МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №4 по дисциплине технологии распознавания образов

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

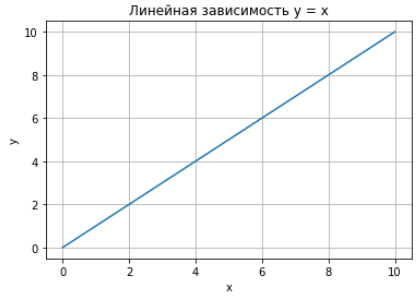
1. Примеры из методических указаний

In [1]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

Построение графика

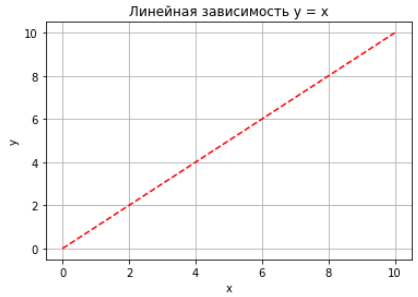
In [4]:
```
import numpy as np
# Независимая (x) и зависимая (y) переменные
x = np.linspace(0, 10, 50)
y = x
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y) # построение графика
```

Out[4]:
```
[<matplotlib.lines.Line2D at 0x7e3b8ec670>]
```



In [5]:
```
# Построение графика
plt.title("Линейная зависимость y = x") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y, "r--") # построение графика
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7e3b93fdc0>]
```



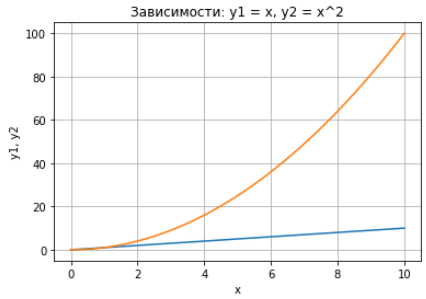
Несколько графиков на одном поле

In [9]:
```
# Линейная зависимость
```

Несколько графиков на одном поле

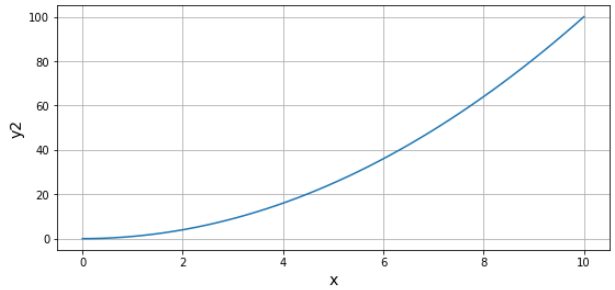
In [9]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# Построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y1, y2") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y1, x, y2) # построение графика
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x7e3ad61bb0>,
 <matplotlib.lines.Line2D at 0x7e3ad61ca0>]
```



Несколько разделенных полей с графиками

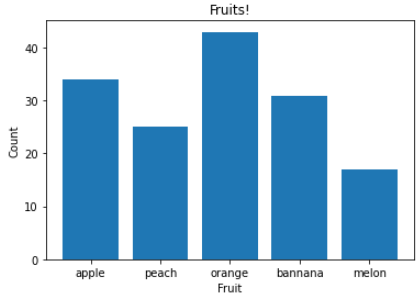
In [10]:
```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# Построение графиков
plt.figure(figsize=(9, 9))
plt.subplot(2, 1, 1)
plt.plot(x, y1) # построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.ylabel("y1", fontsize=14) # ось ординат
plt.grid(True) # включение отображение сетки
plt.subplot(2, 1, 2)
plt.plot(x, y2) # построение графика
plt.xlabel("x", fontsize=14) # ось абсцисс
plt.ylabel("y2", fontsize=14) # ось ординат
plt.grid(True) # включение отображения сетки
```



Построение диаграммы для категориальных данных

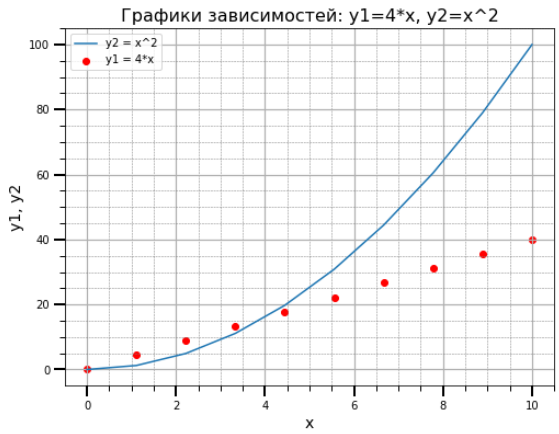
In [11]:
```
fruits = ["apple", "peach", "orange", "bannana", "melon"]
counts = [34, 25, 43, 31, 17]
plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
```

Out[11]:
```
Text(0, 0.5, 'Count')
```



Основные элементы графика

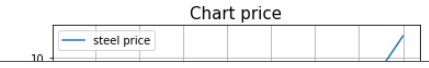
In [12]:
```
import matplotlib.pyplot as plt
from matplotlib.ticker import (MultipleLocator, FormatStrFormatter,

