Лабораторная работа 3

Библиотека `matplotlib`

Есть несколько пакетов для построения графиков. Один из наиболее популярных — `matplotlib`. Графики можно также сохранять в файлы, как в векторных форматах (`eps`, `pdf`, `svg`), так и в растровых (`png`, `jpg`). `matplotlib` позволяет строить двумерные графики практически всех нужных типов, с достаточно гибкой регулировкой их параметров; он также поддерживает основные типы трёхмерных графиков, но для серьёзной трёхмерной визуализации данных лучше пользоваться более мощными специализированными системами.

Некоторые функции отрисовки

- `plt.scatter(x, y, params)` — нарисовать точки с координатами из xx по горизонтальной оси и из yy по вертикальной оси;
- `plt.plot(x, y, params)` — нарисовать график по точкам с координатами из xx по горизонтальной оси и из yy по вертикальной оси. Точки будут соединятся в том порядке, в котором они указаны в этих массивах;
- `plt.fill_between(x, y1, y2, params)` — закрасить пространство между y_1y1 и y_2y2 по координатам из xx;
- `plt.pcolormesh(x1, x1, y, params)` — закрасить пространство в соответствии с интенсивностью yy;
- `plt.contour(x1, x1, y, lines)` — нарисовать линии уровня. Затем нужно применить `plt.clabel`.

Вспомогательные функции

- `plt.figure(figsize=(x, y))` — создать график размера (x,y)(x,y);
- `plt.show()` — показать график;
- `plt.subplot(...)` — добавить подграфик;
- `plt.xlim(x_min, x_max)` — установить пределы графика по горизонтальной оси;
- `plt.ylim(y_min, y_max)` — установить пределы графика по вертикальной оси;
- `plt.title(name)` — установить имя графика;
- `plt.xlabel(name)` — установить название горизонтальной оси;
- `plt.ylabel(name)` — установить название вертикальной оси;
- `plt.legend(loc=...)` — сделать легенду в позиции loc;
- `plt.grid()` — добавить сетку на график;
- `plt.savefig(filename)` — сохранить график в файл.

http://matplotlib.org/gallery.html — тысячи примеров

У функций в `matplotlib` много параметров. Для того, чтобы посмотреть все параметры, можно воспользоваться справкой, например,

`plt.plot?`

1. Простые графики

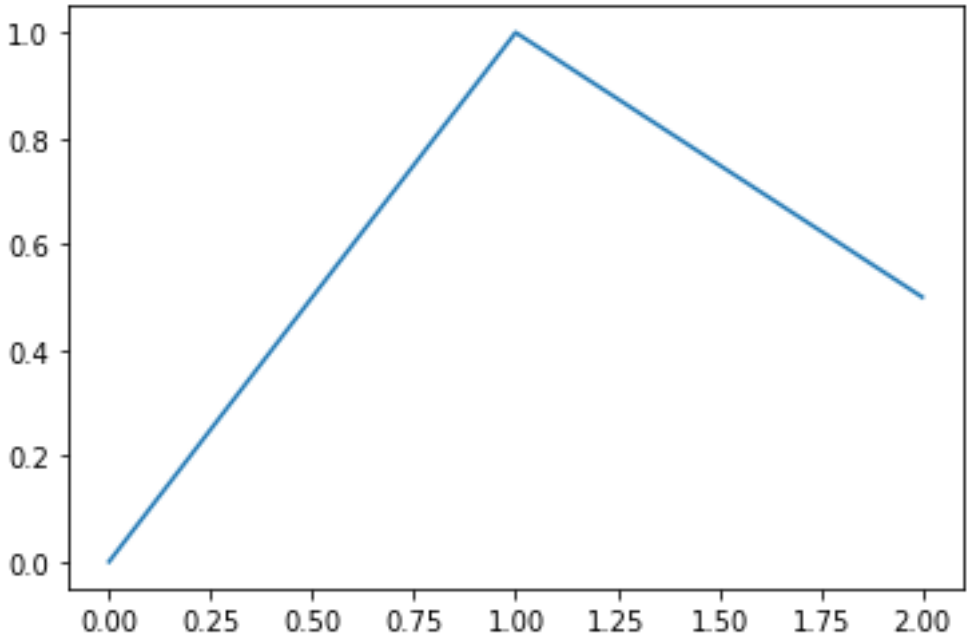
In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

Рисуем график с помощью списка yy-координат; xx-координаты образуют последовательность 0, 1, 2, ...

In [2]:

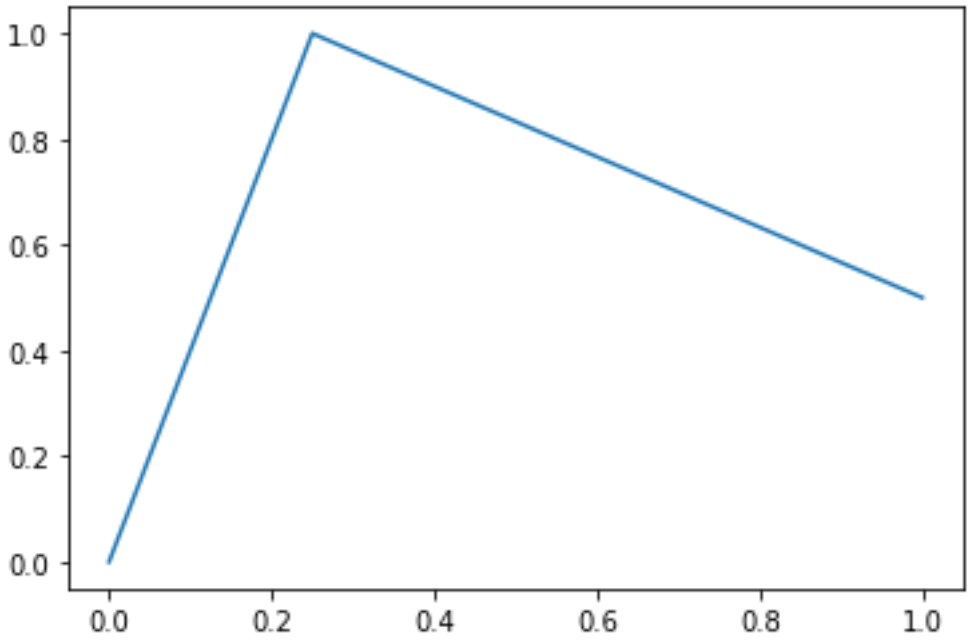
```
plt.figure()
plt.plot([0, 1, 0.5])
plt.show()
```



Списки xx и yy координат точек. Точки соединяются прямыми, т.е. строится ломаная линия.

In [3]:

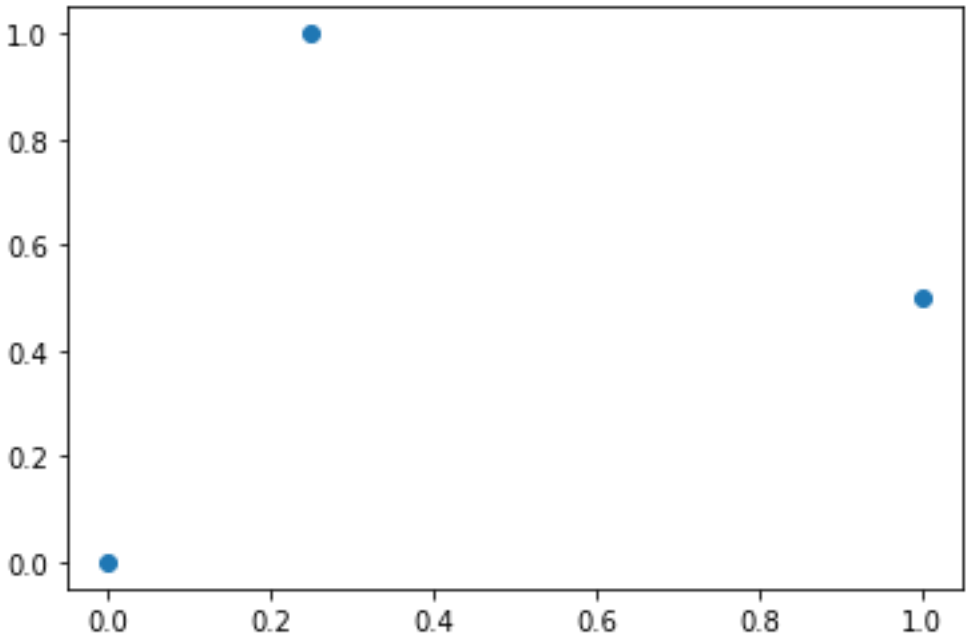
```
plt.figure()
plt.plot([0, 0.25, 1], [0, 1, 0.5])
plt.show()
```



Функция `scatter` просто рисует точки, не соединяя из линиями.

In [4]:

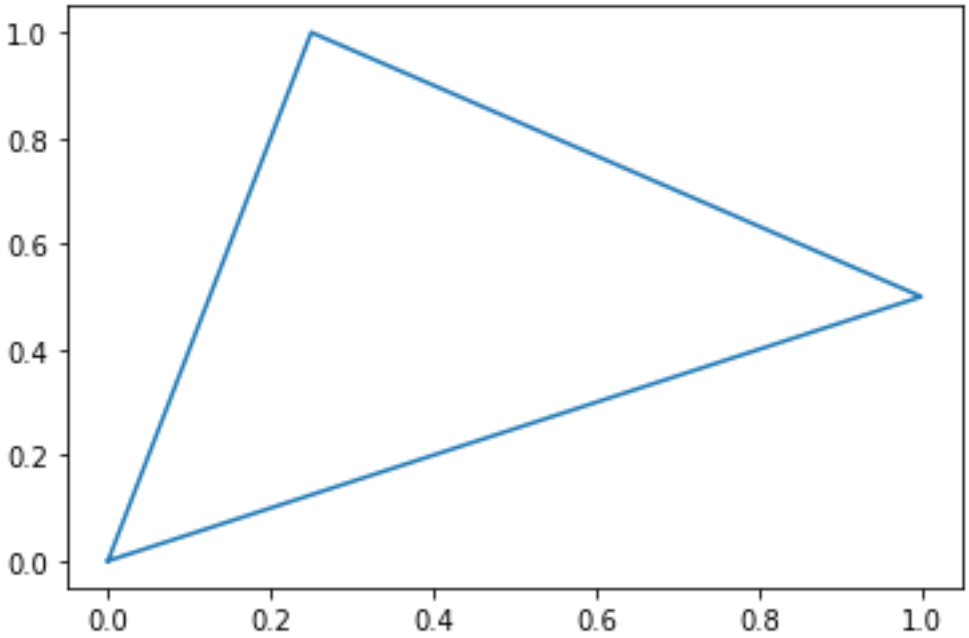
```
plt.figure()
plt.scatter([0, 0.25, 1], [0, 1, 0.5])
plt.show()
```



xx-координаты не обязаны монотонно возрастать. Тут, например, мы строим замкнутый многоугольник.

In [5]:

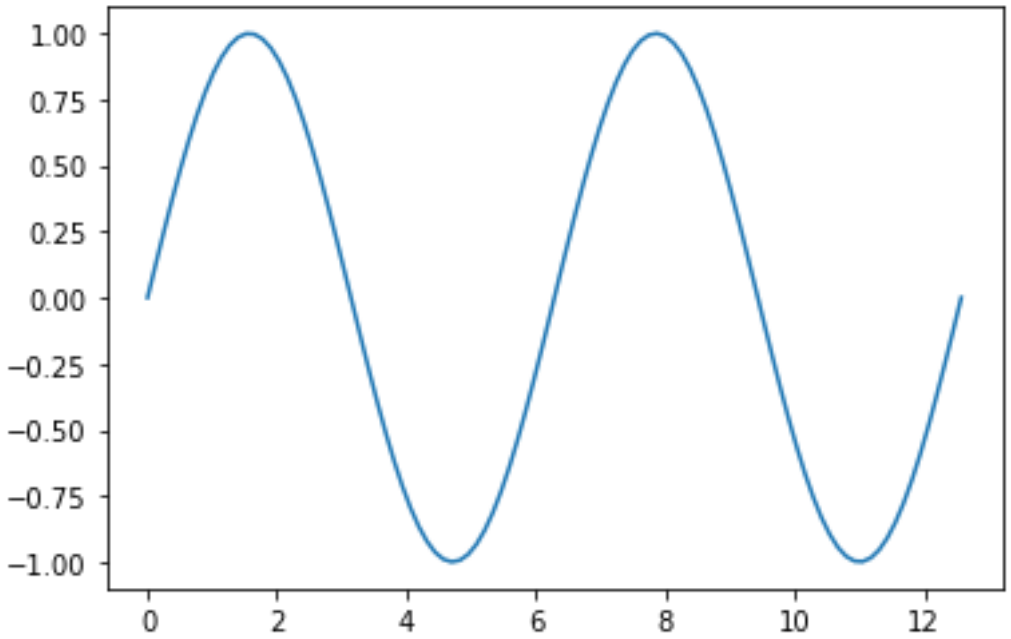
```
plt.figure()
plt.plot([0, 0.25, 1, 0], [0, 1, 0.5, 0])
plt.show()
```



Когда точек много, ломаная неотличима от гладкой кривой.

