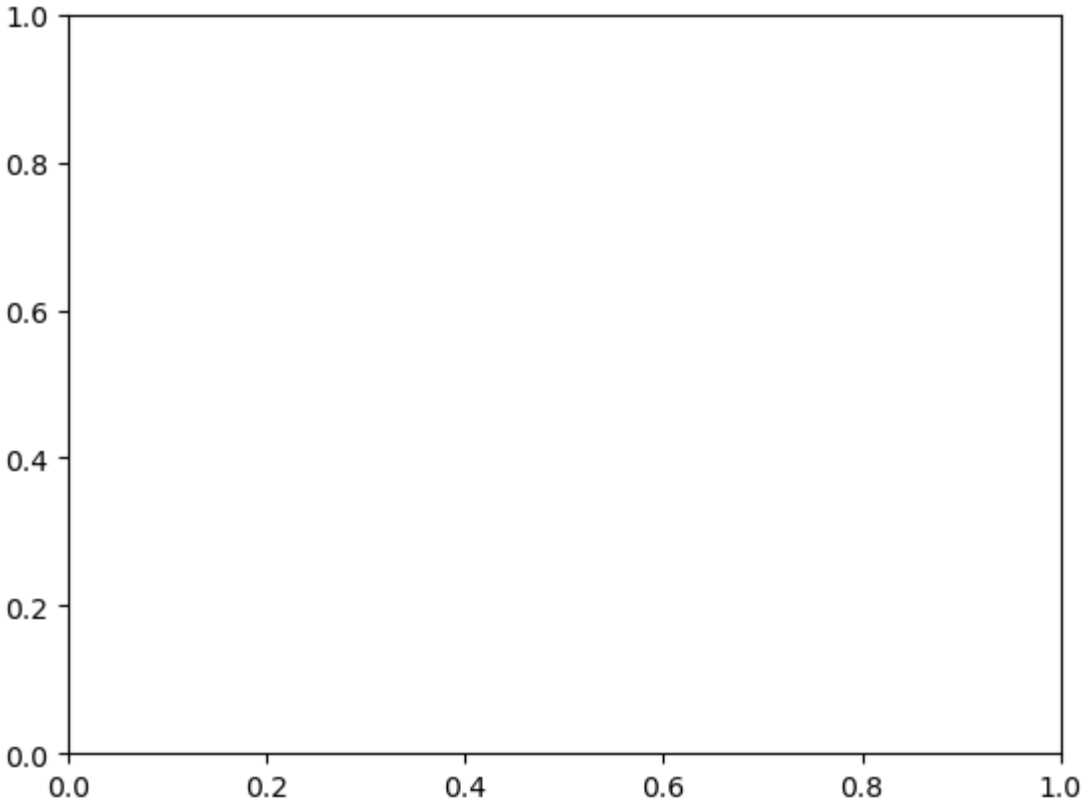
Out[4]:
```
matplotlib.axes._axes.Axes
```



Создать графическое окно с одной графической областью можно также с помощью функции `subplots` модуля `matplotlib.pyplot` .

In [5]:
```
fig, ax = plt.subplots()
type(ax)
```

Out[5]:
```
matplotlib.axes._axes.Axes
```

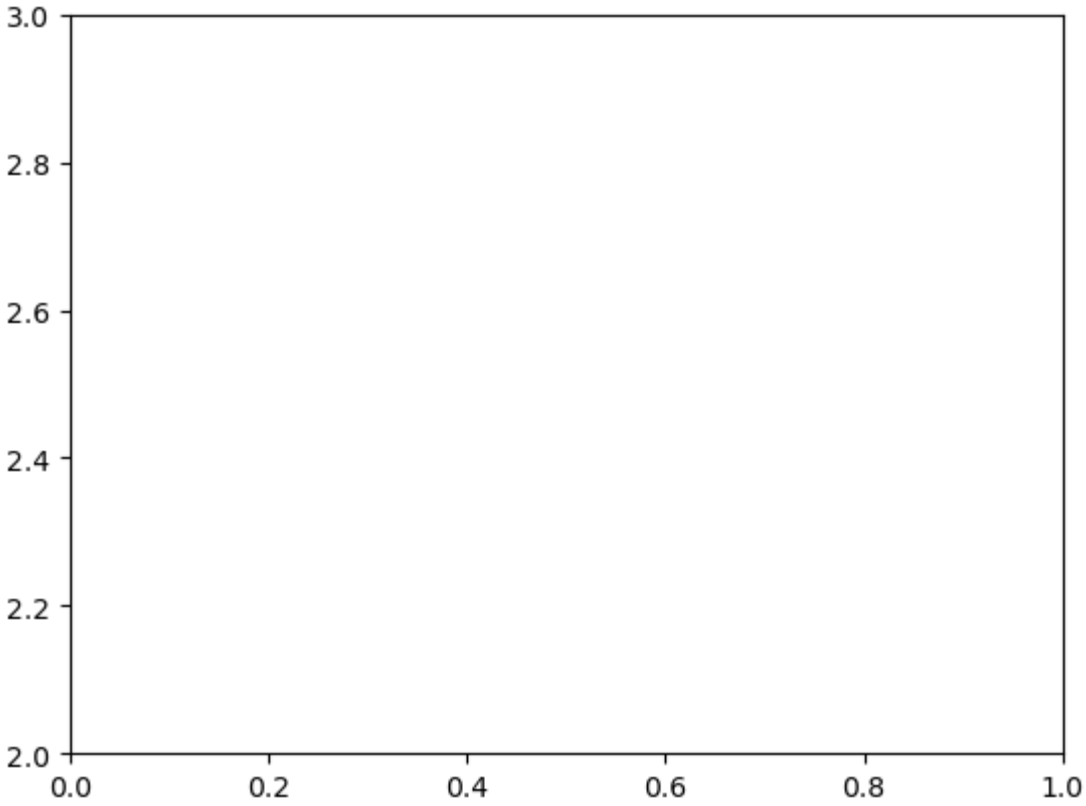


Координатные оси

Каждая графическая область содержит *координатные оси*. При создании графической области координатные оси создаются автоматически.

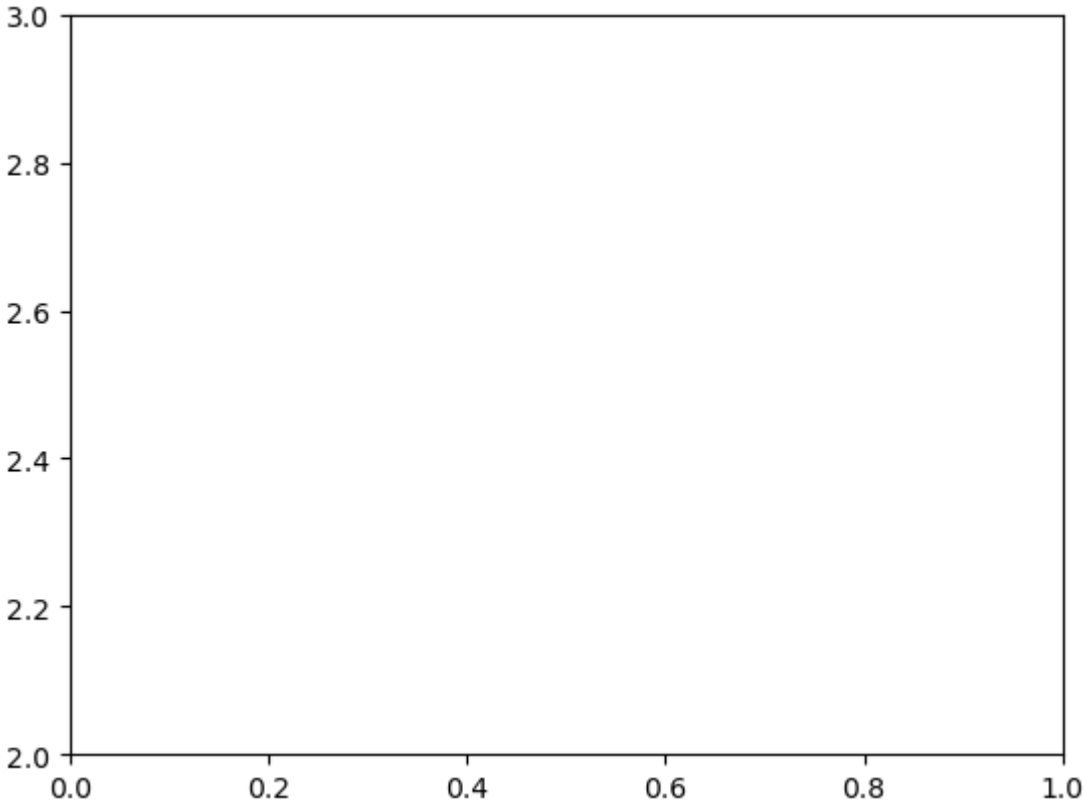
Координатные оси являются экземплярами класса `Axis`. Класс `Axis` расположен в модуле `matplotlib.axis`. Создать координатные оси можно с помощью функции `axis` модуля `matplotlib.pyplot`. В качестве аргумента функции `axis` можно указать список с пределами для горизонтальной и вертикальной координатной осей

In [6]:
```
plt.axis([0, 1, 2, 3]);
```



Функции `xlim` , `ylim` модуля `matplotlib.pyplot` также позволяют задать пределы для горизонтальной и вертикальной координатной осей, соответственно

In [7]:
```
plt.xlim(0,1); plt.ylim(2,3);
```



Иерархическая структура изображения

Графическая область *может* содержать графические объекты и/или текстовые объекты для представления графиков и изображений.

Иерархическая структура `matplotlib` -изображения:

```
экземпляр класса Figure содержит экземплярЫ класса Axes
экземпляр класса Axes содержит экземпляр класса Axis и экземплярЫ классов
Line2D, Text, Patch и др.
```

Явное создание сначала объекта графического окна, затем создание объекта графической области с последующим вызовом методов этих объектов относится к *объектно-ориентированному* подходу построения изображений.

Функция subplot

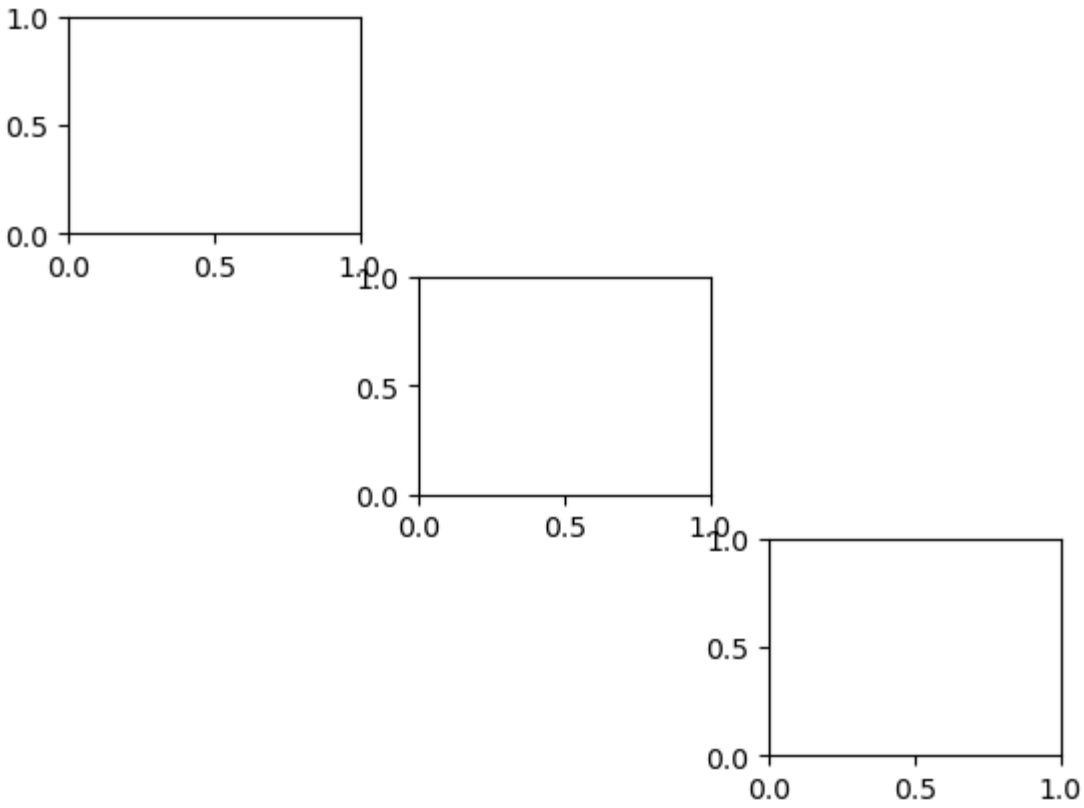
Для создания нескольких графических областей в одном графическом окне можно использовать функцию `subplot` модуля `matplotlib.pyplot` . Функция

```
subplot(nrows, ncols, index)
```

вызывается с тремя аргументами. Аргументы `nrows` и `ncols` задают количество строк и столбцов таблицы в графическом окне, в ячейках которой могут располагаться графические области. Аргумент `index` определяет номер ячейки/ячеек, в которой будет создана графическая область. Ячейка в левом верхнем углу имеет порядковый номер 1. Далее номера ячеек последовательно увеличиваются при движении по таблице слева направо и сверху вниз.

