**ПРАКТИЧЕСКАЯ РАБОТА №3**

**Визуализация в языке программирования Python**

**Цель**: ознакомится с различными библиотеками визуализации данных (matplotlib, pyplot) и особенностями работы с ними в среде программирования Python.

1. **Matplotlib.**

**Matplotlib** — это обширная библиотека для создания статических, анимированных и интерактивных визуализаций на Python. Matplotlib делает простые вещи простыми, а с ложные возможными. Она является одной из наиболее популярных библиотек для визуализации данных. С её помощью можно создать любой график, для этого представлен широкий функционал. Но для создания презентабельных графиков зачастую необходимо настраивать несколько параметров.

В случае использования пакета Anaconda Matplotlib уже входит в состав предустановленных библиотек. В ином случае интересующую библиотеку можно установить через менеджер пакетов **pip**, введя в командной строке операционной системой **pip install matplotlib** (на Рисунке 1 представлен пример установки библиотеки. В данном случае библиотека уже была установлена, поэтому представлен такой вывод)

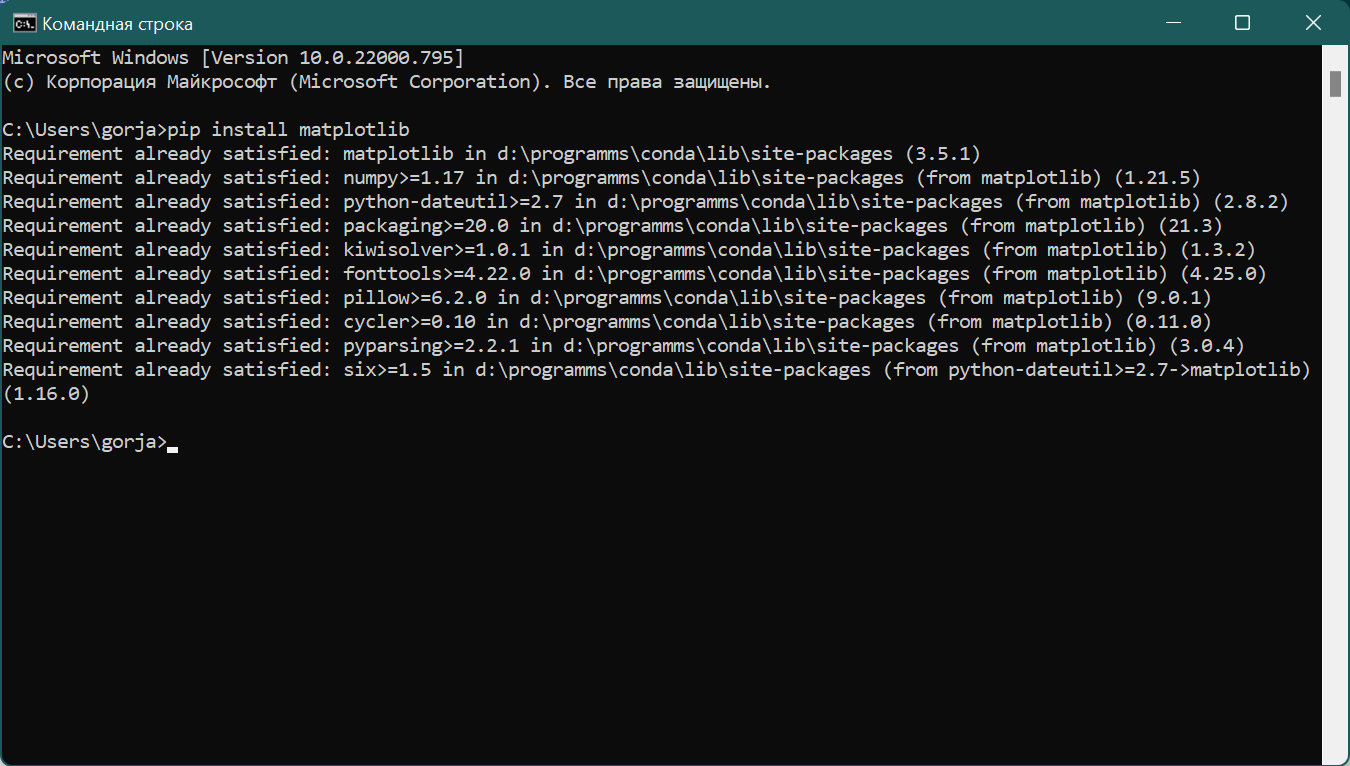


Рисунок 1 - Установка библиотеки Matplotlib

Также при использовании Jupyter Notebook можно установить библиотеку и проверить установлена ли она, например отобразив текущую версию matplotlib.

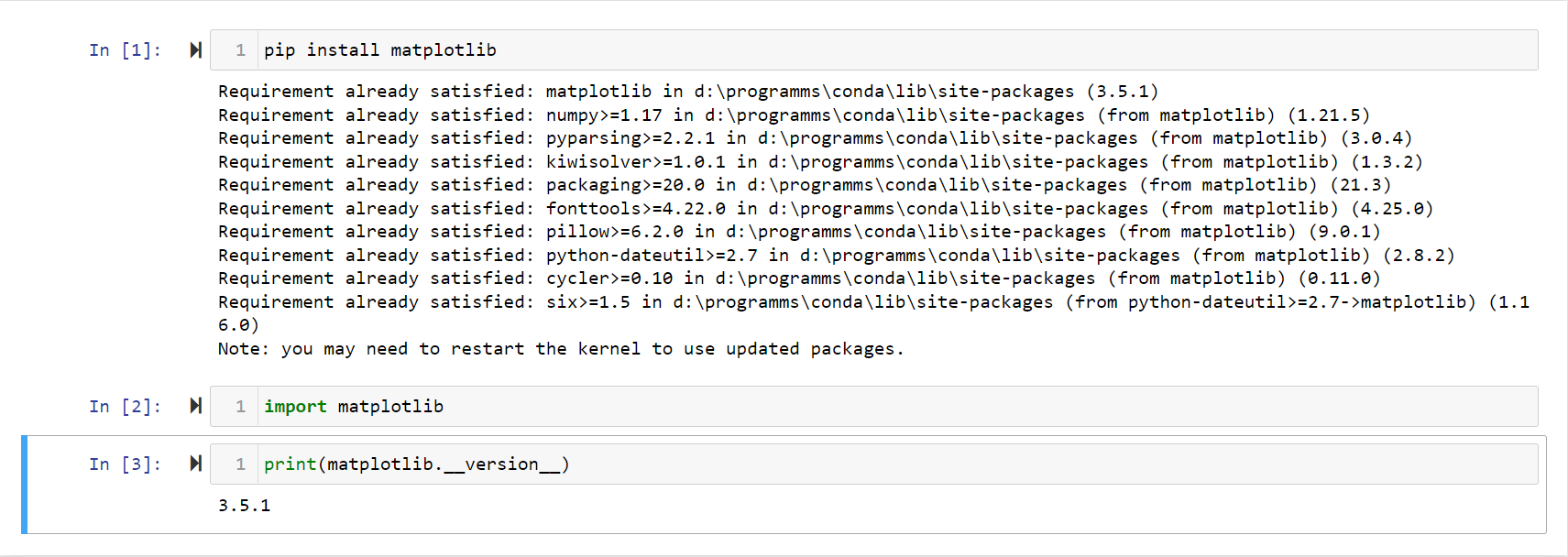


Рисунок 2 - Результат работы программы в Jupyter Notebook

В Matplotlib график разделён на 3 условных части:

1. Рисунок (Figure) — объект самого верхнего уровня, на котором располагаются холст, область рисования, элементы рисунка
2. Область рисования (Axes) — часть изображения, на котором содержатся данные. Может содержать 2 или 3 координатных оси для отображения двух- и трёхмерных данных
3. Координатная ось (Axis) — определяет область изменения данных, например деления и подписи к делениям

Pyplot - интерфейс для построения графиков простых функций.