AutoMinorLocator)

import numpy as np
x = np.linspace(0, 10, 10)
y1 = 4*x
y2 = [i**2 for i in x]
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_title("Графики зависимостей: y1=4*x, y2=x^2", fontsize=16)
ax.set_xlabel("x", fontsize=14)
ax.set_ylabel("y1, y2", fontsize=14)
ax.grid(which="major", linewidth=1.2)
ax.grid(which="minor", linestyle="--", color="gray", linewidth=0.5)
ax.scatter(x, y1, c="red", label="y1 = 4*x")
ax.plot(x, y2, label="y2 = x^2")
ax.legend()
ax.xaxis.set_minor_locator(AutoMinorLocator())
ax.yaxis.set_minor_locator(AutoMinorLocator())
ax.tick_params(which='major', length=10, width=2)
ax.tick_params(which='minor', length=5, width=1)
plt.show()
```

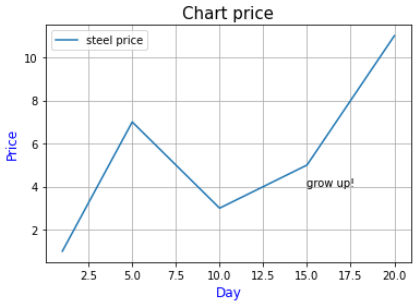


Текстовые надписи на графике

In [13]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, label='steel price')
plt.title('Chart price', fontsize=15)
plt.xlabel('Day', fontsize=12, color='blue')
plt.ylabel('Price', fontsize=12, color='blue')
plt.legend()
plt.grid(True)
plt.text(15, 4, 'grow up!')
```

Out[13]:
```
Text(15, 4, 'grow up!')
```

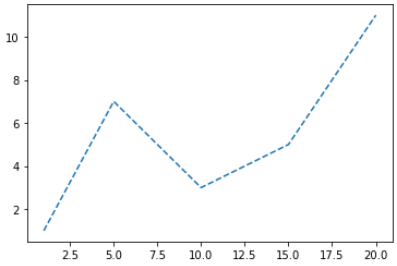




Работа с линейным графиком

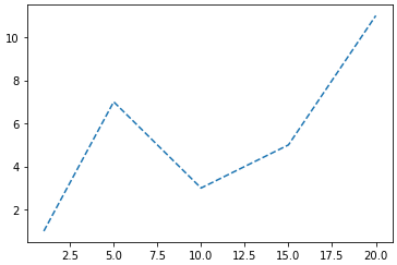
In [14]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--')
```

Out[14]: [<matplotlib.lines.Line2D at 0x7e37fa82e0>]



In [15]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
line = plt.plot(x, y)
plt.setp(line, linestyle='--')
```

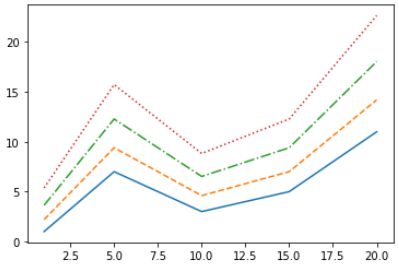
Out[15]: [None]