In [8]:
```
plt.subplot(3,3,1)
plt.subplot(3,3,5)
plt.subplot(3,3,9)
```

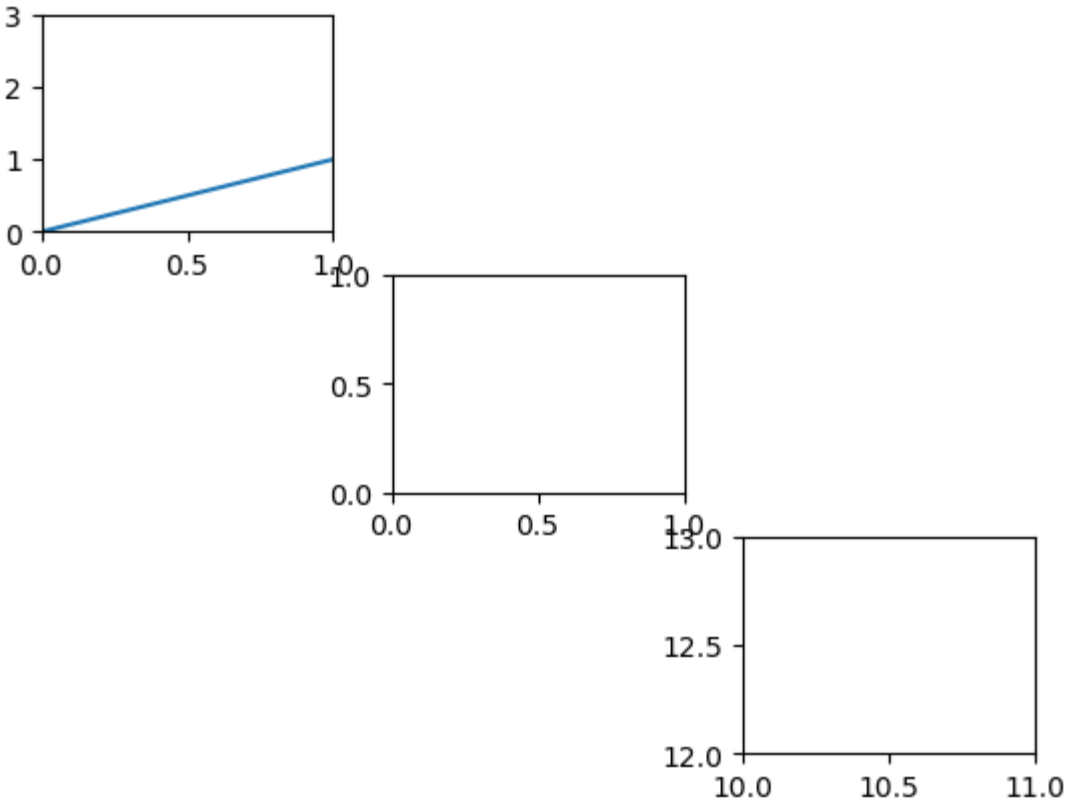
Out[8]:
```
<Axes: >
```



Функция `subplot` также определяет, какая из графических областей будет *текущей* для выполнения действий

In [9]:
```
p = plt.subplot(3,3,1)
plt.axis([0, 1, 0, 3]);
plt.subplot(3,3,5)
plt.subplot(3,3,9)
plt.axis([10, 11, 12, 13]);
p.plot(range(5))
```

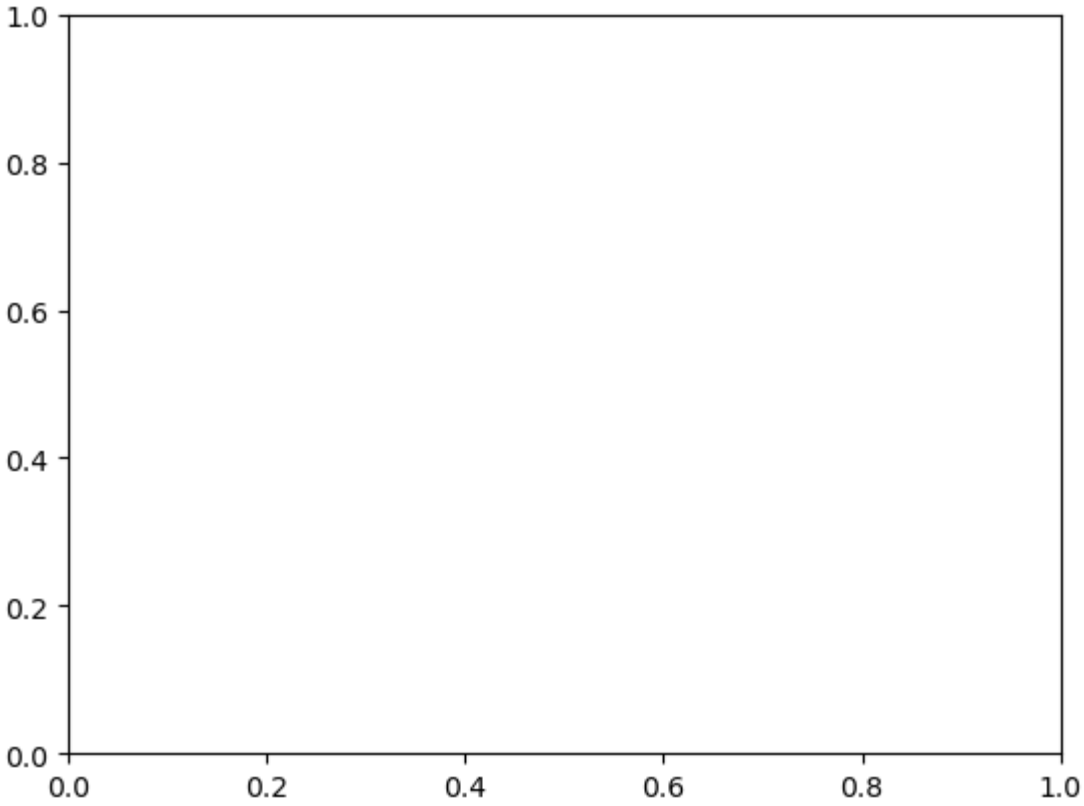
Out[9]:
```
[<matplotlib.lines.Line2D at 0x1b81880e250>]
```



Функция savefig

Функция `savefig` модуля `matplotlib.pyplot` позволяет сохранить *текущее* графическое окно в графические файлы различных форматов (`png` , `pdf` , `gif` , `jpeg` и др.) для дальнейшего использования изображения в других приложениях. Для файла обязательно указание расширения

In [10]:
```
plt.axes()
plt.savefig('myfigure.pdf')
```



11.3 Процедурный подход для построения двумерных графиков

Все функции данного подраздела относятся к *процедурному подходу* построения изображений средствами модуля `matplotlib.pyplot` .

Функция plot

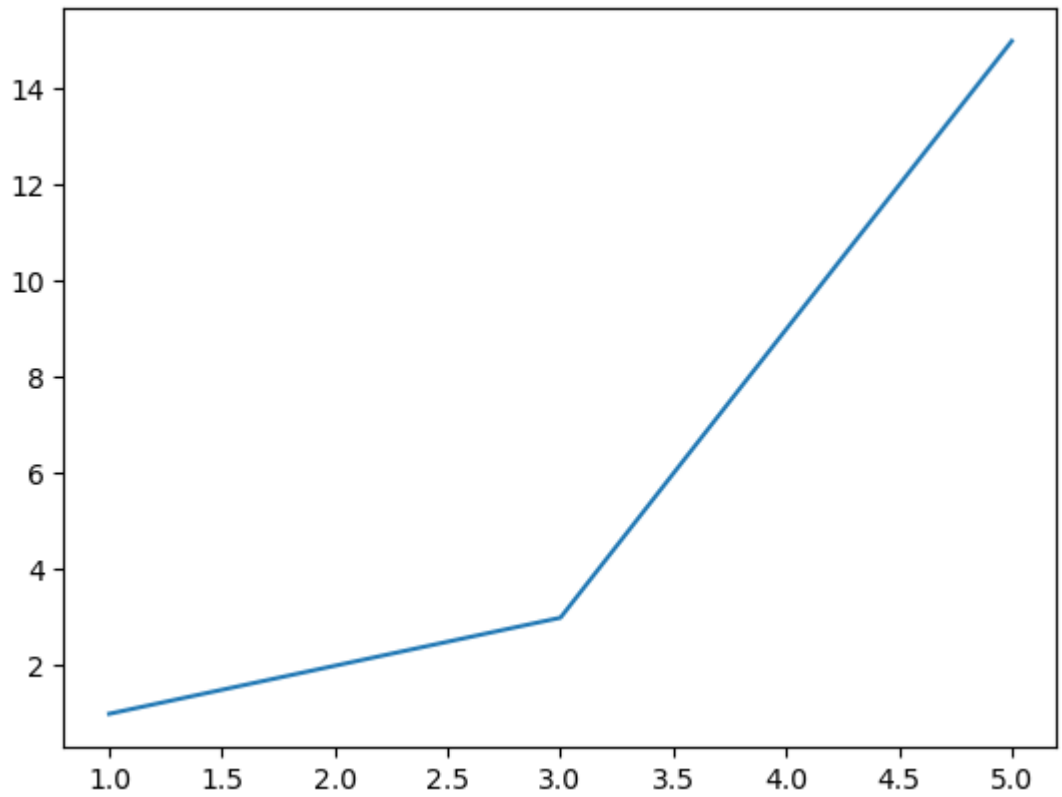
Функция `plot` модуля `matplotlib.pyplot` используется для построения двумерных графиков в *текущей* графической области. Графики строятся по значениям абсцисс и ординат точек, представляющих график. Координаты точек задаются списками или массивами. *Точки графика последовательно соединяются отрезками*.

При вызове функции `plot` *автоматически* создается графическое окно и графическая область, которые будут содержать построенный график. Вызов функции `plot` относится к *процедурному подходу* построения изображений средствами модуля `matplotlib.pyplot` .

Первым аргументом функции `plot` является `numpy`-массив или список абсцисс точек, вторым аргументом является `numpy`-массив или список ординат точек. Размеры первого и второго аргумента должны совпадать.

Внутренне списки с данными преобразуются в `numpy`-массив при вызове функций модуля `matplotlib.pyplot`.

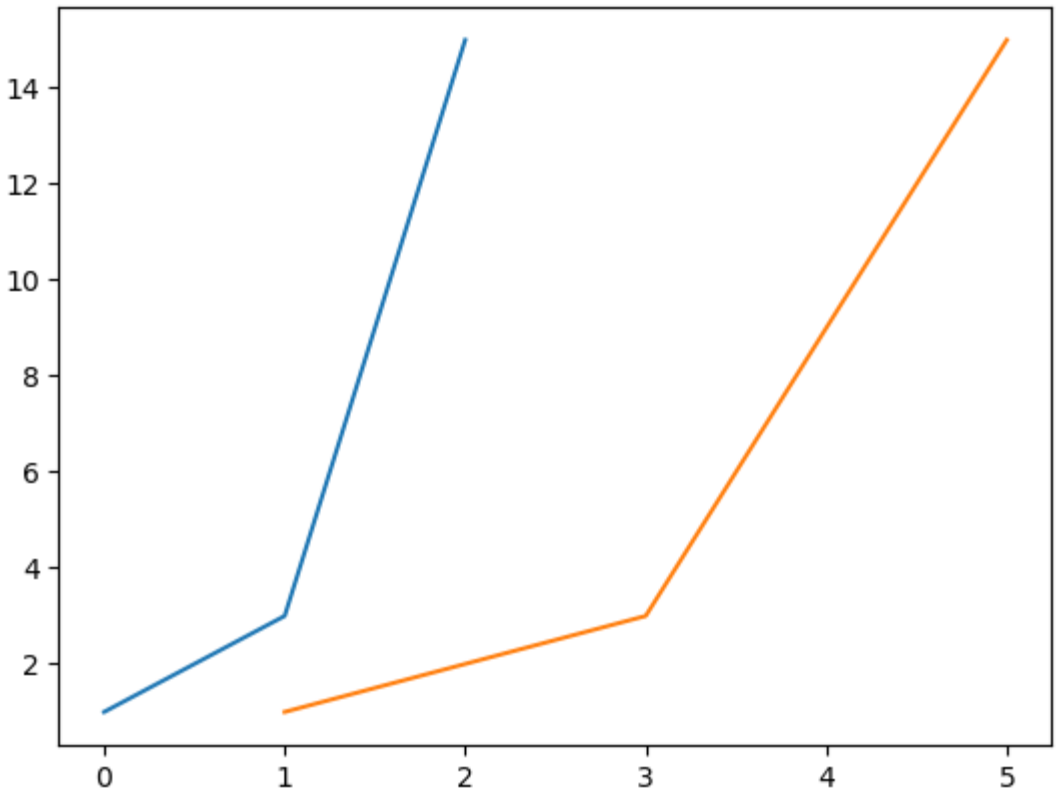
In [11]:
```
x = np.array([1,3,5]); y = np.array([1,3,15]);
plt.plot(x, y);
```



Вызов функции `plot` с одним аргументом в виде `numpy`-массива или списка изображает график по точкам, абсциссами которых являются индексы элементов `numpy`-массива или списка, а ординатами -- значения элементов `numpy`-массива или списка.

