

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Основы работы с пакетом matplotlib»

ОТЧЁТ
по лабораторной работе №4
дисциплины
<<Технологии Распознавания образов>>

Выполнил:

Зиёдуллаев Жавохир

2 курс, группа ПИЖ-б-о-21-1,

011.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

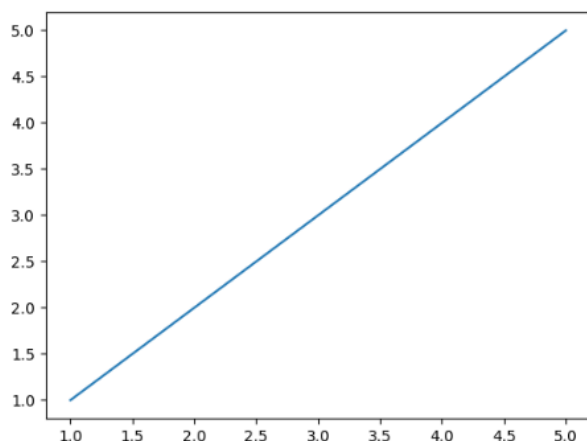
Ставрополь, 2022 г.

Проработка примеров из лабораторной работы:

Основы работы с пакетом matplotlib

Для начала создадим простейший график

```
In [2]: import matplotlib.pyplot as plt
%matplotlib inline
plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])
plt.show()
```



Стоит отметить что для работы в Jupyter Notebook с библиотекой matplotlib нужно обязательно прописывать magic команду %matplotlib inline

Рисунок 1 – Проработка примеров

Основы построения графиков

```
In [3]: 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[3]: [<matplotlib.lines.Line2D at 0x1b025d67050>]
```

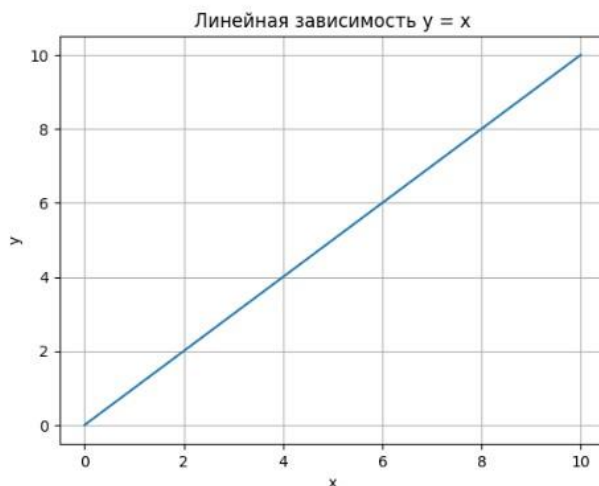


Рисунок 2 – Проработка примеров

```
In [4]: # 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[4]: [<matplotlib.lines.Line2D at 0x1b025a48f90>]
```

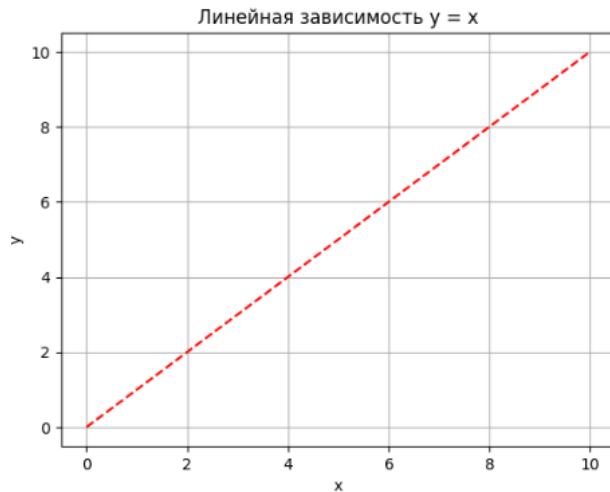


Рисунок 3 – Проработка примеров

Создание нескольких графиков на одном поле

```
In [6]: # 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[6]: [<matplotlib.lines.Line2D at 0x1b0290d76d0>,
<matplotlib.lines.Line2D at 0x1b0290e4910>]
```

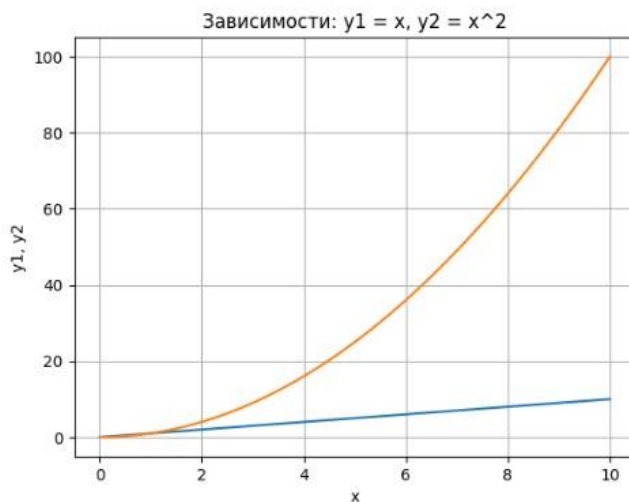


Рисунок 4 – Проработка примеров

Несколько разделенных полей с графиками

```
In [7]: # 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 отображения 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
```