In [16]:
```
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
```

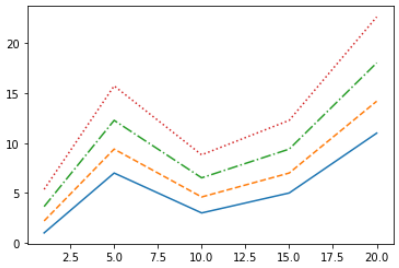
In [16]:
```
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
plt.plot(x, y1, '-', x, y2, '--', x, y3, '-.', x, y4, ':')
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x7e3ad85790>,
 <matplotlib.lines.Line2D at 0x7e3ad856d0>,
 <matplotlib.lines.Line2D at 0x7e3ad85910>,
 <matplotlib.lines.Line2D at 0x7e3ad85c10>]
```



In [17]:
```
plt.plot(x, y1, '-')
plt.plot(x, y2, '--')
plt.plot(x, y3, '-.')
plt.plot(x, y4, ':')
```

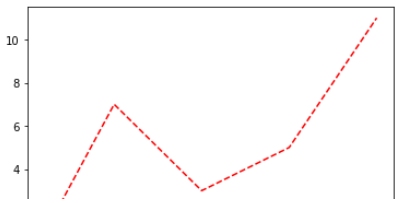
Out[17]:
```
[<matplotlib.lines.Line2D at 0x7e3ae4e6d0>]
```



Цвет линии

In [18]:
```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--r')
```

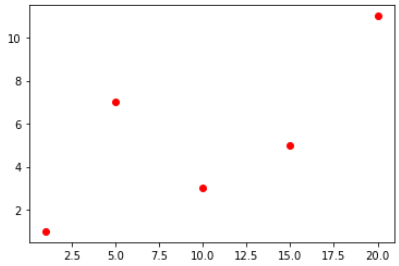
Out[18]:
```
[<matplotlib.lines.Line2D at 0x7e38037d90>]
```



Тип графика

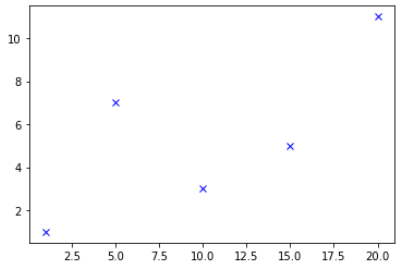
In [19]:
```
plt.plot(x, y, 'ro')
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x7e3aee53a0>]
```



In [20]:
```
plt.plot(x, y, 'bx')
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x7e3af3a0d0>]
```



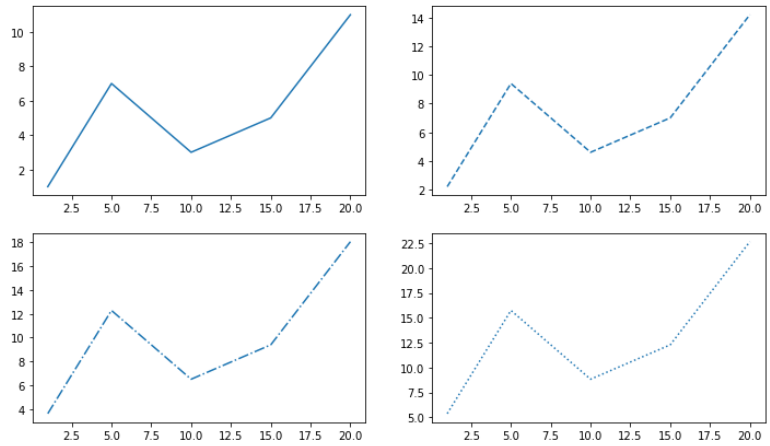
Размещение графиков на разных полях

Работа с функцией subplot()

In [21]:
```
# Исходный набор данных
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
# Настройка размеров подложки
plt.figure(figsize=(12, 7))
# Вывод графиков
plt.subplot(2, 2, 1)
plt.plot(x, y1, '-')
plt.subplot(2, 2, 2)
plt.plot(x, y2, '--')
plt.subplot(2, 2, 3)
plt.plot(x, y3, '-.')
plt.subplot(2, 2, 4)
plt.plot(x, y4, ':')
```