In [6]:

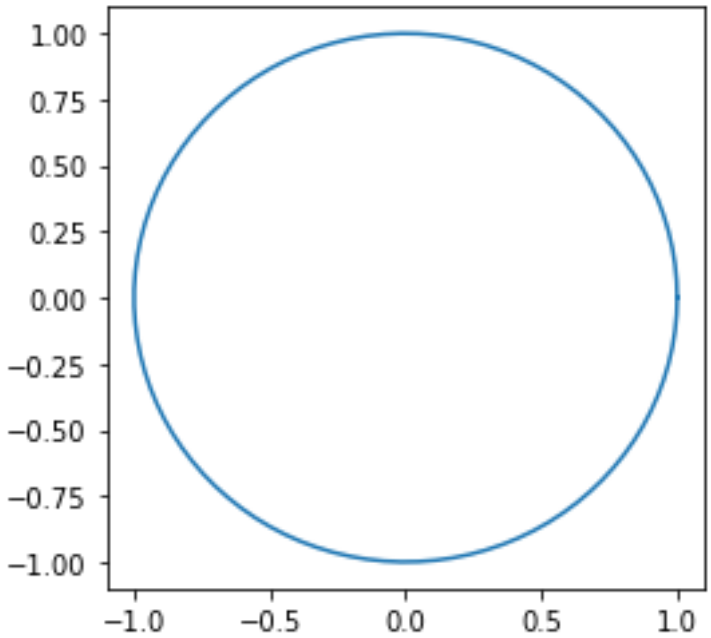
```
x = np.linspace(0, 4 * np.pi, 100)

plt.figure()
plt.plot(x, np.sin(x))
plt.show()
```



Массив xx не обязан быть монотонно возрастающим. Можно строить любую параметрическую линию $x=x(t)$x=x(t), $y=y(t)$y=y(t).

In [7]:

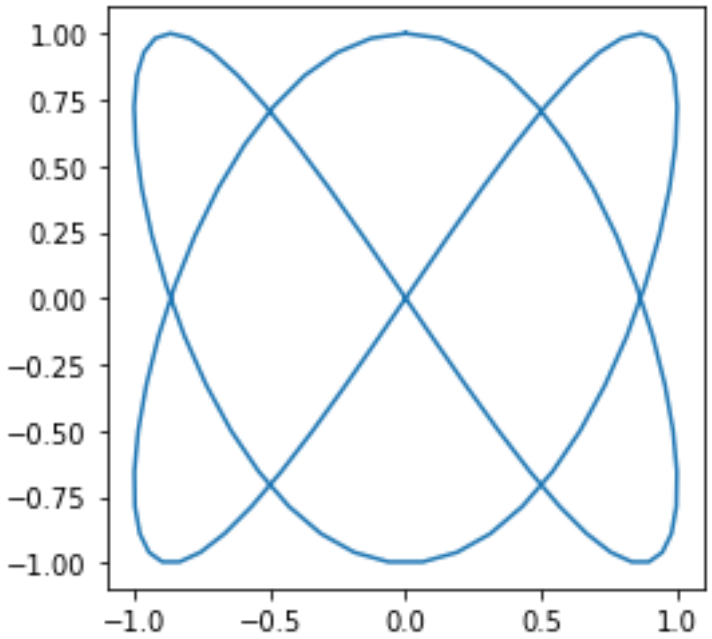
```
t = np.linspace(0, 2 * np.pi, 100)

plt.figure(figsize=(4,4))
plt.plot(np.cos(t), np.sin(t))
plt.show()
```



А вот одна из фигур Лиссажу.

In [8]:

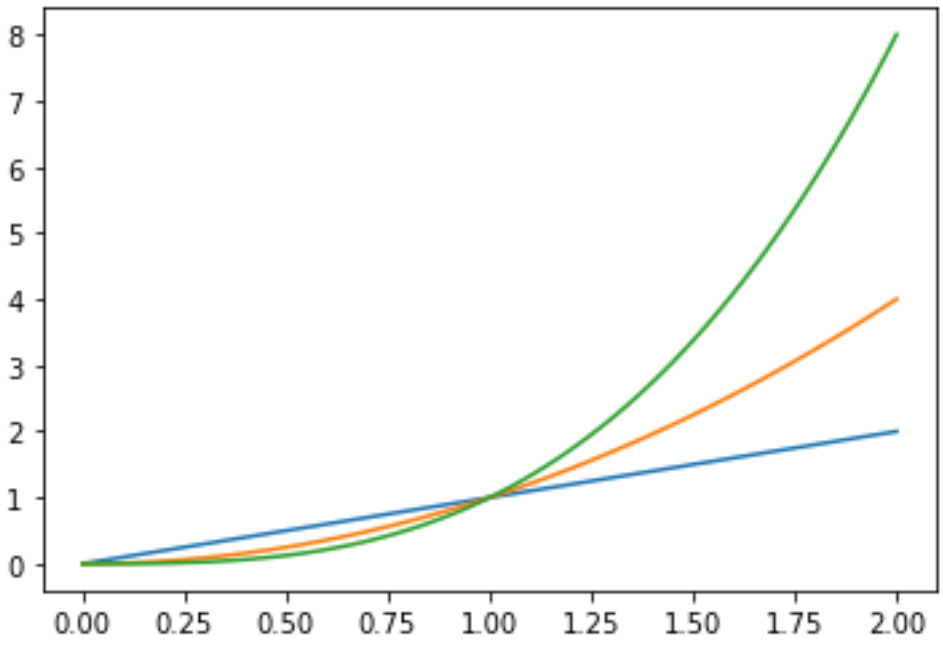
```
plt.figure(figsize=(4,4))
plt.plot(np.sin(2 * t), np.cos(3 * t))
plt.show()
```



Несколько кривых на одном графике. Каждая задаётся парой массивов — xx и yy координаты. По умолчанию, им присваиваются цвета из некоторой последовательности цветов; разумеется, их можно изменить. Вообще говоря, подобным кодом не стоит пользоваться.

In [9]:

```
x = np.linspace(0, 2, 100)

plt.figure()
plt.plot(x, x, x**2, x, x**3)
plt.show()
```

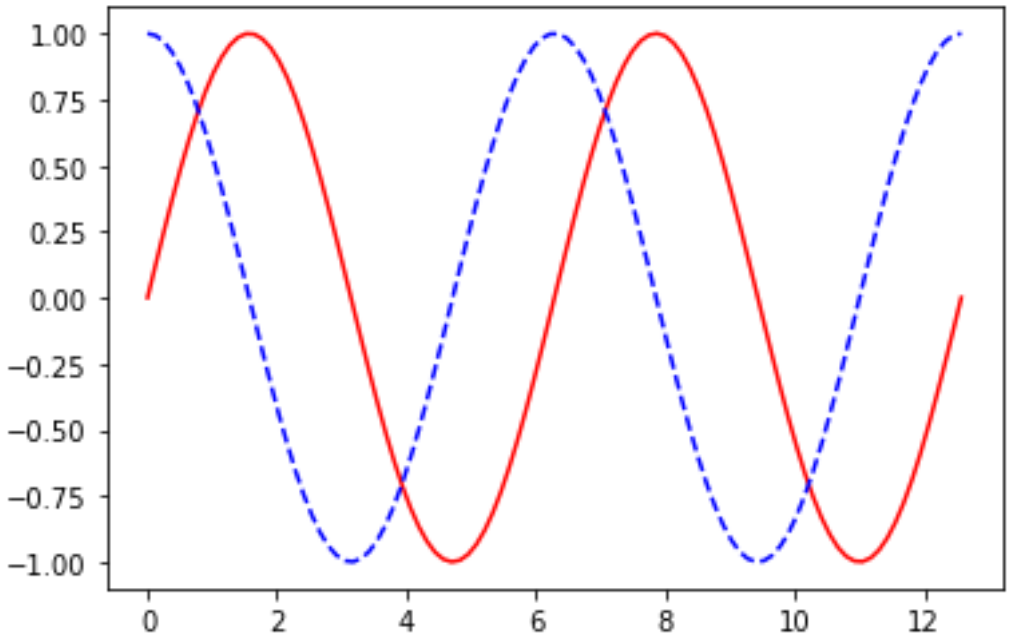


Для простой регулировки цветов и типов линий после пары xx и yy координат вставляется форматная строка. Первая буква определяет цвет (`'r'` — красный, `'b'` — синий и т.д.), дальше задаётся тип линии (`'-'` — сплошная, `'--'` — пунктирная, `'-.'` — штрих-пунктирная и т.д.).

In [10]:

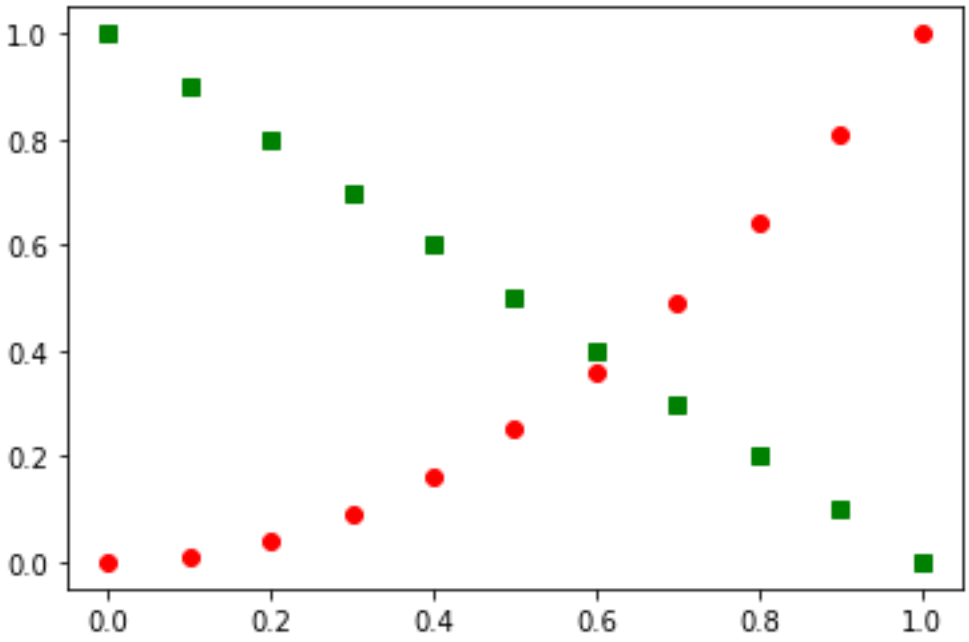
```
x = np.linspace(0, 4 * np.pi, 100)
```

```
plt.figure()
plt.plot(x, np.sin(x), 'r-')
plt.plot(x, np.cos(x), 'b--')
plt.show()
```



Если в качестве "типа линии" указано `'o'`, то это означает рисовать точки кружочками и не соединять их линиями; аналогично, `'s'` означает квадратики. Конечно, такие графики имеют смысл только тогда, когда точек не очень много.

```
In [11]:
x = np.linspace(0, 1, 11)

plt.figure()
plt.plot(x, x ** 2, 'ro')
plt.plot(x, 1 - x, 'gs')
plt.show()
```