In [12]:
```
plt.plot(y)
plt.plot(x, y)
```

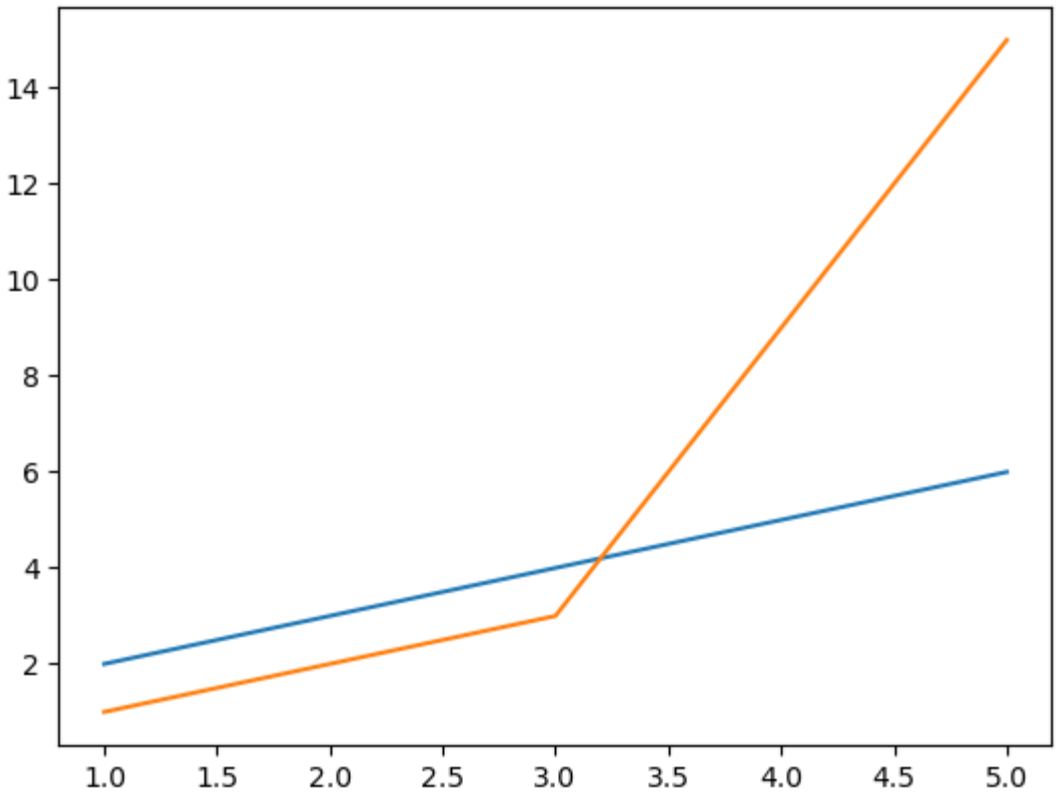
Out[12]:
```
[<matplotlib.lines.Line2D at 0x1b816e13d50>]
```



Несколько графиков в одной графической области можно создать с помощью одиночного вызова функции `plot` . При этом координаты точек графика задаются последовательно друг за другом

In [13]:
```
plt.plot(x, x+1, x, y)
```

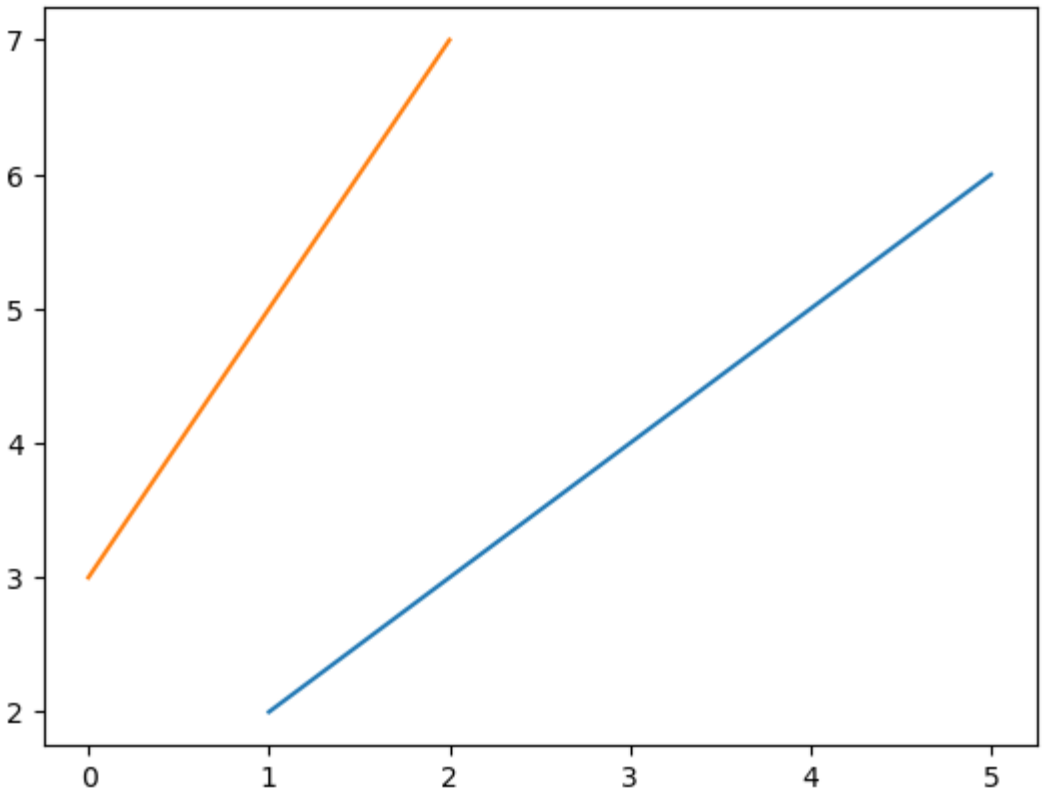
Out[13]:
```
[<matplotlib.lines.Line2D at 0x1b818df1510>,
 <matplotlib.lines.Line2D at 0x1b8175ed090>]
```



Функция `plot` возвращает список экземпляров класса `Line2D` для каждого из графиков в *текущей* графической области. Класс `Line2D` расположен в модуле `matplotlib.lines`

In [14]:
```
line1, line2 = plt.plot(x, x+1, x+2)
type(line1), type(line2)
```

Out[14]:
```
(matplotlib.lines.Line2D, matplotlib.lines.Line2D)
```



Оформление графика

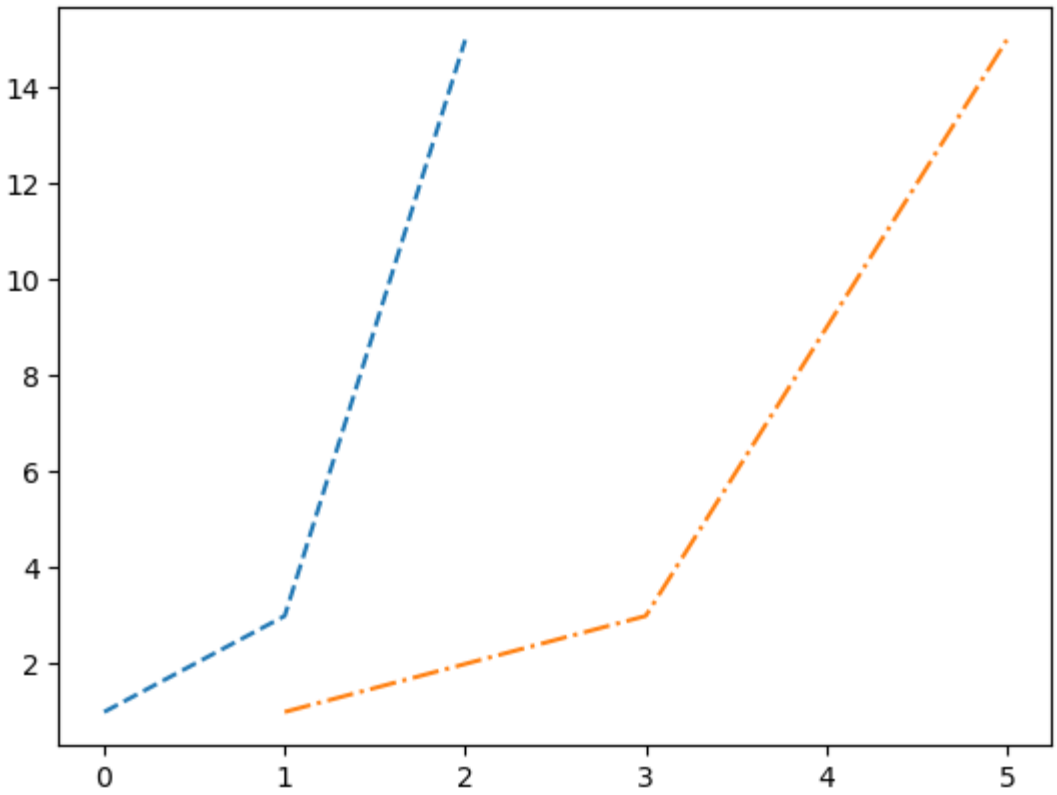
Для графика при его создании с помощью функции `plot` можно задать оформление указанием специальных ключевых аргументов: `linestyle`, `linewidth`, `color`, `marker` и т.д.

Ключевой аргумент `linestyle` задает стиль линии графика:

- `'-'` или `'solid'` -- непрерывная линия (*значение по умолчанию*)
- `'--'` или `'dashed'` -- штриховая линия
- `'-.'` или `'dashdot'` -- штрихпунктирная линия
- `':'` или `'dotted'` -- пунктирная линия
- `'None'` или `' '` или `''` -- не отображать линию

In [15]:
```
plt.plot(y, linestyle='dashed')
plt.plot(x, y, linestyle='dashdot')
```

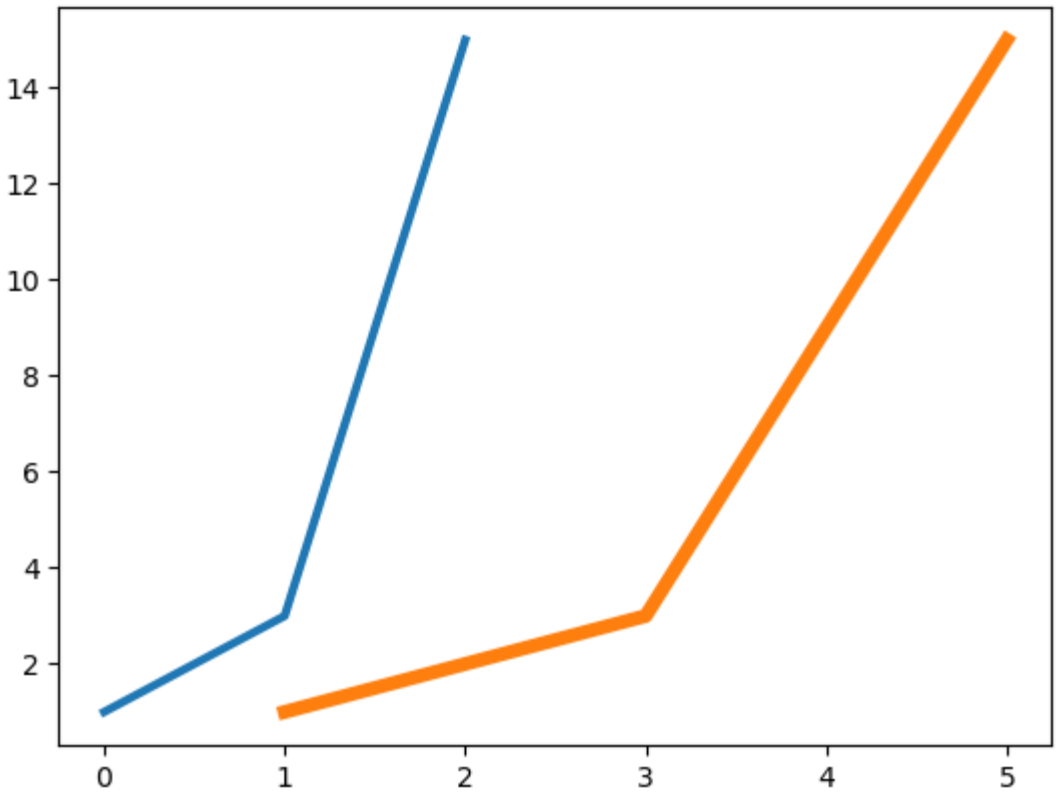
Out[15]:
```
[<matplotlib.lines.Line2D at 0x1b818eed550>]
```



Ключевой аргумент `linewidth` задает толщину линии графика в пунктах. Значением по умолчанию является 1.5 пункта.

In [16]:
```
plt.plot(y, linewidth=3)
plt.plot(x, y, linewidth=5)
```

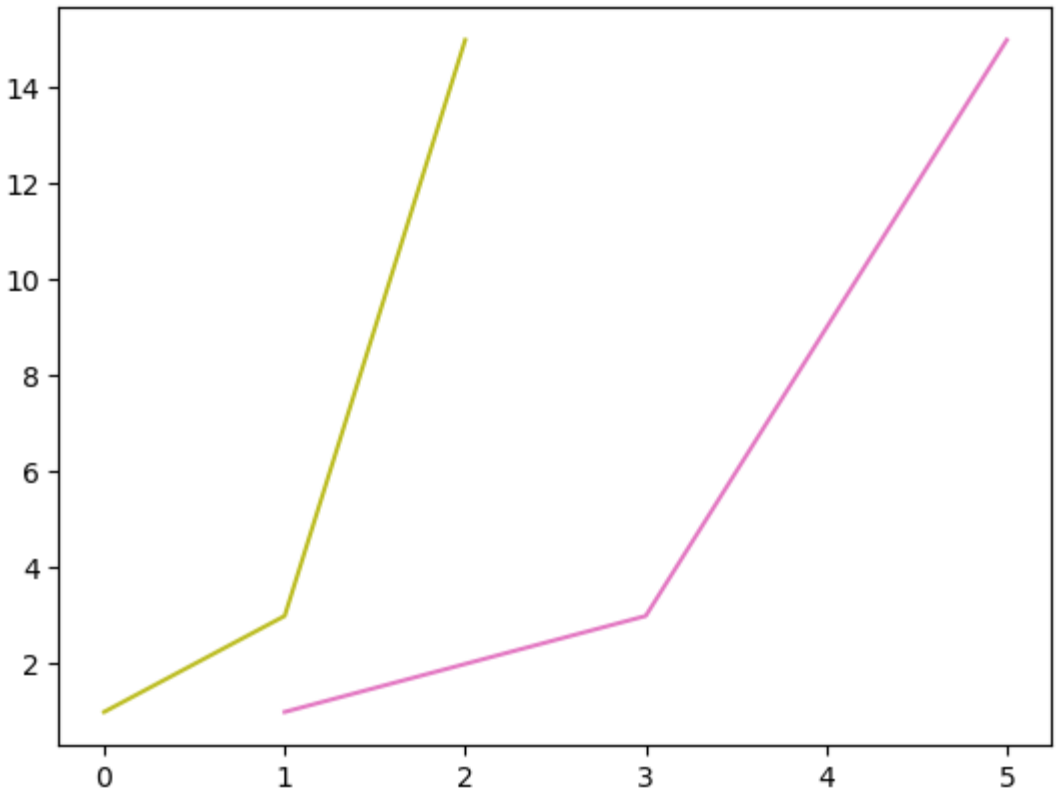
Out[16]:
```
[<matplotlib.lines.Line2D at 0x1b818f644d0>]
```



Ключевой аргумент `color` или `c` задает цвет линии графика. Существует несколько форматов представления цвета. Одним из форматов является использование строк с названием цвета: `'blue'`, `'green'`, `'red'`, `'cyan'`, `'magenta'`, `'yellow'`, `'black'`, `'white'`. Формат для задания менее ярких цветов: `'tab:blue'`, `'tab:pink'`, `'tab:olive'` и т.д.

