Основы системной инженерии

Оглавление

[ТЕМА 14. Введение в библиотеку Pandas. 2](#_Toc168912463)

[14.1. Установка Pandas 2](#_Toc168912464)

[14.2. Основные компоненты Pandas. Знакомство с Series 2](#_Toc168912465)

[14.3. Знакомство с DataFrame 3](#_Toc168912466)

[14.4. Обращение к DataFrame по индексу 3](#_Toc168912467)

[14.5. Арифметические операции в Series 5](#_Toc168912468)

[14.6. Логические операции к Series. 5](#_Toc168912469)

[14.7. Редактирование таблиц. Создание и удаление столбцов. 6](#_Toc168912470)

[14.8. Удаление столбцов или строк 7](#_Toc168912471)

[14.9. Удаление дубликатов строк. 7](#_Toc168912472)

[14.10. Переименование столбцов 7](#_Toc168912473)

[14.11. Подсчет уникальных значение в DataFrame 8](#_Toc168912474)

[14.12. Загрузка данных и знакомство с методами Pandas. 8](#_Toc168912475)

[14.13. Предварительный осмотр данных 9](#_Toc168912476)

[14.14. Подготовка датасэта к работе 10](#_Toc168912477)

[14.15. Работа с числовыми данными 11](#_Toc168912478)

[14.16. Сортировка данных с помощью Pandas 13](#_Toc168912479)

[14.17. Фильтрация данных 14](#_Toc168912480)

[14.18. Работа с пропущенными значениями 15](#_Toc168912481)

[14.19. Группировка данных и агрегирующие функции. Сводные таблицы 16](#_Toc168912482)

[14.20. Функции в Pandas 18](#_Toc168912483)

[ТЕМА 15. Работа с NumPy 19](#_Toc168912484)

[15.1. Что такое NumPy? 19](#_Toc168912485)

[15.2. Установка NumPy. 19](#_Toc168912486)

[15.3. Создание массивов 19](#_Toc168912487)

[15.4. Сортировка элементов 20](#_Toc168912488)

[15.5. Конкатенация 20](#_Toc168912489)

[15.6. Применение Shape и size 21](#_Toc168912490)

[15.7. Преобразование одномерного массива в двумерный 21](#_Toc168912491)

[15.8. Индексирование и срезы массивов 22](#_Toc168912492)

[ТЕМА 16. Работа с Matplotlib и Seaborn 23](#_Toc168912493)

[16.1. Что такое Matplotlib? 23](#_Toc168912494)

[16.2. Установка Matplotlib. 24](#_Toc168912495)

[16.3. Загрузка данных для учебного проекта 24](#_Toc168912496)

[16.4. Базовые графики Matplotlib и построения 25](#_Toc168912497)

[16.5. Вывод двух графиков в Matplotlib 26](#_Toc168912498)

[16.6. Boxplot в Matplotlib 28](#_Toc168912499)

[16.7. Что такое Seaborn? 28](#_Toc168912500)

[16.8. Установка Seaborn. 29](#_Toc168912501)

[16.9. Базовые графики Seaborn. 29](#_Toc168912502)

[16.10. Ядерная оценка плотности. 30](#_Toc168912503)

# ТЕМА 14. Введение в библиотеку Pandas.

## 14.1. Установка Pandas

Pip install pandas

Pip install notebook

Для удобства можно установить anaconda

## 14.2. Основные компоненты Pandas. Знакомство с Series

Основные компоненты Pandas:

* **DataFrame**: двумерная структура данных, представляющая собой таблицу с именованными столбцами. Каждый столбец может содержать данные разного типа (числа, строки и т. д.). DataFrame является основным компонентом Pandas и используется для хранения и обработки данных.
* **Series**: одномерный массив данных, который может быть проиндексирован по одному или нескольким измерениям. Series является базовым компонентом Pandas, из которого строятся более сложные структуры данных, такие как DataFrame.
* **Index**: уникальный идентификатор для каждого элемента в структуре данных Pandas. Индекс может быть числовым или строковым. Он используется для быстрого доступа к элементам и их изменения.
* **Panel**: трёхмерная структура данных, которая представляет собой набор DataFrame, объединённых общим индексом. Panel используется редко и может быть заменена на несколько DataFrame.

**Series** — это одномерный массив данных с метками осей. В отличие от списков Python, Series являются неизменяемыми, что означает, что после создания нельзя изменить размерность или добавить новые элементы. Однако можно изменить значения существующих элементов.

Series используются для представления одномерных массивов данных, таких как временные ряды, цены акций, температуры и т. п. Они могут быть созданы из списков, массивов NumPy и других источников данных.

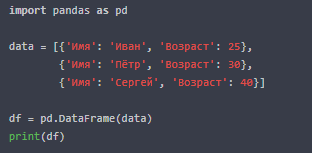


## 14.3. Знакомство с DataFrame

**DataFrame** — это двумерная структура данных, которая представляет собой таблицу с именованными столбцами. Каждый столбец может содержать данные разного типа (числа, строки и т. д.). DataFrame является основным компонентом библиотеки Pandas и используется для хранения и обработки данных.

DataFrame состоит из трёх основных компонентов:

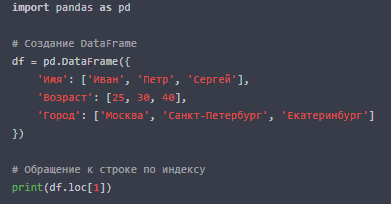
* **Индекс (index)** — уникальный идентификатор для каждой строки DataFrame. Индекс может быть числовым или строковым. Он используется для быстрого доступа к строкам и их изменения.
* **Столбцы (columns)** — именованные столбцы DataFrame, которые содержат данные разных типов. Столбцы могут быть добавлены, удалены или изменены.
* **Данные (data)** — значения ячеек DataFrame, расположенные на пересечении строк и столбцов. Данные могут быть любого типа, который поддерживается библиотекой Pandas.