Рисунки в matplotlib создаются путём последовательного вызова команд. Графические элементы (точки, линии, фигуры и т.д.) наслаиваются одна на другую последовательно. При этом последующие перекрывают предыдущие, если они занимают общее участки на рисунке, но их наслаивание можно регулировать методом **zorder.**

Рисунок создаются при помощи метода **figure()** — желательно с него начинать создавать какой-либо графический объект. Связано это с тем, что работа происходит с определённой областью рисования, в данном случае мы начинаем работу с самого верхнего уровня. Далее выбирается вид графика и для отображения результата используют **show()**. На рисунке 3 представлен пример простого вывода графика

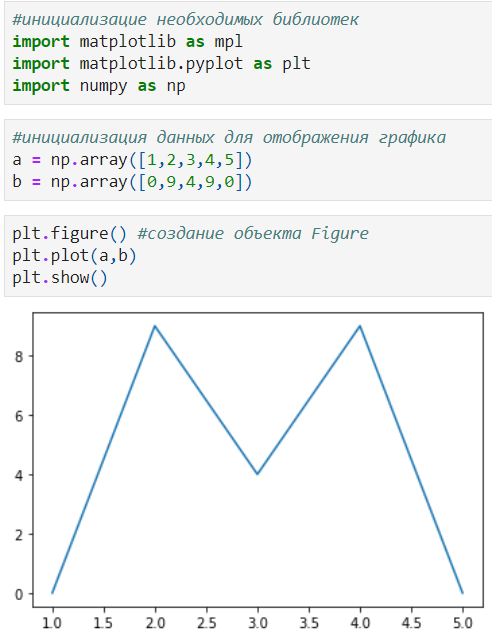


Рисунок 3. Вывод графика Matplotlib в Jyputer Notebook

В Jupyter Notebook есть магические команды, которые позволяют выводить не просто рисунок, а интерактивный график. Такие команды необходимо использовать перед подключением библиотеки, то есть перед **import matplotlib.pyplot as plt**. Одна из команд вывода — **%matplotlib**. В результате отобразится интерактивный график в отдельном окне (Рисунок 4).

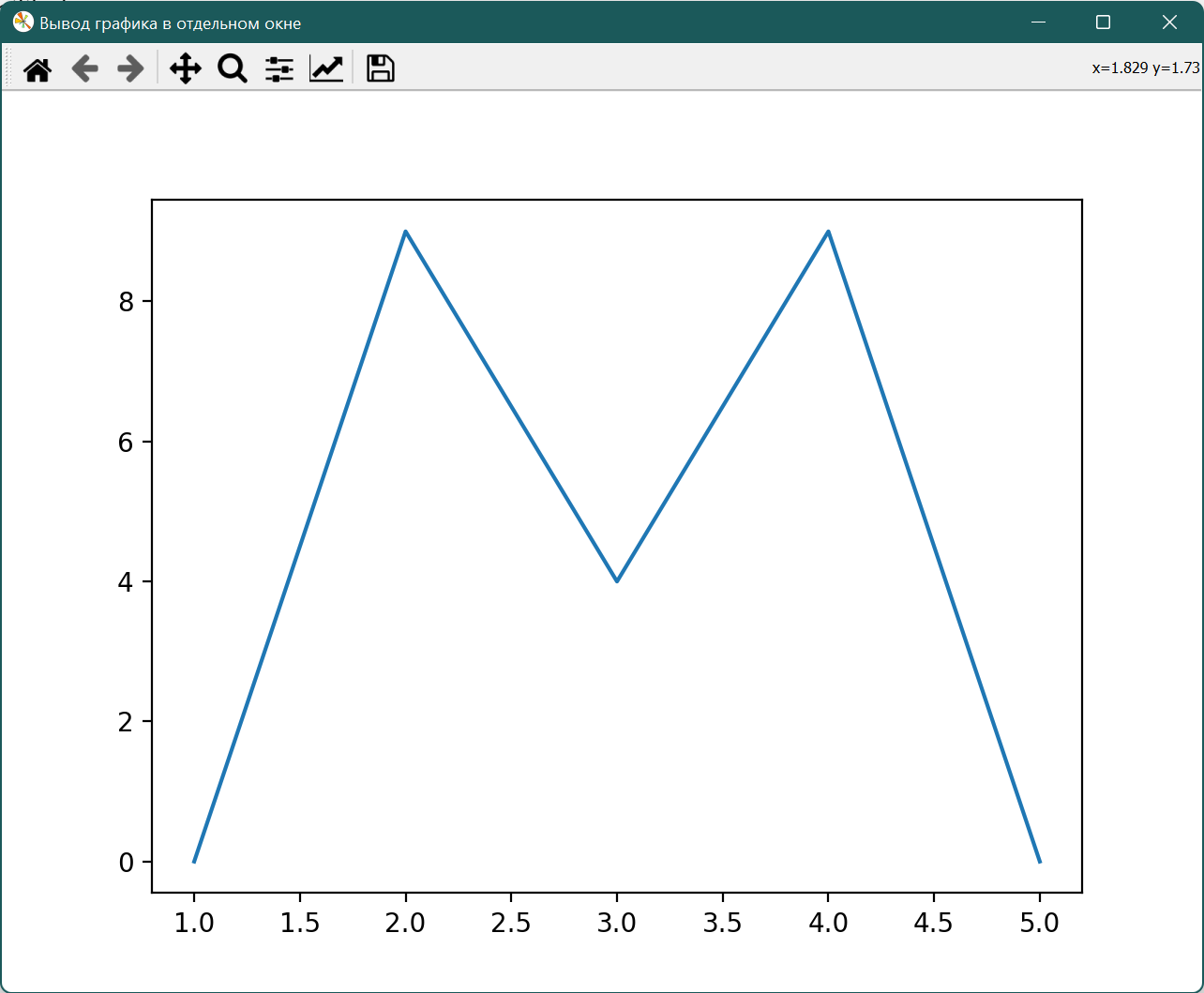


Рисунок 4. Вывод графика в интерактивном режиме в отдельном окне

При данных версиях Python, Jupyter Notebook и matplotlib (Рисунок 5) магическую команду **%matplotlib** (и другие) можно писать и после подключения библиотеки.

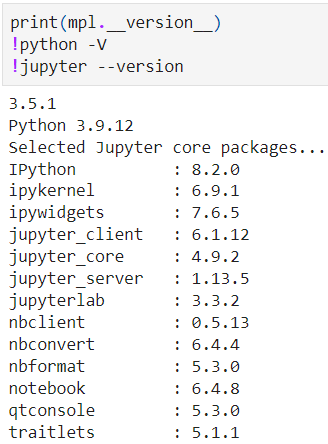


Рисунок 5. Текущие версии

Другим методом отображения интерактивных графиков является **%matplotlib notebook**, в этом случае вывод происходит внутри блокнота (Рисунок 6).

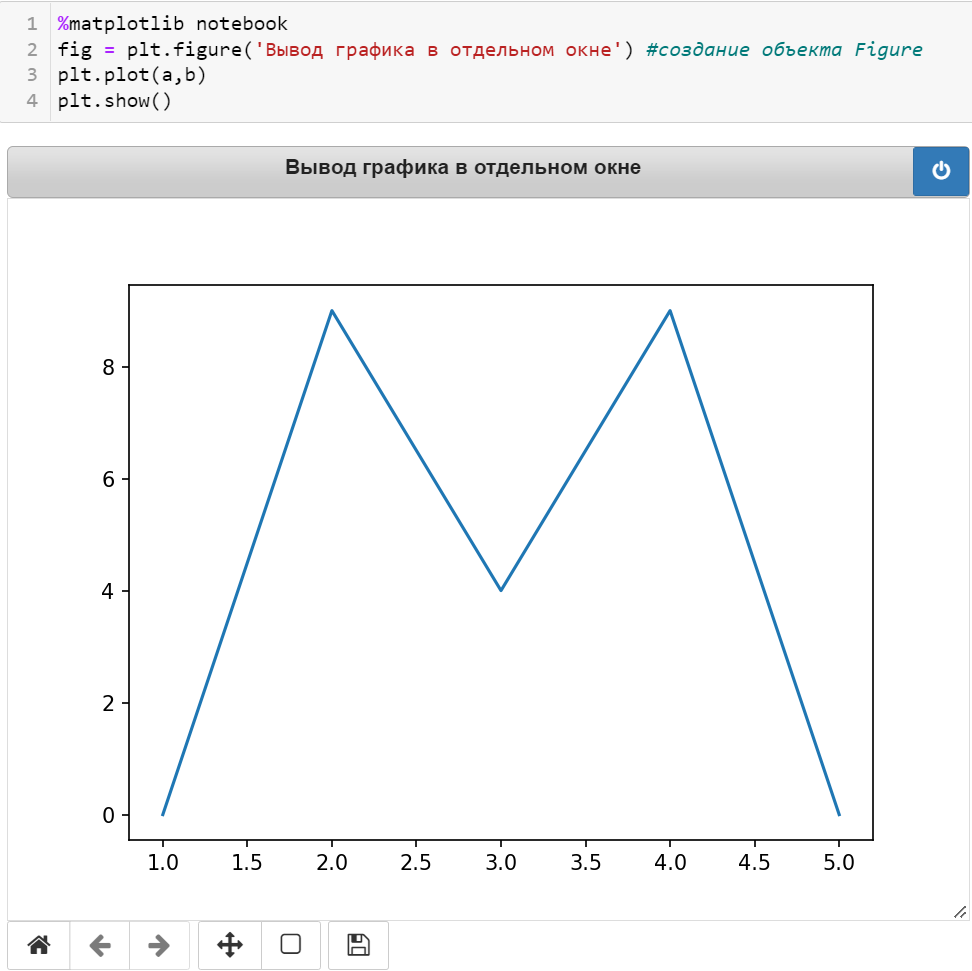


Рисунок 6 - Вывод интерактивного графика внутри Jyputer Notebook

Загрузим данные для дальнейшей работы с графиками (Рисунок 7).