In [17]:
```
plt.plot(y, color='tab:olive')
plt.plot(x, y, c='tab:pink')
```

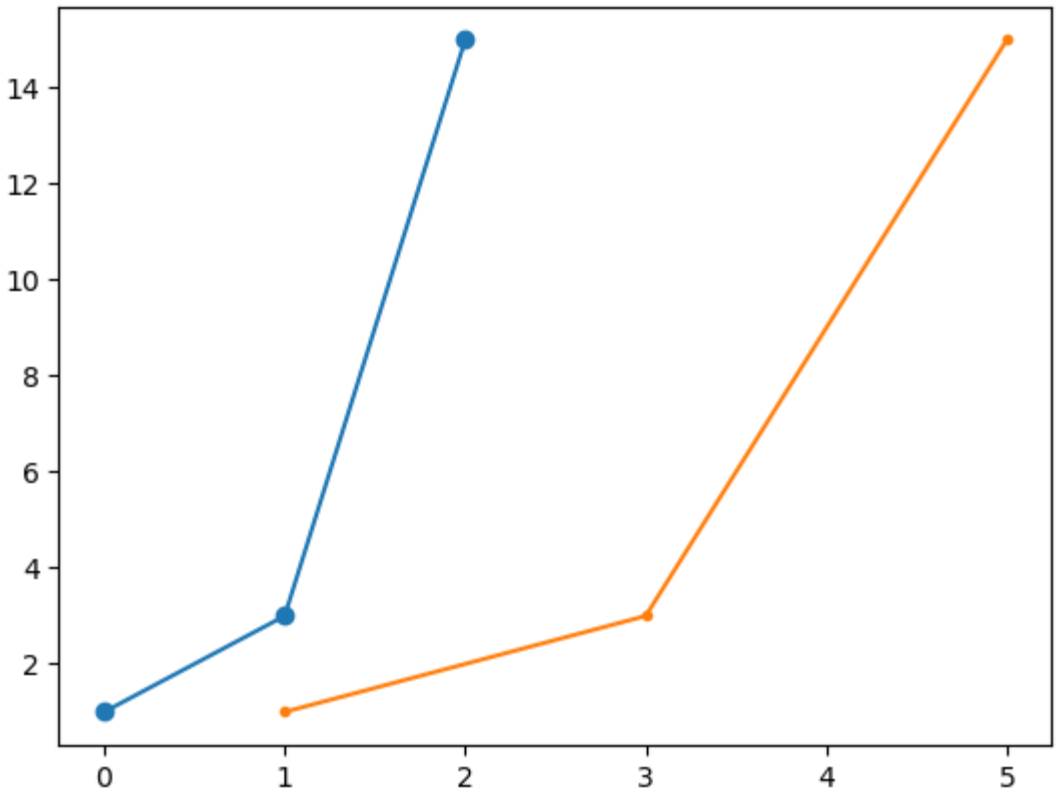
Out[17]:
```
[<matplotlib.lines.Line2D at 0x1b818f75cd0>]
```



Маркер -- это символ, выводимый в каждой точке данных графика. По умолчанию маркеры для точек графика НЕ создаются. Ключевой аргумент `marker` задает тип маркера для точек гафика. Значением ключевого аргумента `marker` является специальный односимвольный строковый объект из одного символа, определяющий конкретный тип маркера. Всего определено 22 способа определения типа маркера. Например, `'o'` задает маркер в виде кружка, `.` -- маркер в виде точки, `'*'` -- маркер в виде звездочки и т.д.

In [18]:
```
plt.plot(y, marker='o')
plt.plot(x, y, marker='.')
```

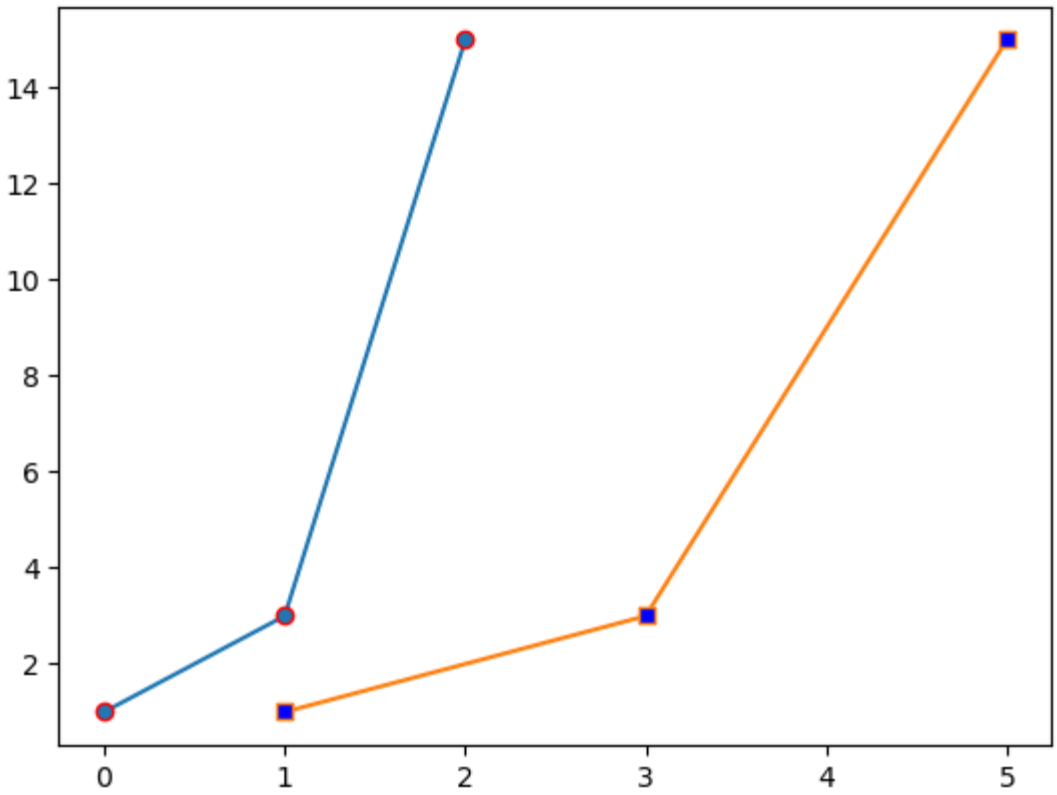
Out[18]:
```
[<matplotlib.lines.Line2D at 0x1b81917ee10>]
```



Свойства маркера задаются дополнительными ключевыми аргументами: цвет границы маркера `markeredgecolor` , толщина границы маркера `markeredgewidth` , цвет заливки маркера `markerfacecolor` , размер маркера в пунктах `markersize`

In [19]:
```
plt.plot(y, marker='o', markeredgecolor='red', markeredgewidth=1)
plt.plot(x, y, marker='s', markerfacecolor='blue', markersize = 6)
```

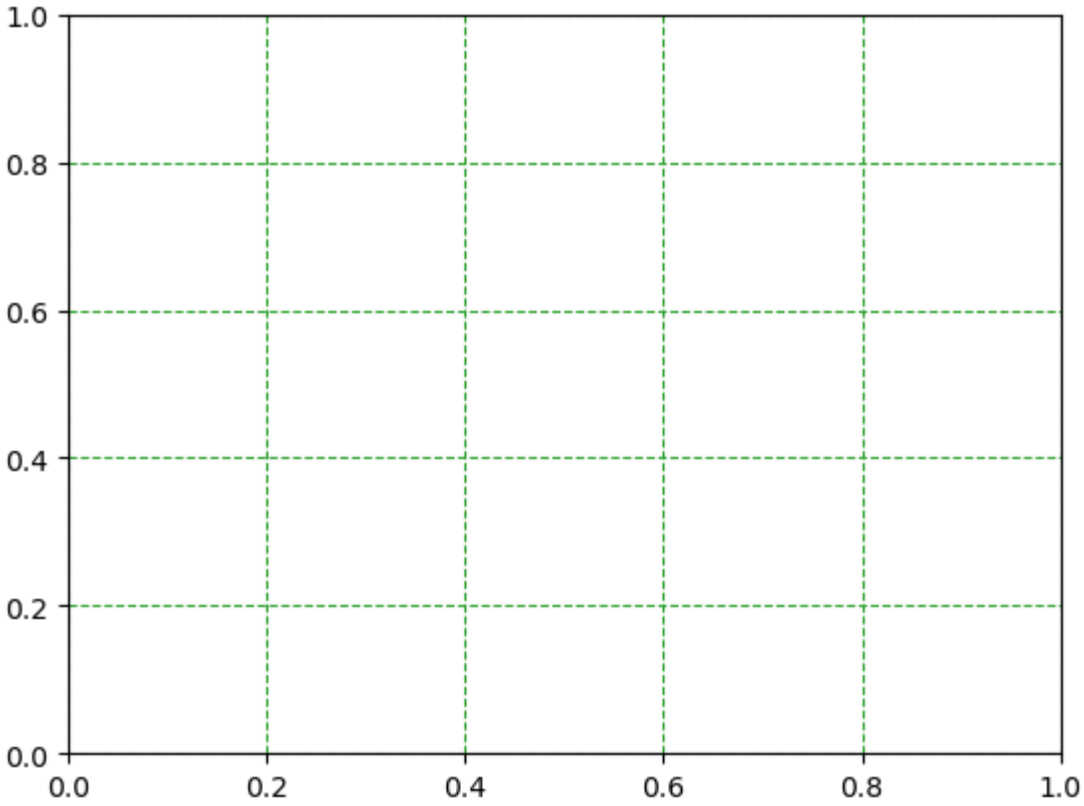
Out[19]:
```
[<matplotlib.lines.Line2D at 0x1b8192043d0>]
```



Функция grid

Функция `grid` модуля `matplotlib.pyplot` отображает сетку в *текущей* графической области с помощью вертикальных и горизонтальных прямых в графической области. Для линий сетки можно задать оформление указанием специальных ключевых аргументов: `linestyle` , `linewidth` , `color` , `marker` и т.д., как и для линий графика

In [20]:
```
plt.grid(True, linestyle='--', color='tab:green')
```



Встроенные стили

В `matplotlib` можно использовать заранее определенные стили оформления для графиков.