## 14.4. Обращение к DataFrame по индексу

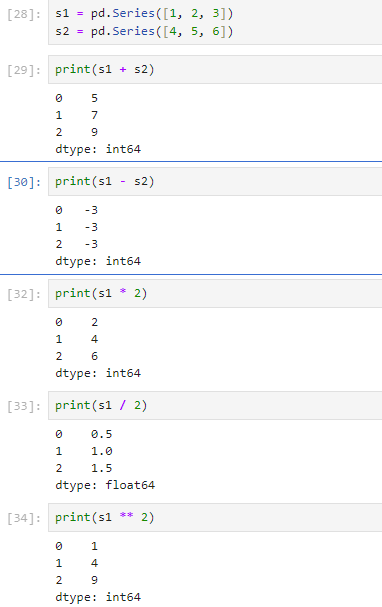
**Обращение по индексу**

Чтобы обратиться к элементу DataFrame по его индексу, можно использовать метод loc. Этот метод принимает в качестве аргумента индекс строки или список индексов строк. Например:





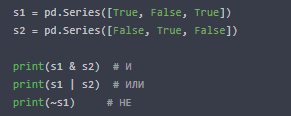
14.5. Арифметические операции в Series

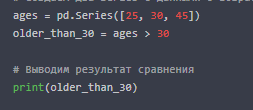


## 14.6. Логические операции к Series.

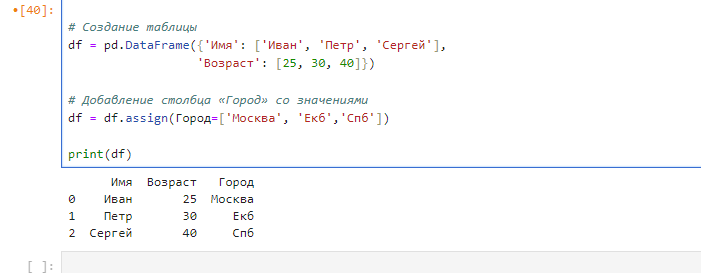
Вот основные логические операции, доступные в Series:

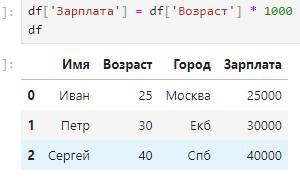
* **Сравнение (comparison)**: сравнение двух или более Series с одинаковыми индексами. Результатом будет новый Series, где каждый элемент равен результату логического сравнения соответствующих элементов исходных Series. Поддерживаются следующие операторы сравнения: ==, !=, <, <=, > и >=.
* **И (AND)**: логическое «И» между двумя Series приводит к созданию нового Series, где каждый элемент равен результату применения логического «И» к соответствующим элементам исходных Series.
* **ИЛИ (OR)**: логическое «ИЛИ» между двумя Series приводит к созданию нового Series, где каждый элемент равен результату применения логического «ИЛИ» к соответствующим элементам исходных Series.
* **НЕ (NOT)**: логическое отрицание Series приводит к созданию нового Series, где каждый элемент равен противоположному значению соответствующего элемента исходного Series.



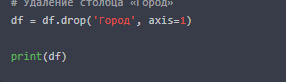


## 14.7. Редактирование таблиц. Создание и удаление столбцов.





Чтобы удалить столбец из таблицы, можно использовать метод drop. Этот метод принимает имя столбца или список имён столбцов в качестве аргумента, а axis=1 показывает что мы удаляем столбцы а не строки:



## 14.8. Удаление столбцов или строк

Для удаления строки из таблицы можно использовать метод drop. Этот метод принимает индекс строки или список индексов строк в качестве аргумента, а axis = 0 говорить что мы будем удалять строки:

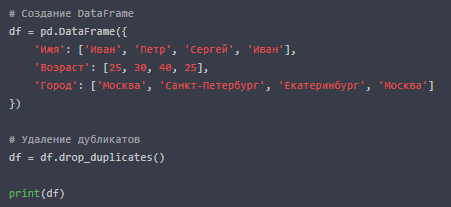


## 14.9. Удаление дубликатов строк.

Самый простой способ удалить дубликаты строк — использовать метод drop\_duplicates. Этот метод принимает параметр keep, который определяет, как обрабатывать дубликаты:

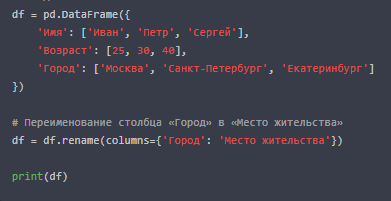
* **keep='first'** — оставить только первую строку из каждой группы дубликатов;
* **keep='last'** — оставить только последнюю строку из каждой группы дубликатов;
* **keep=False** — удалить все дубликаты.

Вот пример использования метода drop\_duplicates:



## 14.10. Переименование столбцов

Самый простой способ переименовать столбец — использовать метод rename. Этот метод принимает словарь, где ключи соответствуют старым именам столбцов, а значения — новым именам:



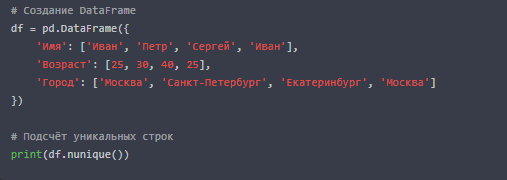
## 14.11. Подсчет уникальных значение в DataFrame

**Метод nunique**

Самый простой способ подсчитать уникальные значения — использовать метод nunique. Этот метод принимает параметр axis, который определяет ось (или измерение) данных, по которой будет производиться подсчет:

* **axis=0** — подсчёт уникальных строк;
* **axis=1** — подсчёт уникальных столбцов.

Вот пример использования метода nunique:



## 14.12. Загрузка данных и знакомство с методами Pandas.

**Загрузка данных**

Для начала работы с Pandas необходимо загрузить данные в DataFrame. Существует несколько способов загрузки данных:

Из файла CSV:



Из URL:



**Знакомство с методами Pandas**

В Pandas существует множество методов для работы с данными. Вот некоторые из них:

head(): выводит первые несколько строк DataFrame.



