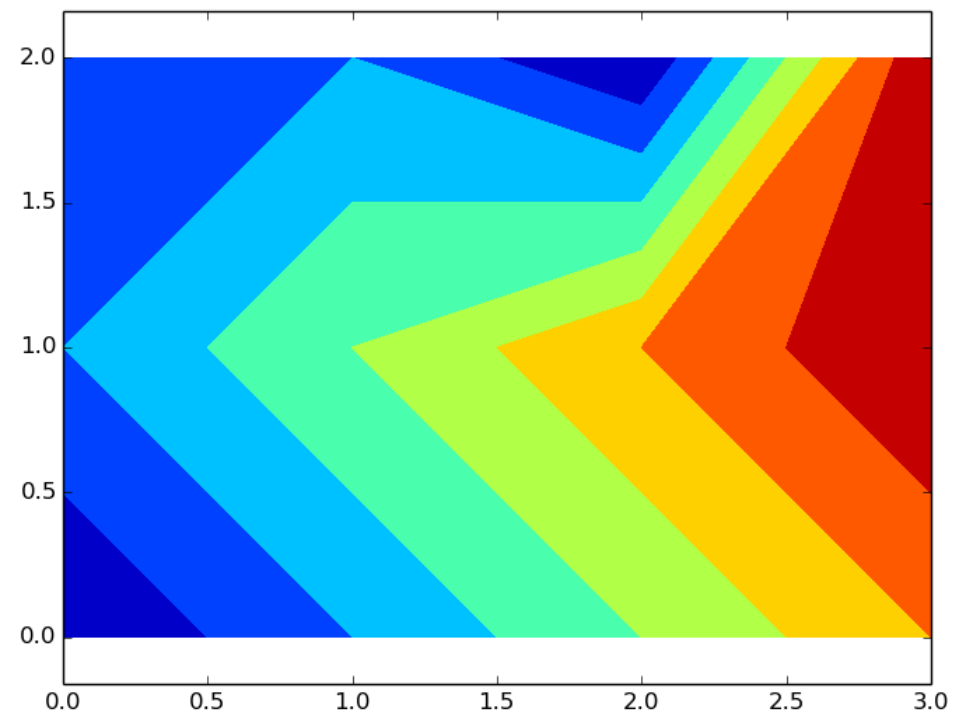
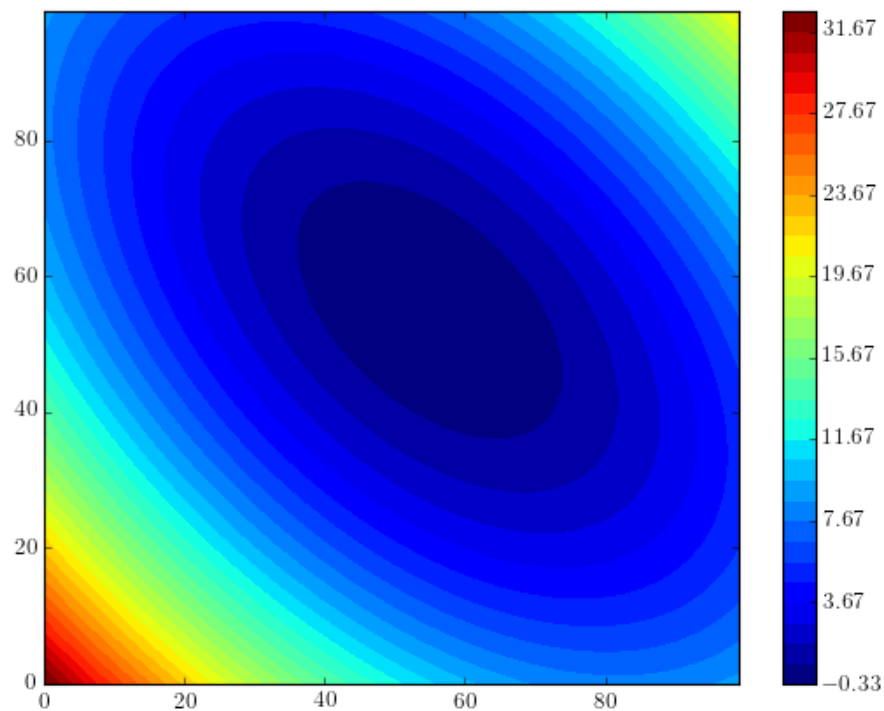


# Библиотека "matplotlib" Python разработка.

---



# Часть 1. Что же это такое?

Библиотека Matplotlib - является одним из самых популярных средств визуализации данных на Python.

- Matplotlib - это библиотека, на языке программирования Python для визуализации данных двумерной и трехмерной графикой.
- Генерируемые в различных форматах изображения могут быть использованы в интерактивной графике, научных публикациях, графическом интерфейсе пользователя, веб-интерфейсе, где требуется построение диаграмм. Она построена на принципах ООП, но имеет процедурный интерфейс "pylab", который предоставляет аналоги команд MATLAB.

Пакет поддерживает многие виды графиков и диаграмм:

- 1)Графики
- 2)Диаграммы рассеяния
- 3)Столбчатые диаграммы
- 4)Круговые диаграммы
- 5)Диаграммы стебель-листья
- 6)Контурные графики
- 7)Поля градиентов
- 8)Спектральные диаграммы

Matplotlib является часть Scientific Python - набора библиотек для научных вычислений и визуализации данных, куда также входят NumPy, SciPy, Pandas и SymPy и т.д.

Также в дополнении хотелось бы сказать, что `background` библиотеки `Matplotlib` написан на языке `C`. Поддерживаемые версии `Python2`, `Python3` и `IPython`. Работает как кроссплатформенная библиотека.