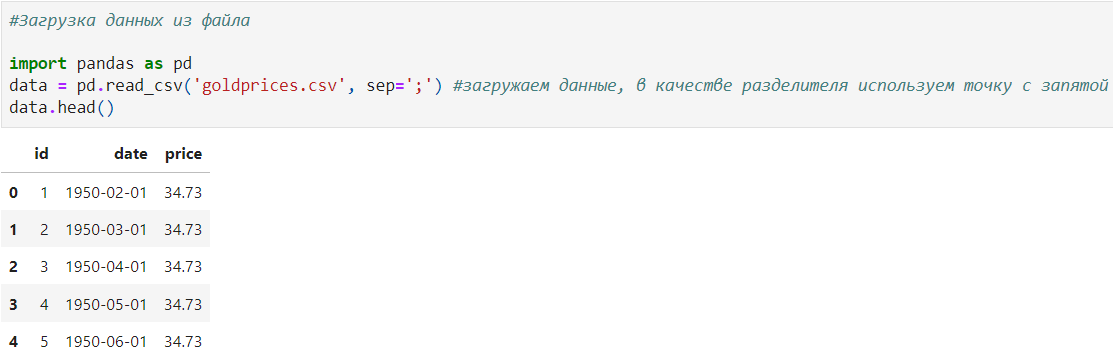


Рисунок 7 - Выгрузка данных из файла

Графики можно настраивать, для этого есть несколько способов:

1. Задать параметры при создании графика, например (Рисунок 8,9):

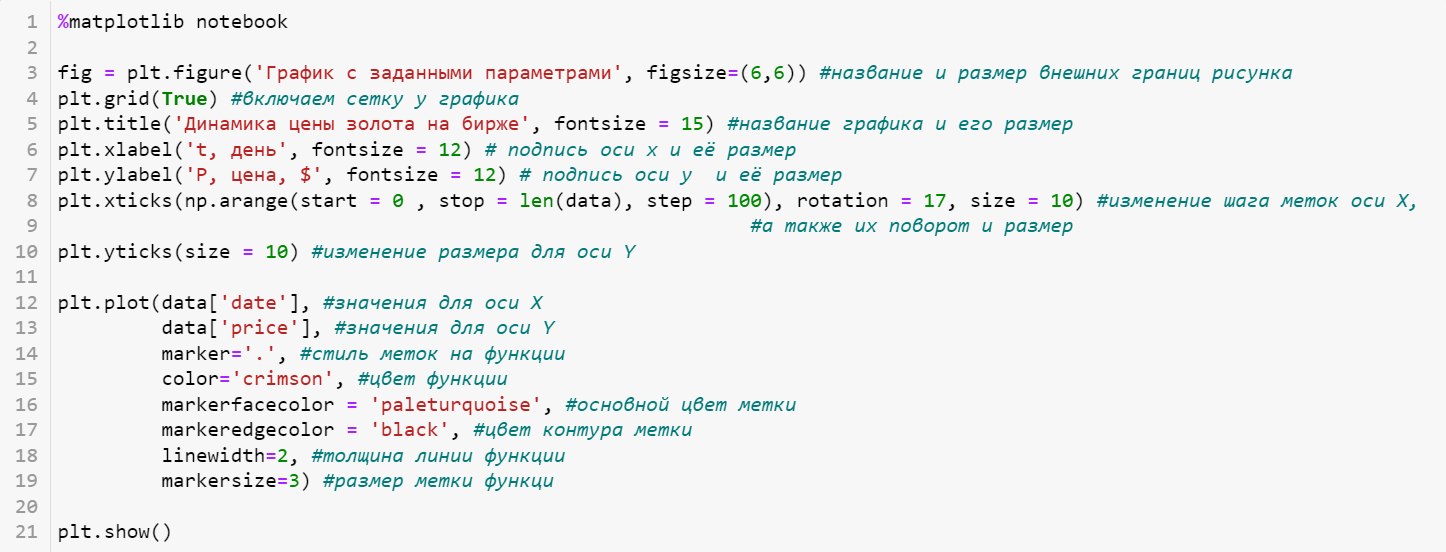


Рисунок 8 - Параметры графика



Рисунок 9 – Результат на основе параметров

1. Задать параметры графика заранее, например (Рисунок 10,11):



Рисунок 10 – Параметры графика



Рисунок 11 - Результат на основе параметров

В таком случае эти параметры буду применены ко всем графикам, которые вы будете создавать далее. Больше настраиваемых параметров вы можете найти на официальном сайте [Matplotlib](https://matplotlib.org/stable/tutorials/introductory/customizing.html).

Для вывода нескольких функций на одном полотне достаточно задать значения для каждой из них (Рисунок 12)

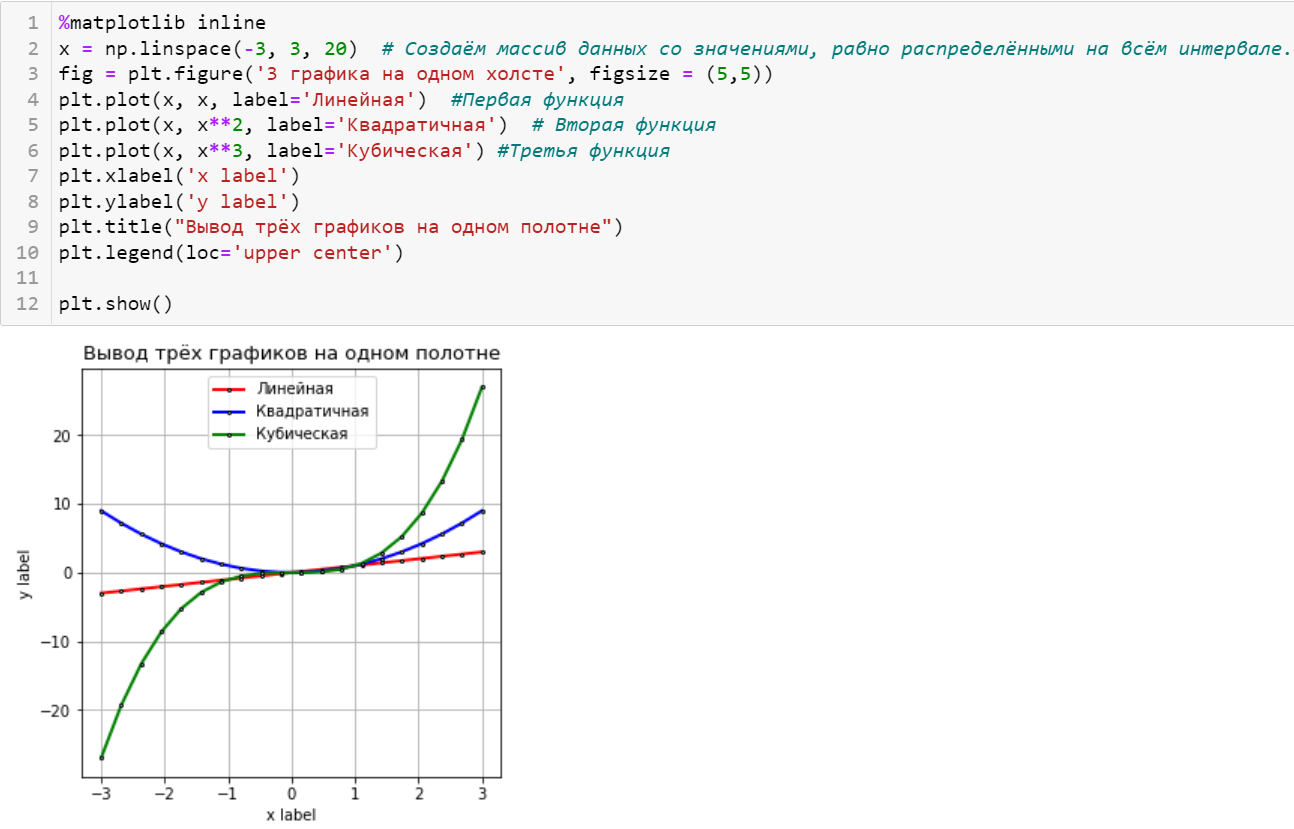


Рисунок 12 - Три функции на одном полотне

Для вывода нескольких функций раздельно необходимо дописать команду **subplots** (Рисунок 12):