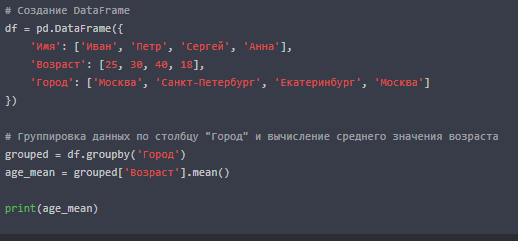
tail(): выводит последние несколько строк DataFrame

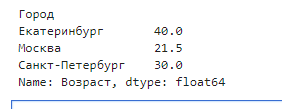


describe(): вычисляет статистические характеристики DataFrame, такие как среднее значение, стандартное отклонение и т. д.



groupby(): группирует данные по одному или нескольким столбцам и выполняет агрегацию.

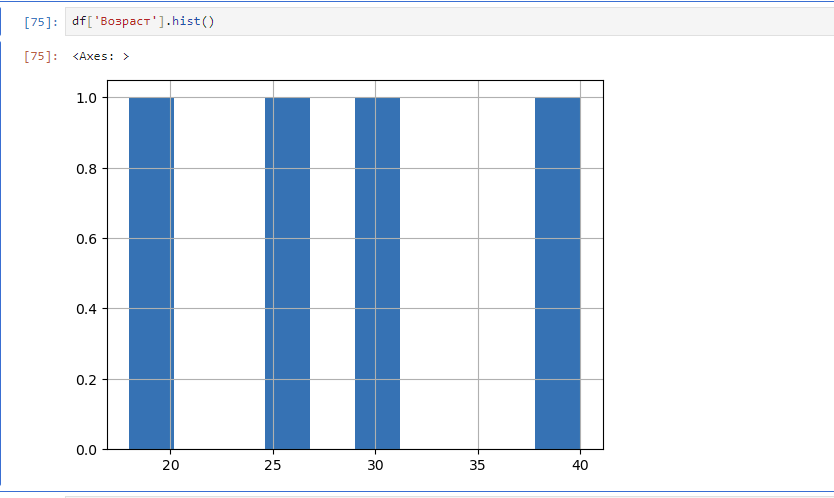




## 14.13. Предварительный осмотр данных

Метод info() выводит информацию о DataFrame, такую как количество строк и столбцов, типы данных и значения null

Визуализация данных является мощным инструментом для предварительного осмотра данных. Она позволяет увидеть закономерности, тренды и аномалии, которые могут быть не видны при просмотре числовых характеристик. В Pandas есть несколько функций для визуализации данных, таких как plot(), hist() и scatter(). Например, вы можете построить гистограмму распределения возраста:

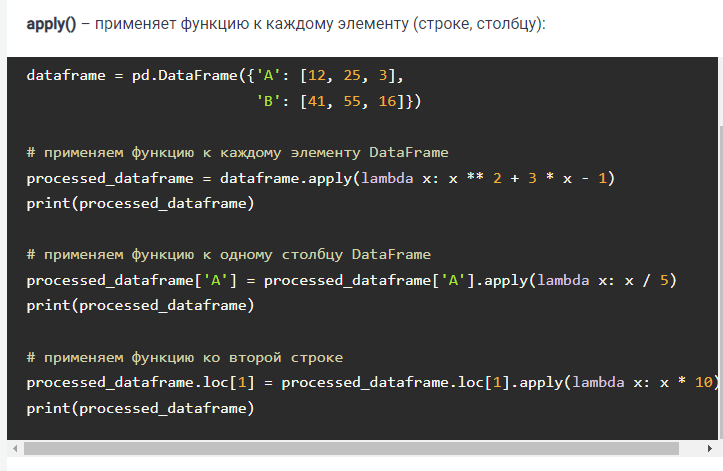


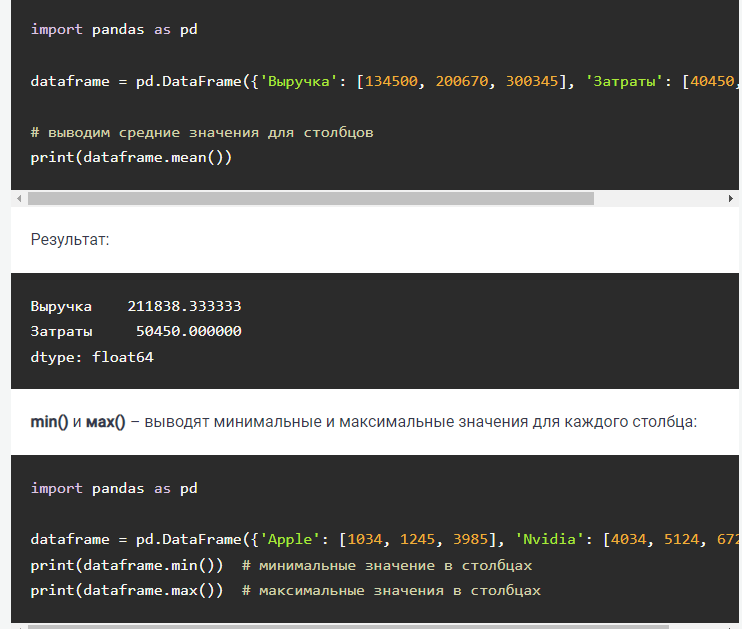
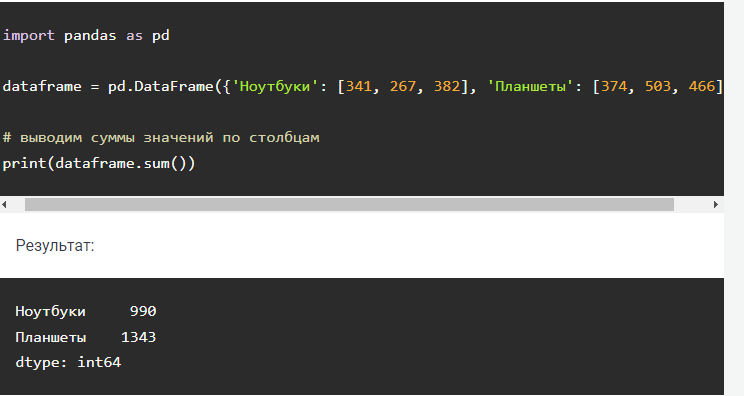
## 14.14. Подготовка датасэта к работе