Все доступные стили можно просмотреть

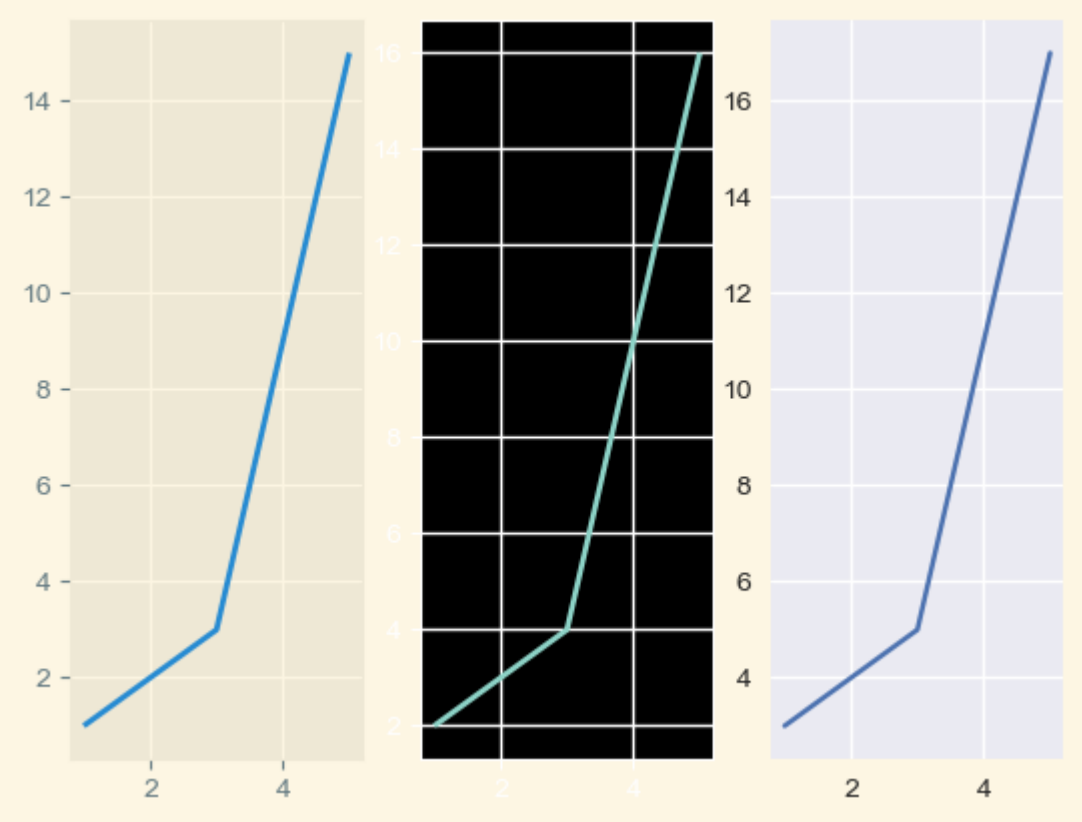
In [21]:
```
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', '
bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale
', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-colorblind', 'seaborn-v0_8-
dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep',
'seaborn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-
pastel', 'seaborn-v0_8-poster', 'seaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seabor
n-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

Для установки стиля нужно вызвать функцию `plt.style.use('имя_стиля')` перед построением графика

In [22]:
```
plt.style.use('Solarize_Light2')
plt.subplot(1,3,1)
plt.plot(x,y)
plt.style.use('dark_background')
plt.subplot(1,3,2)
plt.plot(x,y+1)
plt.style.use('seaborn-v0_8')
plt.subplot(1,3,3)
plt.plot(x,y+2)
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x1b81a25e8d0>]
```



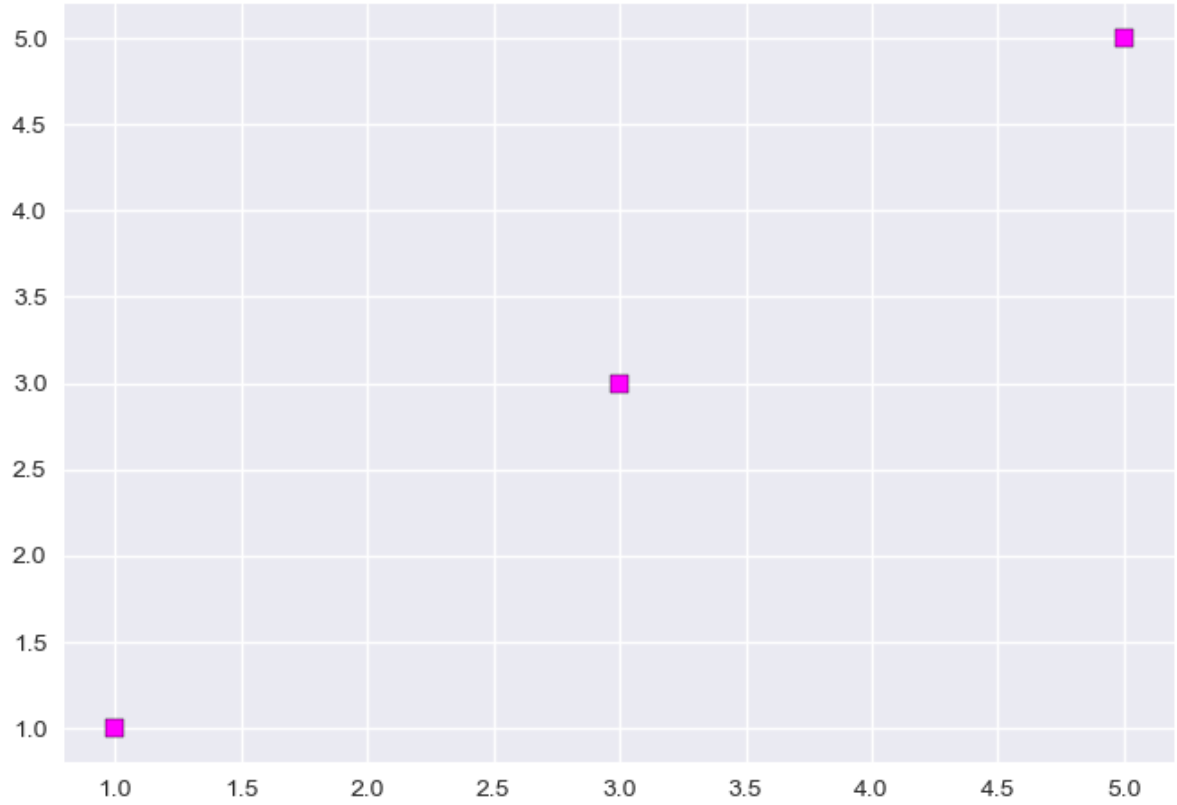
Функции scatter, step, stem, stackplot

Функция `scatter` модуля `matplotlib.pyplot` используется для построения двумерных графиков в *текущей* графической области. Графики строятся по значениям абсцисс и ординат точек, представляющих график. Координаты точек задаются `numpy`-массивами или списками. *Точки графика НЕ соединяются отрезками*.

Синтаксис вызова функции `plot` и функции `scatter` очень похожи. В частности, многие ключевые аргументы для оформления графика, которые задаются при вызове функции `plot`, можно использовать также при вызове функции `scatter`

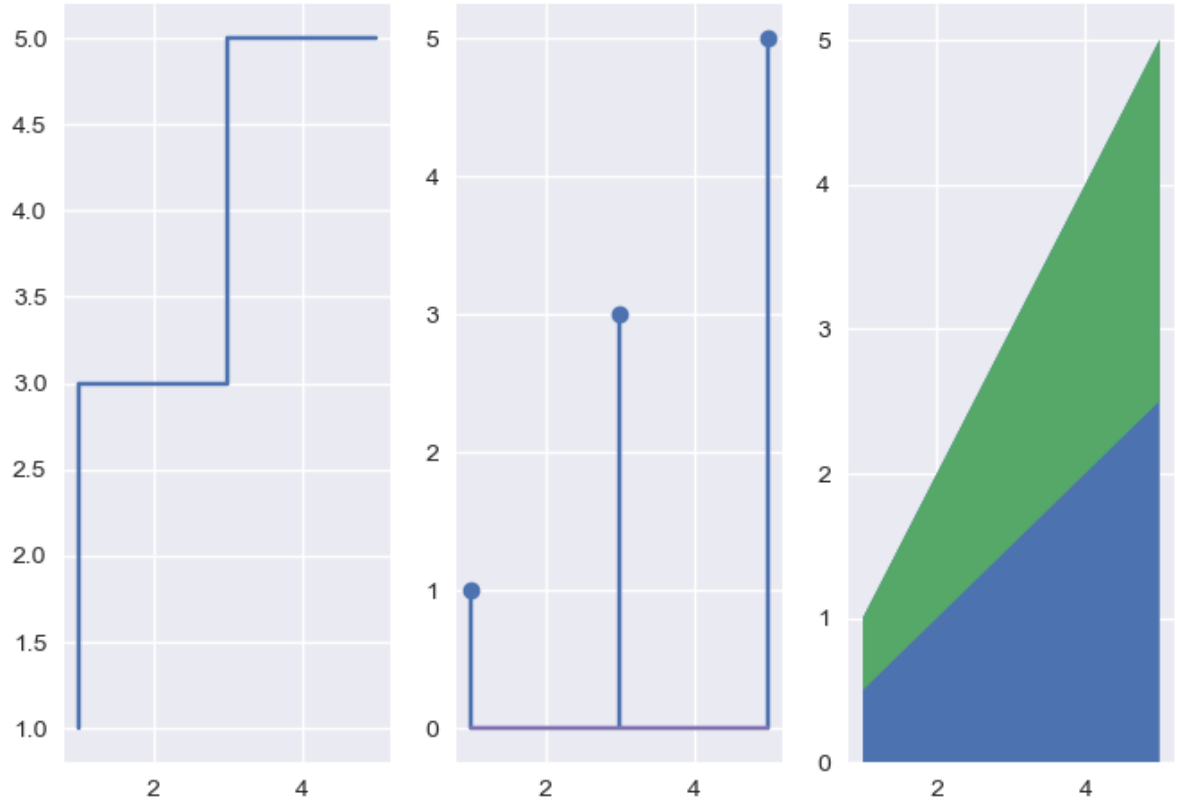
In [23]:
```
plt.scatter(x, x,
            color='magenta', marker='s', edgecolors='black')
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x1b818f75790>
```



In [24]:
```
plt.subplot(1,3,1)
plt.step(x, x)
plt.subplot(1,3,2)
plt.stem(x, x)
plt.subplot(1,3,3)
plt.stackplot(x, x, -x/2)
```

Out[24]:
```
[<matplotlib.collections.PolyCollection at 0x1b81a464310>,
 <matplotlib.collections.PolyCollection at 0x1b81a471410>]
```



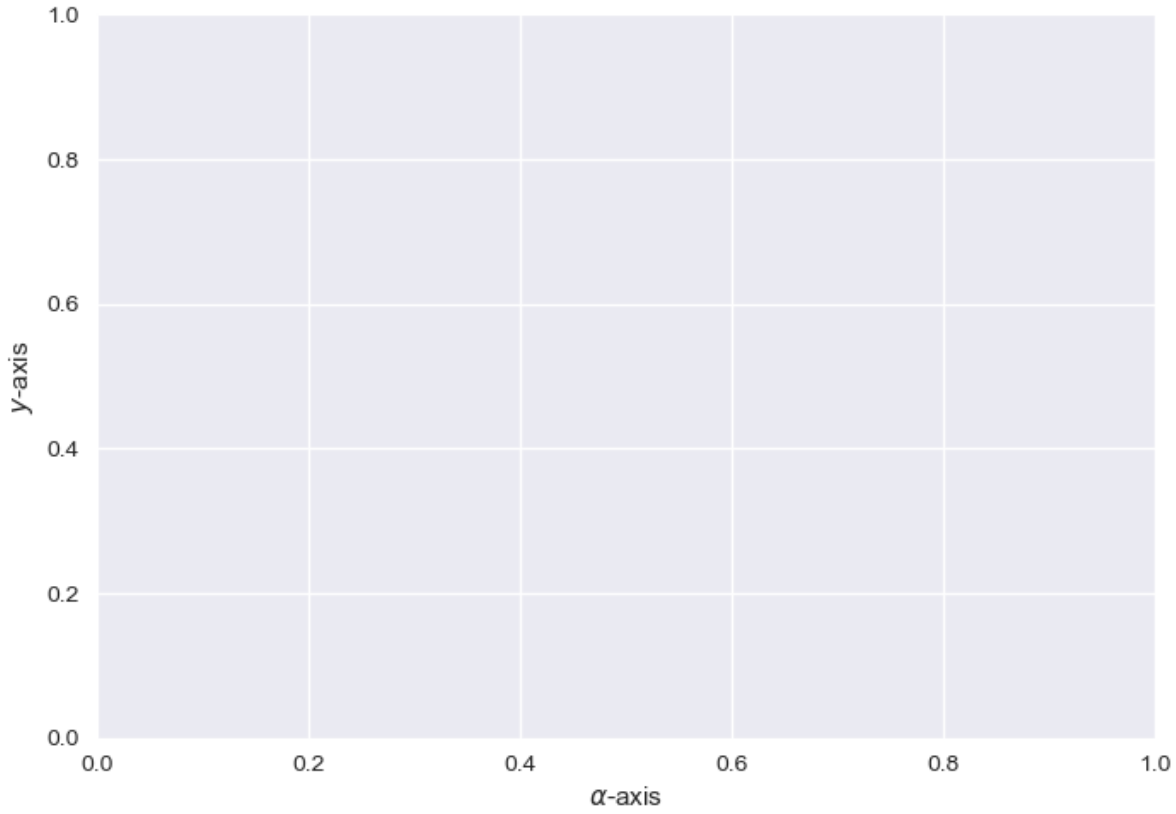
Текстовое объекты в графической области

Добавление текстовых объектов в графическую область делает изображение более информативным.

Функции xlabel, ylabel