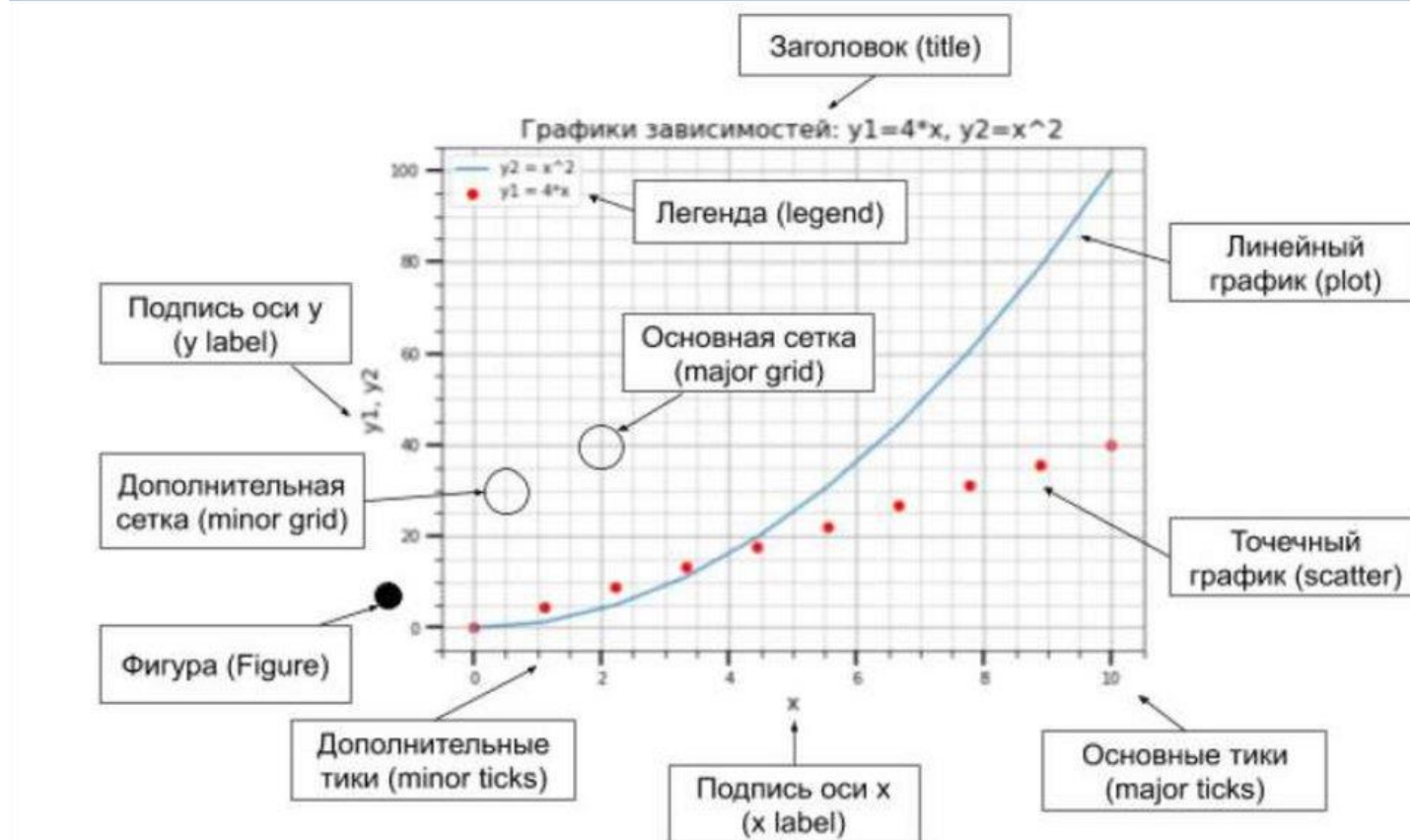
Набор поддерживаемых форматов изображений, векторных и растровых, можно получить из словаря `FigureCanvasBase.filetypes`. Типичные поддерживаемые форматы:

- Encapsulated PostScript (EPS)
- Enhanced Metafile (EMF)
- JPEG
- PDF
- PNG
- Postscript
- RGBA («сырой» формат)
- SVG
- SVGZ
- TIFF



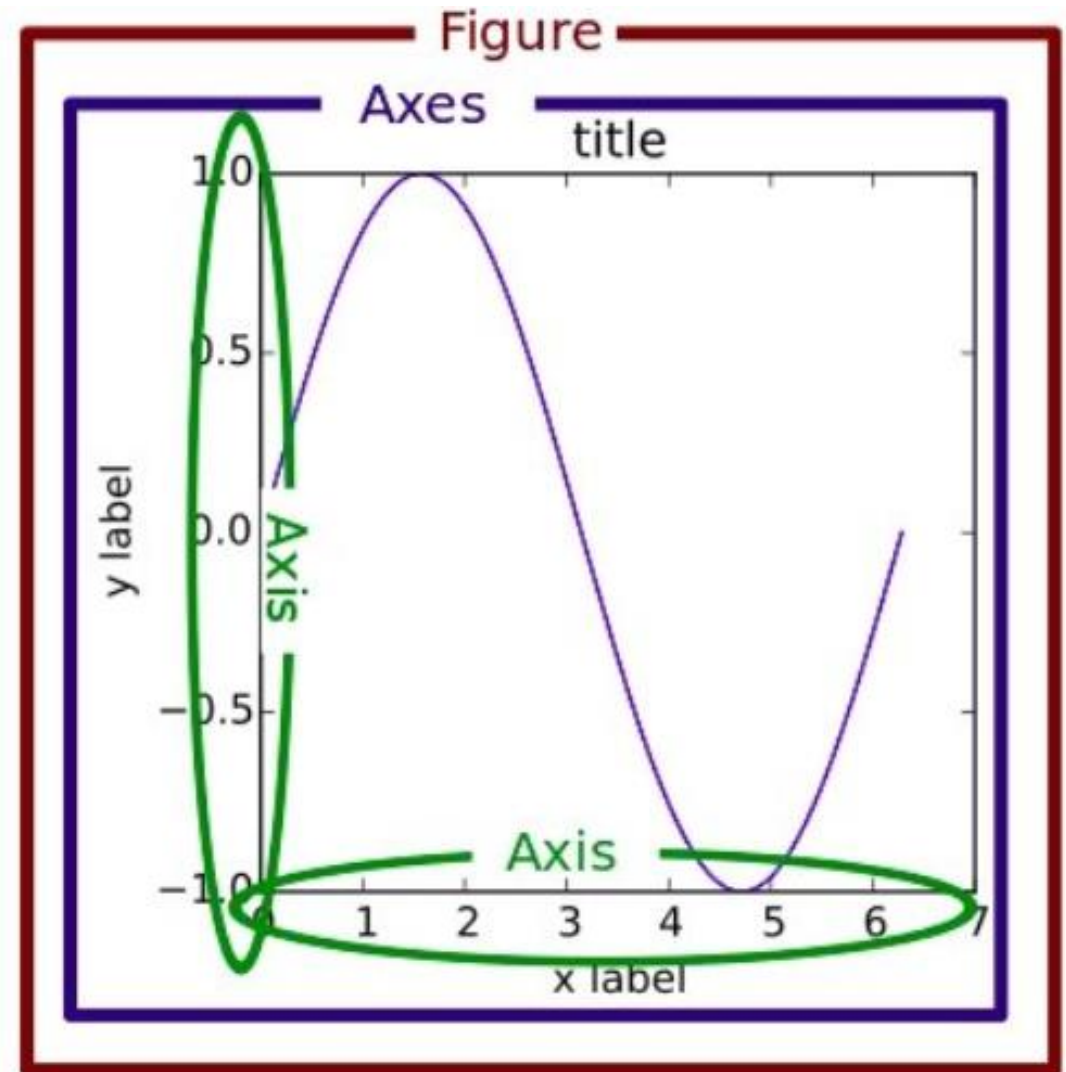
## Часть 2. Основные методы и их реализация.