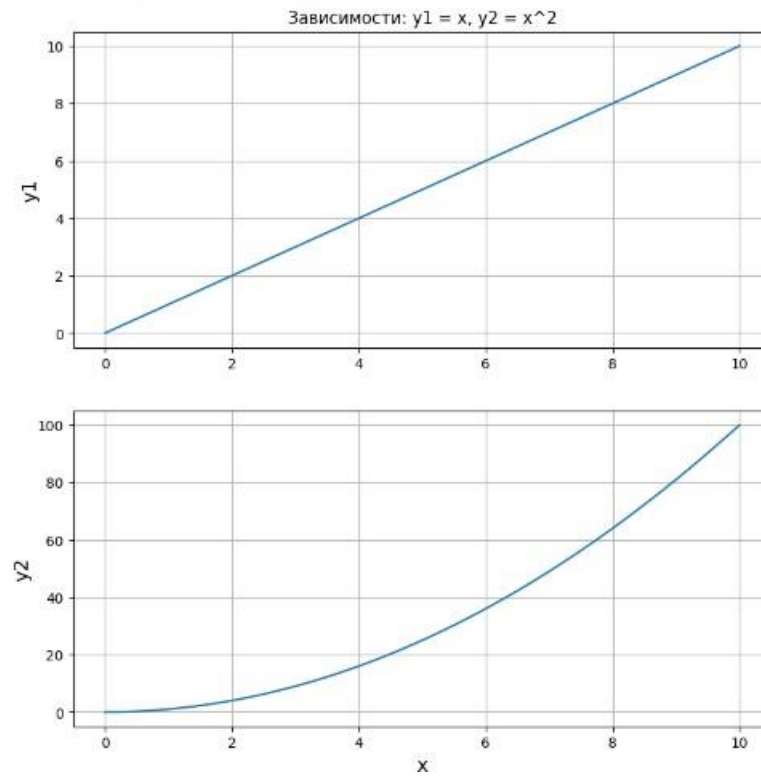


Рисунок 5 – Проработка примеров

Построение диаграммы для категориальных данных

```
In [8]: 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[8]: Text(0, 0.5, 'Count')

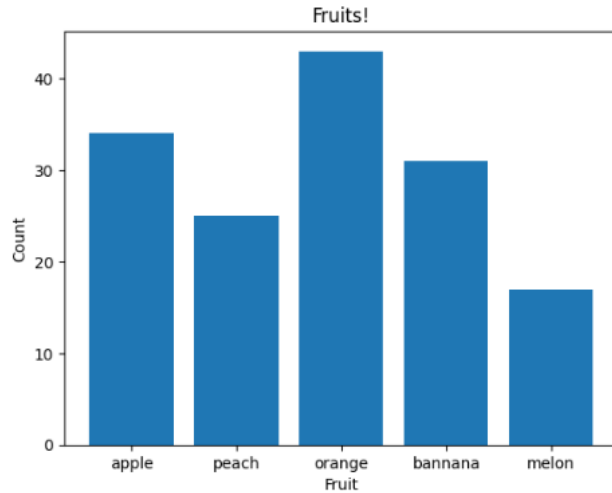


Рисунок 6 – Проработка примеров

```
In [9]: 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()
```

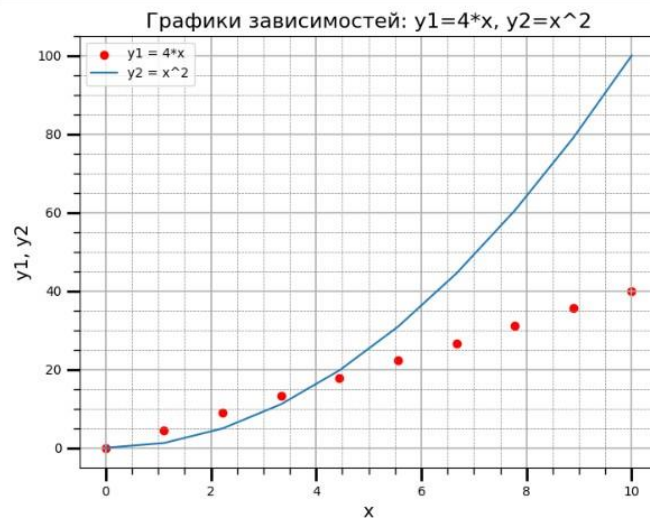


Рисунок 7 – Проработка примеров

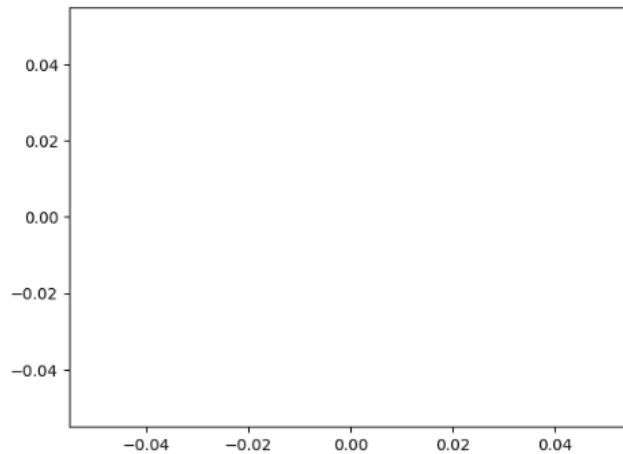
Работа с инструментом pyplot

Построение графиков

Для построения графика используется команда *plot()*