Функции `xlabel` , `ylabel` модуля `matplotlib.pyplot` создают текстовые объекты в *текущей* графической области для представления *подписей координатных осей*. Первый аргумент является обязательным, он задает содержимое текстового объекта в виде строки.

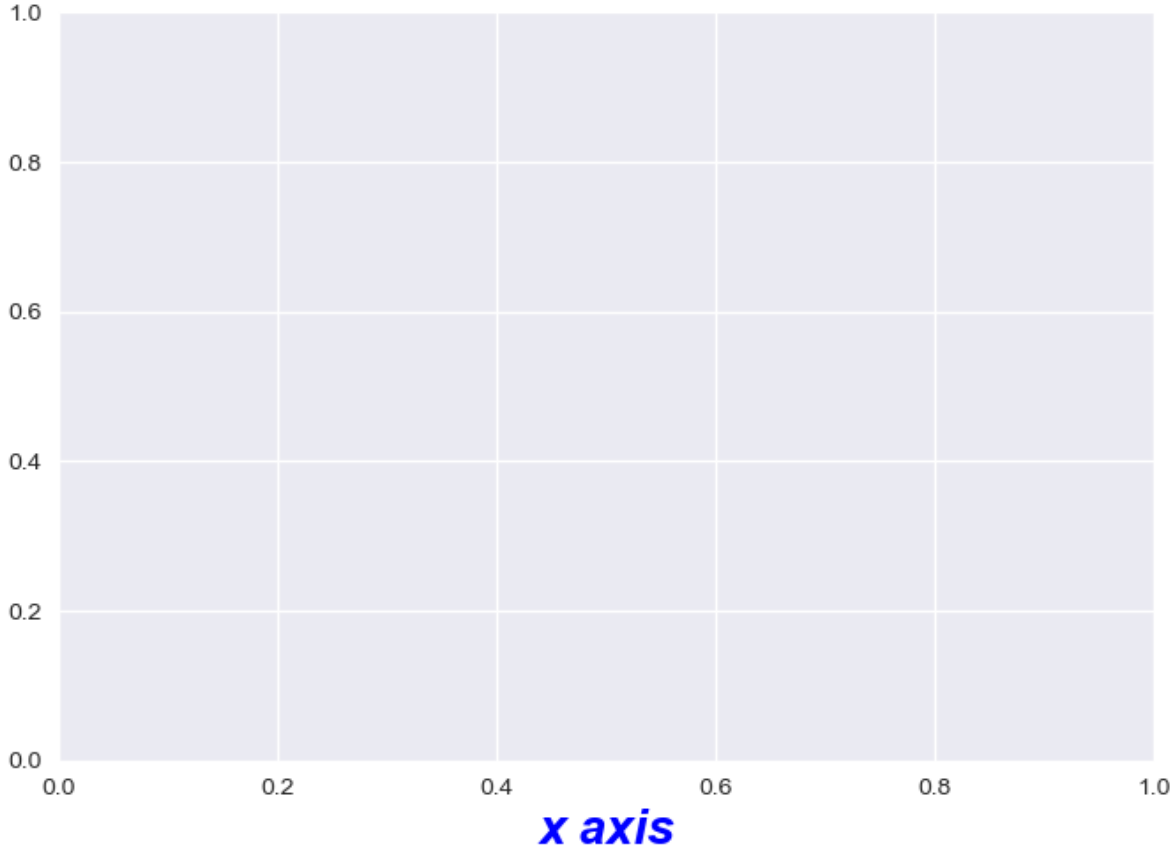
In [25]:
```
plt.xlabel(r'$\alpha$-axis')
plt.ylabel('$y$-axis');
```



Строка, являющаяся содержимым ЛЮБОГО текстового объекта, может быть записана с использованием языка разметки `LaTeX` для представления математического текста. Например, `$\alpha -\frac{\sigma}{2}$` для отображения математического текста $\alpha - \frac{\sigma}{2}$. Перед строковым объектом с использованием языка разметки `LaTeX` стоит указать символ `r`, чтобы символ (`\`) не интерпретировался как часть управляющего символа.

Для ЛЮБОГО текстового объекта в графической области можно задать оформление текста указанием специальных ключевых аргументов: размер шрифта в пунктах `fontsize`, стиль шрифта `fontstyle` (например, `'italic'`, `'oblique'`, `'normal'`), насыщенность шрифта `fontweight` (например, `'light'`, `'regular'`, `'bold'`, `'roman'`), цвет текста `color` и т.д.

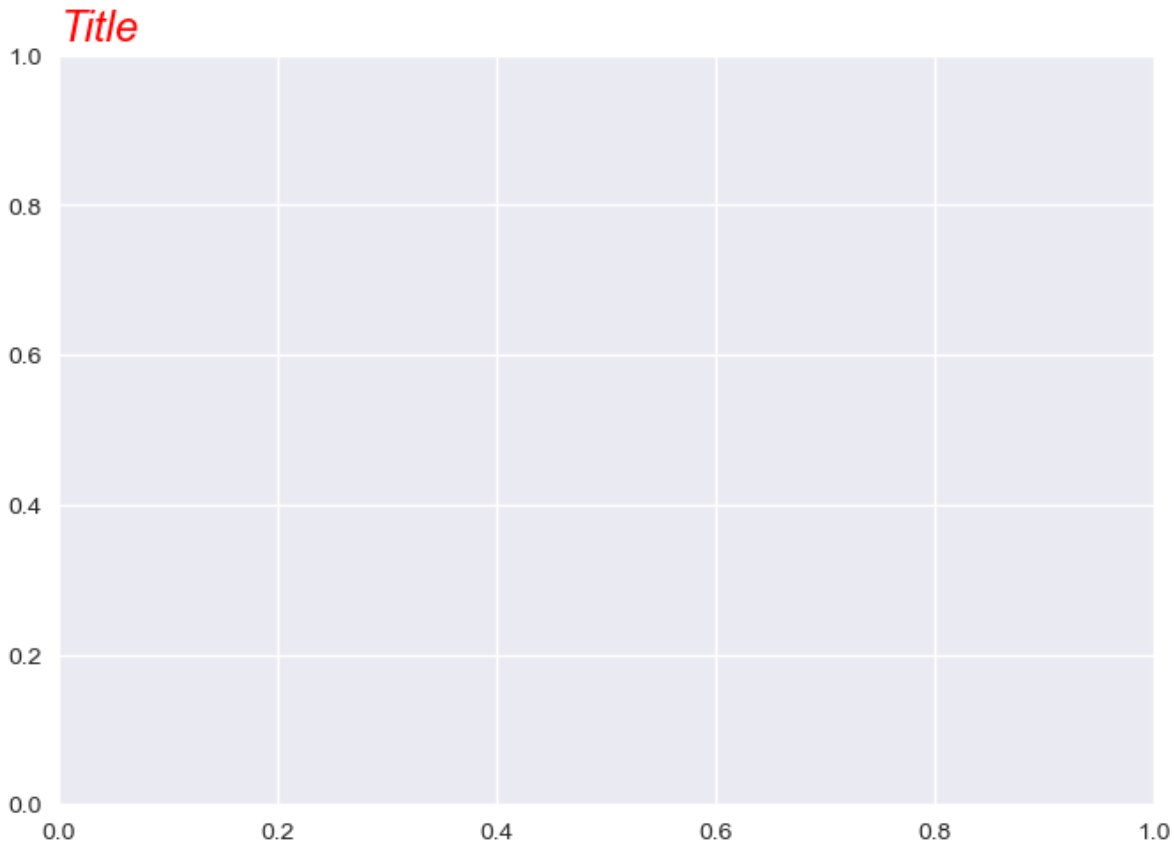
In [26]:
```
plt.xlabel("x axis", fontsize=20, fontstyle='italic',
           fontweight='bold', color='blue');
```



Функция title

Функция `title` модуля `matplotlib.pyplot` создает текстовый объект в *текущей* графической области для представления *заголовка графической области*. Первый аргумент является обязательным, он задает содержимое текстового объекта. Дальнейшие аргументы являются необязательными и передаются по ключу. Специальный ключевой аргумент `loc` со значениями `'center'`, `'left'`, `'right'` определяет выравнивание текста заголовка.

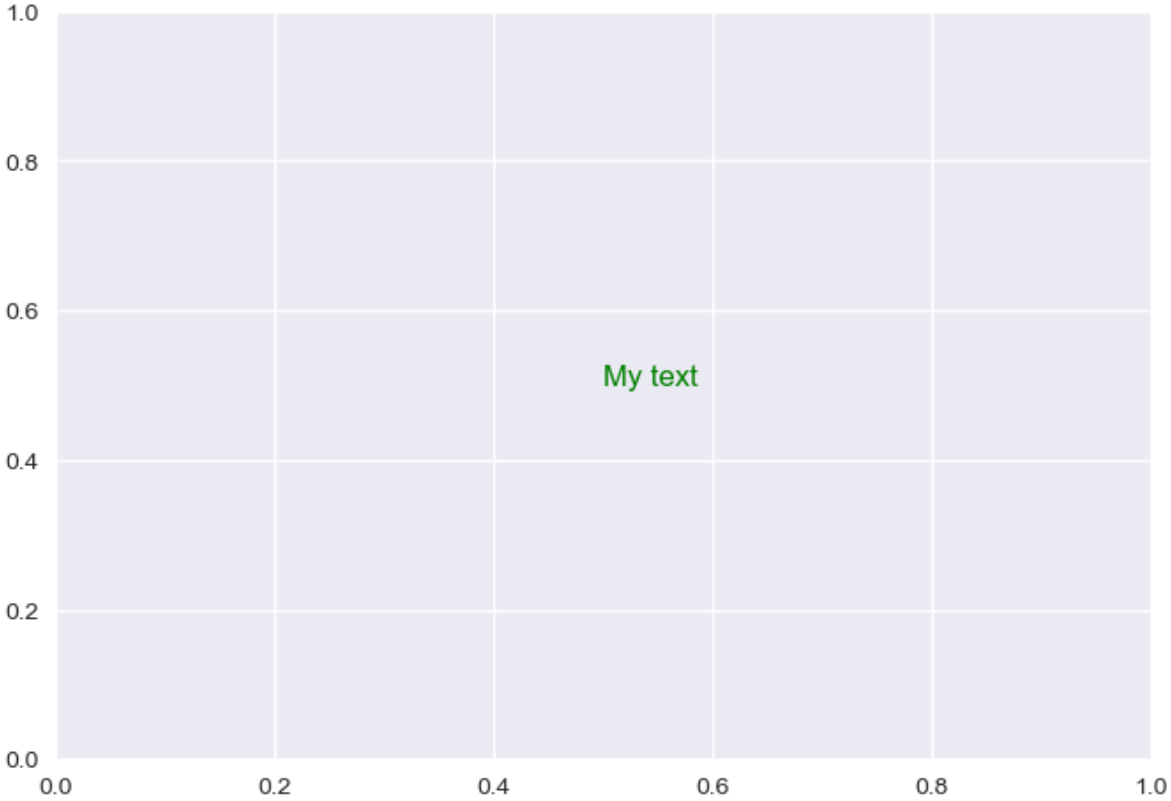
In [27]:
```
plt.title("Title", fontsize='xx-large', fontstyle='oblique',
          fontweight='light', color='red', loc='left');
```



Функция text

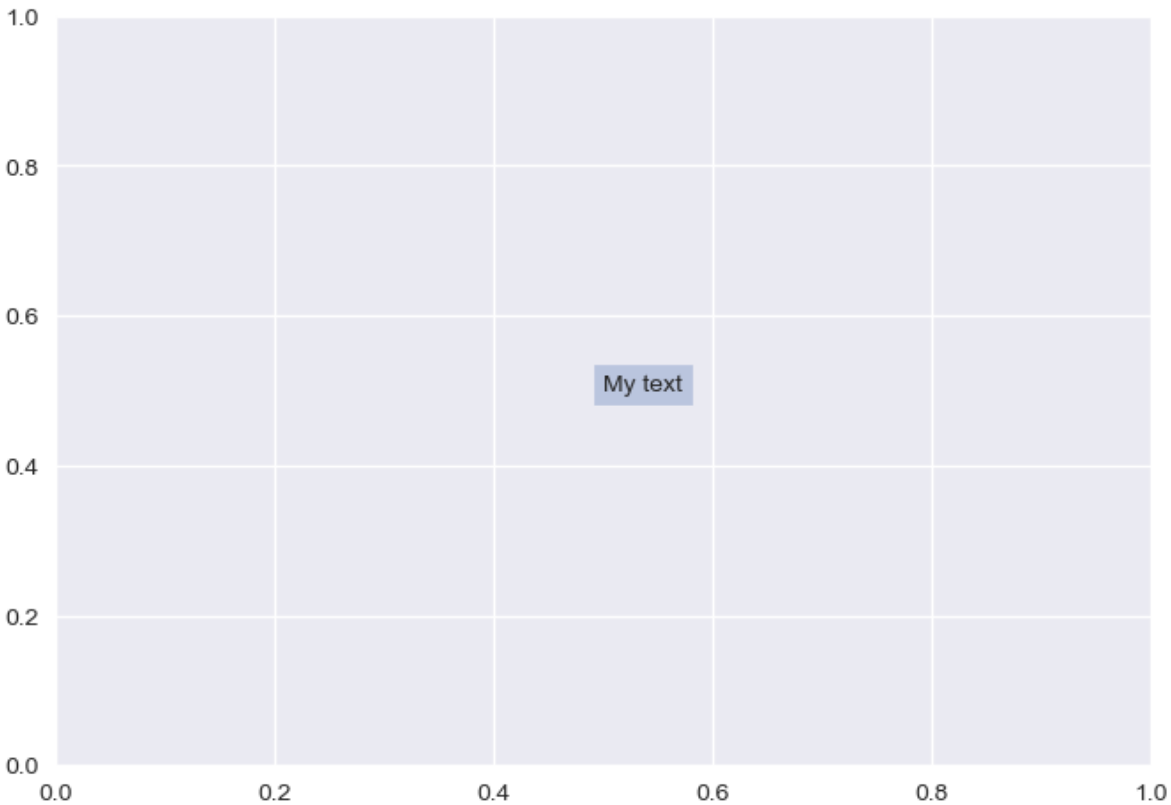
Функция `text` модуля `matplotlib.pyplot` создает текстовый объект в *текущей* графической области. Первые два аргумента функции `text` задают координаты левого нижнего угла текстового объекта, третий аргумент задает содержимое текстового объекта. Дальнейшие аргументы являются необязательными, передаются по ключу и задают оформление текста

In [28]:
```
plt.text(0.5, 0.5, "My text", fontsize='large', fontstyle='normal',
         fontweight='roman', color='green');
```