**Подготовка датасета к работе** — это важный этап в процессе анализа данных, который включает в себя несколько ключевых шагов:

1. **Очистка данных:**
   * Удаление дубликатов и аномальных значений;
   * Обработка пропущенных значений (например, замена на среднее или удаление строк);
   * Преобразование типов данных (например, из строковых в числовые).
2. **Преобразование данных:**
   * Нормализация данных (приведение к единому диапазону значений);
   * Стандартизация данных (преобразование в стандартное нормальное распределение);
   * Кодирование категориальных переменных (например, с помощью One-Hot Encoding).
3. **Визуализация данных:**
   * Построение графиков и диаграмм для понимания структуры и распределения данных.
4. **Анализ данных:**
   * Вычисление статистических характеристик (среднее, медиана, стандартное отклонение и т. д.);
   * Поиск корреляций между переменными.

## 14.15. Работа с числовыми данными



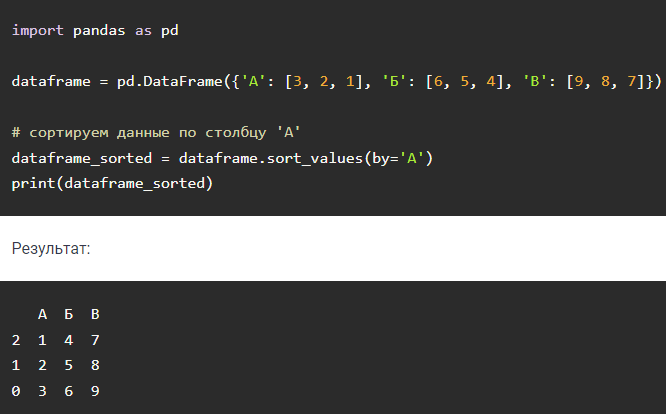


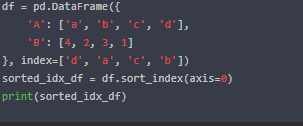
## 14.16. Сортировка данных с помощью Pandas

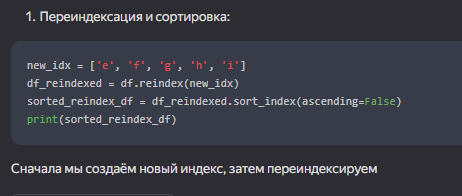
**Методы сортировки:**

* **sort\_values()**: сортирует данные по одному или нескольким столбцам. Этот метод возвращает новый отсортированный DataFrame.
* **sort\_index()**: сортирует строки или столбцы DataFrame по индексу. Этот метод также возвращает новый отсортированный объект.
* **reindex()**: позволяет переиндексировать DataFrame и отсортировать его по новому индексу.

**sort\_values()** выполняет сортировку:







## 14.17. Фильтрация данных

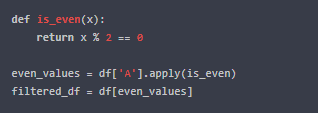
**Использование логического индексирования:**

* Вы можете использовать логические выражения для фильтрации данных в Pandas. Например, чтобы выбрать все строки, где значение в столбце «A» больше 5, вы можете написать:



**Применение функций к столбцам:**

* Если у вас есть функция, которая возвращает логическое значение, вы можете применить её к столбцу DataFrame для фильтрации. Например:



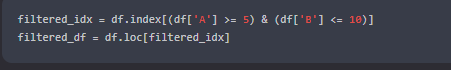
**Использование метода query():**

* Метод query() позволяет писать запросы к DataFrame в стиле SQL. Например:



**Работа с индексами:**

* В Pandas вы можете фильтровать данные, используя индексы. Например:



В данном случае мы выбираем строки, где значения в столбцах A больше или равны 5 и значения в B меньше или равны 10.

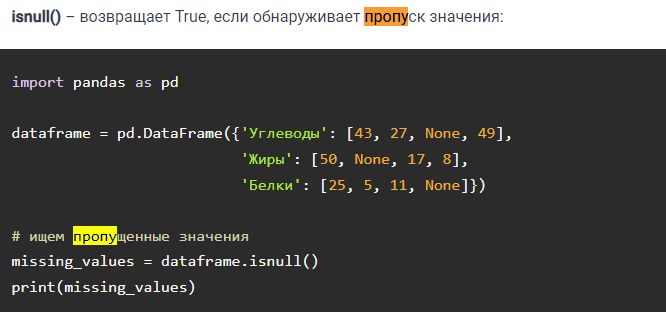
**Группировка и фильтрация:**

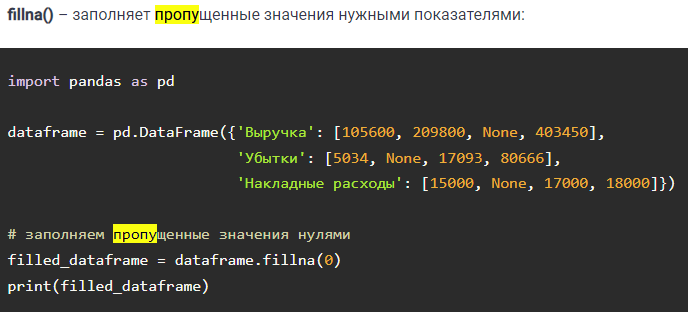
* Если у вас есть DataFrame с несколькими группами, вы можете выполнить фильтрацию внутри каждой группы. Например:



Здесь мы группируем DataFrame по столбцу category и получаем группу group1.

## 14.18. Работа с пропущенными значениями





**Удаление строк с пропущенными значениями (dropna)**: этот метод удаляет строки, в которых есть хотя бы одно пропущенное значение. Это полезно, когда вы хотите работать только с полными данными.