```
In [11]: plt.plot()
```

```
Out[11]: []
```

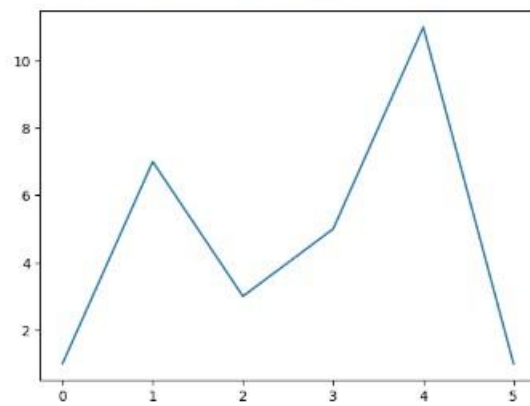


Если передать аргументом для *plot()* список, то значения из этого списка будут отложены по оси ординат (ось *y*), а по оси абсцисс (ось *x*) будут отложены индексы элементов массива:

Рисунок 8 – Проработка примеров

```
In [12]: plt.plot([1, 7, 3, 5, 11, 1])
```

```
Out[12]: [<matplotlib.lines.Line2D at 0x1b02920e218>]
```



Для того чтобы задать значения по осям x и y необходимо указать 2 списка

```
In [13]: plt.plot([1, 5, 10, 15, 20], [1, 7, 3, 5, 11])
```

```
Out[13]: [<matplotlib.lines.Line2D at 0x1b025cd7898>]
```

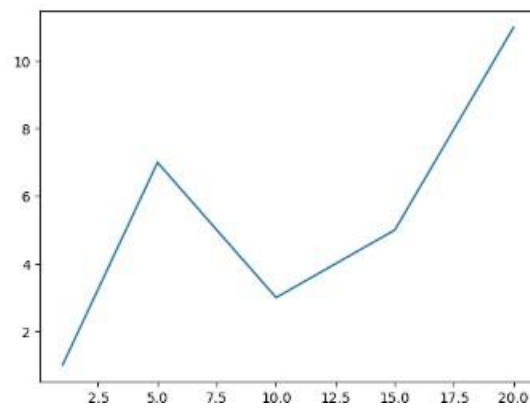


Рисунок 9 – Проработка примеров

Текстовые надписи на графике

Наименование осей

Для задания подписи оси x используется функция `xlabel()`, оси y – `ylabel()`

Функции `xlabel()`/`ylabel()` принимают в качестве аргументов параметры конструктора класса `matplotlib.text.Text`. Пример использования:

```
In [14]: plt.xlabel('Day', fontsize=15, color='blue')
```

```
Out[14]: Text(0.5, 0, 'Day')
```

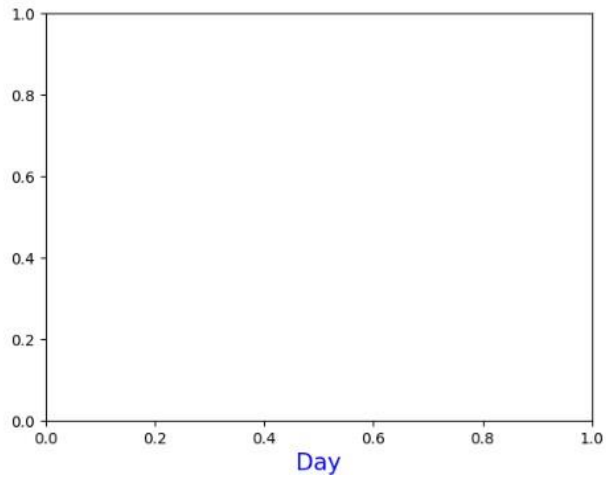


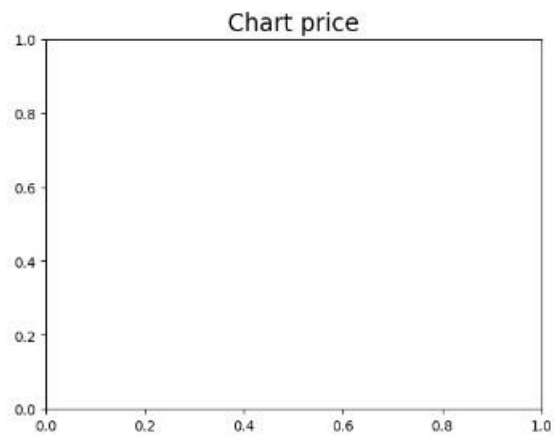
Рисунок 10 – Проработка примеров

Заголовок графика

Для задания заголовка графика используется функция `title()`:

```
In [15]: plt.title('Chart price', fontsize=17)
```

```
Out[15]: Text(0.5, 1.0, 'Chart price')
```



Текстовое примечание

За размещение текста на поле графика отвечает функция `text()`