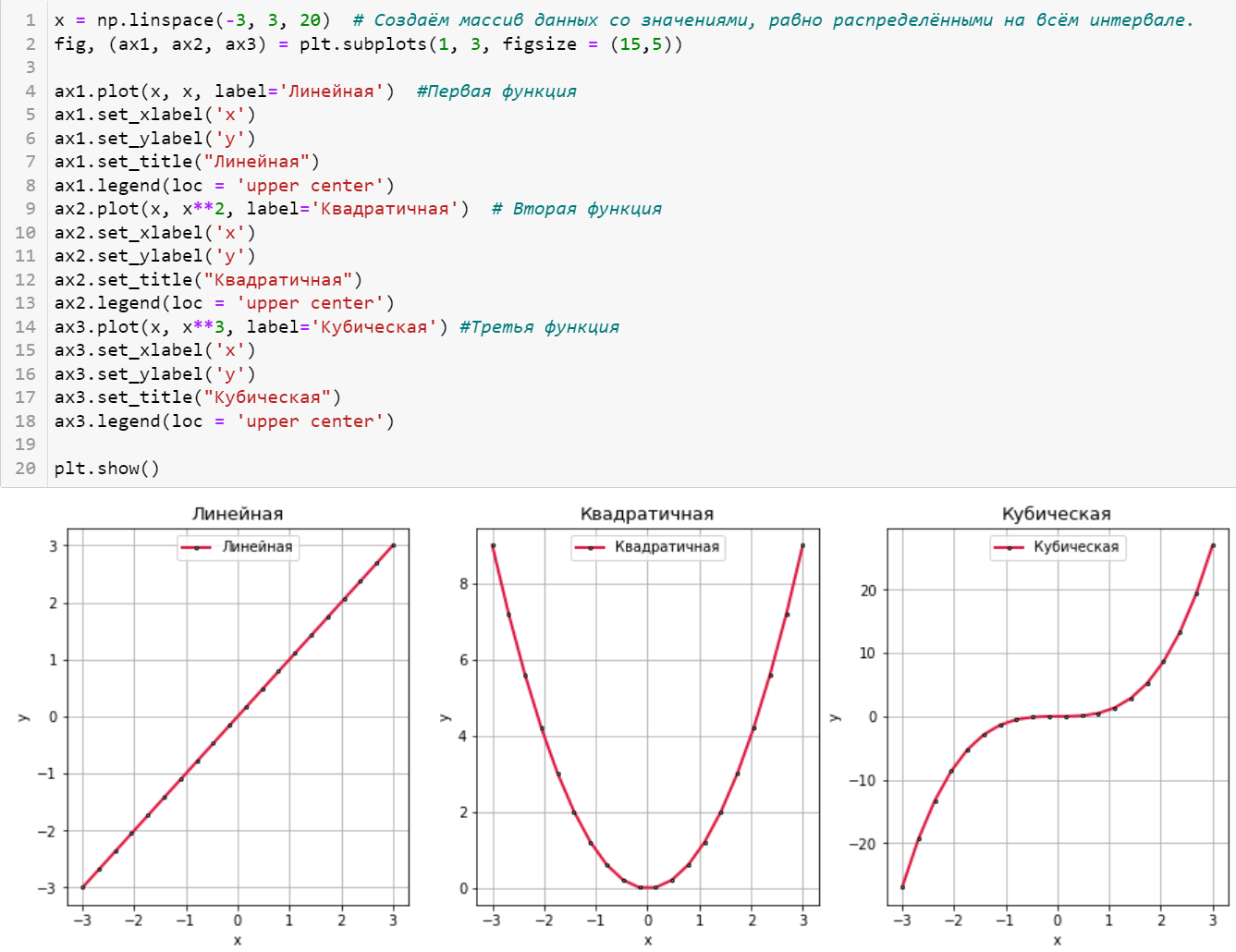


Рисунок 13. Разделённые графики

В скобках, в строчке под номером 2 указанные переменные образуют кортеж, к которому далее идёт обращение для настройки каждого из отдельных графиков, также можно указать одну переменную и обращаться к графику при помощи индекса.

Также можно создавать 3d графики, указав дополнительную ось (Рисунок 14).

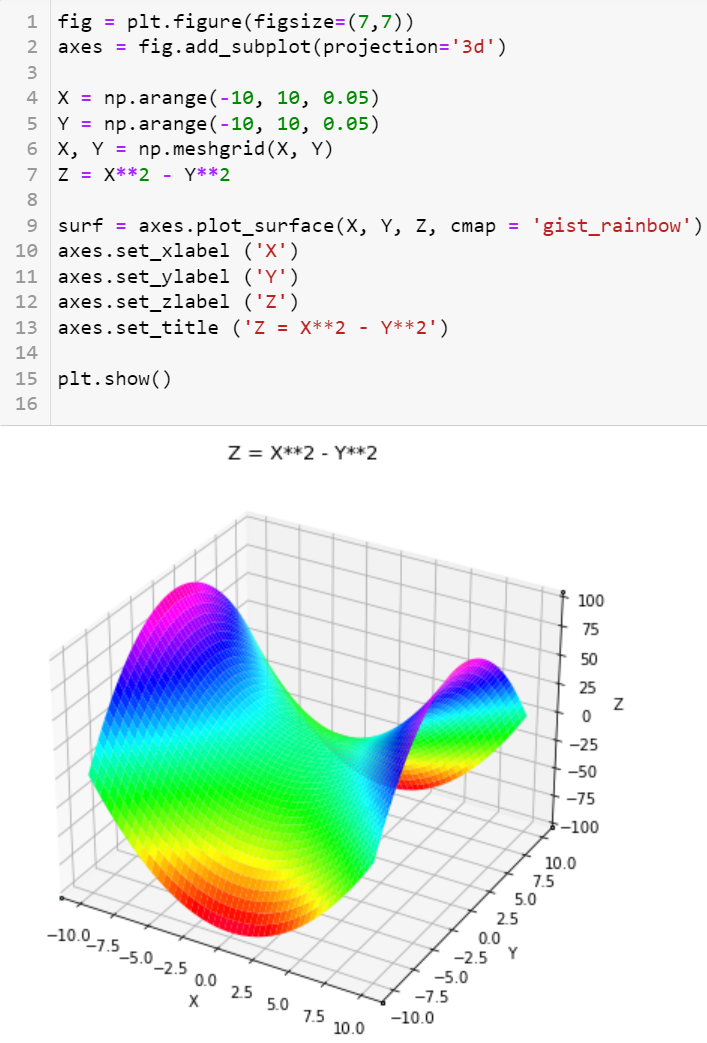


Рисунок 14

Другим распространённым видом графика является **scatter()** (Рисунок 15) Как правило, эта диаграмма требует незначительного изменения внешнего вида. Однако иногда видимость данных сильно ограничена из-за небольшого размера диаграммы, размеров и цветов используемых маркеров.