## 14.19. Группировка данных и агрегирующие функции. Сводные таблицы

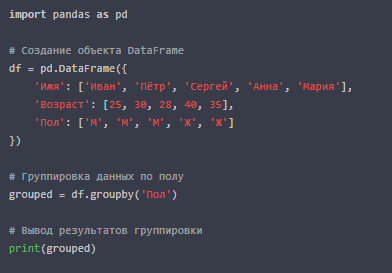
**Группировка данных** — это процесс объединения данных по определённому признаку или критерию. В результате группировки данные становятся более структурированными и удобными для анализа.

В Pandas группировка данных осуществляется с помощью метода groupby(). Этот метод принимает два аргумента:

* объект DataFrame или Series, который содержит данные;
* ключ группировки (столбец или список столбцов), по которому будут группироваться данные.

После выполнения метода groupby() объект становится объектом GroupBy, который можно использовать для выполнения различных операций над данными.

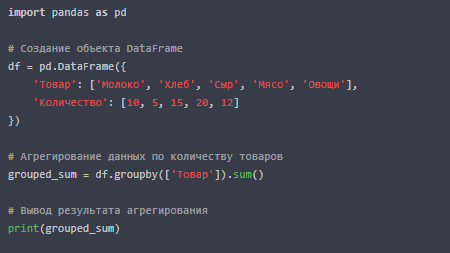
Вот пример группировки данных в Pandas



Этот код создаёт объект DataFrame с данными о людях и группирует данные по полу. Результаты группировки выводятся на экран.

**Агрегирующие функции** — это функции, которые применяются к группам данных для получения сводной информации. В Pandas есть несколько агрегирующих функций, таких как sum(), mean(), count() и другие.

Пример использования агрегирующей функции sum():

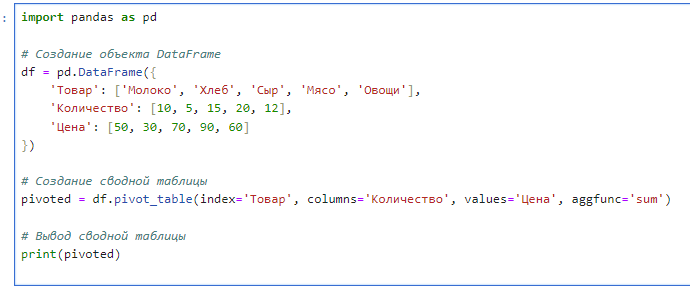


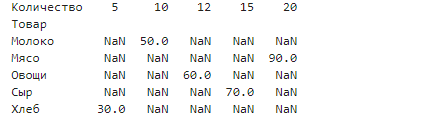
**Сводные таблицы в библиотеке Pandas**

Сводные таблицы — это инструмент для анализа данных, который позволяет быстро и легко получить сводную информацию о данных. В Pandas сводные таблицы создаются с помощью метода pivot\_table(). Этот метод принимает следующие аргументы:

* объект DataFrame или Series, который содержит данные;
* индекс (столбец или список столбцов), по которому будут группироваться данные;
* столбцы (столбец или список столбцов), которые будут использоваться для агрегирования данных;
* функция агрегирования (например, sum(), mean() и т. д.);
* дополнительные параметры (например, fill\_value для заполнения пропущенных значений).

Пример создания сводной таблицы в Pandas:





Этот код создаёт сводную таблицу, которая показывает сумму цен на товары в зависимости от их количества.

**Дополнительные возможности группировки данных и сводных таблиц в Pandas:**

* Группировка данных по нескольким ключам. Метод groupby() может принимать несколько ключей группировки. Это позволяет более точно группировать данные.
* Использование функций агрегирования для нескольких столбцов. Метод pivot\_table() может использовать функции агрегирования для нескольких столбцов одновременно. Это позволяет получить более подробную сводную информацию.
* Настройка параметров сводных таблиц. Метод pivot\_table() имеет множество параметров, которые позволяют настроить вывод сводных таблиц под свои нужды.

## 14.20. Функции в Pandas

**Функции в Pandas** — это инструменты, которые позволяют выполнять различные операции над данными. В библиотеке Pandas есть множество функций для работы с данными, и они могут быть разделены на несколько категорий:

1. **Агрегирующие функции:** эти функции используются для вычисления сводной информации о данных, таких как сумма, среднее значение, количество и т. д. Примеры агрегирующих функций включают sum(), mean(), count() и другие.
2. **Группирующие функции:** они используются для группировки данных по определённому признаку или критерию. Пример группирующей функции — groupby().
3. **Фильтрующие функции:** позволяют выбирать подмножество данных на основе определённых критериев. Например, функция query() позволяет выполнять запросы к данным с использованием синтаксиса SQL-подобных выражений.
4. **Преобразующие функции:** изменяют структуру или формат данных. Например, функции melt() и pivot() преобразуют данные из одного формата в другой.
5. **Статистические функции:** вычисляют статистические показатели, такие как стандартное отклонение, дисперсия и т. п. Примеры статистических функций — std(), var() и др.
6. **Математические функции:** выполняют математические операции над данными, например, умножение, деление и т.д.
7. **Логические функции:** используются для выполнения логических операций над данными, таких как сравнение, проверка условий и т.п.
8. **Текстовые функции:** работают с текстовыми данными, такими как поиск подстрок, замена символов и т.д.
9. **Функции работы с датами и временем:** обрабатывают информацию о дате и времени, например, вычисление разницы между двумя датами, форматирование даты и т.п.

Это лишь некоторые из множества функций, доступных в библиотеке Pandas. Для получения более подробной информации рекомендуется обратиться к официальной документации Pandas или использовать онлайн-ресурсы, такие как Stack Overflow или документация Python.

# ТЕМА 15. Работа с NumPyНачало формы

## 15.1. Что такое NumPy?

NumPy — это библиотека для научных вычислений на Python. Она предоставляет инструменты для работы с многомерными массивами и матрицами, а также функции для математических операций над ними.