```
In [16]: plt.text(1, 1, 'type: Steel')
```

```
Out[16]: Text(1, 1, 'type: Steel')
```

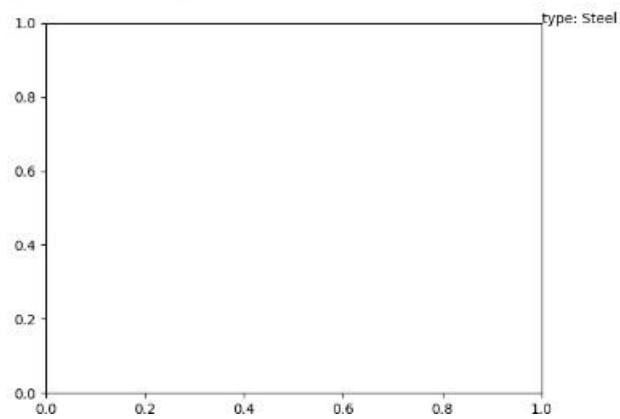


Рисунок 11 – Проработка примеров

Легенда

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

Создадим график с использованием выше указанных функций

```
In [17]: 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[17]: Text(15, 4, 'grow up!')
```

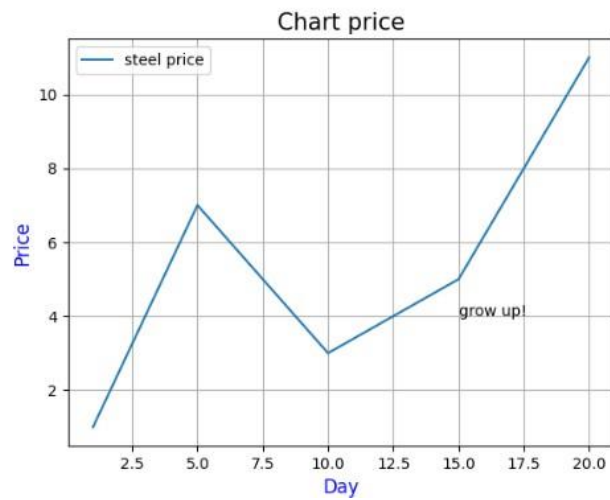


Рисунок 12 – Проработка примеров

Работа с линейным графиком

Параметры, которые отвечают за отображение графика можно задать непосредственно в самой функции `plot()`

```
In [21]: plt.plot(x, y, color='red')
```

```
Out[21]: [<matplotlib.lines.Line2D at 0x1b02ad88950>]
```

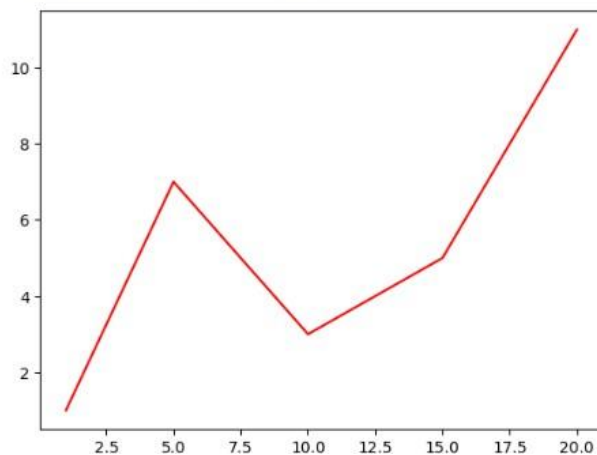


Рисунок 13 – Проработка примеров

Стиль линии графика

Стиль линии графика задается через параметр *linestyle*, который может принимать различные значения

```
In [22]: x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
plt.plot(x, y, '--')
```

```
Out[22]: [<matplotlib.lines.Line2D at 0x1b02ad0c8d0>]
```

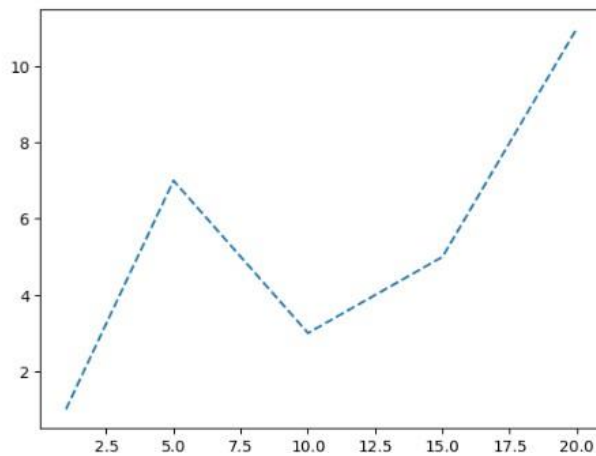
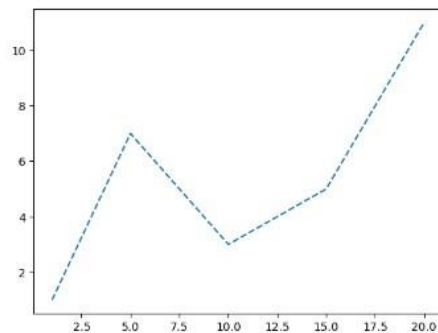


Рисунок 14 – Проработка примеров