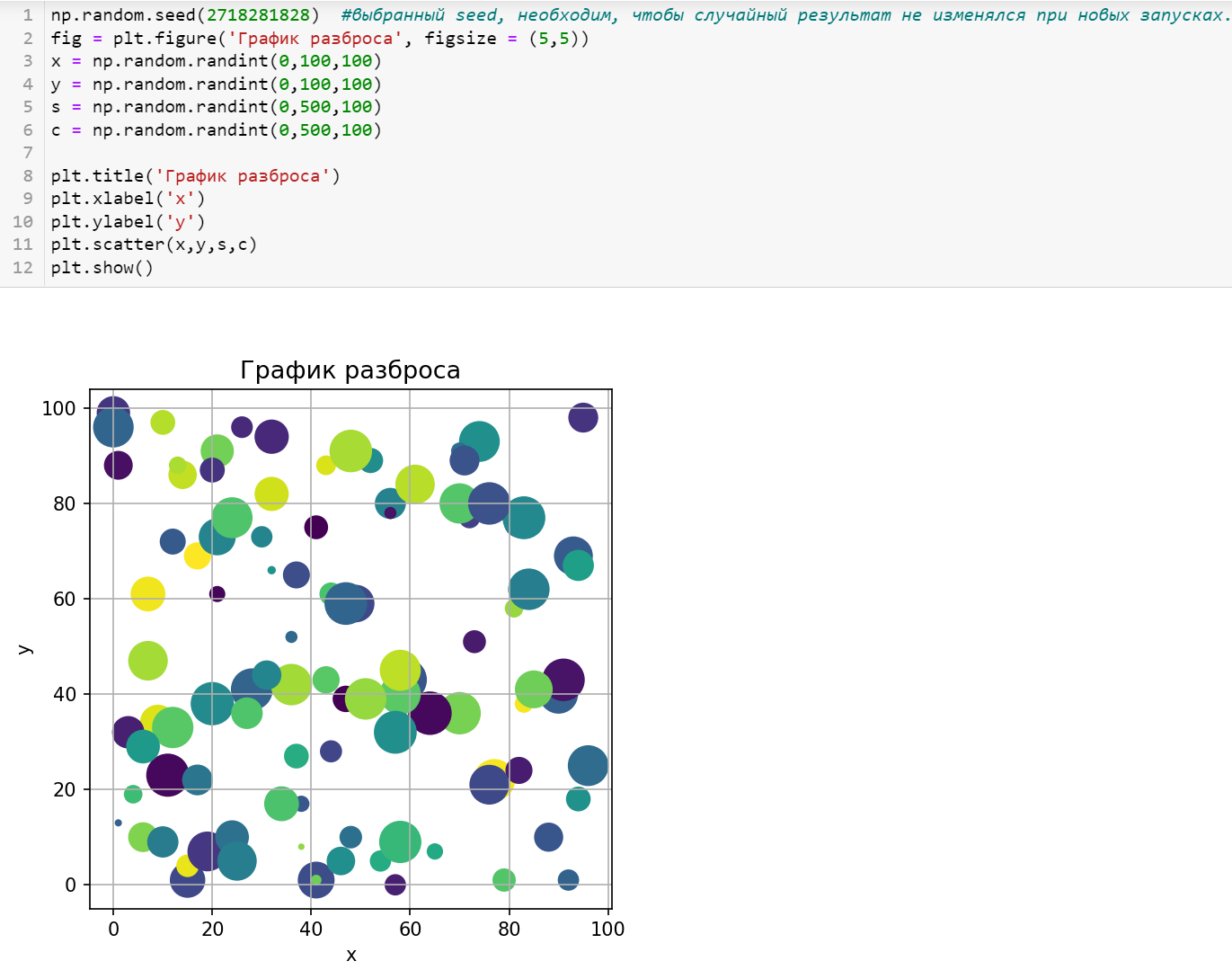


Рисунок 15

В данном случае график является четырёхмерным. Первые две размерности определяются осями, третья — размером окружностей, а четвёртая — цветом.

Ещё один часто используемая визуализация — столбчатые диаграммы **bar()**. В основном они используется для отображения категориальных данных. Простой пример отображения вертикальной и горизонтальной диаграммы представлен на Рисунке 16.

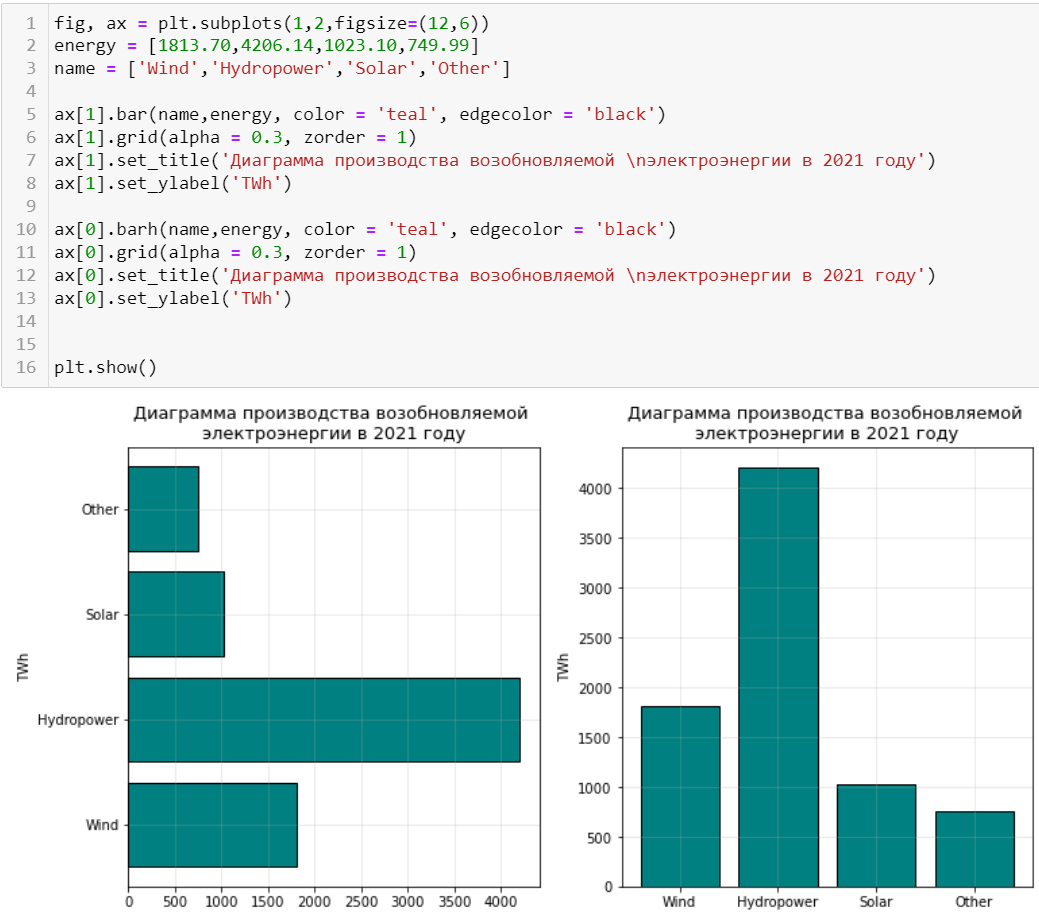


Рисунок 16.

1. Plotly

Это библиотека с открытым исходным кодом, которую можно использовать для визуализации данных. Plotly поддерживает различные типы графиков и вы с большой вероятностью получите оптимальный для восприятия результат, не настраивая дополнительно параметры. Также компания разрабатывает и предоставляет библиотеки научного построения графиков для Arduino, Julia, MATLAB, Perl, Python, R и REST, что позволяет работать и в других языках программирования.

Перед началом работы необходимо установить библиотеку командой **!pip install plotly (!conda install plotly)**, так как она не входит в стандартный пакет. После этого вы можете убедиться, что пакет установился корректно (Рисунок 17).

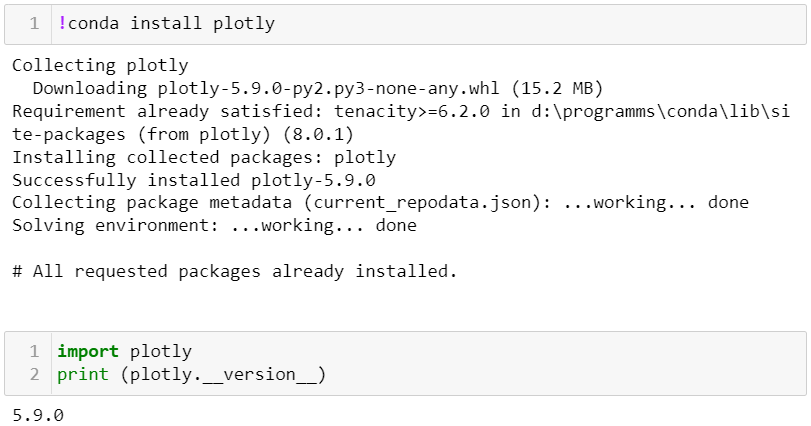


Рисунок 17

Для работы с Plotly понадобятся и дополнительные модули, входящие в библиотеку, поэтому извлечём их и будем обращаться к ним через другие переменные для удобства работы.

**import plotly.graph\_objs as go** — содержит объекты (рисунок, макет, данные и определение графиков, таких как диаграмма рассеяния, линейный график), которые отвечают за создание графиков.

**import plotly.express as px** — модуль plotly.express может создать весь рисунок за один раз. Он автоматически использует graph\_objects и возвращает graph\_objects.

Попробуем вывести рабочую область



Рисунок 18

Как мы видим, здесь уже есть отличие с matplotlib. В первом случае создаётся просто полотно без осей. В Plotly сразу создаётся рабочая плоскость с двумя осями.

Попробуем отобразить предыдущий график. В случае, если необходимо быстро отрисовать график, и проанализировать показатели, можно воспользоваться экспресс-модулем plotly.express, который самостоятельно создаёт графические объекты. Он позволяет уменьшить количество кода, но при этом необходимо предобработать данные. Для этого надо указать тип графика, а в скобках передать его параметры. В частности, к какому объекту обратиться, чтобы взять данные. В нашем случае это DataFrame. И указать набор данных для осей абсцисс и ординат, которые будут использоваться как подписи данных и в маркерах, при наведении на точки графика.