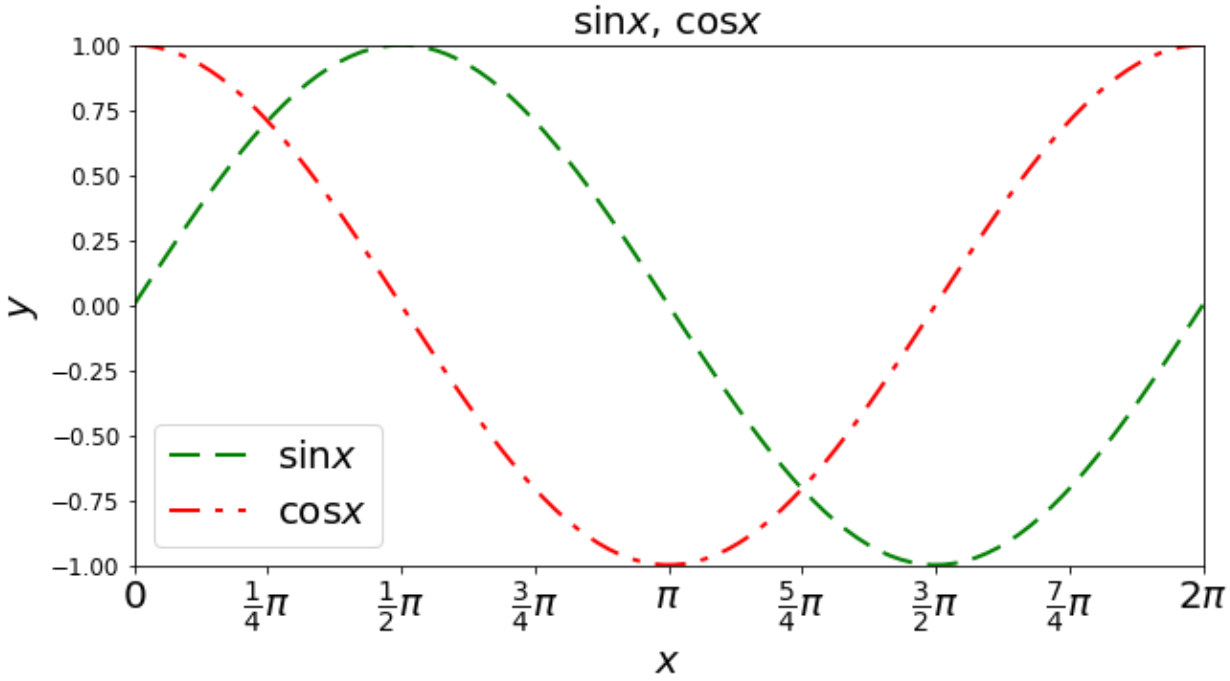


Вот пример настройки почти всего, что можно настроить. Можно задать последовательность засечек на оси xx (и yy) и подписи к ним, в которых, как и в других текстах.. Задать подписи

осей xx и yy и заголовок графика. Во всех текстовых элементах можно задать размер шрифта. Можно задать толщину линий и штрихи. В примере ниже на графике косинуса рисуется штрих длины 8, потом участок длины 4 не рисуется, потом участок длины 2 рисуется, потом участок длины 4 опять не рисуется, и так по циклу; поскольку толщина линии равна 2, эти короткие штрихи длины 2 фактически выглядят как точки. Можно задать подписи к кривым (legend); где разместить эти подписи тоже можно регулировать.

In [12]:

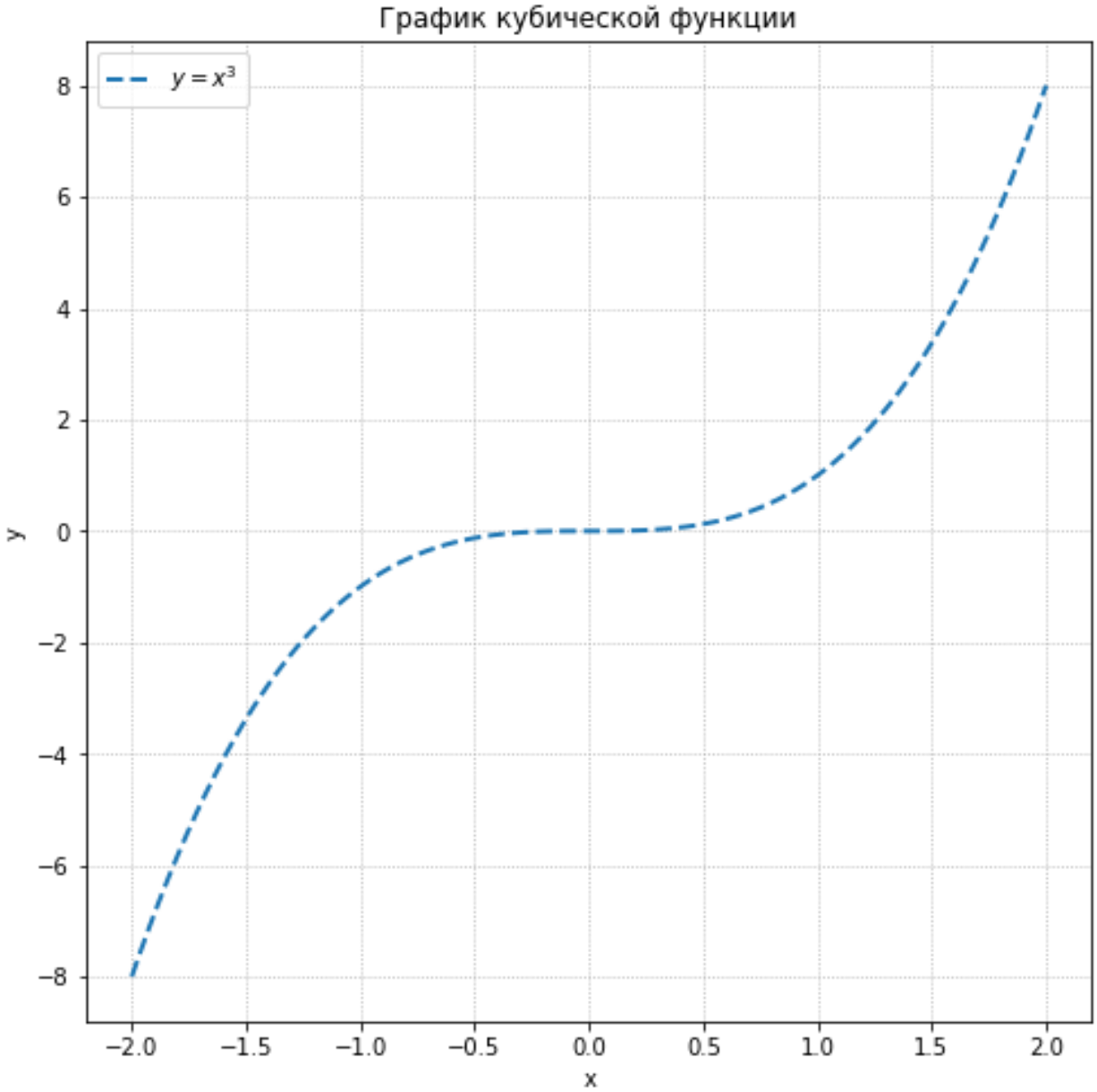
```
x = np.linspace(0, 2 * np.pi, 100)

plt.figure(figsize=(10, 5))
plt.plot(x, np.sin(x), linewidth=2, color='g', dashes=[8, 4], label=r'$\sin x$')
plt.plot(x, np.cos(x), linewidth=2, color='r', dashes=[8, 4, 2, 4], label=r'$\cos x$')
plt.axis([0, 2 * np.pi, -1, 1])
plt.xticks(np.linspace(0, 2 * np.pi, 9),  # Где сделать отметки
       ('0',r'$\frac{1}{4}\pi$',r'$\frac{1}{2}\pi$',  # Как подписать
        r'$\frac{3}{4}\pi$',r'$\pi$',r'$\frac{5}{4}\pi$',
        r'$\frac{3}{2}\pi$',r'$\frac{7}{4}\pi$',r'$2\pi$'),
       fontsize=20)
plt.yticks(fontsize=12)
plt.xlabel(r'$x$', fontsize=20)
plt.ylabel(r'$y$', fontsize=20)
plt.title(r'$\sin x$, $\cos x$', fontsize=20)
plt.legend(fontsize=20, loc=0)
plt.show()
```



In [13]:

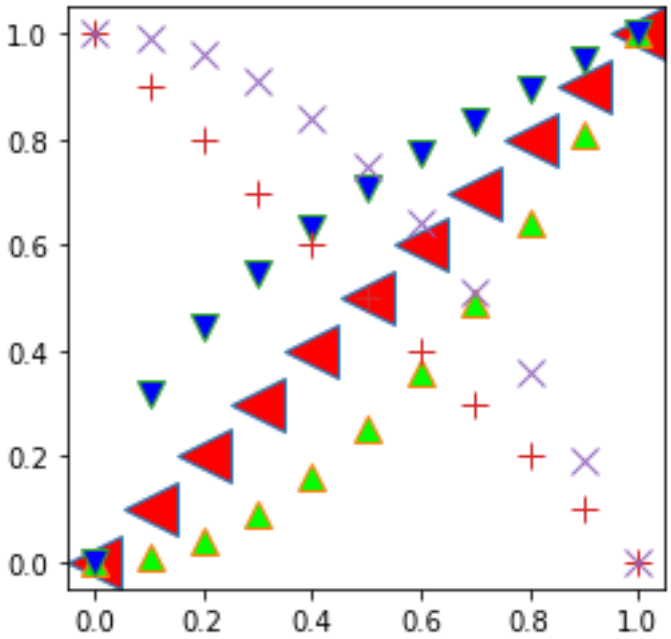
```
x = np.linspace(-2, 2, 100)

plt.figure(figsize=(8, 8))
plt.plot(x, x**3, linestyle='--', lw=2, label='$y=x^3$')
plt.xlabel('x'), plt.ylabel('y')
plt.legend()
plt.title('График кубической функции')
```

```
plt.grid(ls=':')
plt.show()
```



Если `linestyle=''`, то точки не соединяются линиями. Сами точки рисуются маркерами разных типов. Тип определяется строкой из одного символа, который чем-то похож на нужный маркер. В добавок к стандартным маркерам, можно определить самодельные.

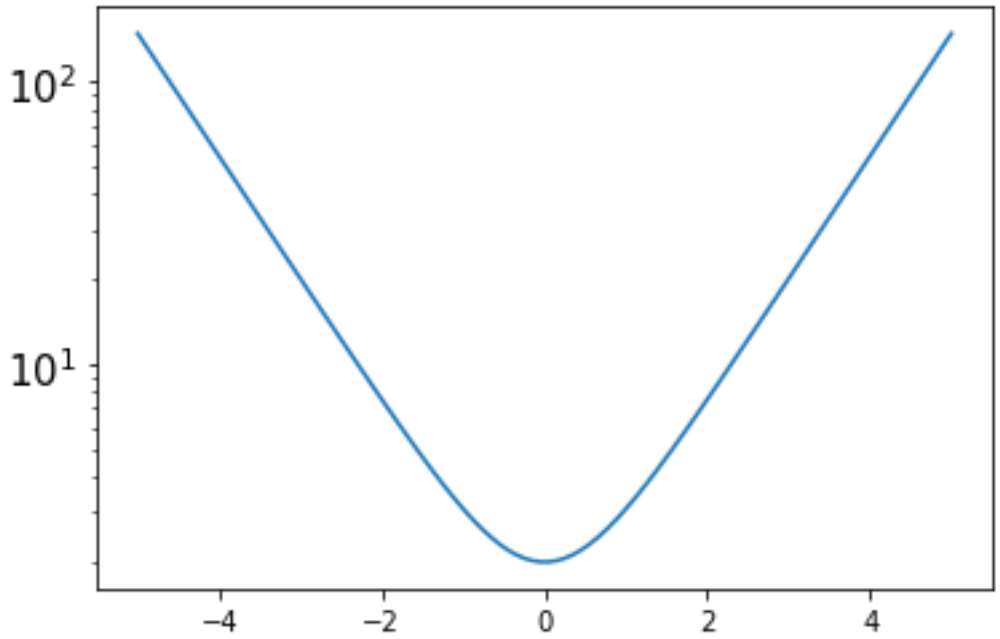
In [14]:
```
x = np.linspace(0, 1, 11)

plt.figure(figsize=(4, 4))
plt.plot(x, x, linestyle='', marker='<', markersize=20, markerfacecolor='#FF0000')
plt.plot(x, x ** 2, linestyle='', marker='^', markersize=10, markerfacecolor='#00FF00')
plt.plot(x, x ** (1/2), linestyle='', marker='v', markersize=10, markerfacecolor='#0000FF')
plt.plot(x, 1 - x, linestyle='', marker='+', markersize=10, markerfacecolor='#0F0F00')
plt.plot(x, 1 - x ** 2, linestyle='', marker='x', markersize=10, markerfacecolor='#0F000F')
plt.axis([-0.05, 1.05, -0.05, 1.05])
plt.show()
```



Если yy меняется на много порядков, то удобно использовать логарифмический масштаб по yy.