NumPy позволяет эффективно манипулировать большими объёмами данных и выполнять сложные вычисления. Библиотека широко используется в научных исследованиях, машинном обучении и анализе данных.

Основные возможности NumPy:

* **Создание и манипулирование массивами:** NumPy предоставляет возможность создавать и изменять многомерные массивы и матрицы. Это позволяет легко работать с данными, представленными в табличной форме.
* **Математические операции:** библиотека включает в себя множество функций для выполнения математических операций, таких как сложение, вычитание, умножение и деление. Эти функции могут быть применены к массивам и матрицам.
* **Индексация и срезы:** NumPy поддерживает различные способы индексации и срезов массивов, что позволяет легко выбирать подмножества данных.
* **Универсальные функции (ufuncs):** библиотека предоставляет набор универсальных функций, которые могут применяться к элементам массива или матрицы. Например, можно вычислить среднее значение элементов массива или найти максимум и минимум.
* **Интеграция с другими библиотеками:** NumPy может быть интегрирован с другими популярными библиотеками, такими как SciPy, Matplotlib и Pandas, для создания мощных инструментов анализа данных.

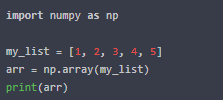
## 15.2. Установка NumPy.

Для использования NumPy в Python необходимо установить библиотеку с помощью менеджера пакетов, такого как pip или conda. После установки можно импортировать NumPy и начать использовать его функции и классы. А в anaconda она уже встроена и остается только импортировать ее.

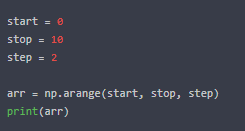
## 15.3. Создание массивов

В NumPy есть несколько способов создания массивов:

* **Функция array()**: принимает последовательность в качестве аргумента и возвращает массив. Последовательность может быть списком, кортежем или строкой.
* **Литеральная запись**: квадратные скобки [ ] используются для создания массива. Внутри скобок через запятую перечисляются элементы массива.
* **Генерация массива**: можно использовать функцию arange() для генерации числового диапазона. Функция zeros() создаёт массив из нулей, а функция ones() — массив из единиц.
* **Чтение данных из файла**: функция loadtxt() считывает данные из текстового файла и преобразует их в массив.





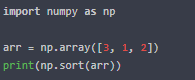




## 15.4. Сортировка элементов

NumPy предоставляет несколько функций для сортировки элементов массива. Вот некоторые из них:

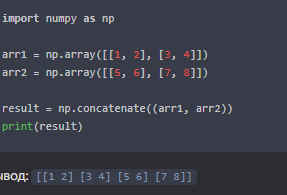
1. **sort()**: сортирует элементы массива по возрастанию или убыванию. По умолчанию используется возрастающий порядок.
2. **argsort()**: возвращает индексы, которые бы отсортировали массив в порядке возрастания. Это полезно, если вы хотите сохранить исходные значения элементов, но изменить их порядок.
3. **lexsort()**: выполняет лексикографическую сортировку по нескольким осям. Эта функция полезна, когда у вас есть многомерные массивы и вы хотите отсортировать их по определённому столбцу или строке.
4. **partition()**: разделяет элементы массива на две части таким образом, что все элементы до определённой позиции будут меньше или равны заданному значению, а все остальные — больше.
5. **searchsorted()**: находит индексы, в которые можно вставить элемент так, чтобы



## 15.5. Конкатенация

Конкатенацией называется операция соединения двух или более массивов в один. В NumPy есть несколько способов конкатенации массивов:

1. **Функция concatenate()**: принимает в качестве аргументов два или более массива и объединяет их в один массив. Массивы должны иметь одинаковую размерность, за исключением оси конкатенации.
2. **Оператор «+»**: можно использовать для объединения двух массивов одинаковой размерности. Однако этот способ может быть менее эффективным, чем функция concatenate().
3. **Функции hstack() и vstack()**: используются для горизонтальной и вертикальной конкатенации соответственно. Эти функции объединяют массивы вдоль первой оси (hstack) или второй оси (vstack).

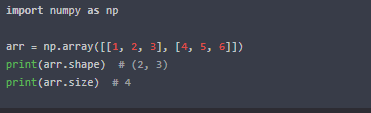


## 15.6. Применение Shape и size

В библиотеке NumPy для работы с многомерными массивами есть две функции: shape и size. Они используются для определения размера массива и его формы.

1. **shape**: возвращает кортеж, который содержит размер массива по каждой оси. Например, если массив двумерный, то shape вернёт кортеж из двух элементов, представляющих количество строк и столбцов. Если массив одномерный, shape вернет один элемент, равный длине массива.
2. **size**: возвращает общее количество элементов в массиве. Это число можно получить, умножив размеры массива по всем осям.

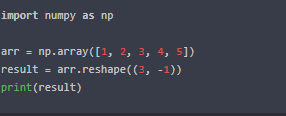
Вот пример использования этих функций



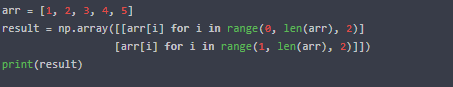
## 15.7. Преобразование одномерного массива в двумерный

В NumPy есть несколько способов преобразования одномерного массива (вектора) в двумерный массив (матрицу). Вот некоторые из них:

1. **Функция reshape()**: принимает в качестве аргументов одномерный массив и кортеж, который определяет размеры двумерного массива. Например, чтобы преобразовать вектор размера N в матрицу размера MxN, можно использовать функцию reshape следующим образом:



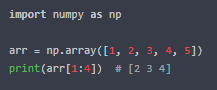
1. **Литеральная запись**: можно создать двумерный массив из одномерного, используя квадратные скобки [ ] и разделив элементы на строки и столбцы с помощью запятых. Например:



## 15.8. Индексирование и срезы массивов

NumPy предоставляет мощные инструменты для работы с многомерными массивами, включая индексирование и срезы. Вот основные понятия и методы, связанные с этой темой:

1. **Индексация**: это процесс доступа к элементам массива по их индексу или позиции. В NumPy индексация начинается с 0, что означает, что первый элемент массива имеет индекс 0.
2. **Срезы**: это способ выбора подмножества элементов массива на основе индексов. Срез может быть одномерным (для одномерных массивов) или многомерным (для многомерных массивов).
3. **Одномерные срезы**: для одномерного массива можно использовать квадратные скобки [ ] для выбора элементов на основе их индексов. Например, чтобы выбрать элементы с индексами 1, 2 и 3, можно написать:



Здесь срез arr[1:4] выбирает элементы с индексом от 1 до 3 (исключая 4).