```
In [23]: x = [1, 5, 10, 15, 20]
y = [1, 7, 3, 5, 11]
line = plt.plot(x, y)
plt.setp(line, linestyle='--')
```

```
Out[23]: [None]
```



```
In [24]: 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[24]: [<matplotlib.lines.Line2D at 0x1b02ae12f00>,
<matplotlib.lines.Line2D at 0x1b02ae1c2d0>,
<matplotlib.lines.Line2D at 0x1b02ae44a90>,
<matplotlib.lines.Line2D at 0x1b02ae47900>]
```

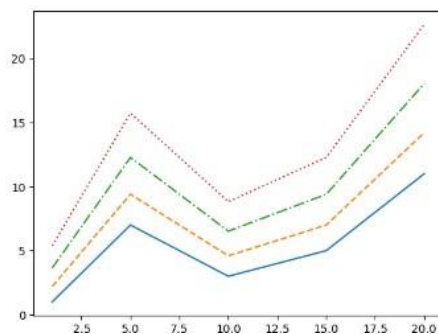


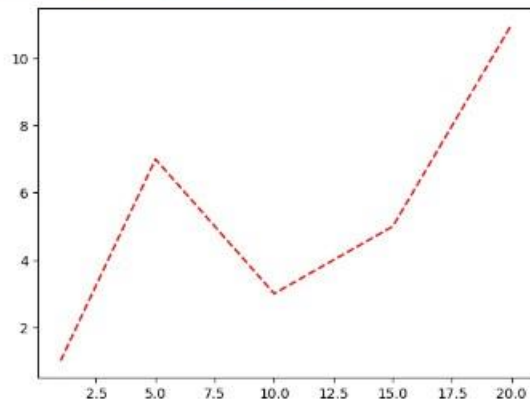
Рисунок 15 – Проработка примеров

Цвет линии

Задание цвета линии графика производится через параметр *color*

```
In [25]: x = [1, 5, 10, 15, 20]  
y = [1, 7, 3, 5, 11]  
plt.plot(x, y, '--r')
```

```
Out[25]: [
```



```
In [26]: x = [1, 5, 10, 15, 20]  
y = [1, 7, 3, 5, 11]  
plt.plot(x, y, '--g')
```

```
Out[26]: [
```

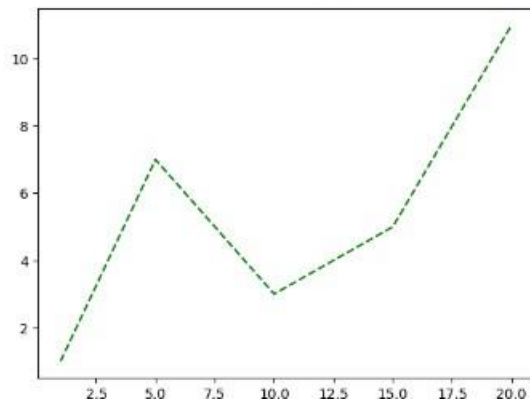


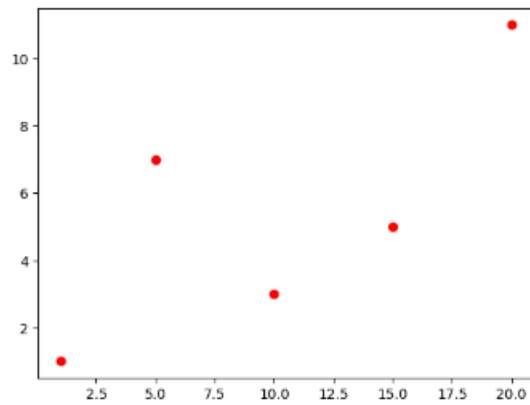
Рисунок 16 – Проработка примеров

Тип графика

Примеры различных видов графиков:

```
In [27]: plt.plot(x, y, 'ro')
```

```
Out[27]: [<matplotlib.lines.Line2D at 0x1b02aee0d50>]
```



```
In [28]: plt.plot(x, y, 'bx')
```

```
Out[28]: [<matplotlib.lines.Line2D at 0x1b02bcb5790>]
```

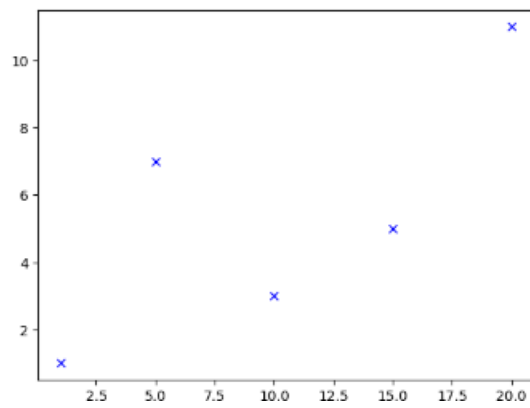


Рисунок 17 – Проработка примеров

Размещение графиков на разных полях

Рассмотрим размещение графиков при помощи функций `subplot()` и `subplots()`

Работа с функцией `subplot()`