In [15]:

```
x = np.linspace(-5, 5, 100)

plt.figure()
plt.plot(x, np.exp(x) + np.exp(-x))
plt.yscale('log')
plt.yticks(fontsize=15)
plt.show()
```

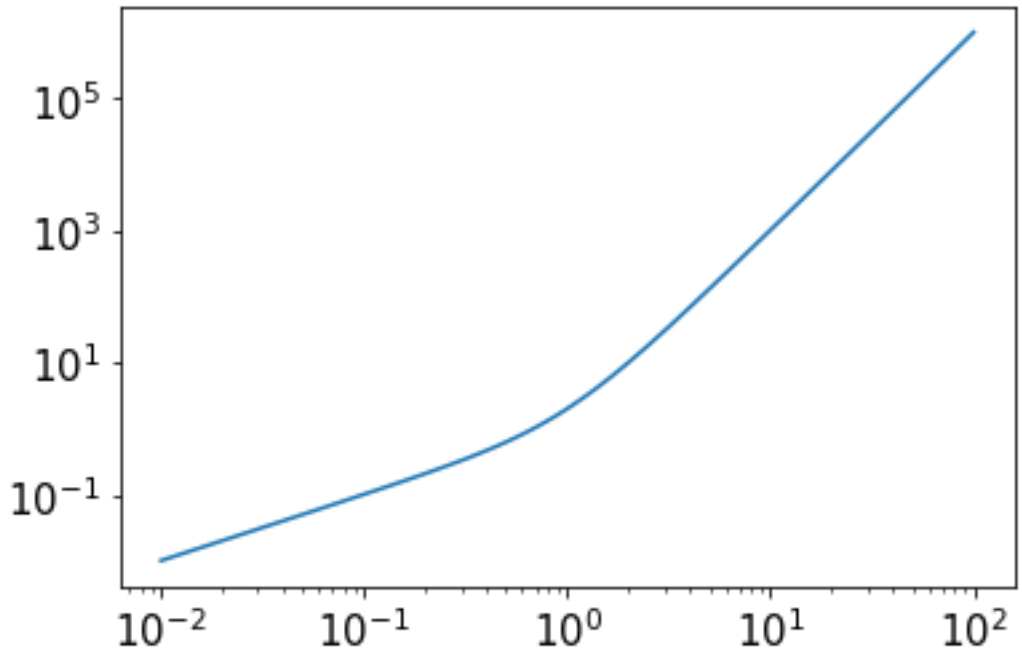


Можно задать логарифмический масштаб по обоим осям.

In [16]:

```
x = np.logspace(-2, 2, 100)

plt.figure()
```

```
plt.plot(x, x + x ** 3)
plt.xscale('log'), plt.xticks(fontsize=15)
plt.yscale('log'), plt.yticks(fontsize=15)
plt.show()
```

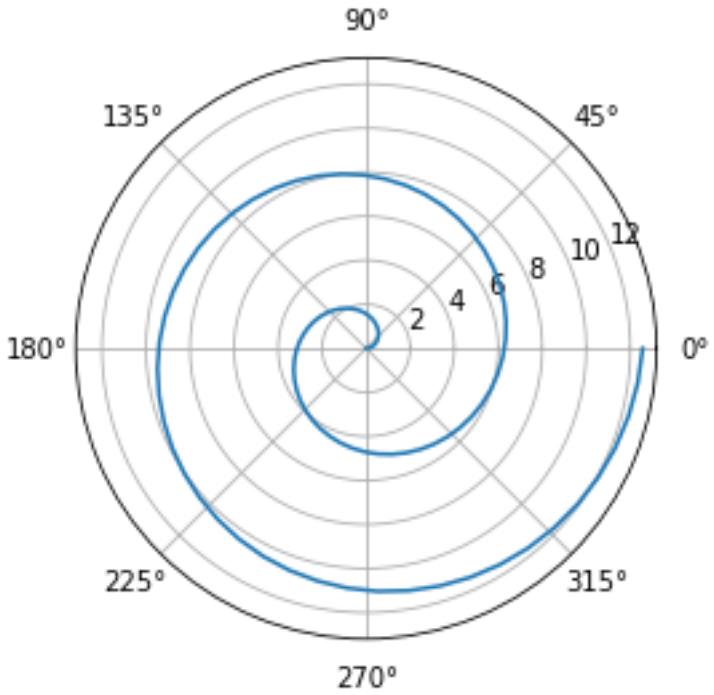


2. Более сложные графики

2.1. Полярные координаты

Первый массив — φφ, второй — rr. Вот спираль.

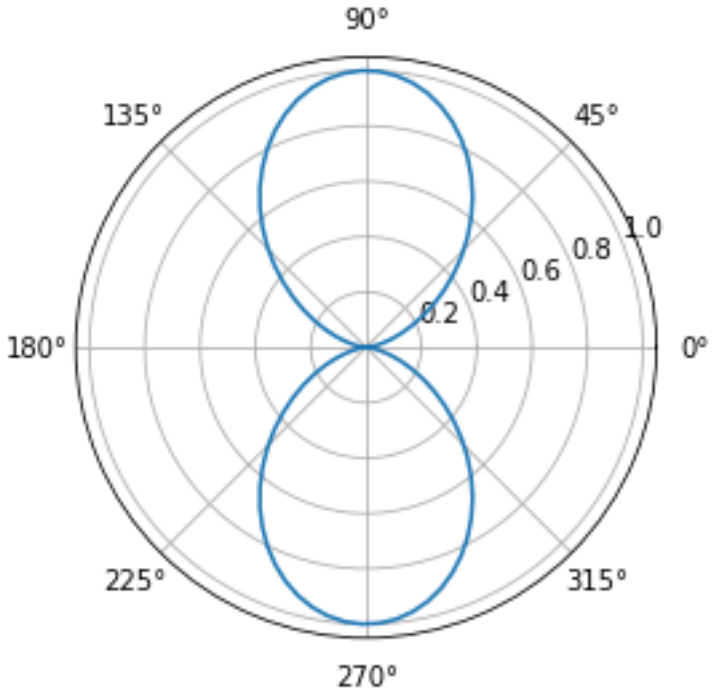
```
In [17]:
t = np.linspace(0, 4 * np.pi, 100)

plt.figure()
plt.polar(t, t)
plt.show()
```



А это угловое распределение пионов в e^+e^-e+e− аннигиляции.

In [18]:

```
phi = np.linspace(0, 2 * np.pi, 100)

plt.figure()
plt.polar(phi, np.sin(phi) ** 2)
plt.show()
```



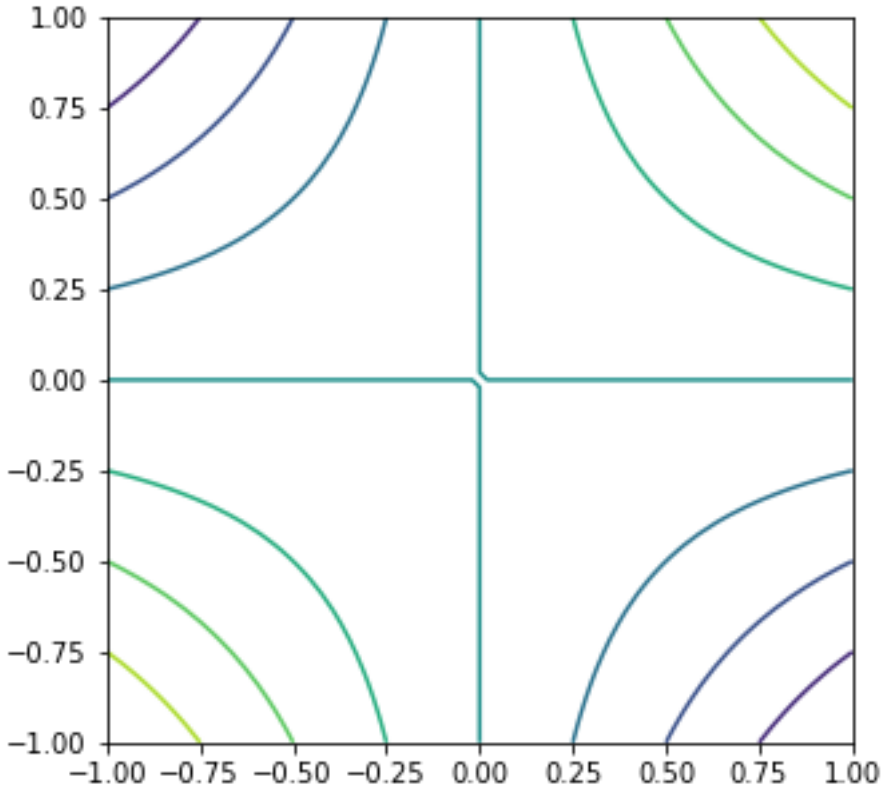
2.2. Контурные графики

Пусть мы хотим изучить поверхность $z=xy$z=xy. Вот её горизонтали.

In [19]:

```
x = np.linspace(-1, 1, 50)
y = x
```

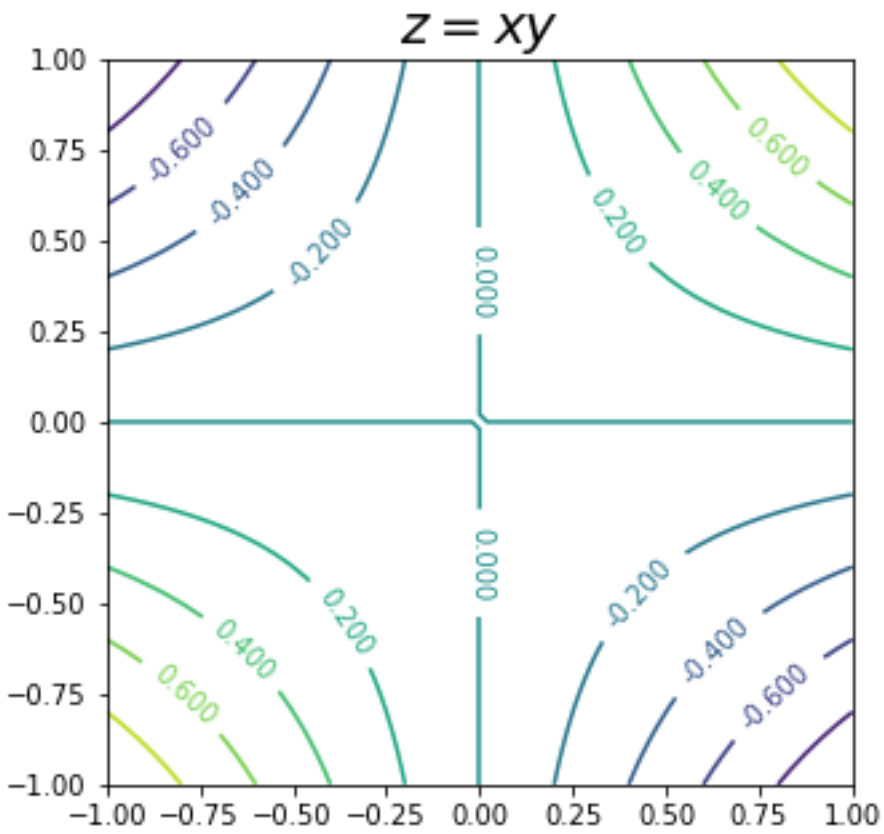
```
z = np.outer(x, y)

plt.figure(figsize=(5,5))
plt.contour(x, y, z)
plt.show()
```



Что-то их маловато. Сделаем побольше и подпишем.