# Matplotlib - Основные элементы графика



# Иерархия объектов в Matplotlib

- Объект **Figure** – это самый важный внешний контейнер для графики matplotlib, который может включать в себя несколько объектов Axes.
- Вы можете рассматривать объект **Figure** как похожий на ящик контейнер, содержащий один или несколько объектов Axes (настоящих графиков). Под объектами Axes, в порядке иерархии расположены меньшие объекты, такие как индивидуальные линии, отметки, легенды и текстовые боксы. Практически каждый «элемент» диаграммы – это собственный манипулируемый объект Python, вплоть до ярлыков и отметок:



# Структурированные и неструктурированные подходы

- Ранее, мы использовали **import matplotlib.pyplot as plt** для импорта модуля pyplot из matplotlib и назвали его plt.
- Практически все функции pyplot, такие как **plt.plot()**, так или иначе, ссылаются на нынешний существующий объект Figure и нынешний объект Axes, или создают их, если какой-либо из них не существует.
- Структурный интерфейс делает свои вызовы с **plt.plot()** и другими высшими функциями pyplot. Существует только один объект Figure или Axes, который вы используете за данное время, и вам не нужна явная ссылка на этот объект;

# Начало работы.

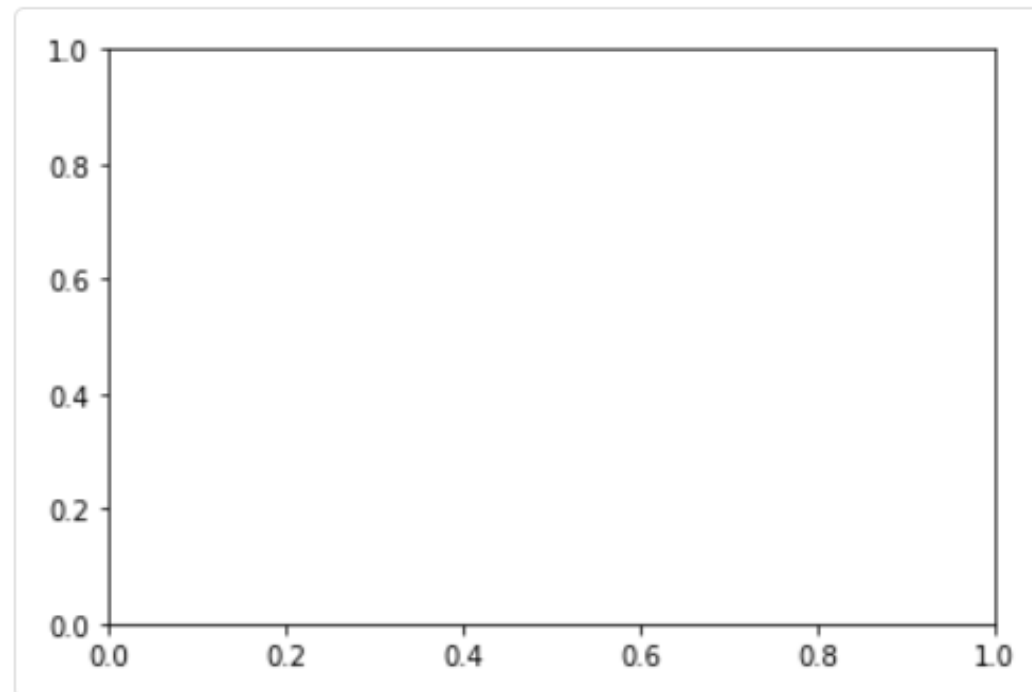
- Давайте попробуем выполним следующий код:

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

fig = plt.figure()
ax = fig.add_subplot(111)

plt.show()
```

В строке `fig = plt.figure()` мы создали область *Figure* (экземпляр класса *figure*). В строке `ax = fig.add_subplot(111)` мы добавили к *Figure* область *Axes*. Вообще, было бы правильнее использовать `fig.add_axes`, но в данном случае `fig.add_subplot(111)` намного удобнее, в конце концов `subplot` просто размещает *Axes* на сетке *Figure*. Обратите внимание на параметр, который мы передаем `111` - это первая строка, первый столбец и первая (единственная) ячейка на сетке *Figure*.



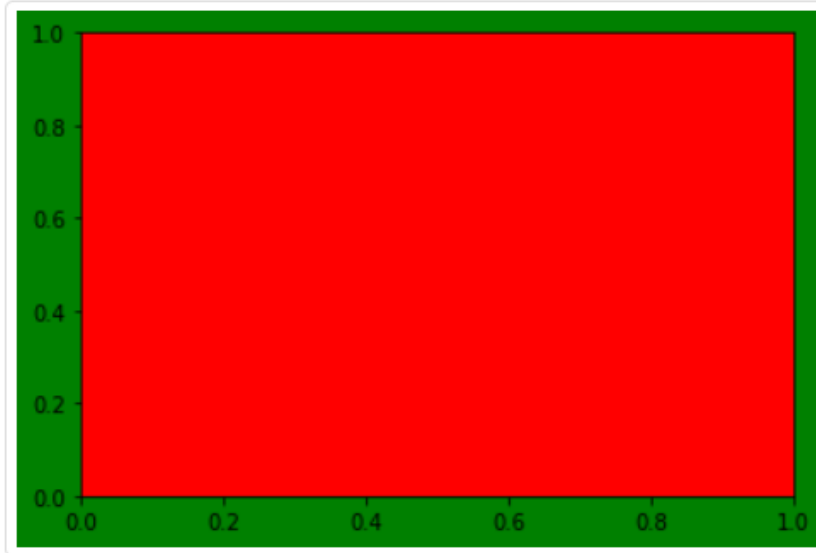
- В том что *Figure* и *Axes* это разные области можно легко убедиться если изменить их цвет:

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)

fig.set(facecolor = 'green')
ax.set(facecolor = 'red')

plt.show()
```



Кстати, *Axes* должна принадлежать только одной области *Figure*. Как правило, всегда сначала создается область *Figure*, а затем с помощью `add_subplot()` в *Figure* размещается одна или несколько областей *Axes*.