Ключевой аргумент `bbox` позволяет установить фон текстового объекта и, в частности, определить прозрачность `'alpha'` для фонового цвета

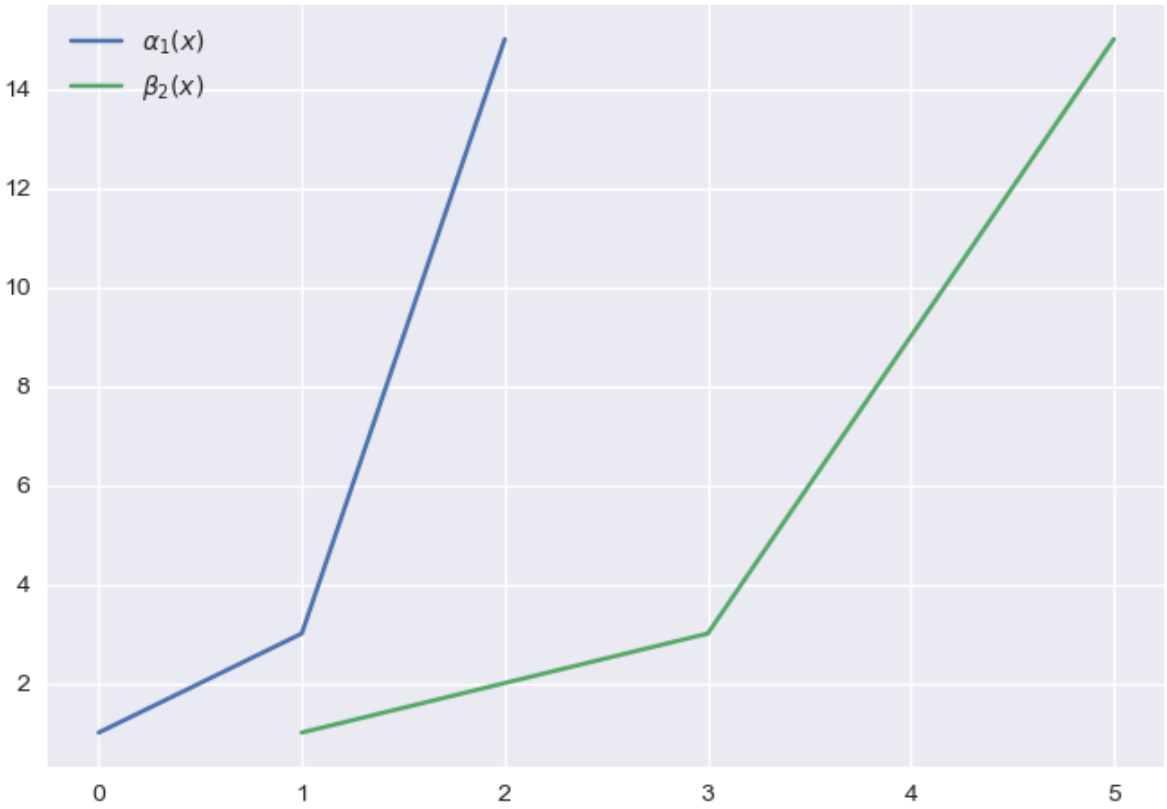
In [29]:
```
plt.text(0.5, 0.5, "My text", bbox={'alpha':0.3});
```



Функция legend

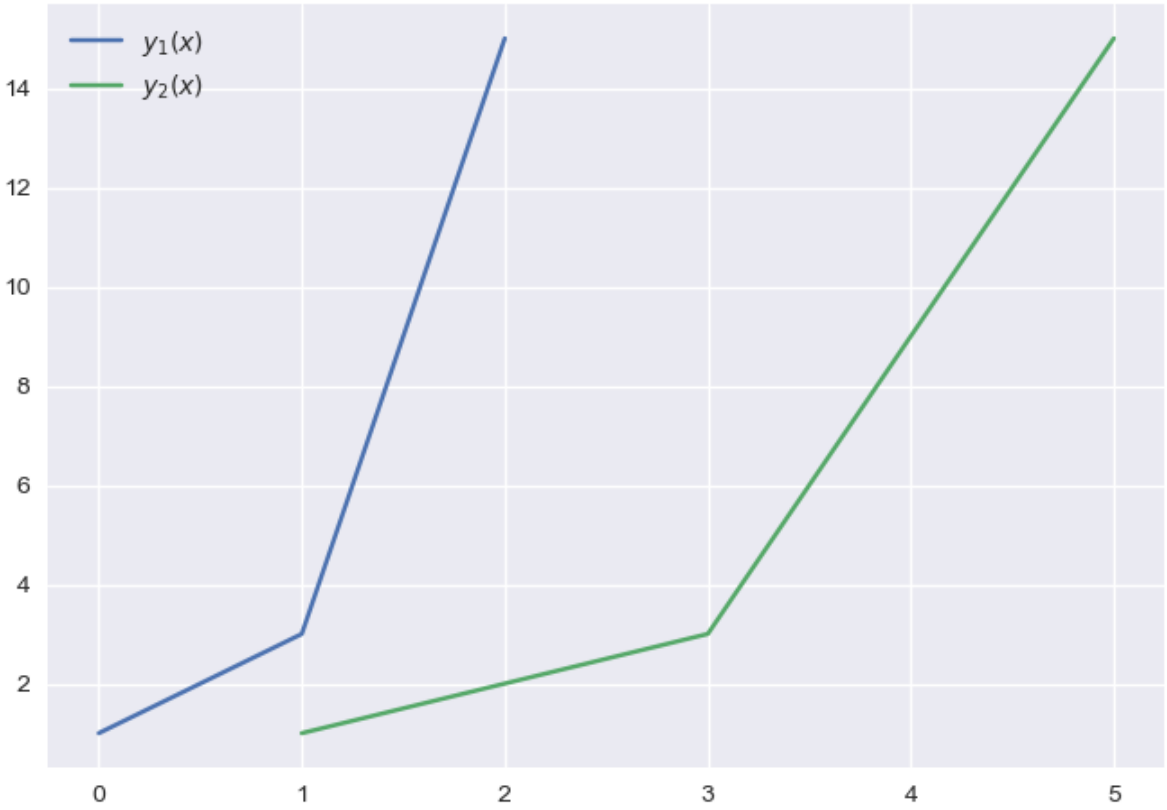
Функция `legend` модуля `matplotlib.pyplot` создает текстовый объект в *текущей* графической области для представления *подписей графиков*. При создании графиков необходимо указание ключевого аргумента `label`. Значение ключевого аргумента задает подпись графика в виде строкового объекта, которая будет отображаться в легенде после вызова функции `legend`.

In [30]:
```
plt.plot(y, label=r'$\alpha_1(x)$')
plt.plot(x, y, label=r'$\beta_2(x)$')
plt.legend();
```



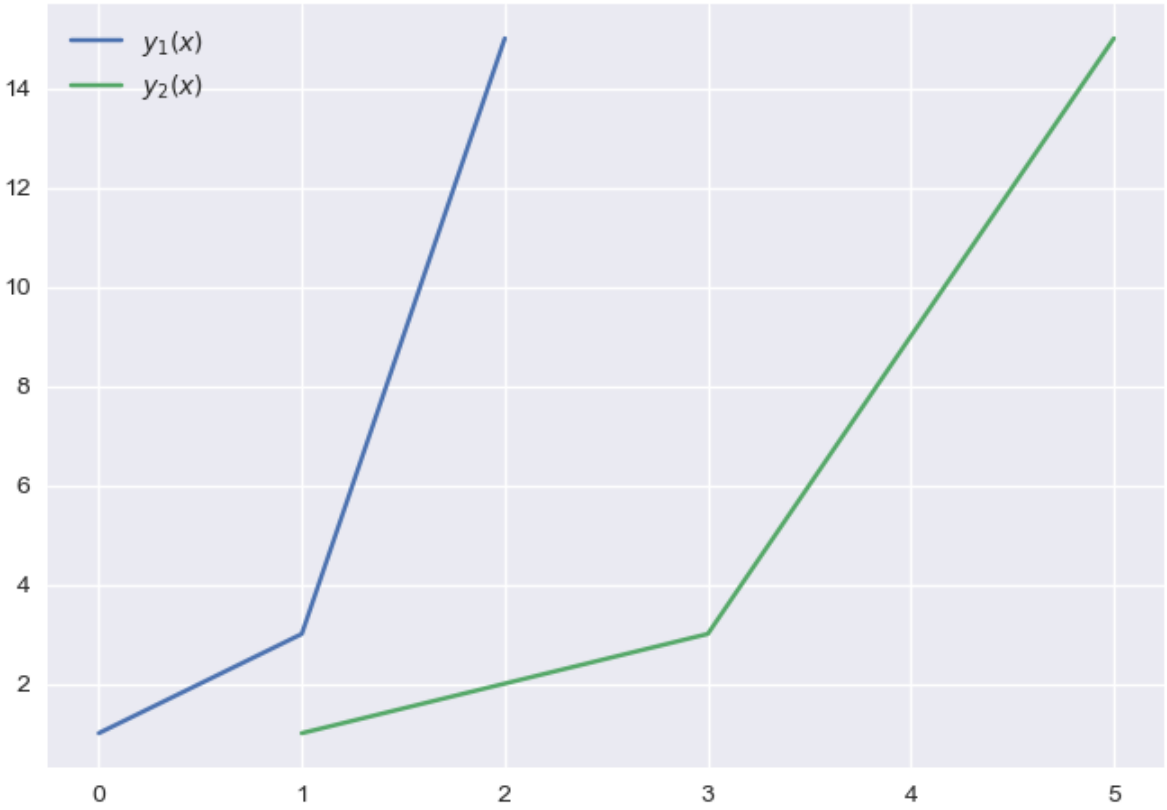
Текст подписи к графикам можно указать непосредственно при вызове функции `legend`. В этом случае функция `legend` вызывается с одним аргументом в виде списка с подписями к графикам

In [31]:
```
plt.plot(y)
plt.plot(x, y)
plt.legend(['$y_1(x)$', '$y_2(x)$']);
```



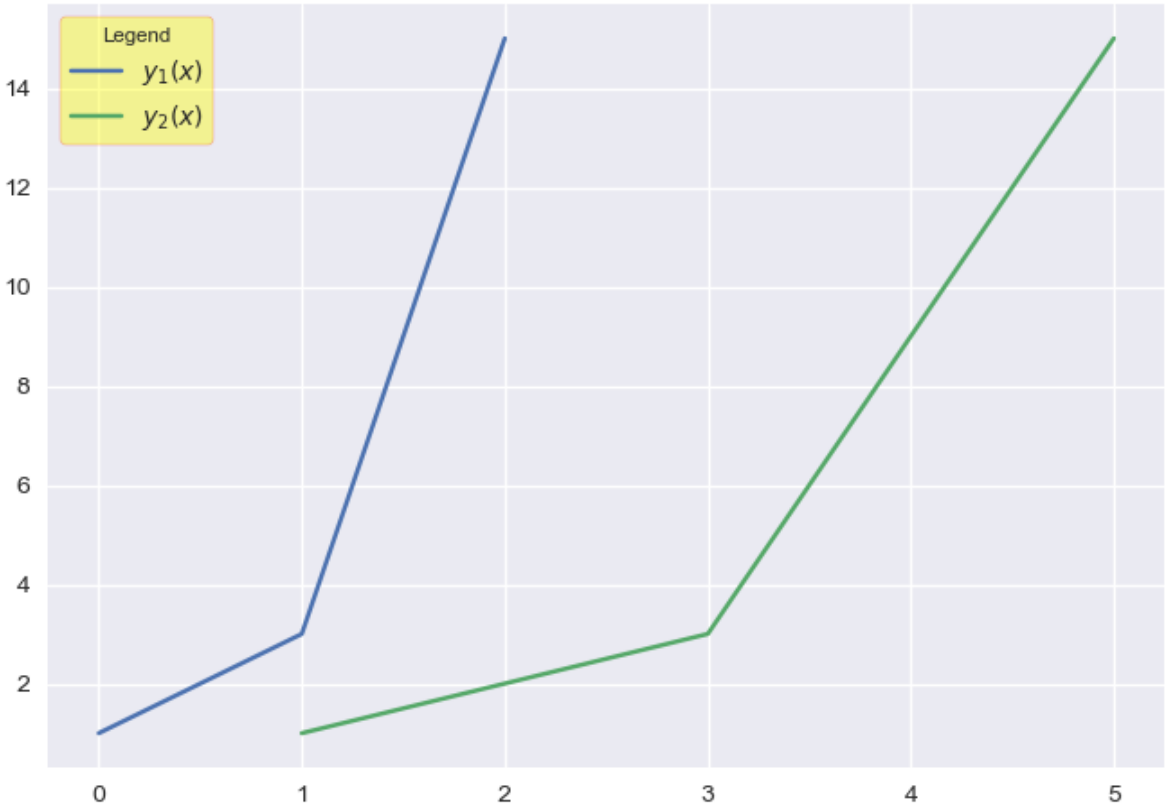
По умолчанию легенда располагается в верхнем левой углу графической области. Специальный ключевой аргумент `loc` со значениями `'best'`, `'upper right'`, `'upper left'`, `'lower left'`, `'lower right'`, `'right'`, `'center left'`, `'center right'`, `'lower center'`, `'upper center'`, `'center'` определяет расположение легенды в графической области.

In [32]:
```
plt.plot(y)
plt.plot(x, y)
plt.legend(['$y_1(x)$', '$y_2(x)$'], loc='best');
```



Оформление легенды можно задать с помощью специальных ключевых аргументов: размер шрифта `fontsize`, отображение рамки `frameon`, прозрачность `framealpha`, цвет заливки `facecolor`, цвет рамки `edgecolor`, текст заголовка `title`, размер шрифта заголовка `title_fontsize` и т.д.

In [33]:
```
plt.plot(y)
plt.plot(x, y)
plt.legend(['$y_1(x)$', '$y_2(x)$'],
           fontsize='medium', frameon=True, framealpha=0.4,
           facecolor='yellow', edgecolor='red',
           title='Legend', title_fontsize='small');
```

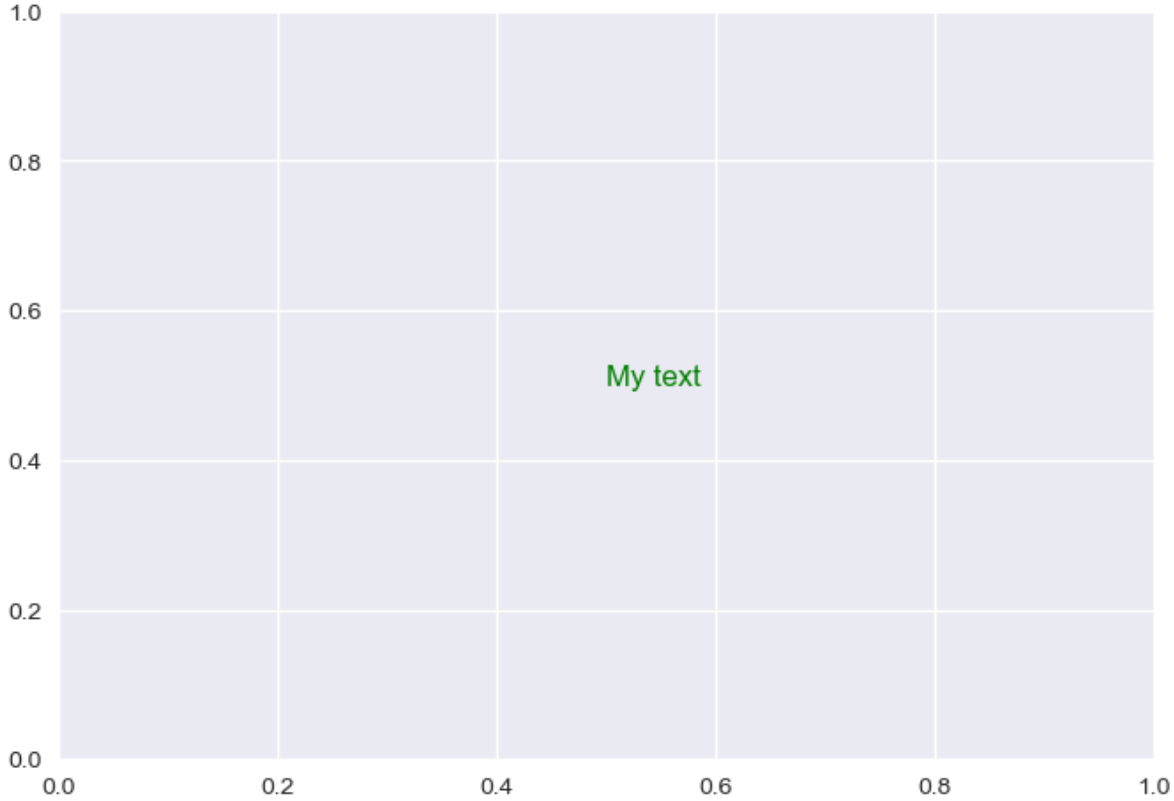


Все функции данного подраздела относятся к *процедурному подходу* построения изображений средствами модуля `matplotlib.pyplot` .

Функции создания текстовых объектов `xlabel` , `ylabel` , `title` , `text` , возвращают экземпляры класса `Text` . Класс `Text` расположен в модуле `matplotlib.text`