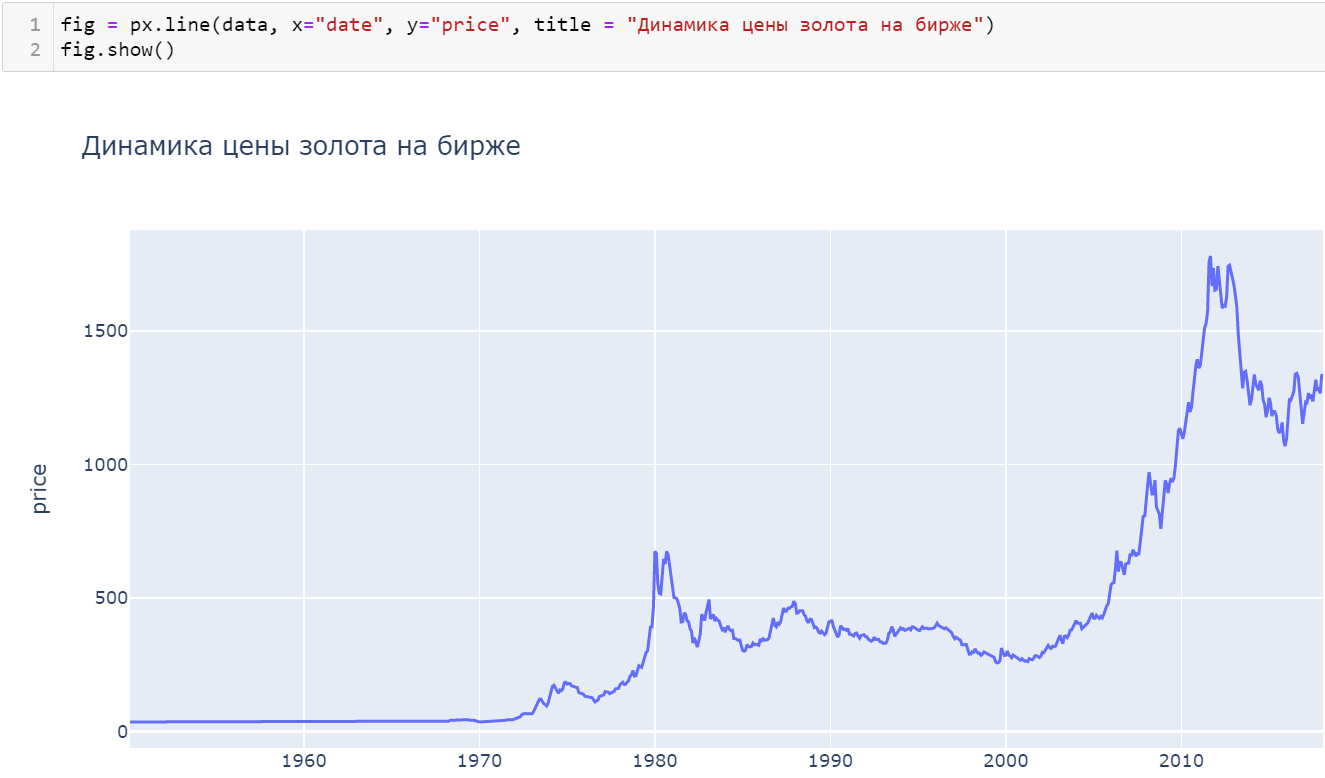
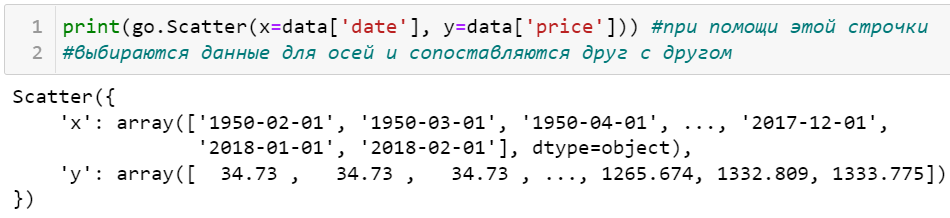


Рисунок 19

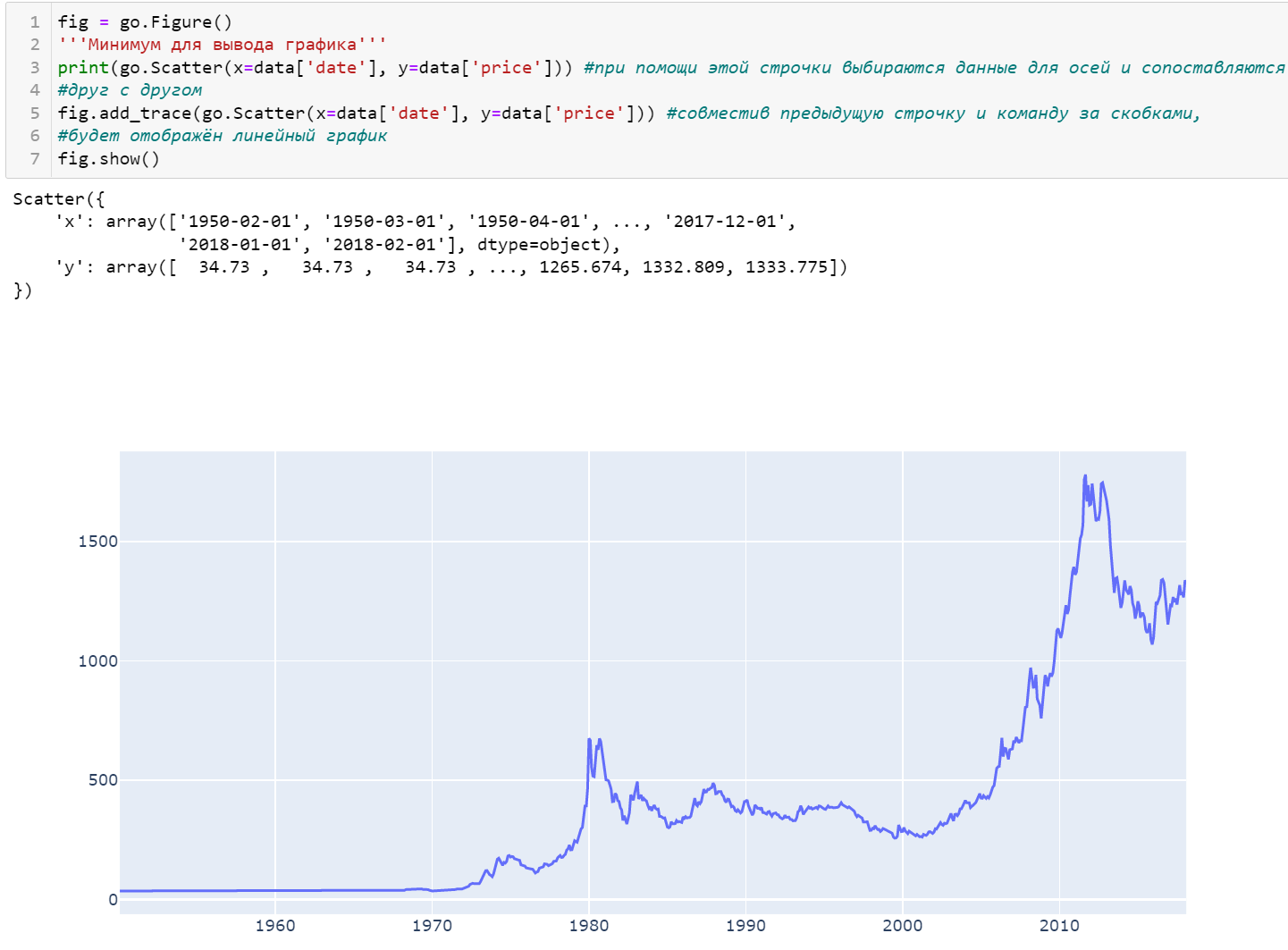
У Plotly есть возможности инструмента наведения, которые позволяют нам обнаруживать любые отклонения или аномалии в большом количестве точек данных. В данном случае отображается интерактивный график, с которым вы сразу можете взаимодействовать. При наведении курсора на функцию, вам отображаются координаты каждой из точек.

Но в экспресс-модуле Plotly может не хватать функционала для работы с графиками, например, при создании соосных объектов, фасеточных графиков с разными типами данных, дашбордов (создание нескольких графиков на одном графическом окне), также нельзя создать 3D фигуры с его помощью. Поэтому попробуем отобразить самый простой график рассеяния, используя graph\_objs. Посмотрим, как в нём сопоставляются данные:



Мы видим, что данная структура напоминает словарь в языке Python. То есть фигуры в Plotly могут быть представлены в виде словаря, как и другие настройки. Но обычно используются словари не в чистом виде, а используются специальные графические объекты (graph objects), которые применяются для представления фигур и имеют преимущества перед словарями, например обеспечивается проверка на ошибку. При неверном введение имени свойства или его значения, будет вызвано исключение, понятное пользователю, а также поддерживаются вспомогательные функции более высокого уровня для обновления уже построенных фигур.