```
In [29]: # 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[29]: [<matplotlib.lines.Line2D at 0x1b02bfac2d0>]
```

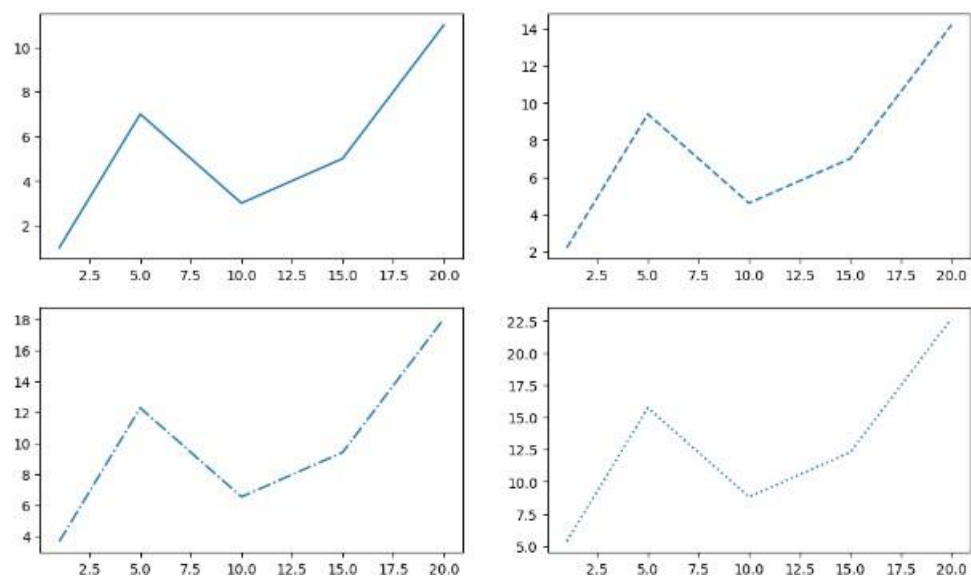


Рисунок 18 – Проработка примеров

Работа с функцией subplots()

```
In [30]: 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[30]: [<matplotlib.lines.Line2D at 0x1b02c3049d0>]
```

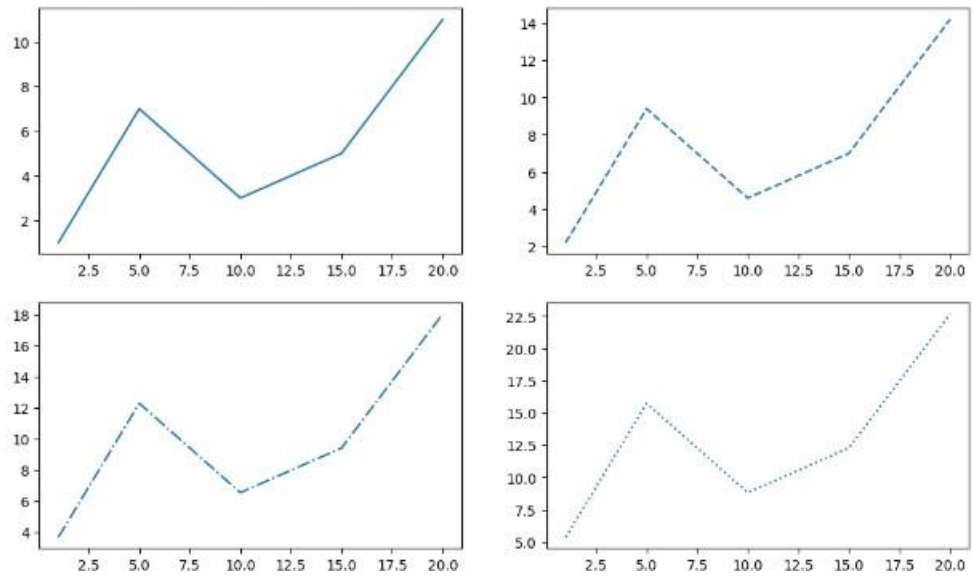


Рисунок 19 – Проработка примеров

Контрольные вопросы

1. Как осуществляется установка пакета 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.

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
```

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

```
import matplotlib.pyplot as plt  
%matplotlib inline
```

3. Как отобразить график с помощью функции plot ?

```
import matplotlib.pyplot as plt  
plt.plot([1, 2, 3, 4, 5], [1, 2, 3, 4, 5])  
plt.show()
```

4. Как отобразить несколько графиков на одном поле?