А теперь обратите внимание на то, как с помощью метода `set()` мы изменили цвет *Figure* и *Axes*. По сути, это самый быстрый способ устанавливать параметры, но он не самый явный. Давайте установим параметры явно.

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111) # We'll explain the "111" later. Basically, 1 row and 1 column.

fig.set_facecolor('green')
ax.set(facecolor = 'red')

plt.show()
```



Тоже самое мы можем проделать и с Axes. Кстати, именно Axes вам придется видоизменять чаще всего, поэтому давайте установим побольше параметров для данной области:

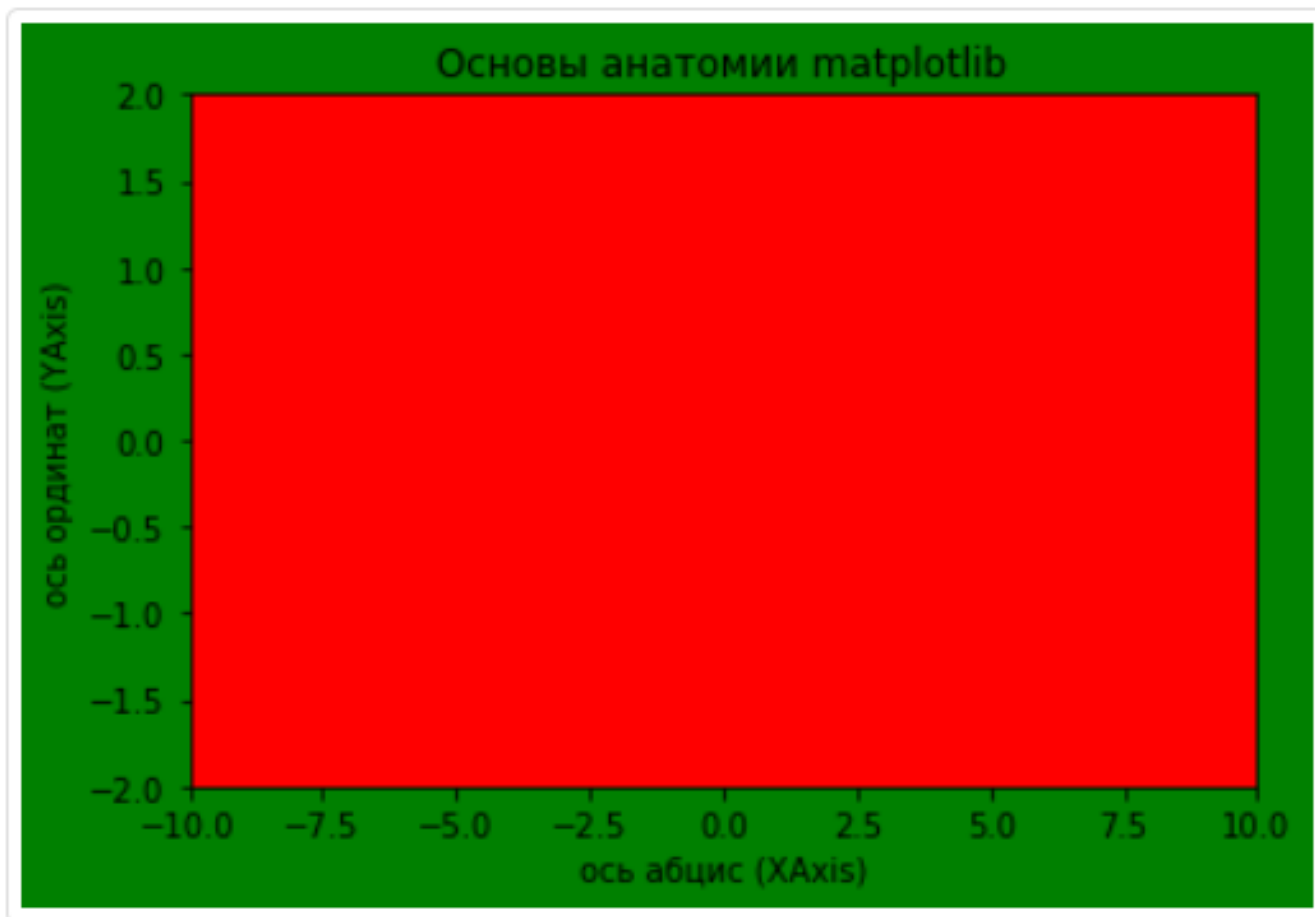
```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111) # We'll explain the

fig.set_facecolor('green')

ax.set_facecolor('red')
ax.set_xlim([-10, 10])
ax.set_ylim([-2, 2])
ax.set_title('Основы анатомии matplotlib')
ax.set_xlabel('ось абцисс (XAxis)')
ax.set_ylabel('ось ординат (YAxis)')