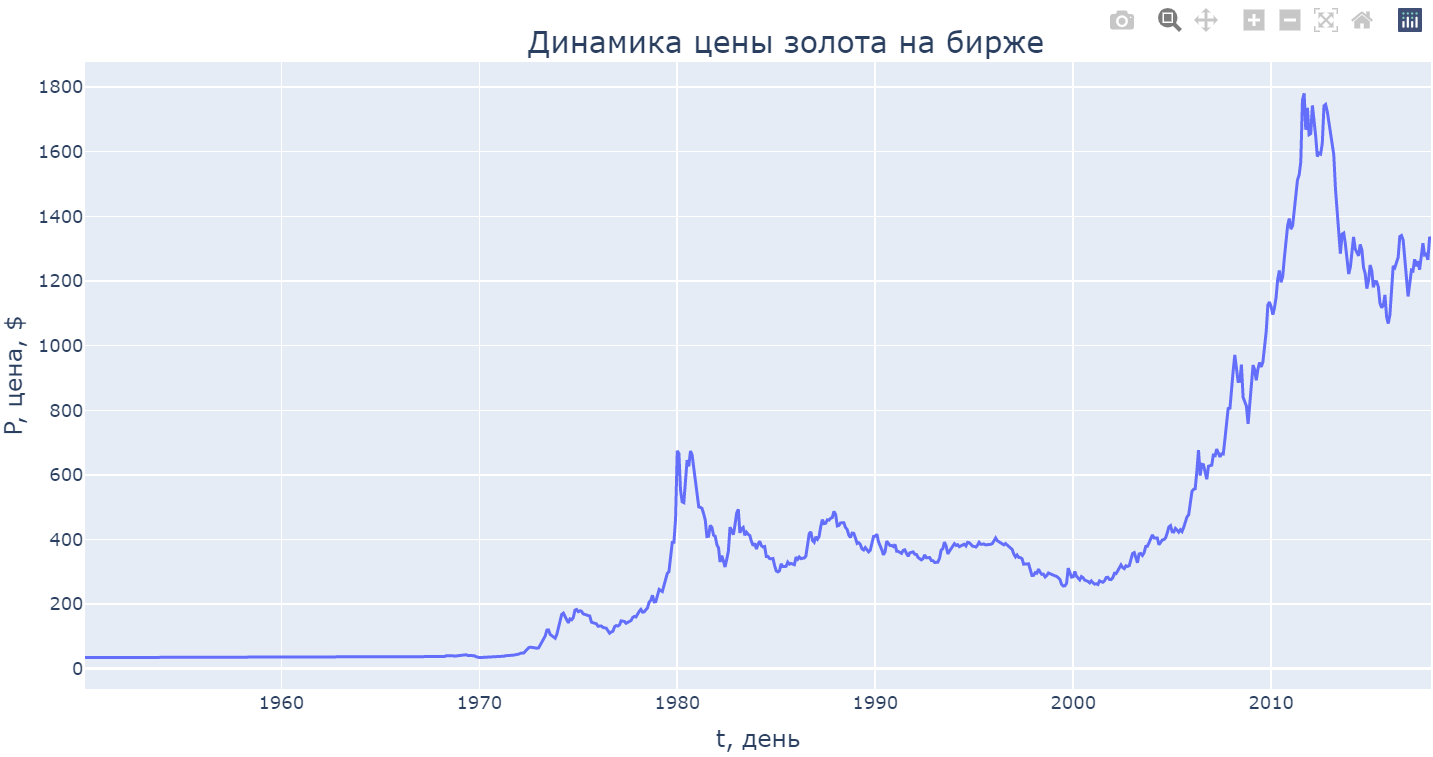
Теперь добавим команду для добавления графического объекта к фигуре (add\_trace) и команду отображения графика (fig.show()).



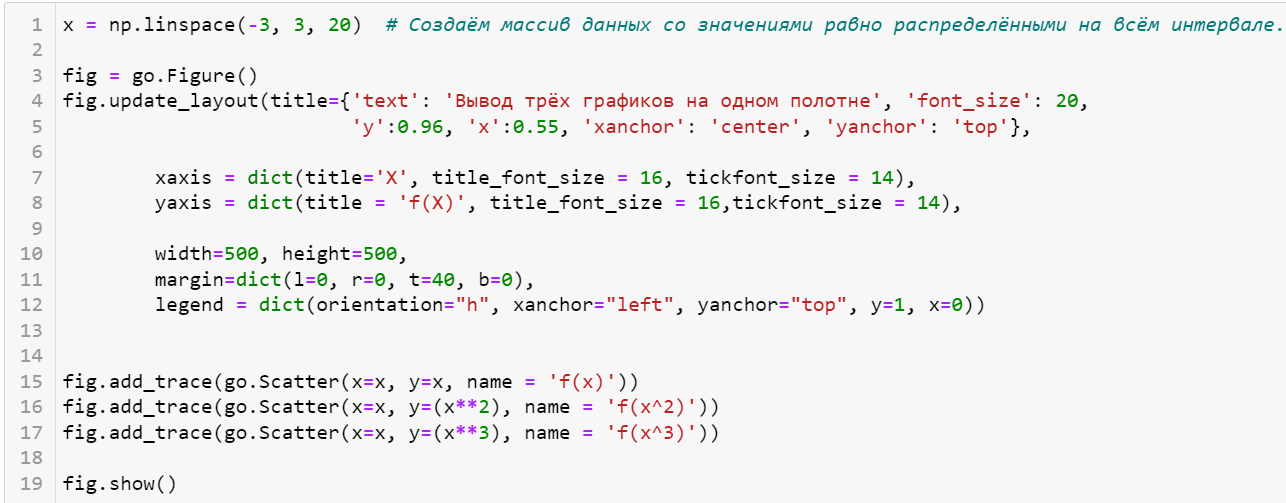
Обновим макет графика, используя дополнительные параметры.   
На скриншоте ниже в 8 строчке вводится команда изменения макета, а в скобках указываются название графика, расположение названия (по оси y и x, а также его команды для его выравнивания по центру сверху с размером текста, равным 20). На 11 и 12 строках указываются названия и размер текста для осей x и y. На 13 строке размеры рисунка (в данном случае высота равна 500 пиксеоей, а ширина выбирается автоматически, занимая ширину рабочей зоны Jupyter). На 14 строчке кода изменяются начальные отступы рисунка слева, справа, сверху и снизу соответственно, так как изначально в Plotly имеются отступы, которые снижают размеры графика, особенно сильно он виден сверху (рисунок предыдущий).

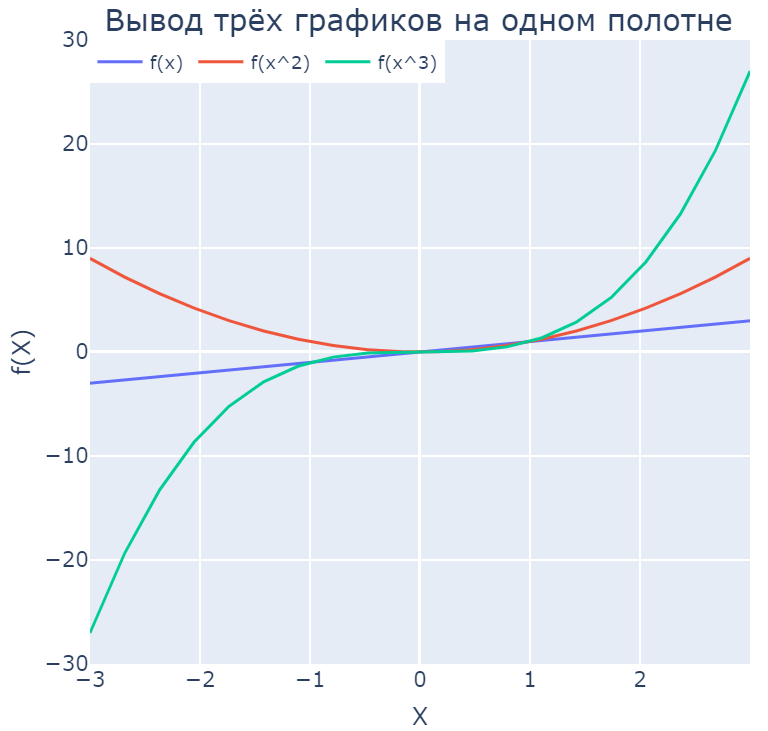


Отобразим получившийся результат.