Out[21]:
```
[<matplotlib.lines.Line2D at 0x7e3b026c40>]
```

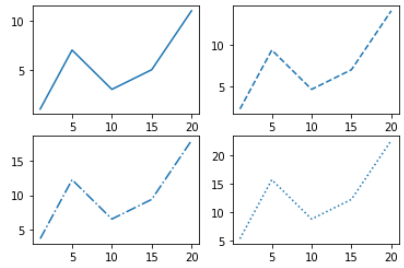
Out[21]: [<matplotlib.lines.Line2D at 0x7e3b026c40>]



Второй вариант использования subplot():

In [22]:
```
# Вывод графиков
plt.subplot(221)
plt.plot(x, y1, '-')
plt.subplot(222)
plt.plot(x, y2, '--')
plt.subplot(223)
plt.plot(x, y3, '-.')
plt.subplot(224)
plt.plot(x, y4, ':')
```

Out[22]: [<matplotlib.lines.Line2D at 0x7e3ca0c820>]

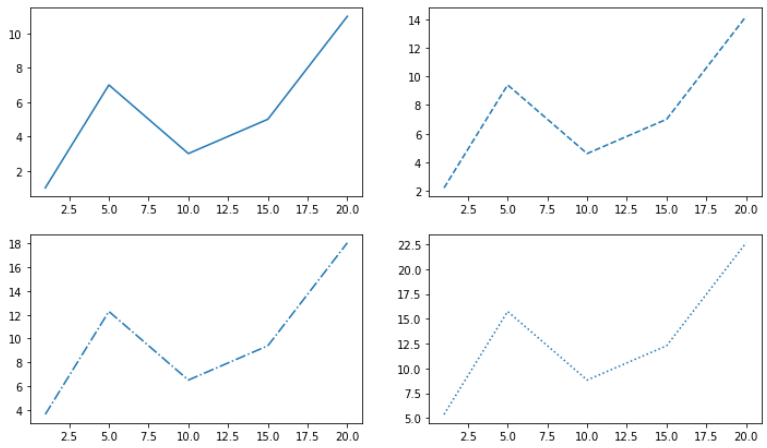


Работа с функцией subplots()

In [23]:
```
fig, axs = plt.subplots(2, 2, figsize=(12, 7))
axs[0, 0].plot(x, y1, '-')
axs[0, 1].plot(x, y2, '--')
axs[1, 0].plot(x, y3, '-.')
axs[1, 1].plot(x, y4, ':')
```

Работа с функцией subplots()

```
In [23]: fig, axs = plt.subplots(2, 2, figsize=(12, 7))
         axs[0, 0].plot(x, y1, '-')
         axs[0, 1].plot(x, y2, '--')
         axs[1, 0].plot(x, y3, '-.')
         axs[1, 1].plot(x, y4, ':')

Out[23]: [<matplotlib.lines.Line2D at 0x7e3cb1e730>]
```



3. Ответы на контрольные вопросы

1. Как осуществляется установка пакета matplotlib?

Варианты установки matplotlib

Существует два основных варианта установки этой библиотеки: в первом случае вы устанавливаете пакет Anaconda, в состав которого входит большое количество различных инструментов для работы в области машинного обучения и анализа данных (и не только); во втором – установить Matplotlib самостоятельно, используя менеджер пакетов.

Установка matplotlib через менеджер pip

Второй вариант – это воспользоваться менеджером pip и установить Matplotlib самостоятельно, для этого введите в командной строке вашей операционной системы следующие команды:

$ python -m pip install -U pip

$ python -m pip install -U matplotlib

Первая из них обновит ваш pip, вторая установит matplotlib со всеми необходимыми зависимостями.

2. Какая "магическая" команда должна присутствовать в ноутбуках Jupyter для корректного отображения графиков matplotlib?

%matplotlib inline

3. Как отобразить график с помощью функции plot ?

plt.plot(x, y)

```
plt.show()
```

4. Как отобразить несколько графиков на одном поле?

```
# Линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x
# Квадратичная зависимость
y2 = [i**2 for i in x]
# Построение графика
plt.title("Зависимости: y1 = x, y2 = x^2") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y1, y2") # ось ординат
plt.grid() # включение отображение сетки
plt.plot(x, y1, x, y2) # построение графика
```

В приведенном примере в функцию plot() последовательно передаются два массива для построения первого графика и два массива для построения второго, при этом, как вы можете заметить, для обоих графиков массив значений независимой переменной x один и то же.