plt.show()
```



И еще, напоследок, график который мы создали - это просто издевательство над восприятием человека. Такие графики можно делать только для примера! Создание отличных графиков - это целая наука (или искусство), у которой даже есть название *инфографика*.

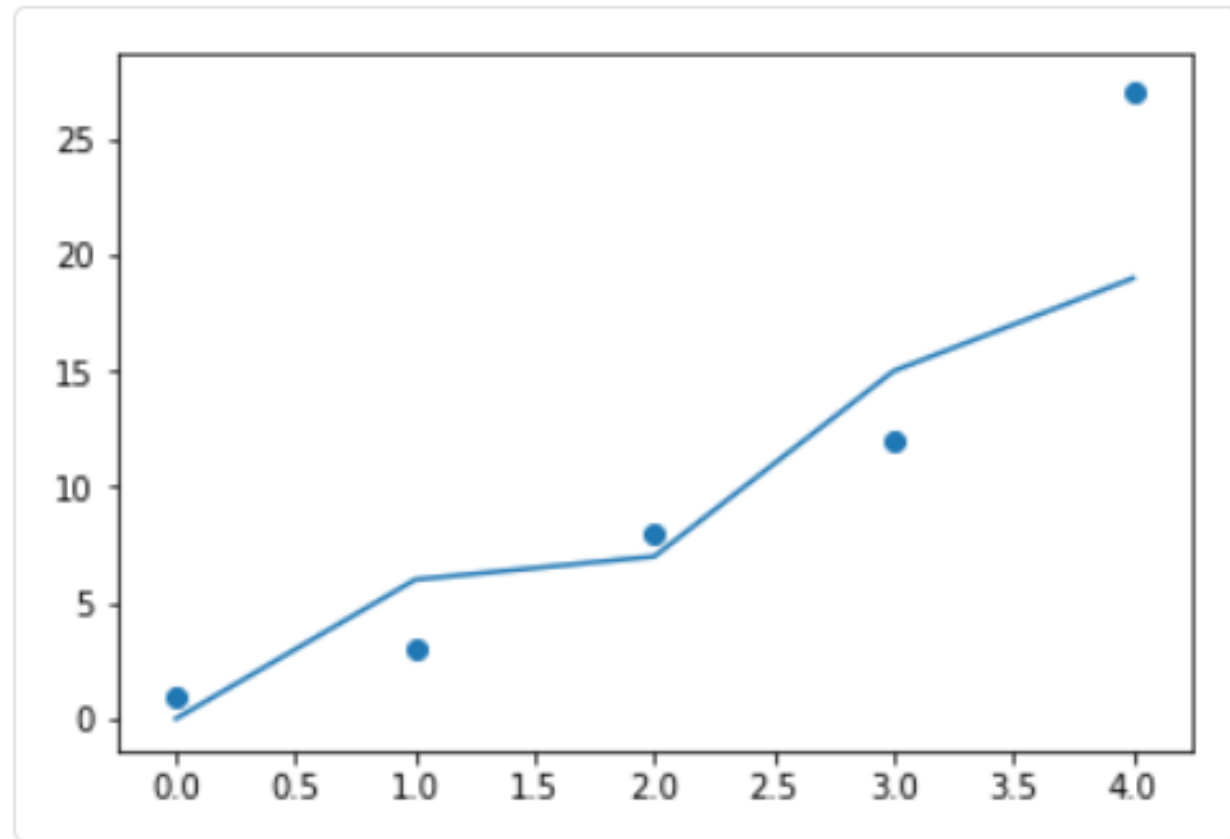
# Отображение данных на графике

- Большинство графиков: линий, гистограмм, круговых диаграмм т.е. отображений данных происходит на Axes. Поэтому, для рисования на Axes необходимо использовать какой-нибудь из его методов. К слову сказать, этих методов целая куча, но мы сосредоточимся всего на двух: `plot` и `scatter`.
- `plot` рисует точки соединенные линиями;
- `scatter` просто рисует точки
- Давайте построим простой график на котором будет присутствовать отображение одних данных точками, а других линиями:

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([0, 1, 2, 3, 4], [0, 6, 7, 15, 19])
ax.scatter([0, 1, 2, 3, 4], [1, 3, 8, 12, 27])

plt.show()
```

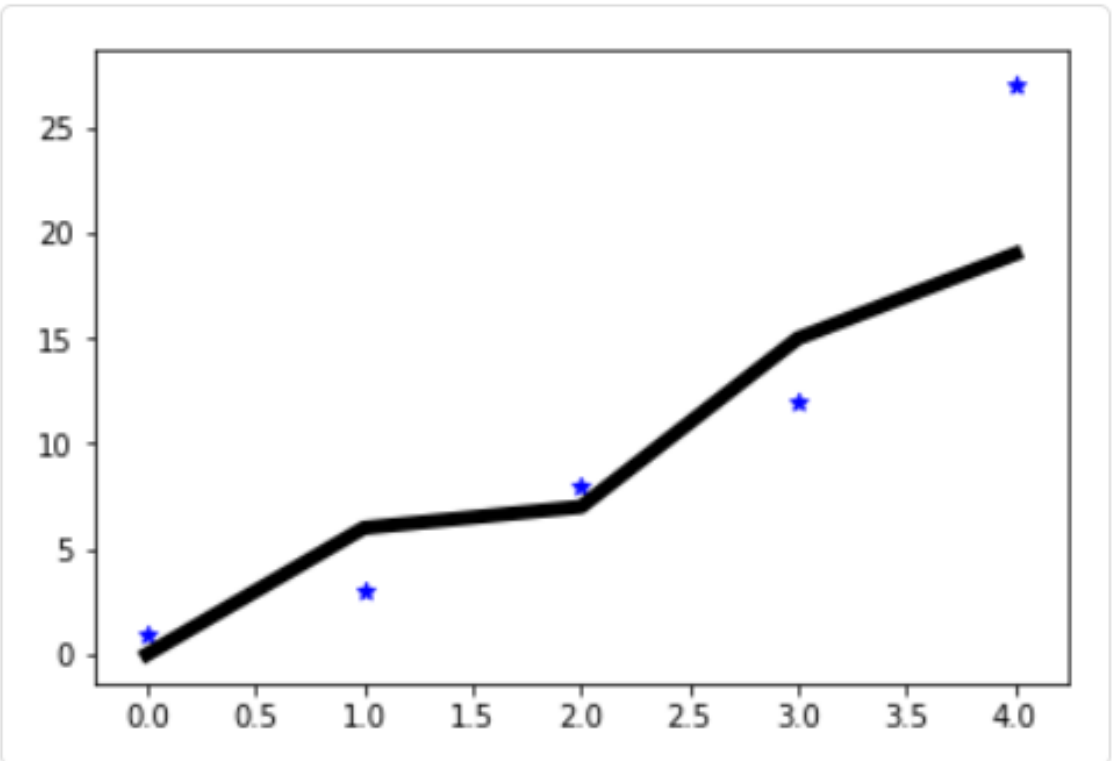


# Нарисованные данные так же поддерживают самые разные параметры внешнего вида:

```
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot([0, 1, 2, 3, 4], [0, 6, 7, 15, 19], color = 'black', linewidth = 5)
ax.scatter([0, 1, 2, 3, 4], [1, 3, 8, 12, 27], color = 'blue', marker = '*')

plt.show()
```



# Несколько Axes на одной Figure

- Очень часто, нам необходимо размещать несколько графиков рядом друг с другом. Это проще всего сделать используя `plt.subplots()`. Но давайте для начала разберем следующий пример:

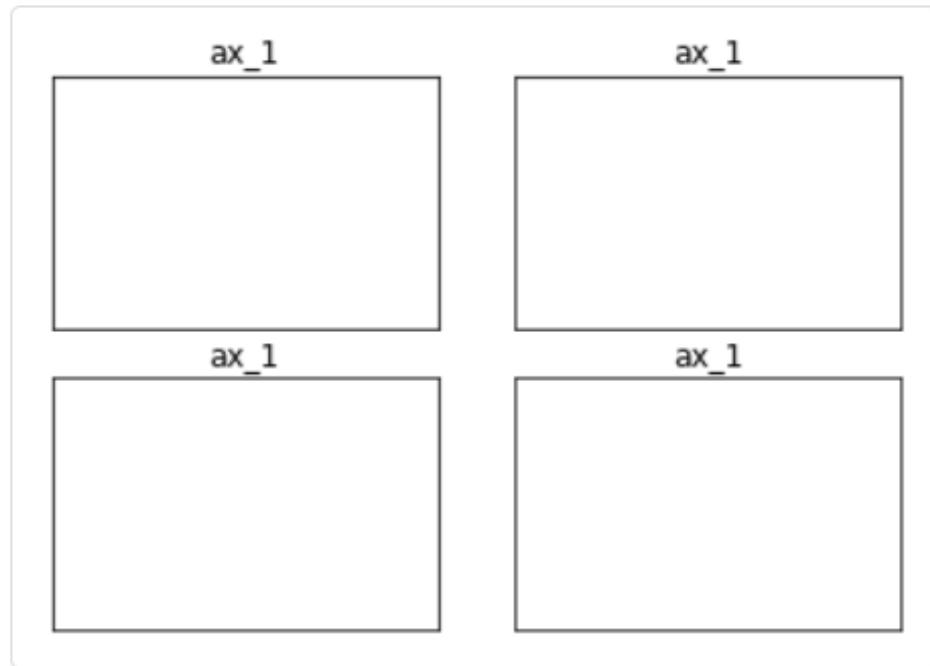
```
import matplotlib.pyplot as plt

fig = plt.figure()

ax_1 = fig.add_subplot(2, 2, 1)
ax_2 = fig.add_subplot(2, 2, 2)
ax_3 = fig.add_subplot(2, 2, 3)
ax_4 = fig.add_subplot(2, 2, 4)

ax_1.set(title = 'ax_1', xticks=[], yticks=[])
ax_2.set(title = 'ax_2', xticks=[], yticks=[])
ax_3.set(title = 'ax_3', xticks=[], yticks=[])
ax_4.set(title = 'ax_4', xticks=[], yticks=[])

plt.show()
```



# Matplotlib Некоторые функции отрисовки

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



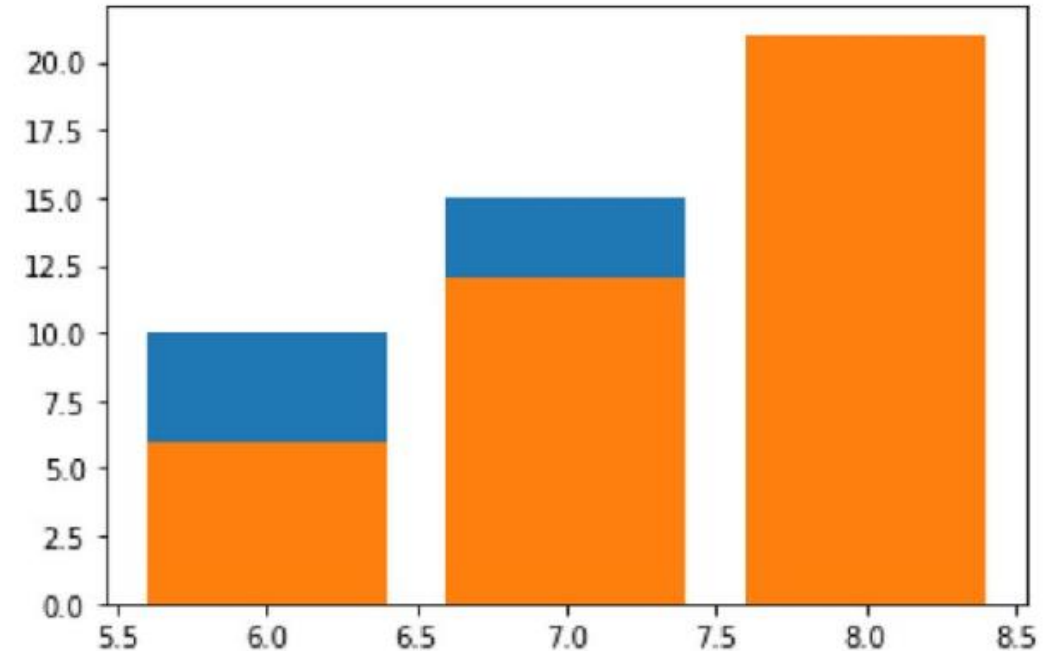
# Matplotlib Вспомогательные функции

- `plt.figure(figsize=(x, y))` — создать график размера (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.

Для визуализации категориальных данных хорошо подходят столбчатые диаграммы.

- Для их построения используются функции:
- `bar()` - вертикальная столбчатая диаграмма
- `barh()` - горизонтальная столбчатая диаграмма

```
import matplotlib.pyplot as plt  
plt.bar([6, 7, 8], [10, 15, 21])  
plt.bar([6, 7, 8], [6, 12, 21])  
plt.show()
```

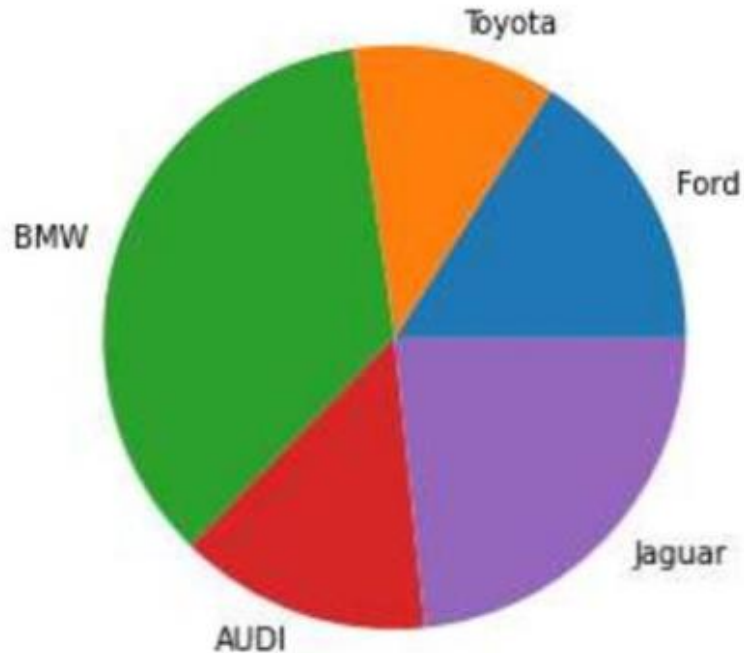




# Круговые диаграммы.

- Для построения круговых диаграмм в Matplotlib используется функция `pie()`

```
import matplotlib.pyplot as plt
vals = [24, 17, 53, 21, 35]
labels = ['Ford', 'Toyota', 'BMW', 'AUDI', 'Jaguar']
fig, ax = plt.subplots()
ax.pie(vals, labels=labels)
ax.axis('equal')
```





## *Модуль `sqlite3` и его назначение*

Работа с базами данных предполагает использование структурированного языка запросов – **SQL (Structured Query Language)**, который ориентирован на выполнение операций над данными.  
Для выполнения **SQL-запросов** применяется приложение «**sqlite3.exe**», позволяющее работать с SQLite из командной строки

Указанное приложение загружается с сайта: <http://www.sqlite.org/download.html>, где в разделе «Precompiled Binaries for Windows» необходимо выбрать архив, соответствующий разрядности установленной на компьютер операционной системы, загрузить его и распаковать.