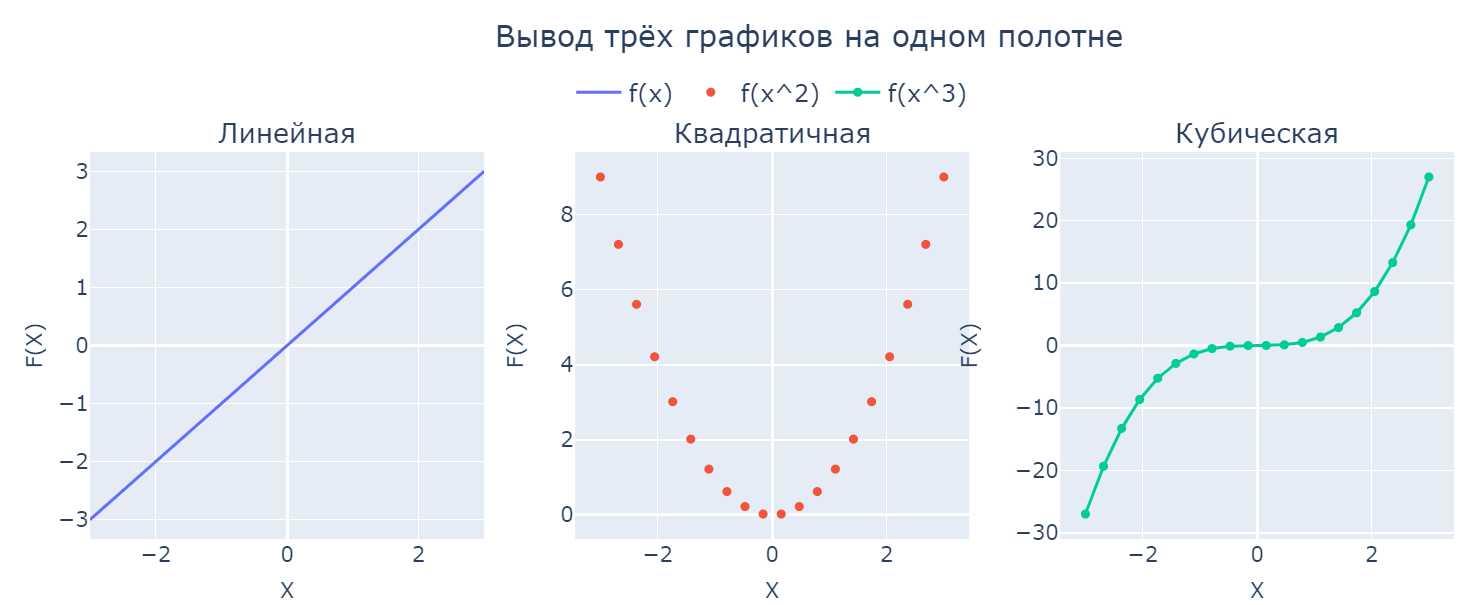
Теперь отобразим 3 графика на одном рисунке. Ниже представлен код. На скриншоте видно, что параметры для заголовка, осей и другие можно задавать через словарь указывая какой объект вы хотите изменить (заголовок, оси и т.д.) и параметры для этого объекта (название, размер текста, положение и т.д.). Для отображения дополнительных функций на одном рисунке достаточно добавлять trace с соответствующими массивами для осей.



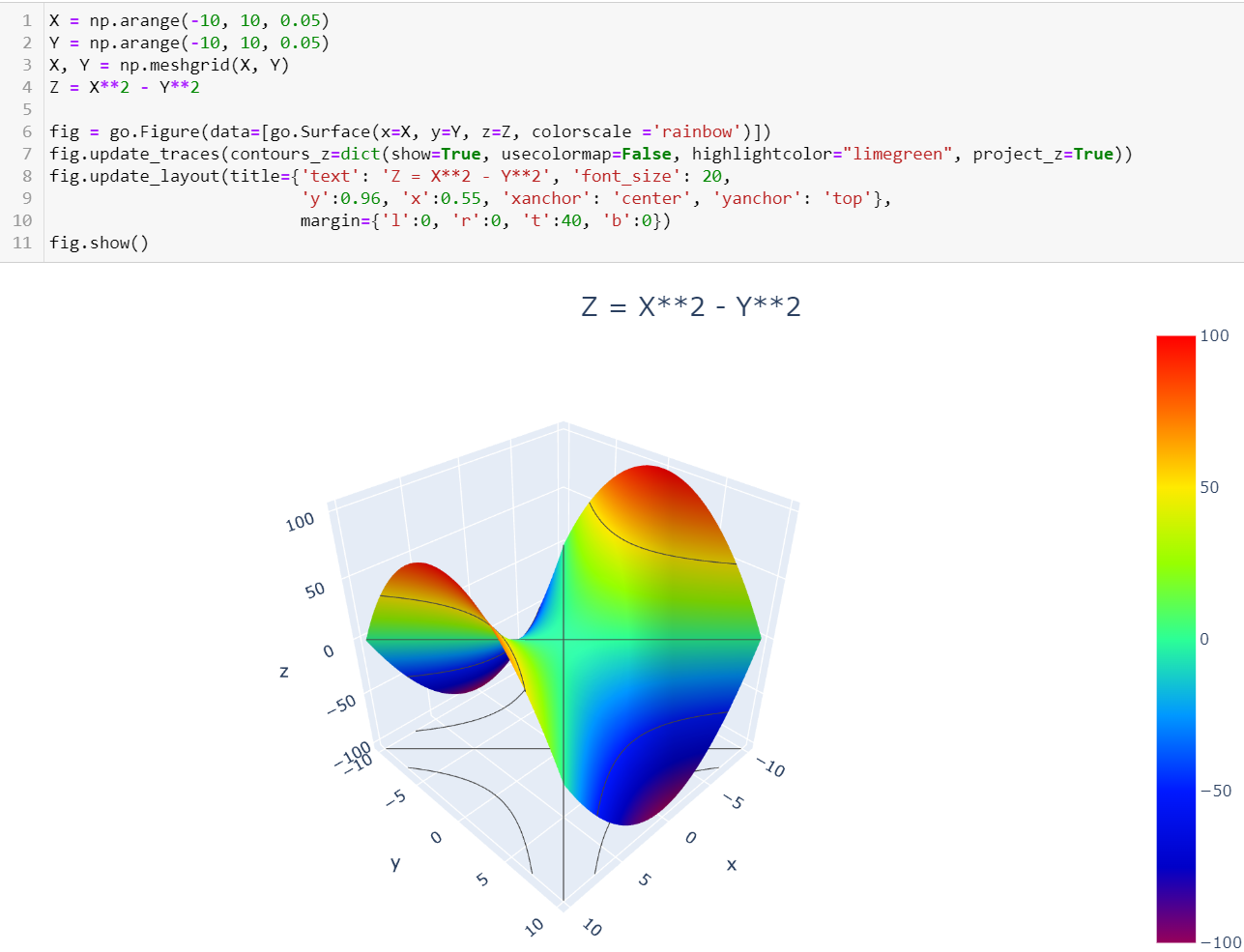


Разделим графики. Аналогично с matplotlib создаём subplots. Созданные ячейки можно объединять, если необходимо изменить их размеры. Для каждого графика можно индивидуально настроить оси, что показано на примере изменения настроек оси X (выбирая строку и колонку ячейки). Или применить настройки ко всем ячейкам (изменение параметров оси Y). Метод add\_trace также принимает номер строки и столбца, для отображения графика в определённой ячейке с данными, передаваемыми в оси, и соответствующими настройками, например для изменения типа линий.





Если необходимо создать трёхмерный график рассеяния, то достаточно обратиться к методу **px.scatter\_3d** и передать в него параметры x, y и z. Ниже представлен пример кода для создания графика поверхности.



Далее представлен код для создания столбчатой диаграммы в Plotly. В экспресс-модуле также можно задать параметры для отображения значений столбцов и значение цвета в зависимости от переданных значений.



К дополнительным возможностям данной библиотеки стоит отнести, что можно размещать интерактивные графики на сайте, с которыми можно взаимодействовать. А также создавать анимации, размещать кнопки, ползунки (слайдеры), выплывающие списки и работать с различными типами визуализации данных. На [сайте Plotly](https://plotly.com/python/) представлены примеры всех графиков и всего функционала, предоставляющего возможности тонкой настройки для создания дашбордов и графиков, понятных для всех.

**Задание**

1. Найти и выгрузить многомерные данные с использованием библиотеки pandas.
2. Вывести информацию о данных при помощи методов **.info()**, .**head()**. Проверить данные на наличие пустых значений. В случае их наличия удалить данные строки или интерполировать пропущенные значения. При необходимости дополнительно предобработать данные для дальнейшей работы с ними.
3. Построить столбчатую диаграмму (**.bar**) с использованием библиотеки Plotly со следующими параметрами:
   1. По оси Х указать дату или название, по оси У указать количественный показатель.
   2. Сделать так, чтобы столбец принимал цвет в зависимости от количественного показателя
   3. Отобразить заголовок диаграммы, разместив его по центру, с 20 размером текста
   4. Добавить подписи для осей X и Y с размером текста, равным 16. Для оси абсцисс развернуть метки так, чтобы они читались снизу вверх. Для оси ординат шаг меток участить в 1.5 раза
   5. Размер текста меток осей сделать равным 14
   6. Обрезать график снизу по минимальному значению количественного показателя, вычтя из него 10%.
4. Построить круговую диаграмму, использовав данные и оформление из предыдущего графика
5. Построить линейный график с накопленными значениями количественного показателя от даты (названия).
   1. Сделать график с линиями и маркерами, цвет линии 'crimson', цвет точек 'darkblue', цвет границ точек 'black'.
   2. Добавить легенду на график в нижнем левом углу
   3. Добавить сетку на график, сделать цвет 'azure'
   4. Остальное оформление сохранить с предыдущих графиков.
6. Построить ящик с усами для количества продаж, сохранив оформление с предыдущих графиков
7. Постараться создать аналогичные графики с использованием библиотеки matplotlib.
8. На основе проделанной работы составить отчёт с описанием и скриншотами полученных результатов, сделать выводы о выбранном организации (процессе) на основе полученных графиков, сравнить библиотеки.