In [20]:

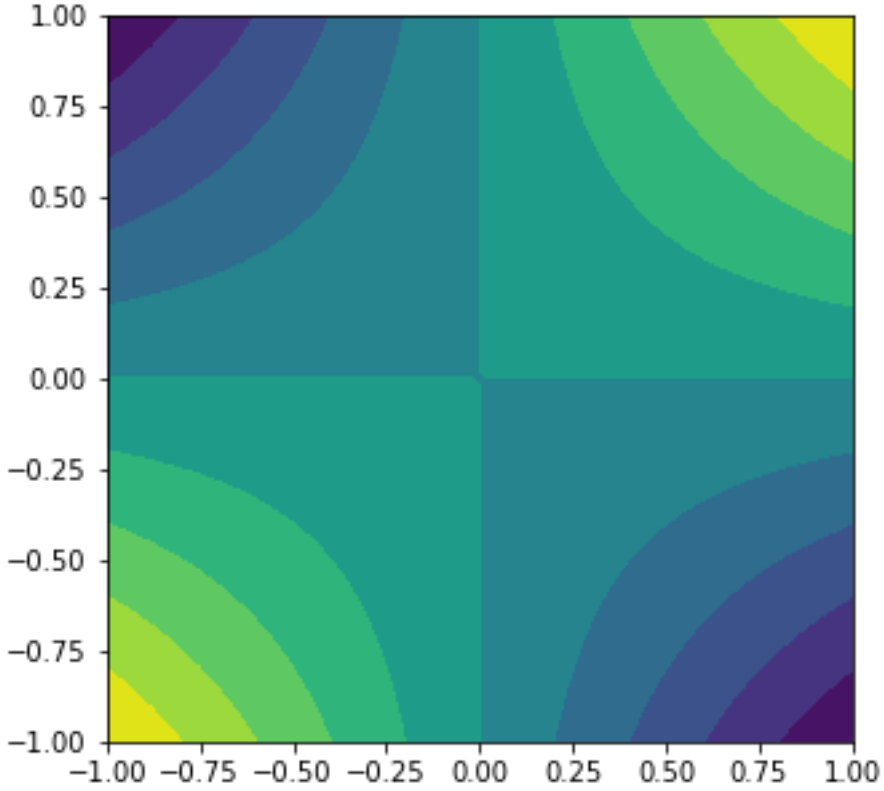
```
plt.figure(figsize=(5,5))
curves = plt.contour(x, y, z, np.linspace(-1, 1, 11))
plt.clabel(curves)
plt.title(r'$z=xy$', fontsize=20)
plt.show()
```



А здесь высота даётся цветом, как на физических географических картах. Функция `colorbar` показывает соответствие цветов и значений zz.

In [21]:

```
plt.figure(figsize=(5,5))
plt.contourf(x, y, z, np.linspace(-1, 1, 11))
# plt.colorbar()
plt.show()
```

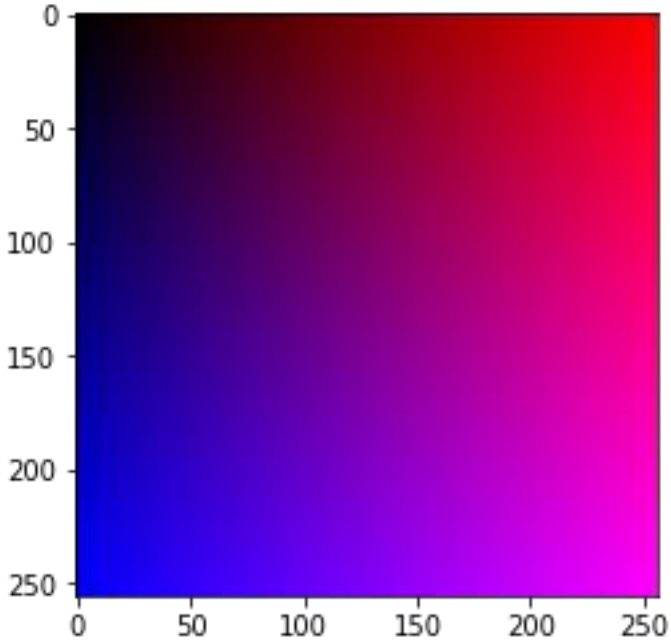


2.3. Images (пиксельные картинки)

Картинка задаётся массивом `z`: `z[i,j]` — это цвет пикселя `i,j`, массив из 3 элементов (`rgb`, числа от 0 до 1).

In [22]:

```
n = 256
u = np.linspace(0, 1, n)
x, y = np.meshgrid(u, u)
z = np.zeros((n, n, 3))
z[:, :, 0] = x
z[:, :, 2] = y

plt.figure()
plt.imshow(z)
plt.show()
```



Можно загрузить картинку из файла. Это будет обычный `numpy.array`. Размерность картинки 280×280280×280. По последней координате цвета RGB и прозрачность.

In [23]:

```
picture = plt.imread('python.png')
print(type(picture), picture.shape)

plt.imshow(picture)
plt.axis('off')
plt.show()
```

```
<class 'numpy.ndarray'> (2000, 2000, 4)
```

2.4. Трёхмерная линия

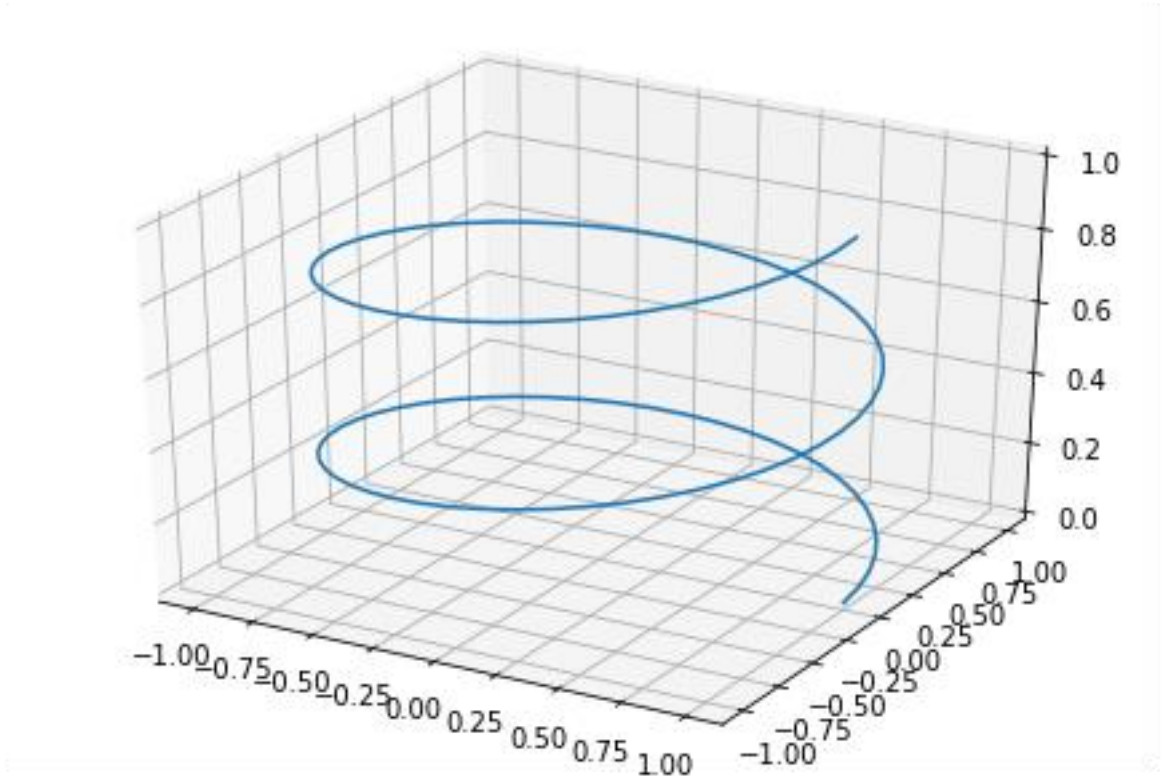
Задаётся параметрически: $x=x(t)$x=x(t), $y=y(t)$y=y(t), $z=z(t)$z=z(t).

In [24]:

```
t = np.linspace(0, 4 * np.pi, 100)
x = np.cos(t)
y = np.sin(t)
z = t / (4 * np.pi)
```

In [25]:

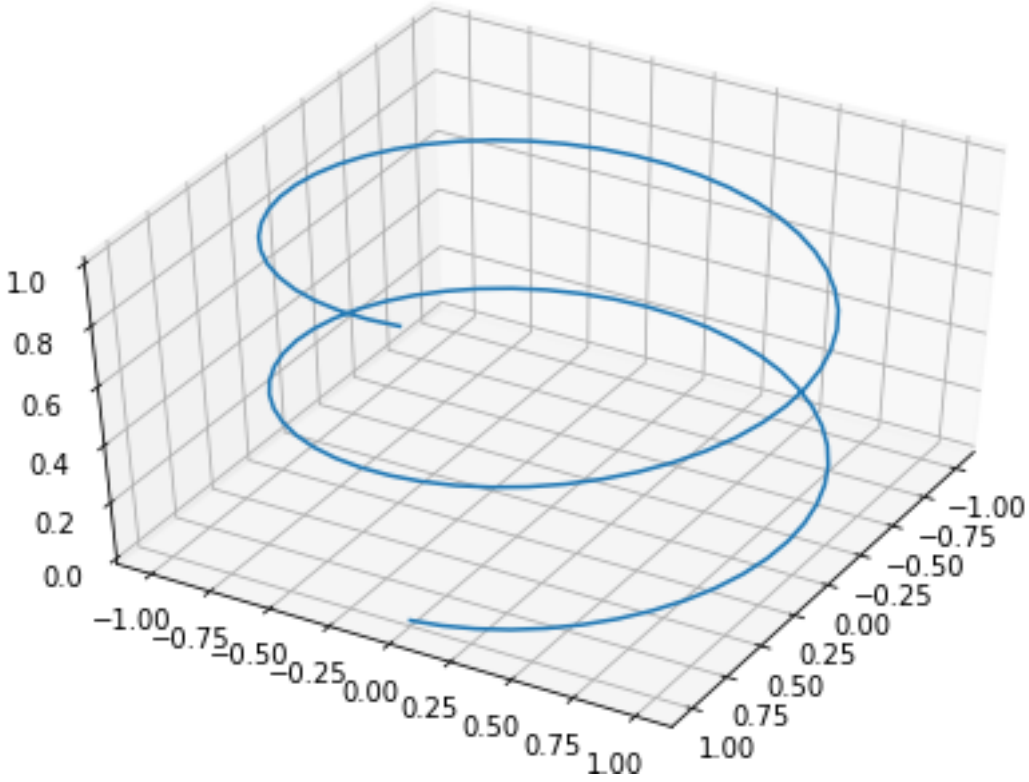
```
fig = plt.figure()
ax = Axes3D(fig)
ax.plot(x, y, z)
plt.show()
```



Можно задать, с какой стороны мы смотрим.

In [26]:

```
fig = plt.figure()
ax = Axes3D(fig)
ax.elev, ax.azim = 45, 30
ax.plot(x, y, z)
plt.show()
```

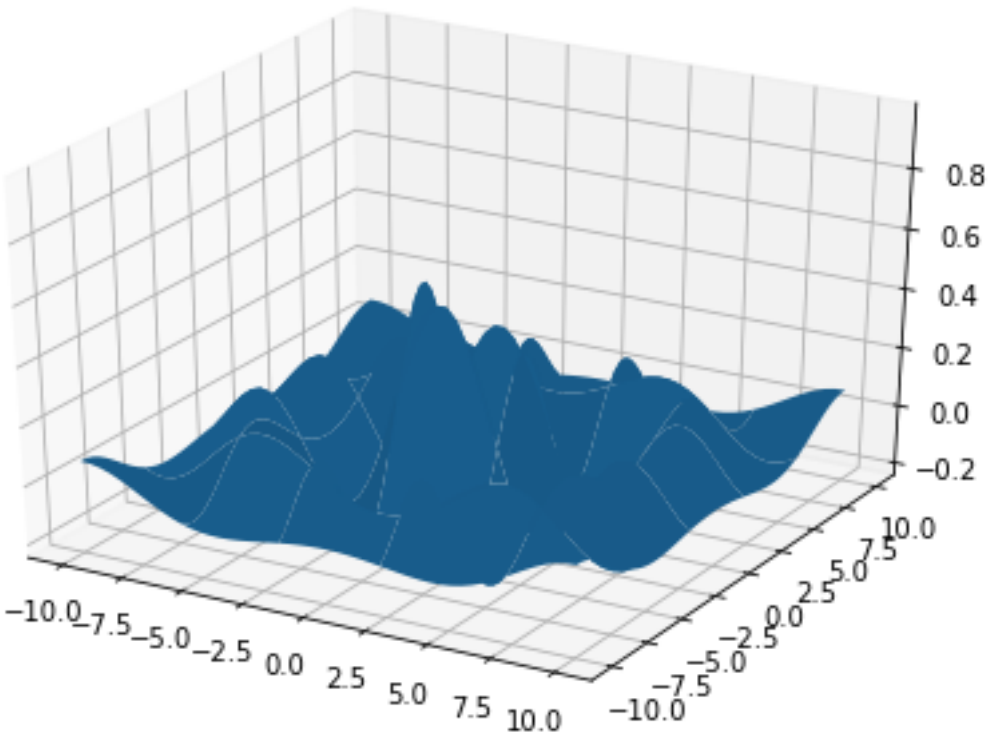


2.5. Поверхности

Все поверхности параметрические: $x=x(u,v)$x=x(u,v), $y=y(u,v)$y=y(u,v), $z=z(u,v)$z=z(u,v). Если мы хотим построить явную поверхность $z=z(x,y)$z=z(x,y), то удобно создать массивы $x=u$x=u и $y=v$y=v функцией `meshgrid`.