После чего нужно скопировать хранящийся в распакованном архиве файл «**sqlite3.exe**» в каталог, предназначенный для дальнейшей работы, в данном случае таковым является предварительно созданный каталог «C:\lesson».

В меню «Пуск» в строке поиска следует ввести команду «**cmd**» и кликнуть по появившейся иконке, в результате чего откроется окно с приглашением для ввода команд.



## Модуль *sqlite3* и его назначение

1. Нужно перейти в каталог «*C:\lesson*», выполнив команду «*cd C:\lesson*». В командной строке появится приглашение: «*C:\lesson>*». По умолчанию в консоли используется кодировка «*cp 866*». Для изменения кодировки на «*cp 1251*» нужно ввести команду:

*chcp 1251*

2. Необходимо изменить название шрифта, поскольку точечные шрифты не поддерживают кодировку «*Windows-1251*». Для этого следует кликнуть правой кнопкой мыши на заголовке окна и из контекстного меню выбрать пункт «*Свойства*». Перейти на вкладку «*Шрифт*» открывшегося окна и выбрать пункт «*Lucida Console*», также можно изменить размер шрифта. После чего нужно нажать на кнопку «*ОК*», для сохранения всех изменений. Проверить правильность установки кодировки можно посредством команды «*chcp*». Результат выполнения должен иметь следующий вид:

*C:\lesson>chcp*

*Текущая кодовая страница: 1251*

После выполнения всех указанных действий можно переходить к созданию новой базы данных, что осуществляется командой:

*C:\lesson>sqlite3.exe onedb.db*

```
C:\Windows\system32\cmd.exe - sqlite3.exe onedb.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>chcp 1251
Текущая кодовая страница: 1251

C:\lesson>sqlite3.exe onedb.db
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
sqlite> _
```



## *Модуль `sqlite3` и его назначение*

Если файл «*onedb.db*» не существует, то будет создана и открыта для дальнейшей работы база данных с таким именем. В случае, если такая база данных уже существует, то она просто откроется без удаления содержимого.

Строка «*sqlite>*» здесь является приглашением для ввода команд. Каждая команда завершается точкой с запятой.

```
C:\Windows\system32\cmd.exe - sqlite3.exe onedb.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>chcp 1251
Текущая кодовая страница: 1251

C:\lesson>sqlite3.exe onedb.db
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
sqlite> _
```

*SQLite* позволяет использовать однострочные и многострочные комментарии:

`sqlite> -- Это однострочный комментарий`

`sqlite> /* Это многострочный комментарий */`



## Создание и заполнение базы данных в *sqlite3*

```
C:\Users\солнышко>cd C:\lesson
C:\lesson>chcp 1251
Текущая кодовая страница: 1251
C:\lesson>sqlite3.exe one.db
sqlite> CREATE TABLE buyer(
...> id_buyer INTEGER PRIMARY KEY
AUTOINCREMENT,
...> name_buyer TEXT);
sqlite> CREATE TABLE supplier(
...> id_supplier INTEGER PRIMARY KEY
AUTOINCREMENT,
...> name_supplier TEXT);
sqlite> CREATE TABLE product(
...> id_buyer INTEGER,
...> id_supplier INTEGER,
...> name_product TEXT);
```

```
sqlite> INSERT INTO buyer
(name_buyer)
...> VALUES ("Иванов");
sqlite> INSERT INTO supplier
(name_supplier)
...> VALUES ("Петров");
sqlite> INSERT INTO product
(id_buyer, id-supplier, name_buyer)
...> VALUES (1, 1, "телевизор");
```



## *Модуль `sqlite3` и его назначение*

Модуль *`sqiite3`* позволяет работать с базой данных SQLite, он входит в состав стандартной библиотеки Python и в дополнительной установке не нуждается. Подключение данного модуля выполняется следующим образом:

```
>>> import sqlite3
```

### **ПОСЛЕДОВАТЕЛЬНОСТЬ РАБОТЫ С БАЗОЙ ДАННЫХ ВЫГЛЯДИТ СЛЕДУЮЩИМ ОБРАЗОМ:**

1. Производится подключение к базе данных с помощью функции *`connect`* (). Функция возвращает объект соединения, с помощью которого осуществляется дальнейшая работа с базой данных;
2. Создается объект-курсор;
3. Выполняются SQL-запросы и обрабатываются результаты. Перед выполнением первого запроса, который изменяет записи (*`INSERT`*, *`REPLACE`*, *`UPDATE`* и *`DELETE`*), автоматически запускается транзакция;
4. Завершается транзакция или отменяются все изменения в рамках транзакции;
5. Закрывается объект-курсор;.
6. Закрывается соединение с базой данных.



## *Открытие базы данных с помощью модуля `sqlite3`*

Для создания и открытия базы данных служит функция *`connect ()`*

Для закрытия базы данных служит функция *`close ()`*

```
>>> import os # импорт модуля
>>> os.getcwd() # просмотр рабочего каталога
'C:\\Program Files\\Python38'
>>> os.chdir("C:\\lesson\\") # изменение рабочего каталога
>>> os.getcwd() # просмотр рабочего каталога
'C:\\lesson'
>>> import sqlite3 # импорт модуля
>>> con = sqlite3.connect("one.db") # открытие базы данных
>>> con.close () # закрытие базы данных
```



## Создание базы данных с помощью модуля *sqlite3*

Для создания и открытия базы данных служит функция *connect ()*

Для закрытия базы данных служит функция *close ()*

```
>>> import os
>>> os.chdir("C:\\lesson\\")
>>> import sqlite3
>>> con = sqlite3.connect("new_file.db") # создание базы данных
>>> cursor = con.cursor() # создание объекта-курсора
>>> cursor.execute ("\"\"\"CREATE TABLE groups(number INTEGER,
name TEXT, old TEXT)\"\"") # создание таблицы
<sqlite3.Cursor object at 0x0000000002FEAB20>
```