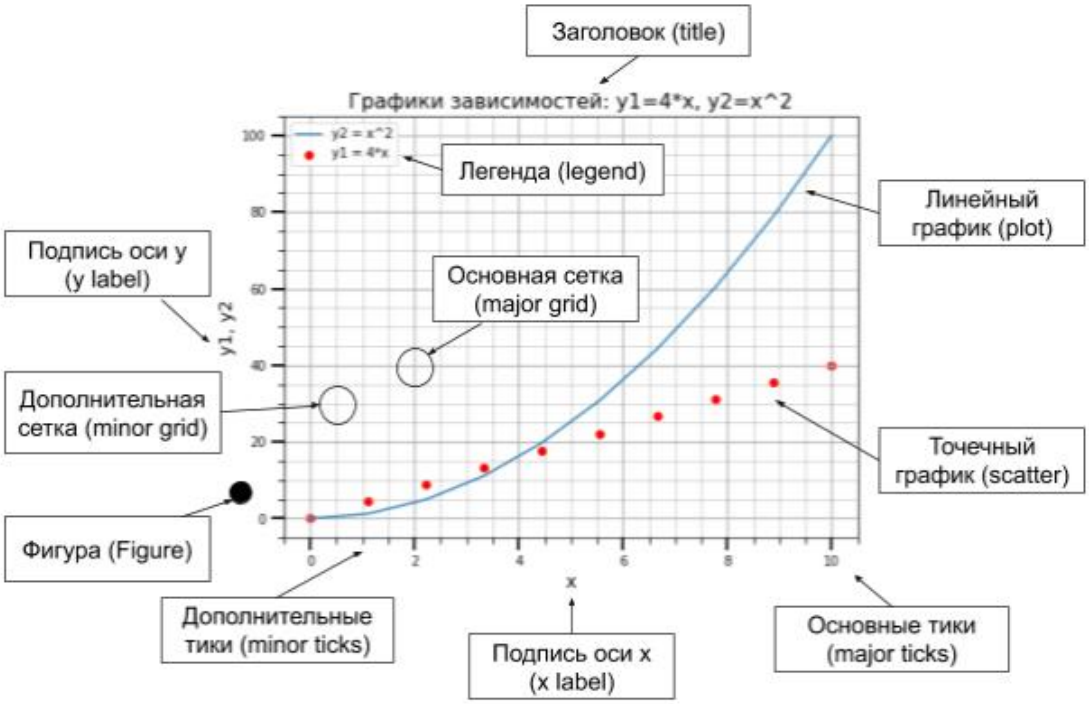
5. Какой метод Вам известен для построения диаграмм категориальных данных?

Построим диаграмму, на которой будет отображаться количество фруктов в магазине:

```
fruits = ["apple", "peach", "orange", "bannana", "melon"]
counts = [34, 25, 43, 31, 17]
plt.bar(fruits, counts)
plt.title("Fruits!")
plt.xlabel("Fruit")
plt.ylabel("Count")
```

Для вывода диаграммы мы использовали функцию bar().

6. Какие основные элементы графика Вам известны?



Корневым элементом при построения графиков в системе Matplotlib является Фигура (Figure). Все, что нарисовано на рисунке выше является элементами фигуры. Рассмотрим ее составляющие более подробно.

График

На рисунке представлены два графика – линейный и точечный. Matplotlib предоставляет огромное количество различных настроек, которые можно использовать для того, чтобы придать графику вид, который вам нужен: цвет, толщина и тип линии, стиль линии и многое другое, все это мы рассмотрим в ближайших статьях.

Оси

Вторым, после непосредственно самого графика, по важности элементом фигуры являются оси. Для каждой оси можно задать метку (подпись), основные (major) и дополнительные (minor) тики, их подписи, размер и толщину, также можно задать диапазоны по каждой из осей.

Сетка и легенда

Следующими элементами фигуры, которые значительно повышают информативность графика являются сетка и легенда. Сетка также может быть основной (major) и дополнительной (minor).

Каждому типу сетки можно задавать цвет, толщину линии и тип. Для отображения сетки и легенды используются соответствующие команды.

7. Как осуществляется управление текстовыми надписями на графике?