In [27]:

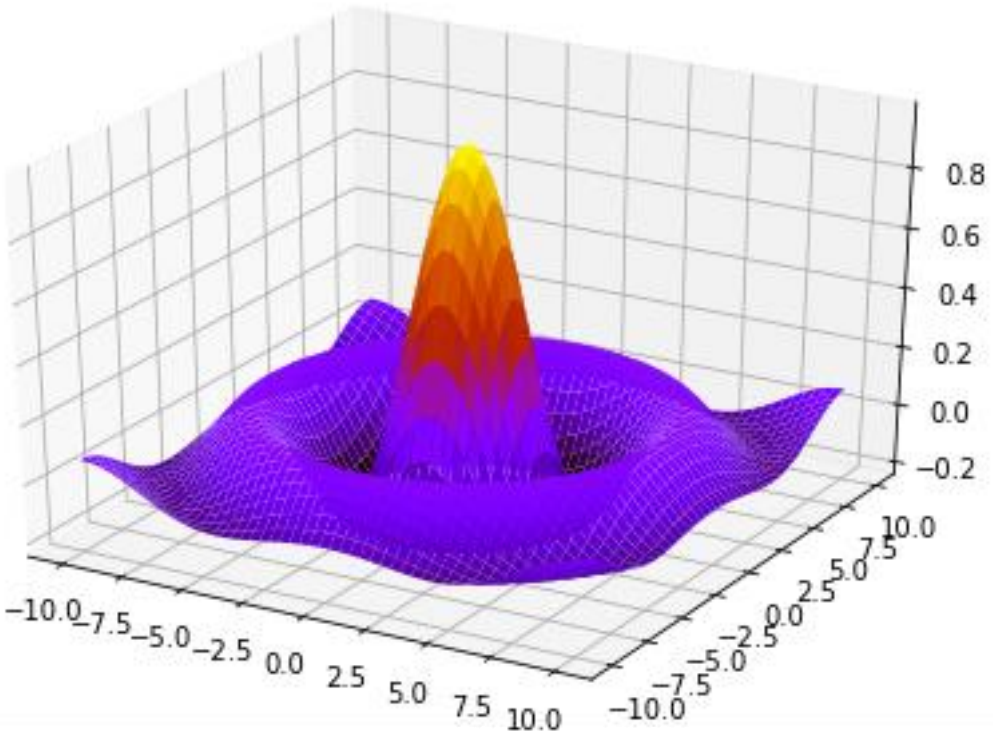
```
X = 10
N = 50
u = np.linspace(-X, X, N)
x, y = np.meshgrid(u, u)
r = np.sqrt(x ** 2 + y ** 2)
z = np.sin(r) / r

fig = plt.figure()
ax = Axes3D(fig)
ax.plot_surface(x, y, z, rstride=10, cstride=10)
plt.show()
```



Есть много встроенных способов раскраски поверхностей. Так, в методе `gnuplot` цвет зависит от высоты zz.

In [28]:

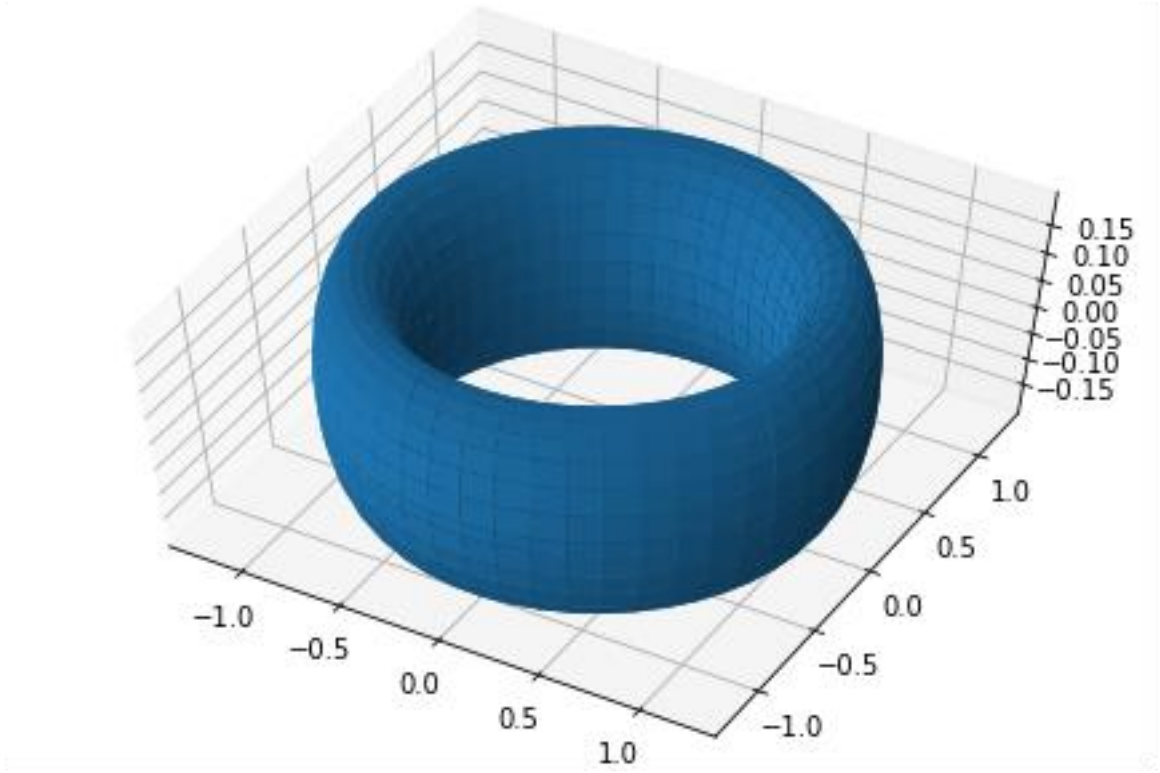
```
fig = plt.figure()
ax = Axes3D(fig)
ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='gnuplot')
plt.show()
```



Построим бублик — параметрическую поверхность с параметрами θθ, φφ.

In [29]:

```
t = np.linspace(0, 2 * np.pi, 50)
th, ph = np.meshgrid(t, t)
r = 0.2
x, y, z = (1 + r * np.cos(ph)) * np.cos(th), (1 + r * np.cos(ph)) * np.sin(th), r * np.sin(ph)

fig = plt.figure()
ax = Axes3D(fig)
ax.elev = 60
ax.plot_surface(x, y, z, rstride=2, cstride=1)
plt.show()
```

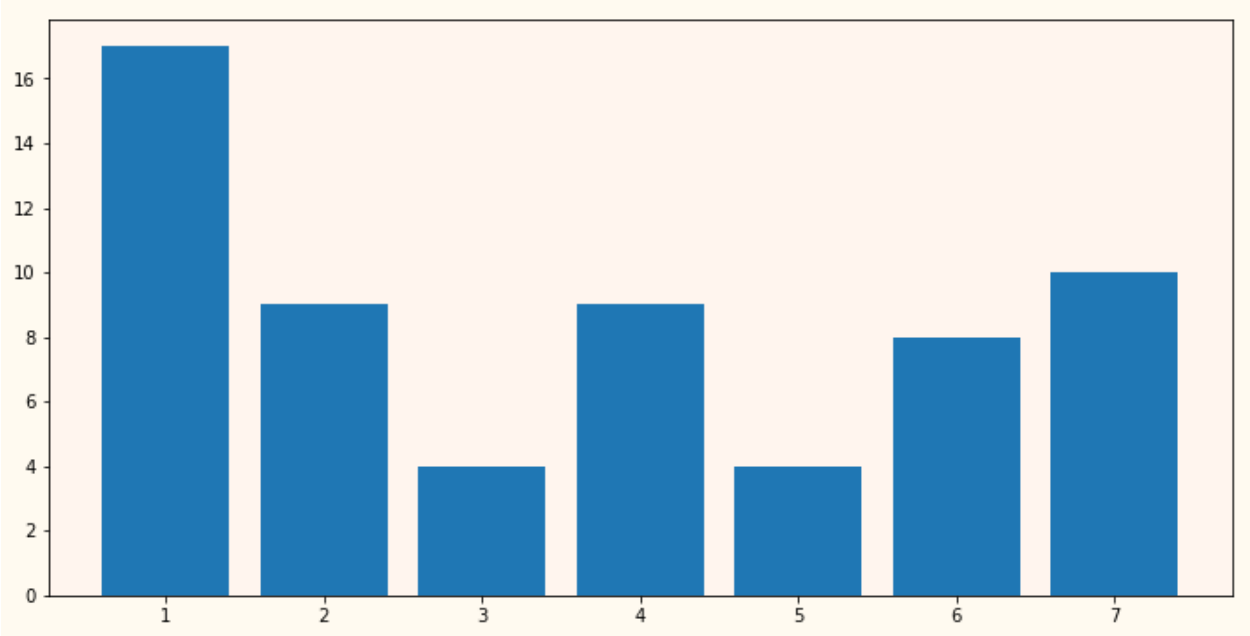


Столбчатая диаграмма(гистограмма):

In [30]:

```
x = np.arange(1, 8)
y = np.random.randint(1, 20, size = 7)

fig, ax = plt.subplots()

ax.bar(x, y)

ax.set_facecolor('seashell')
```

```
fig.set_facecolor('floralwhite')
fig.set_figwidth(12)    #  ширина Figure
fig.set_figheight(6)    #  высота Figure


plt.show()
```

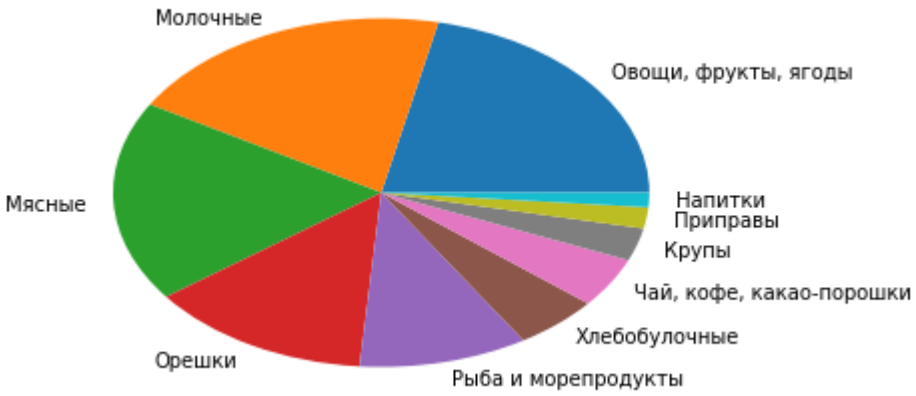


Столбчатая диаграмма(гистограмма):

In [31]:

```
labels = ['Овощи, фрукты, ягоды',
      'Молочные',
      'Мясные',
      'Орешки',
      'Рыба и морепродукты',
      'Хлебобулочные',
      'Чай, кофе, какао-порошки',
      'Крупы',
      'Приправы',
      'Напитки']

values = [21.79,
      20.18,
      18.82,
      13.64,
      10.30,
      5.18,
      4.69,
      3.06,
      1.98,
      0.35]


fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels)
```

```
plt.show()
```



Задание

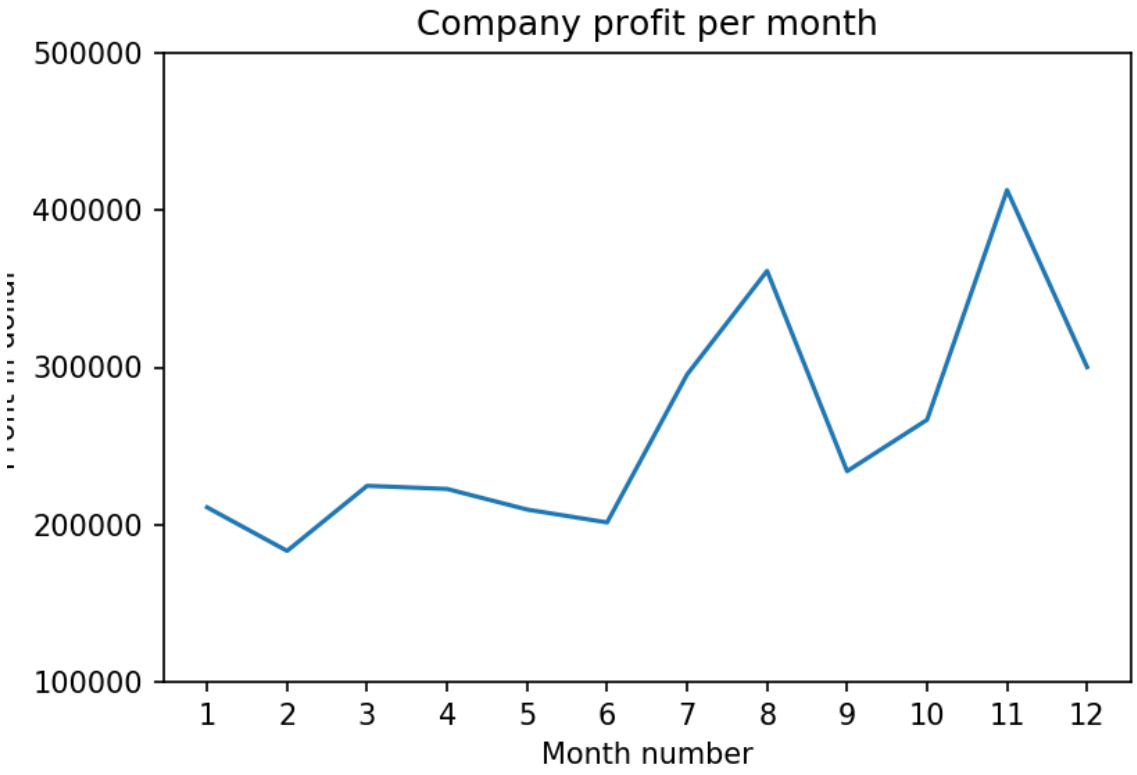
Данные о продажах хранятся в файле company_sales_data.csv

1. Необходимо показать общую прибыль за все месяцы в виде линейного графика.