После создания объекта соединения необходимо создать **ОБЪЕКТ-КУРСОР**. Все дальнейшие запросы должны производиться через этот объект. Создание объекта-курсора производится с помощью метода *cursor()*

### ДЛЯ ВЫПОЛНЕНИЯ ЗАПРОСА К БАЗЕ ДАННЫХ ПРЕДНАЗНАЧЕНЫ СЛЕДУЮЩИЕ МЕТОДЫ ОБЪЕКТА-КУРСОРА:

- 1) *close ()* – закрывает объект-курсор;
- 2) *executescript ()* – выполняет несколько SQL-запросов за один раз;
- 3) *execute ()* – выполняет один SQL-запрос;
- 4) *executemany()* – выполняет SQL-запрос несколько раз, при этом подставляя значения из последовательности





## Создание запросов с помощью модуля *sqlite3*

```
>>> cursor.execute("""INSERT INTO groups VALUES (12, 'Иванов', 18)""") # вставка данных в
таблицу
<sqlite3.Cursor object at 0x0000000002FEAB20>
>>> con.commit() # сохранение изменений
>>> groups = [(13, 'Петров', 19), (14, 'Сидоров', 21), (15, 'Степанов', 25)] # Вставка
множества данных в таблицу, используя безопасный метод "?"
>>> cursor.executemany("INSERT INTO groups VALUES (?, ?, ?)", groups)
<sqlite3.Cursor object at 0x0000000002FEAB20>
>>> con.commit() # сохранение изменений
```

```
C:\Windows\system32\cmd.exe - sqlite3.exe new_file.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>sqlite3.exe new_file.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM groups;
12|Иванов|18
13|Петров|19
14|Сидоров|21
15|Степанов|25
sqlite> _
```



## Редактирование записей

```
>>> import sqlite3
>>> con = sqlite3.connect("new_file.db")
>>> cursor = con.cursor()
>>> sql = """
UPDATE groups
SET name = 'Иванов'
"""
>>> cursor.execute(sql)
<sqlite3.Cursor object at 0x0000000002FEACE0>
>>> con.commit()
```

```
C:\Windows\system32\cmd.exe - sqlite3.exe new_file.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>sqlite3.exe new_file.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM groups;
12|Иванов|18
13|Петров|19
14|Сидоров|21
15|Степанов|25
sqlite> _
```

```
C:\Windows\system32\cmd.exe - sqlite3.exe new_file.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

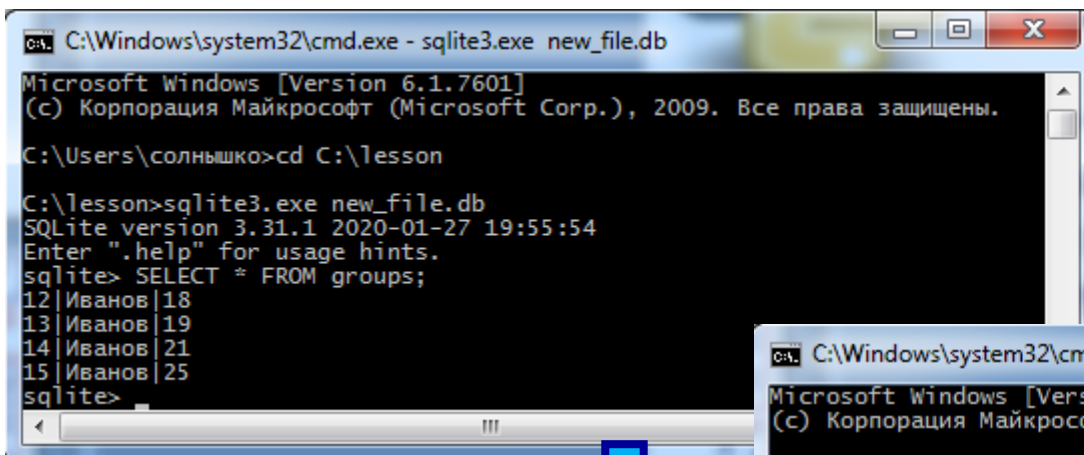
C:\Users\солнышко>cd C:\lesson

C:\lesson>sqlite3.exe new_file.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM groups;
12|Иванов|18
13|Иванов|19
14|Иванов|21
15|Иванов|25
sqlite> _
```



## Удаление записей

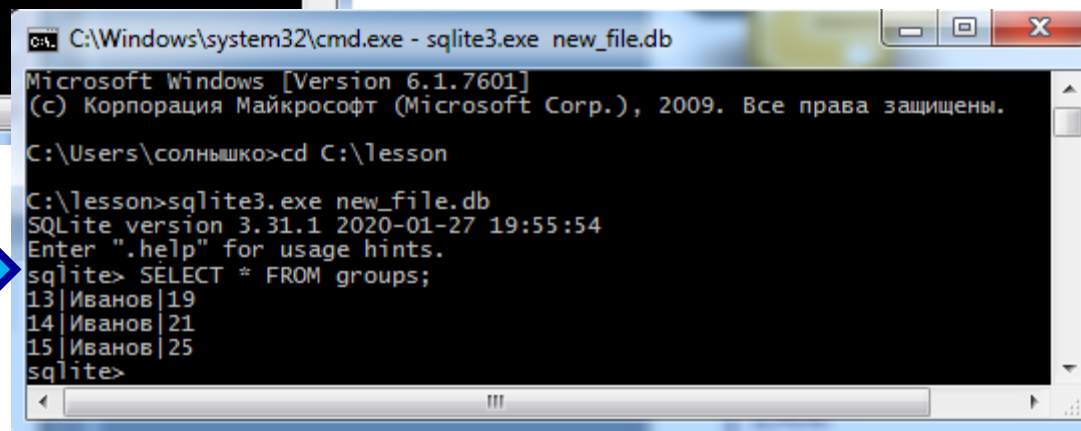
```
>>> import sqlite3
>>> con = sqlite3.connect("new_file.db")
>>> cursor = con.cursor()
>>> sql = "DELETE FROM groups WHERE old = 18"
>>> cursor.execute(sql)
<sqlite3.Cursor object at 0x0000000002FBB8F0>
>>> con.commit()
```



```
C:\Windows\system32\cmd.exe - sqlite3.exe new_file.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>sqlite3.exe new_file.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM groups;
12|Иванов|18
13|Иванов|19
14|Иванов|21
15|Иванов|25
sqlite>
```



```
C:\Windows\system32\cmd.exe - sqlite3.exe new_file.db
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\солнышко>cd C:\lesson

C:\lesson>sqlite3.exe new_file.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> SELECT * FROM groups;
13|Иванов|19
14|Иванов|21
15|Иванов|25
sqlite>
```