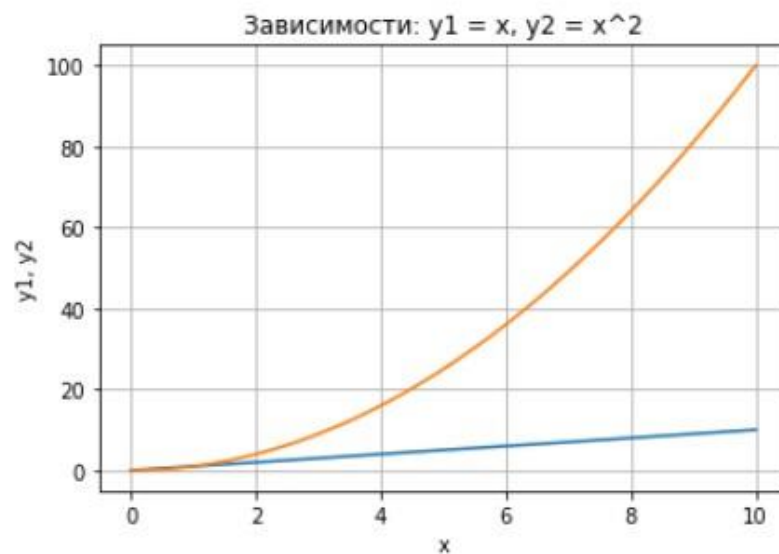
Построим несколько графиков на одном поле, для этого добавим квадратичную зависимость:

```
# линейная зависимость
x = np.linspace(0, 10, 50)
y1 = x

# квадратичная зависимость
y2 = [i**2 for i in x]
```

```
# Построение графика
plt.title("Зависимости:  $y_1 = x$ ,  $y_2 = x^2$ ") # заголовок
plt.xlabel("x") # ось абсцисс
plt.ylabel("y1, y2") # ось ординат
plt.grid() # включение отображения сетки

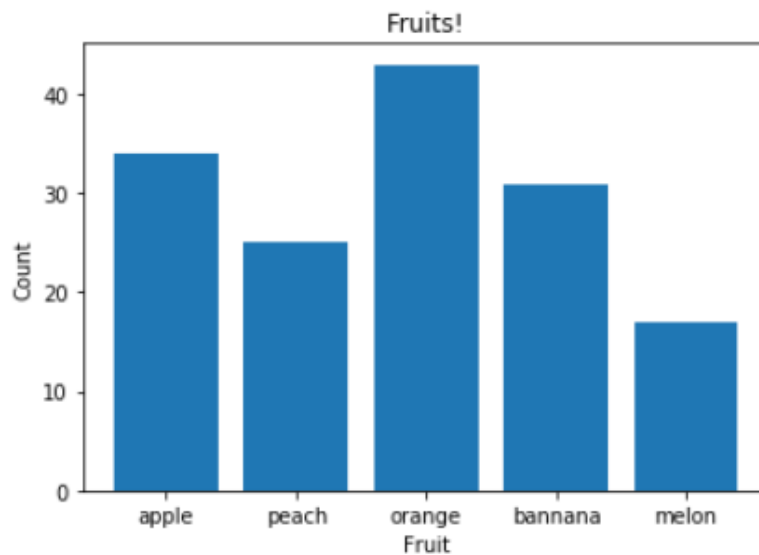
plt.plot(x, y1, x, y2) # построение графика
```



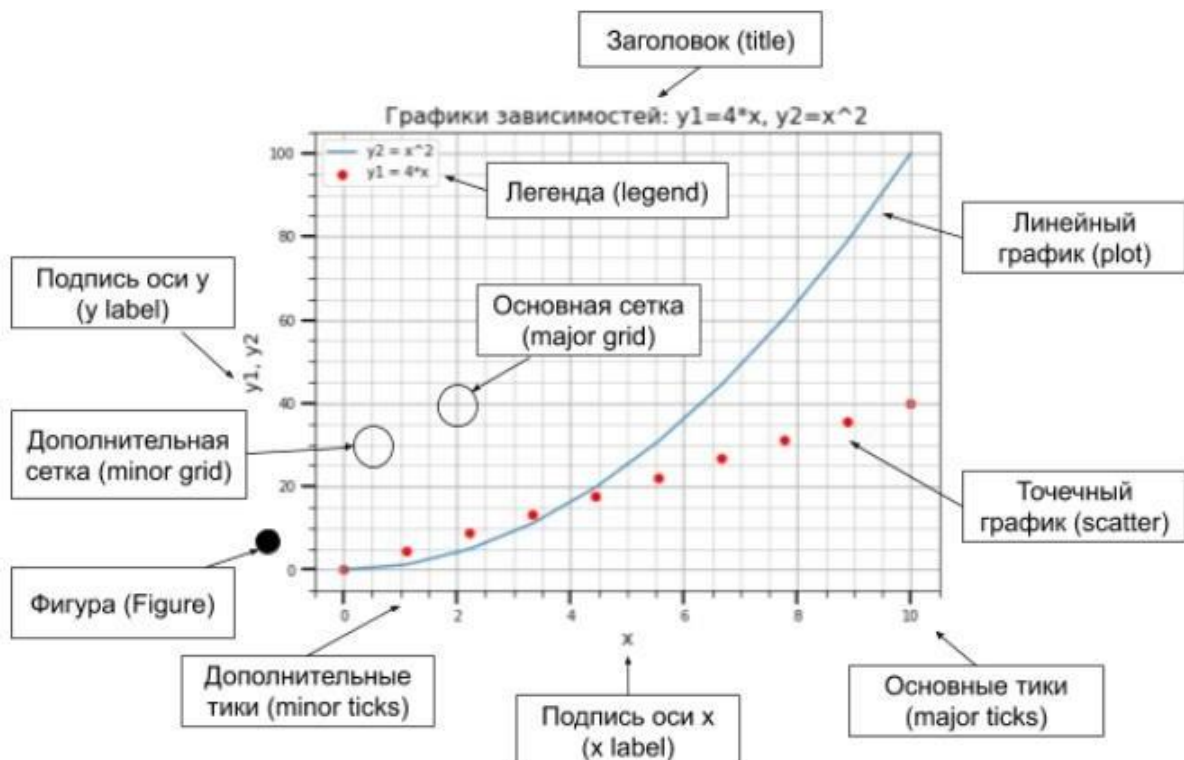
5. Какой метод Вам известен для построения диаграмм категориальных данных?

```
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")
```



6. Какие основные элементы графика Вам известны?