1. **Многомерные срезы**: для многомерного массива можно также использовать квадратные скобки, но с несколькими индексами. Например, для двумерного массива можно выбрать строку и столбец следующим образом:



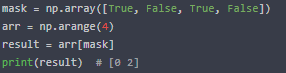
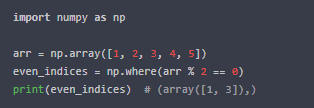
1. **Отрицательные индексы**: в NumPy можно использовать отрицательные индексы для отсчёта от конца массива. Например:



Этот код выводит последний элемент массива arr.

1. **Шаг**: при использовании срезов можно указать шаг, который определяет, как часто выбираются элементы. Например:



1. **Срез по маске**: можно использовать логический массив в качестве маски для выбора элементов из другого массива. Например:
2. 
3. 

# ТЕМА 16. Работа с Matplotlib и Seaborn

## 16.1. Что такое Matplotlib?

**Matplotlib** — это библиотека для визуализации данных на Python. Она предоставляет набор инструментов для создания графиков, диаграмм и других видов визуализации.

Matplotlib позволяет:

* строить графики различных типов (линейные, столбчатые, круговые диаграммы и т. д.);
* настраивать внешний вид графиков (цвет, размер, стиль линий и т. п.);
* сохранять графики в различных форматах (PNG, JPEG, SVG и др.).

Основные элементы Matplotlib:

1. **Рисунок (Figure)** — это окно или холст, на котором отображаются графики. Рисунок может содержать один или несколько графиков.
2. **Оси (Axes)** — это область внутри рисунка, где строится график. Оси могут быть двумерными (для построения графиков в декартовой системе координат) или трёхмерными (для объёмных графиков).
3. **График (Plot)** — это линия, точка, столбец или другой объект, который отображается на осях. Графики могут быть разных типов (линии, гистограммы, круговые диаграммы и т.д.).
4. **Легенда (Legend)** — это описание того, что представляют собой различные графики на рисунке. Легенда может быть расположена в любом месте на рисунке.
5. **Метки (Labels)** — это текст, который описывает оси и графики на рисунке. Метки могут быть расположены на осях, рядом с графиками или в легенде.
6. **Подзаголовки (Subplots)** — это способ размещения нескольких графиков на одном рисунке. Подзаголовки позволяют разделить рисунок на несколько областей и отобразить в каждой области свой график.
7. **Стиль (Style)** — это набор параметров, которые определяют внешний вид графика. Стиль может включать в себя цвет, толщину линии, тип шрифта и другие параметры.
8. **Сохранение (Saving)** — это функция, которая позволяет сохранить рисунок в файл. Сохранение может выполняться в различных форматах, таких как PNG, JPEG, PDF и др.

## 16.2. Установка Matplotlib.

Для использования Matplotlib необходимо установить библиотеку с помощью менеджера пакетов, такого как pip или conda. После установки можно импортировать Matplotlib и начать создавать графики и диаграммы. А в anaconda она уже встроена и остается только импортировать ее.

## 16.3. Загрузка данных для учебного проекта

Для успешного выполнения учебного проекта, связанного с анализом данных, необходимо иметь доступ к данным. Данные могут быть получены из различных источников, таких как:

* **Собственные данные**: если у вас есть собственные данные, которые вы хотите использовать в проекте, то вам необходимо загрузить их на свой компьютер или в облачное хранилище.
* **Открытые данные**: существует множество открытых данных, доступных для использования в учебных целях. Вы можете найти открытые данные на сайтах государственных органов, научных организаций и других источников.
* **Онлайн-сервисы**: некоторые онлайн-сервисы предоставляют доступ к данным для учебных проектов. Например, вы можете использовать сервисы для анализа социальных сетей, геоданных или финансовых данных.

Вот несколько шагов, которые необходимо выполнить для загрузки данных:

1. **Выбор источника данных**: определите, какие данные вы хотите использовать. Это может быть набор данных, который вы уже собрали, или данные из открытого источника.
2. **Получение доступа к данным**: если данные находятся в открытом доступе, то вы можете просто скачать их с сайта. Если данные защищены паролем или другими ограничениями, вам может потребоваться зарегистрироваться на сайте или получить разрешение от владельца данных.
3. **Формат данных**: данные могут быть представлены в различных форматах, таких как CSV, JSON, XML и другие. Выберите формат, который наиболее подходит для вашего проекта.
4. **Размер данных**: учтите размер данных при выборе формата. Большие наборы данных могут занимать много места на вашем компьютере или требовать больше времени для загрузки.
5. **Обработка данных**: после загрузки данных вам может потребоваться обработать их перед использованием. Например, вы можете удалить ненужные столбцы или строки, преобразовать данные в другой формат или очистить данные от ошибок.
6. **Хранение данных**: решите, где вы будете хранить данные. Вы можете сохранить их на своем компьютере, в облачном хранилище или на сервере.
7. **Безопасность данных**: обеспечьте безопасность данных, чтобы предотвратить несанкционированный доступ или потерю данных. Используйте надежные пароли и шифрование данных, если это необходимо.
8. **Документация**: сохраните документацию о данных, включая источник данных, формат данных, размер данных и любые другие важные сведения. Это поможет вам в будущем при работе с данными.