По оси X - Номер месяца

По оси Y - Общая прибыль

График должен выглядеть следующим образом.



2. Необходимо показать общий объем продаж (количество товаров) за все месяцы в виде линейного графика.

По оси X - Номер месяца

По оси Y – Количество проданных товаров

Требования к графику:

График должен быть пунктирной, красного цвета

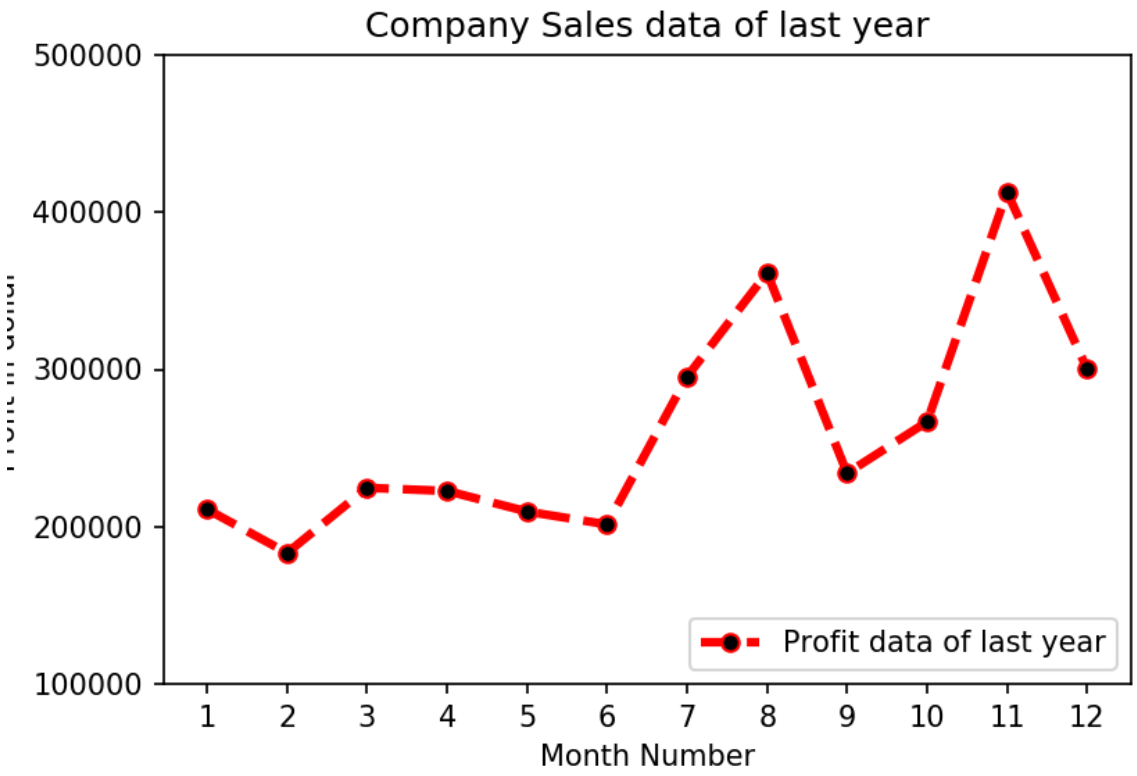
На графике должна быть легенда в правом нижнем углу.

Точки на графике должны быть отмечены круглым маркером.

Граница маркера должна быть красного цвета

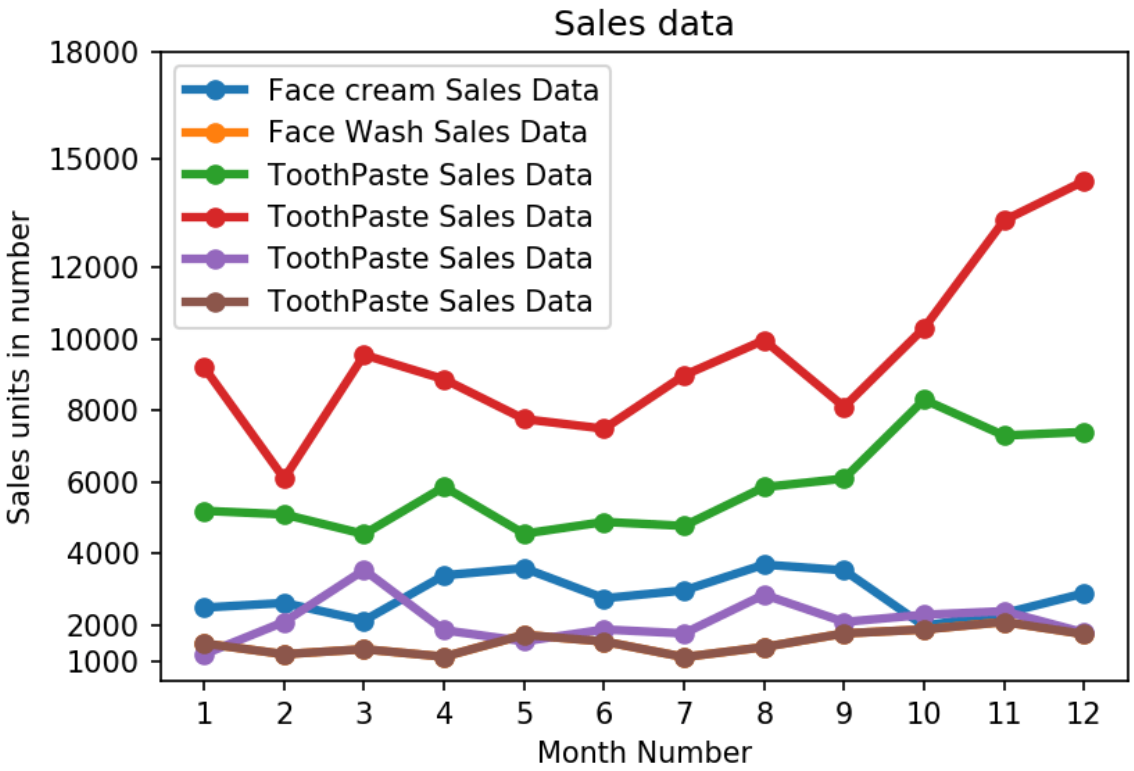
Ширина линии – 3

График должен выглядеть следующим образом.



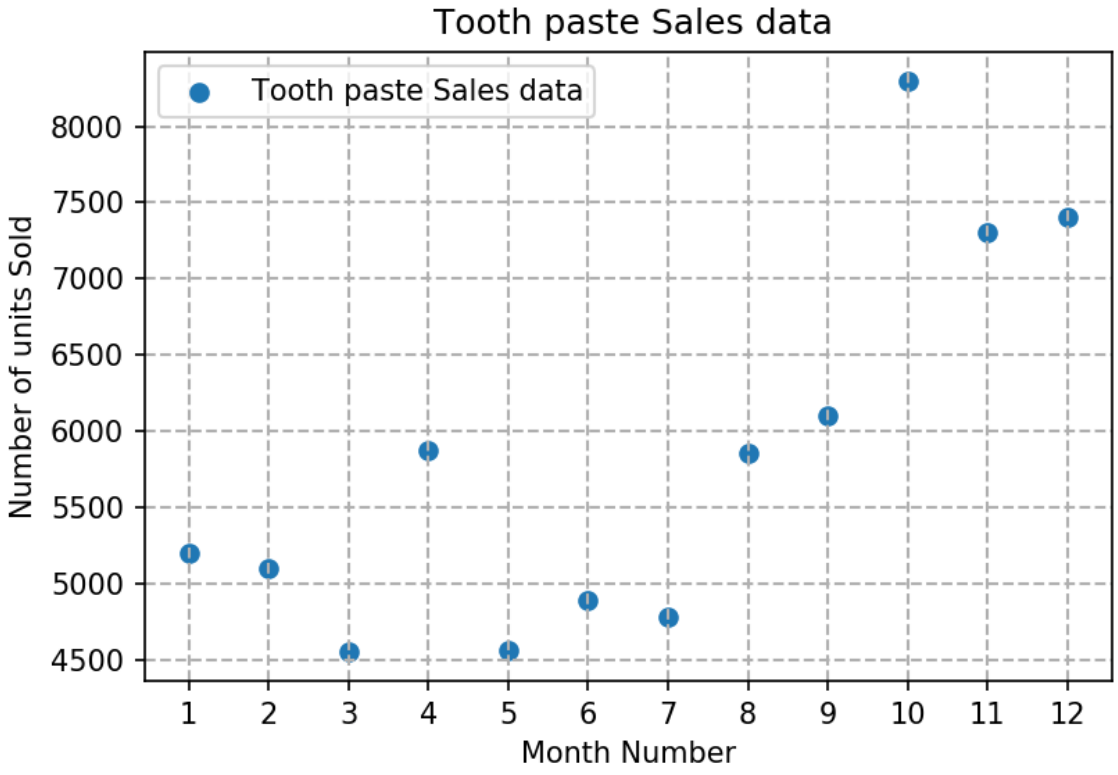
3. Необходимо показать количество проданных товаров каждого вида за все месяцы на одном графике.

График должен выглядеть следующим образом.



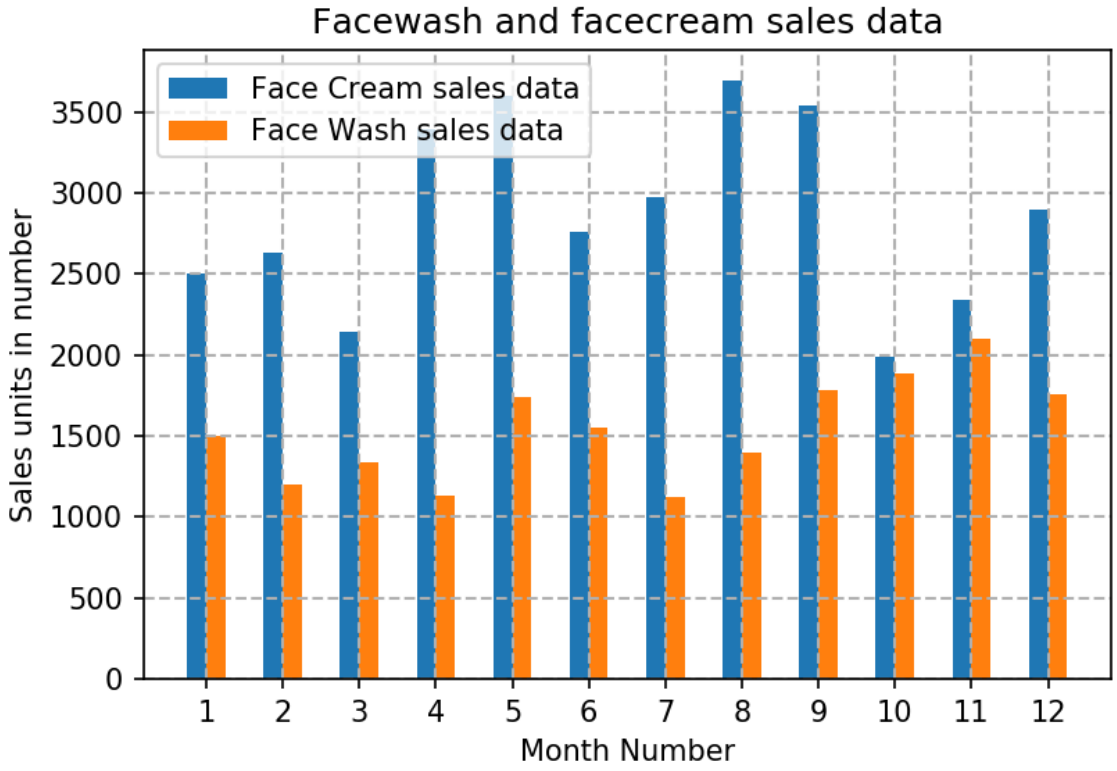
4. Необходимо показать данные о продажах зубной пасты за каждый месяц на диаграмме рассеяния (scatter plot). Необходимо добавить сплошную сетку на график.