Наиболее часто используемые текстовые надписи на графике это:

- наименование осей;
- наименование самого графика;
- текстовое примечание на поле с графиком;
- легенда.

Рассмотрим кратко данные элементы..

Наименование осей

Для задания подписи оси x используется функция xlabel(), оси y – ylabel(). Разберемся с аргументами данных функций. Функции xlabel()/ylabel() принимают в качестве аргументов параметры конструктора класса matplotlib.text.Text. Пример использования:

plt.xlabel('Day', fontsize=15, color='blue')

Аргументов у этих функций довольно много и они позволяют достаточно тонко настроить внешний вид надписей. В рамках этого урока мы только начинаем знакомиться с инструментом pyplot поэтому не будем приводить весь список.

Заголовок графика

Для задания заголовка графика используется функция title():

plt.title('Chart price', fontsize=17)

Для функции title() также доступны параметры конструктора класса matplotlib.text.Text, часть из них представлена в описании аргументов функций xlabel() / ylabel().

Текстовое примечание

За размещение текста на поле графика отвечает функция text(), которой вначале передаются координаты позиции надписи, после этого – текст самой надписи.

plt.text(1, 1, 'type: Steel')

Легенда

Легенда будет размещена на графике, если вызвать функцию legend().

Разместим на уже знакомом нам графике необходимый набор подписей.

x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

```
plt.plot(x, y, label='steel price')
plt.title('Chart price', fontsize=15)
plt.xlabel('Day', fontsize=12, color='blue')
plt.ylabel('Price', fontsize=12, color='blue')
plt.legend()
plt.grid(True)
plt.text(15, 4, 'grow up!')
```

К перечисленным опциям мы добавили сетку, которая включается с помощью функции grid(True).

8. Как осуществляется управление легендой графика?

9. Как задать цвет и стиль линий графика?

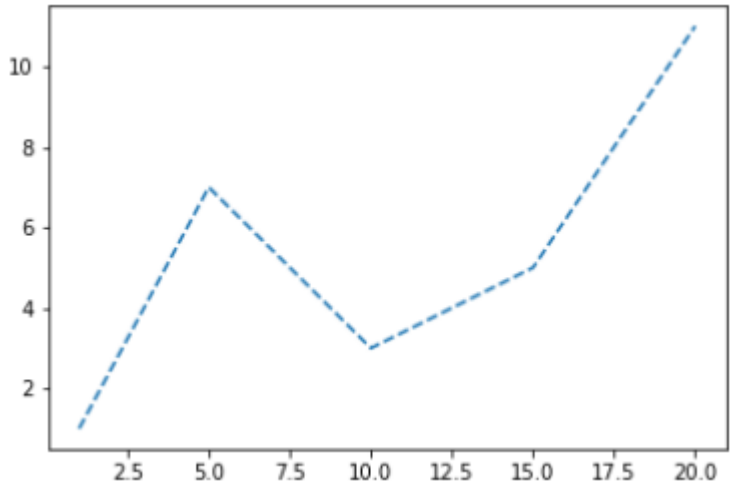
Стиль линии графика

Стиль линии графика задается через параметр linestyle, который может принимать значения из приведенной ниже таблицы.

Значение параметра	Описание
'-' или *'solid'*	Непрерывная линия
'–' или *'dashed'*	Штриховая линия
'-.' или *'dashdot'*	Штрихпунктирная линия
':' или *'dotted'*	Пунктирная линия
'None' или ' ' или ''	Не отображать линию

Стиль линии можно передать сразу после указания списков с координатами без указания, что это параметр linewidth.

```
x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--')
```



Либо можно воспользоваться функцией setp():

x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

line = plt.plot(x, y)

plt.setp(line, linestyle='--')

Цвет линии

Задание цвета линии графика производится через параметр color (или c, если использовать сокращенный вариант). Значение может быть представлено в одном из следующих форматов:

RGB или RGBA кортеж значений с плавающей точкой в диапазоне [0, 1] (пример: (0.1, 0.2, 0.3) RGB или RGBA значение в hex формате (пример: ‘#0a0a0a’)

строковое представление числа с плавающей точкой в диапазоне [0, 1] (определяет цвет в шкале серого) (пример: ‘0.7’)