In [34]:
```
txt = plt.text(0.5, 0.5, "My text", fontsize='large', fontstyle='normal',
               fontweight='roman', color='green');
type(txt)
```

Out[34]:
```
matplotlib.text.Text
```



11.4 Объектно-ориентированный подход для построения изображений

В пакете `matplotlib` определен абстрактный базовый класс `Artist`, от которого наследуются классы `Figure`, `Axes`, `Axis`, `Line2D`, `Text`, `Patch` и т.д. для создания графических объектов.

Двумерные фигуры определяются с помощью классов: `Circle`, `Ellipse`, `Rectangle`, `Polygon` и т.д., которые наследуются от класса `Patch`.

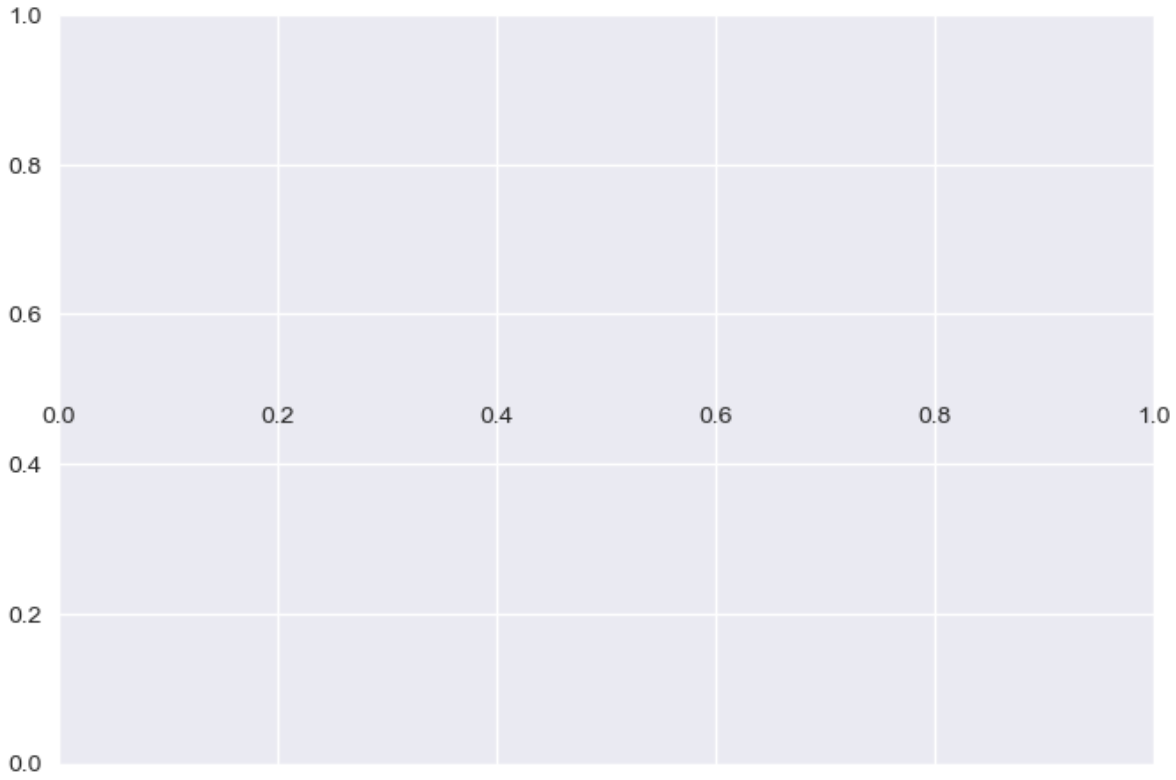
Основные возможности построения изображения при использовании *объектно-ориентированного подхода* реализованы с помощью методов для объектов графических областей.

In [35]:
```
len(dir(plt.Axes))
```

Out[35]:
```
369
```

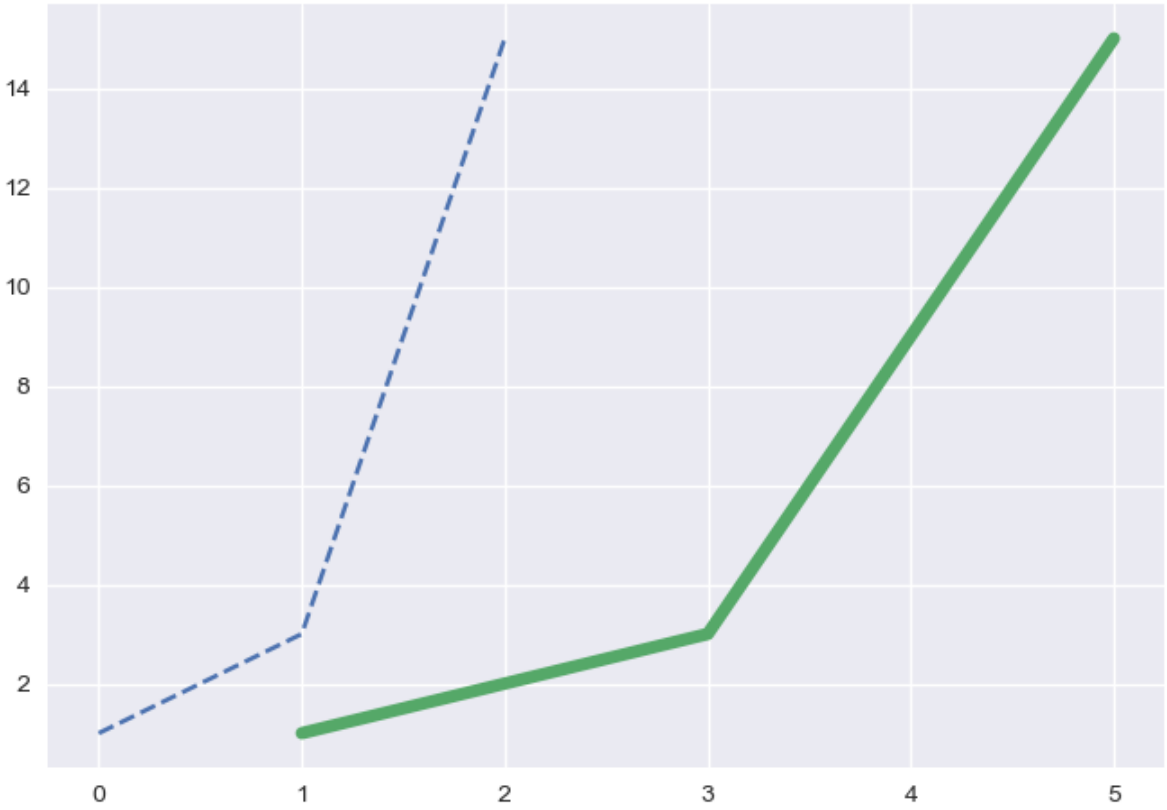
Для многих функций модуля `matplotlib.pyplot` существуют аналогичные методы класса `Axes`. Например, `ax.axis()`, `ax.set_xlim`, `ax.set_ylim`, `ax.plot()`, `ax.grid()`, `ax.set_xlabel()`, `ax.set_ylabel()`, `ax.set_title()`, `ax.text()`, `ax.legend()`, `ax.set()` и др.

In [36]:
```
fig = plt.gcf()
ax = plt.gca()
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')
ax.spines['bottom'].set_position(('data',0.5))
#ax.spines['left'].set_position(('data',0.5))
```



Оформление графика можно задать с помощью специальных ключевых аргументов: `linestyle`, `linewidth`, `color`, `marker` и т.д., при вызове функции `plot`. Аналогичные действия можно выполнить вызовом соответствующих методов для объекта типа `Line2D`, представляющего график

In [37]:
```
l1, = plt.plot(y)
l2, = plt.plot(x, y)
l1.set_linestyle('dashed'); l2.set_linewidth(5)
```



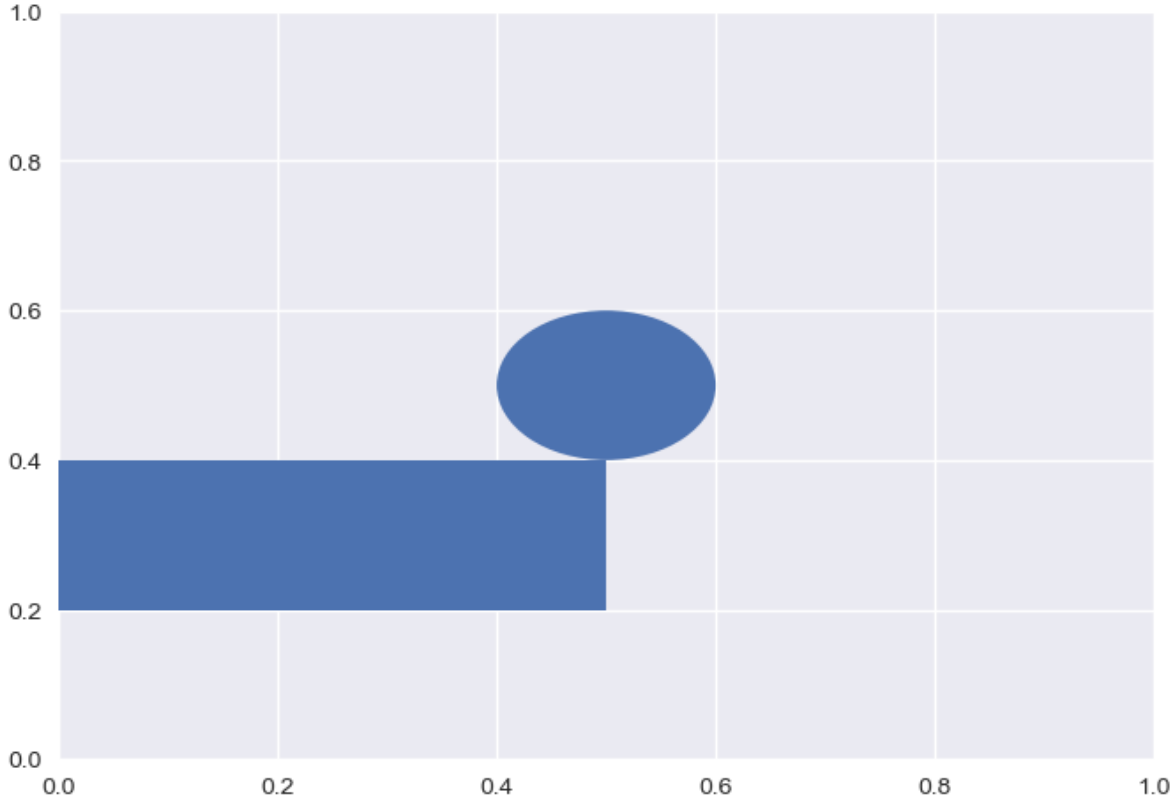
Объекты двумерных фигур создаются с помощью конструкторов соответствующих классов. Добавление двумерных фигур в графическую область осуществляется с помощью вызова метода `add_patch` для объекта графической области

In [38]:
```
ax = plt.axes()

circle = plt.Circle((0.5,0.5),0.1)
rectangle = plt.Rectangle((0,0.2), 0.5, 0.2)

ax.add_patch(circle)
ax.add_patch(rectangle)
```

Out[38]:
```
<matplotlib.patches.Rectangle at 0x1b818db42d0>
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0