График должен выглядеть следующим образом.



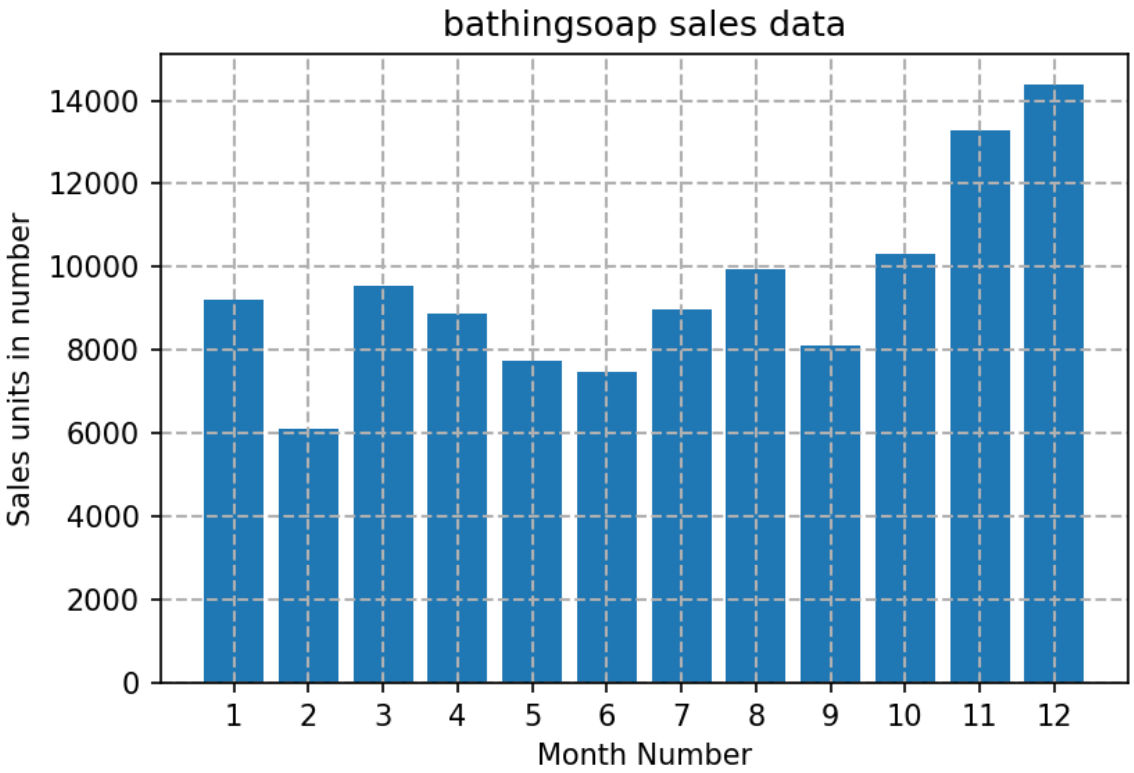
5. Необходимо показать данные о продажах кремов и средств для мытья лица в гистограмме. Гистограмма должна отображать количество проданных единиц в месяц для каждого продукта.

График должен выглядеть следующим образом.



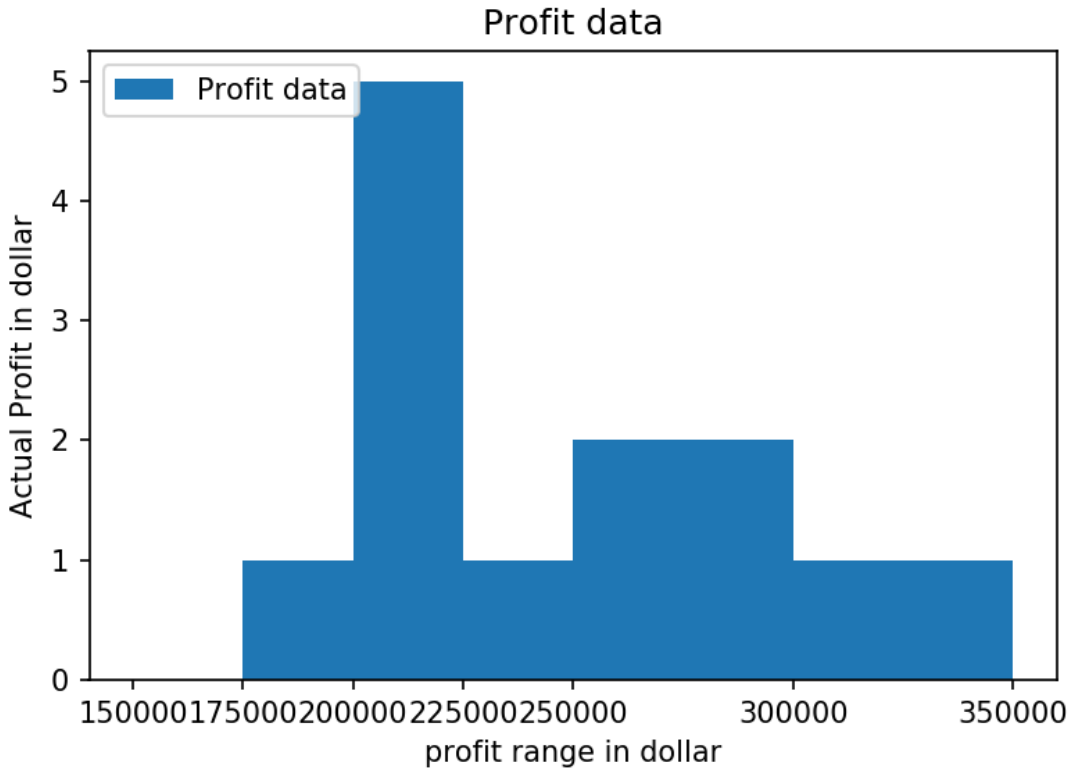
6. Необходимо показать данные о продажах мыла для купания за все месяцы в виде гистограммы. Сохранить полученный график.

График должен выглядеть следующим образом.



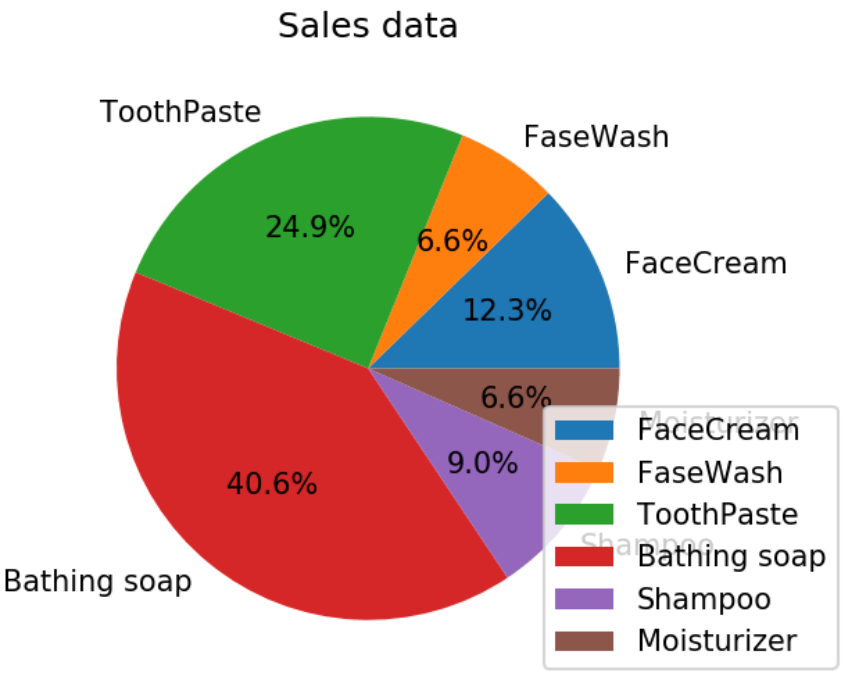
7. Необходимо сосчитать общую прибыль каждого месяца и отразить ее с помощью гистограммы, которая показывает, какой диапазон прибыли является самым распространенным.

График должен выглядеть следующим образом.



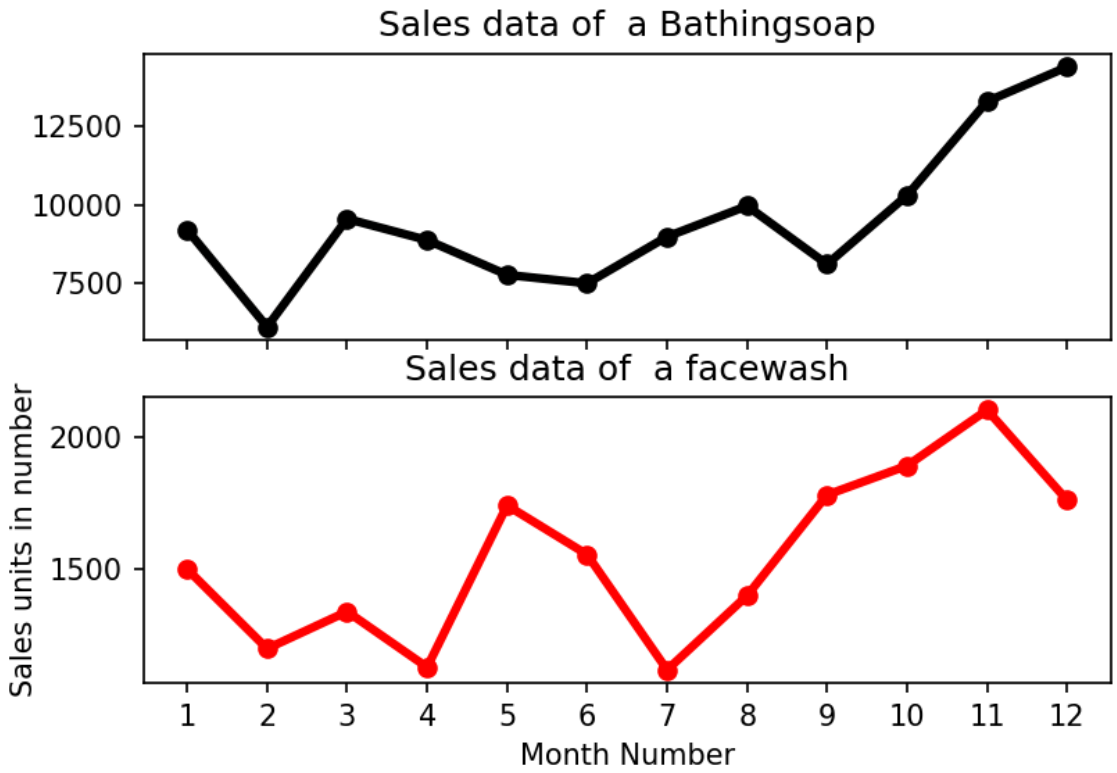
8. Необходимо подсчитать общие данные о продажах для каждого продукта и отразить их с помощью круговой диаграммы. На круговой диаграмме отображается количество проданных единиц каждого продукта в процентном отношении.

График должен выглядеть следующим образом.



9. Необходимо показать данные о продажах средства для мытья лица в сравнении с мылом для купания за все месяцы с помощью вспомогательного графика (subplot).

График должен выглядеть следующим образом.



10. Необходимо показать все данные о продажах продуктов в виде надставленных областей (stack plot).

График должен выглядеть следующим образом.