символ из набора {‘b’, ‘g’, ‘r’, ‘c’, ‘m’, ‘y’, ‘k’, ‘w’}

имя цвета из палитры X11/CSS4

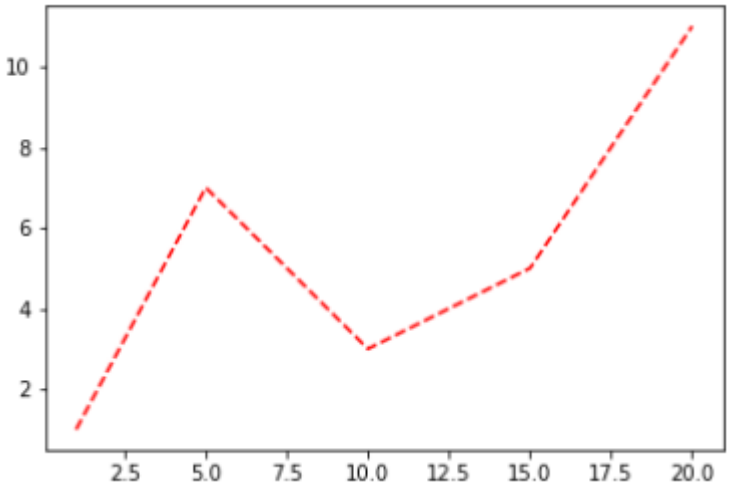
цвет из палитры xkcd(https://xkcd.com/color/rgb/), должен начинаться с префикса ‘xkcd:’

цвет из набора Tableau Color (палитра T10), должен начинаться с префикса ‘tab:’

Если цвет задается с помощью символа из набора {‘b’, ‘g’, ‘r’, ‘c’, ‘m’, ‘y’, ‘k’, ‘w’}, то он может быть совмещен со стилем линии в рамках параметра fmt функции plot().

Например штриховая красная линия будет задаваться так: ‘–r’, а штрих пунктирная зеленая так ‘-.g’

```
x = [1, 5, 10, 15, 20]

y = [1, 7, 3, 5, 11]

plt.plot(x, y, '--r')
```



10. Как выполнить размещение графика в разных полях?

Существуют три основных подхода к размещению нескольких графиков на разных полях:

использование функции subplot() для указания места размещения поля с графиком;

использование функции subplots() для предварительного задания сетки, в которую будут укладываться поля;

использование GridSpec, для более гибкого задания геометрии размещения полей с графиками в сетке.

Работа с функцией subplot()

Самый простой способ представить графики в отдельных полях – это использовать функцию subplot() для задания их мест размещения. До этого момента мы не работали с Фигурой (Figure) напрямую, значения ее параметров, задаваемые по умолчанию, нас устраивали. Для решения текущей задачи придется один из параметров – размер подложки, задать вручную. За это отвечает аргумент figsize функции figure(), которому

присваивается кортеж из двух float элементов, определяющих высоту и ширину подложки.

После задания размера, указывается местоположение, куда будет установлено поле с графиком с помощью функции subplot(). Чаще всего используют следующие варианты вызова subplot:

plt.plot(x, y, 'bx')

subplot(nrows, ncols, index)

nrows: int

Количество строк.

ncols: int

Количество столбцов.

index: int

Местоположение элемента.

subplot(pos)