7. Как осуществляется управление текстовыми надписями на графике?

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')
```

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')
```

8. Как осуществляется управление легендой графика?

Легенда

Легенда будет размещена на графике, если вызвать функцию *legend()*, в рамках данного урока мы не будем рассматривать аргументы этой функции.

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!')
```

9. Как задать цвет и стиль линий 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, '--')
```

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

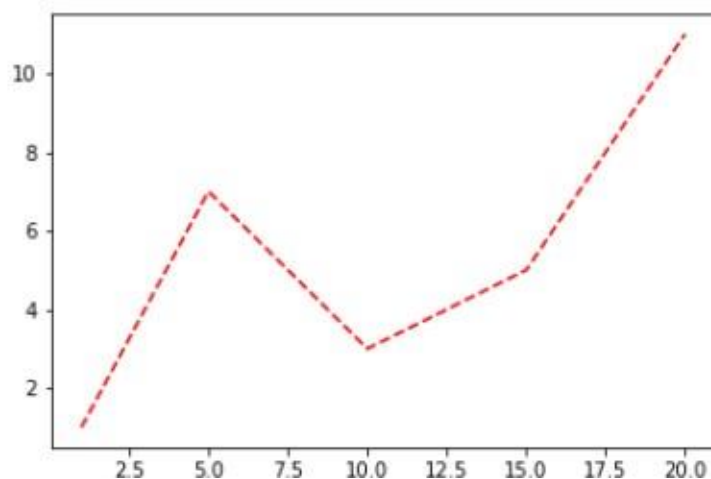
- *RGB* или *RGBA* кортеж значений с плавающей точкой в диапазоне [0, 1] (пример: (0.1, 0.2, 0.3))
- *RGB* или *RGBA* значение в *hex* формате (пример: '#0a0a0a')
- строковое представление числа с плавающей точкой в диапазоне [0, 1] (определяет цвет в шкале серого) (пример: '0.7')
- символ из набора {'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?

Размещение графиков на 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.

В этом уроке будут рассмотрены первые два подхода.

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`:

`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]

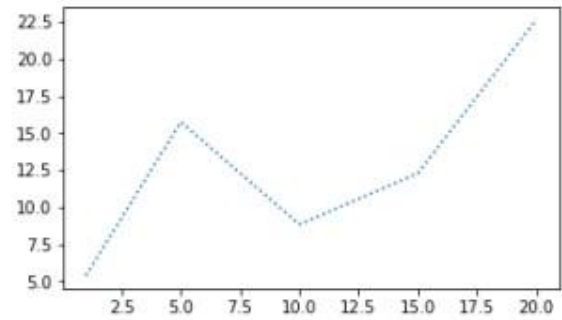
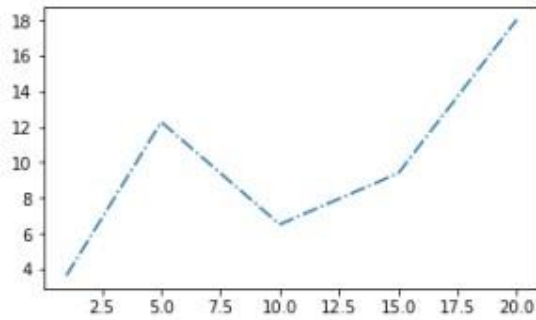
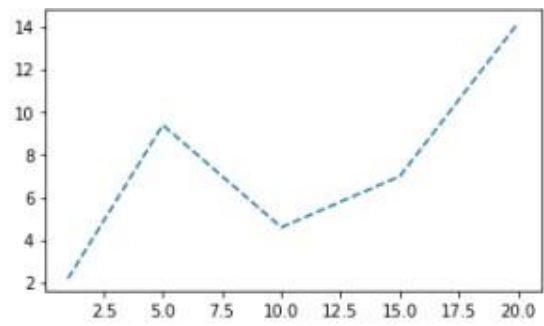
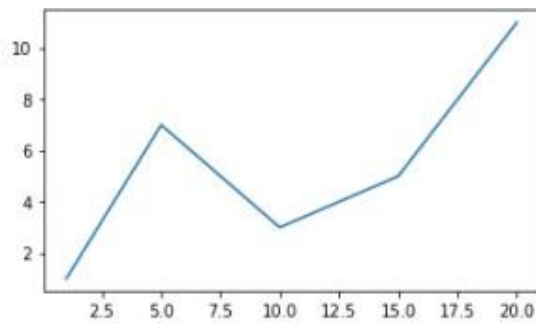
# Настройка размеров подложки
plt.figure(figsize=(12, 7))

# 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 использования `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, ':')
```

Результат функции `subplot()`