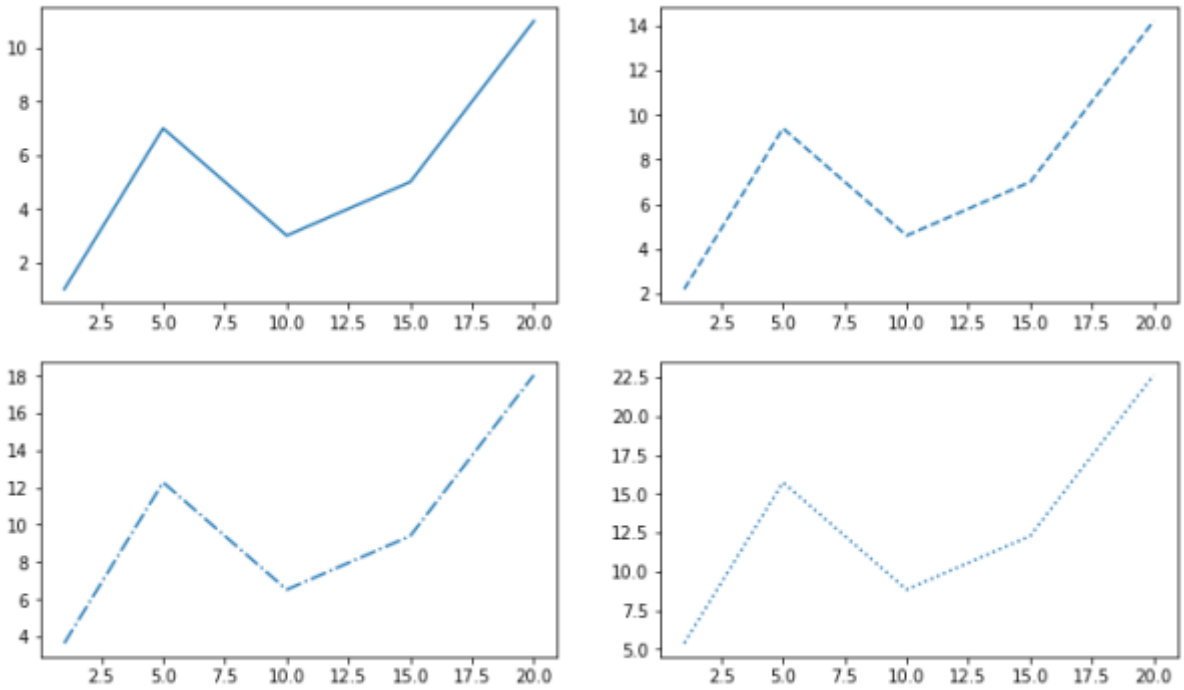
pos:int

Позиция, в виде трехзначного числа, содержащего информацию о количестве строк, столбцов и индексе, например 212, означает подготовить разметку с двумя строками и одним столбцов, элемент вывести в первую позицию второй строки. Этот вариант можно использовать, если количество строк и столбцов сетки не более 10, в ином случае лучше обратиться к первому варианту.

Рассмотрим на примере работу с данными функциями:

```
# Исходный набор данных
x = [1, 5, 10, 15, 20]
y1 = [1, 7, 3, 5, 11]
y2 = [i*1.2 + 1 for i in y1]
y3 = [i*1.2 + 1 for i in y2]
y4 = [i*1.2 + 1 for i in y3]
# Настройка размеров подложки
plt.figure(figsize=(12, 7))
# Вывод графиков
```

```
plt.subplot(2, 2, 1)
plt.plot(x, y1, '-')
plt.subplot(2, 2, 2)
plt.plot(x, y2, '--')
plt.subplot(2, 2, 3)
plt.plot(x, y3, '-.')
plt.subplot(2, 2, 4)
plt.plot(x, y4, ':')
```



Второй вариант subplot():

```
# Вывод графиков
plt.subplot(221)
plt.plot(x, y1, '-')
plt.subplot(222)
plt.plot(x, y2, '--')
plt.subplot(223)
plt.plot(x, y3, '-.')
plt.subplot(224)
```

```
plt.plot(x, y4, ':')
```

Работа с функцией subplots()

Одно из неудобств использования последовательного вызова функций subplot() заключается в том, что каждый раз приходится указывать количество строк и столбцов сетки. Для того, чтобы этого избежать, можно воспользоваться функцией subplots(), из всех ее параметров, нас пока интересуют только первые два, через них передается количество строк и столбцов сетки. Функция subplots() возвращает два объекта, первый – это Figure, подложка, на которой будут размещены поля с графиками, второй – объект или массив объектов Axes, через которые можно получить полных доступ к настройке внешнего вида отображаемых элементов.

Решим задачу вывода четырех графиков с помощью функции subplots():

```
fig, axs = plt.subplots(2, 2, figsize=(12, 7))

axs[0, 0].plot(x, y1, '-')

axs[0, 1].plot(x, y2, '--')

axs[1, 0].plot(x, y3, '-.')

axs[1, 1].plot(x, y4, ':')
```

Результат аналогичный предыдущему.