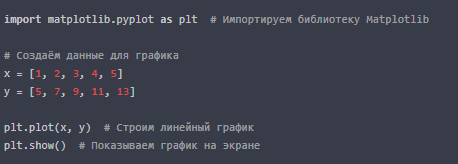
## 16.4. Базовые графики Matplotlib и построения

В Matplotlib есть несколько базовых графиков, которые можно использовать для построения различных типов данных:

1. **Линейный график (line plot)** — отображает зависимость между двумя переменными в виде линии. Линейные графики используются для отображения тенденций, изменений во времени и т. д.
2. **Столбчатый график (bar plot)** — представляет данные в виде столбцов. Столбчатые графики используются для сравнения значений, распределения и т. п.
3. **Круговая диаграмма (pie chart)** — показывает долю каждого значения в общем объёме. Круговые диаграммы используются для демонстрации пропорций, соотношения и т. д.
4. **Гистограмма (histogram)** — отображает распределение значений в виде гистограммы. Гистограммы используются для анализа частот, плотности и т. п.
5. **Диаграмма рассеяния (scatter plot)** — показывает связь между двумя переменными. Диаграммы рассеяния используются для выявления корреляции, зависимости и т. д.
6. **График плотности (density plot)** — похож на гистограмму, но показывает плотность распределения значений. Графики плотности используются для оценки распределения, вероятности и т. п.
7. **Тепловая карта (heatmap)** — представляет двумерные данные в виде цветовой карты. Тепловые карты используются для визуализации матриц, таблиц и т. д.
8. **Boxplot (ящик с усами)** — показывает распределение данных с помощью квартилей. Boxplots используются для анализа выбросов, медианы и т. п.
9. **Контурный график (contour plot)** — визуализирует функцию двух переменных в виде линий уровня. Контурные графики используются для представления поверхностей, градиентов и т. д.
10. **Трёхмерный график (3D plot)** — позволяет отображать данные в трёхмерном пространстве. Трёхмерные графики используются для визуализации объёмов, поверхностей и т. п.

Для построения графиков в Matplotlib необходимо выполнить следующие шаги:

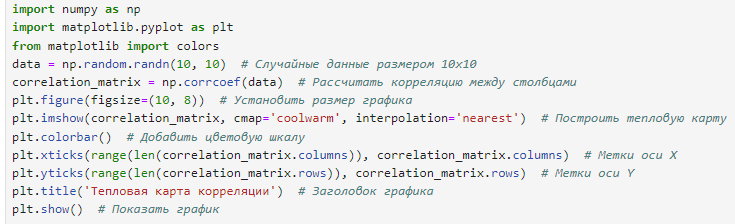
* Импортировать библиотеку Matplotlib.
* Создать объект Figure или использовать уже существующий.
* Добавить оси Axes к объекту Figure.
* Построить график с помощью функций из модуля pyplot.
* Сохранить график в файл или показать его на экране.



Этот код создаст линейный график зависимости y от x. Вы можете изменить значения x и y, чтобы построить свой график.

Matplotlib предоставляет множество возможностей для настройки графиков. Вы можете изменять цвета, стили линий, метки осей и другие параметры. Для этого используются функции из модуля pyplot. Например, вы можете добавить заголовок к графику с помощью функции title():

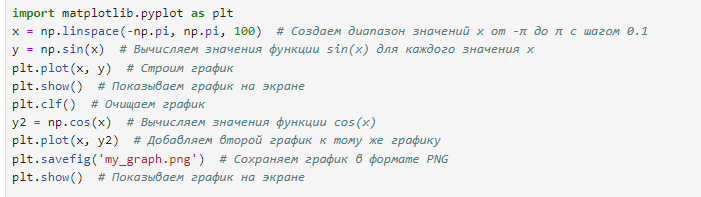




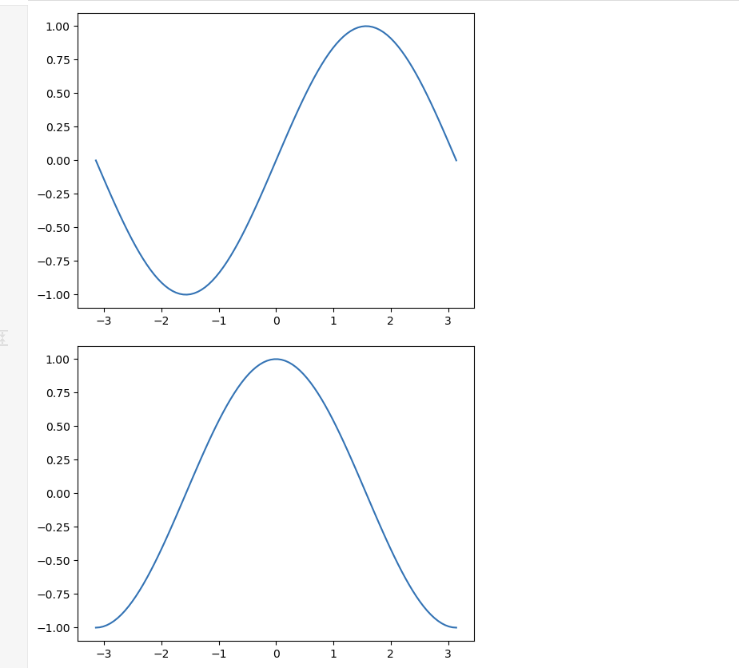
Построение тепловой карты

## 16.5. Вывод двух графиков в Matplotlib

1. **Импортируйте необходимые библиотеки**:
2. **Создайте данные для графиков**:  
   Предположим, у вас есть два набора данных, которые вы хотите визуализировать. Например, это могут быть две функции, которые вы хотите сравнить.
3. **Постройте первый график**:  
   Вы можете построить первый график с помощью метода plot из Matplotlib. В этом примере мы построим график функции $sin(x)$:
4. **Сохраните график или покажите его на экране**:  
   После того как вы построили первый график, вы можете сохранить его в файл или показать на экране с помощью функции show:
5. **Очистите график**:  
   Перед тем как построить второй график, необходимо очистить предыдущий график. Для этого можно использовать метод clf (clear figure):
6. **Постройте второй график**:  
   Теперь вы можете построить второй график с использованием тех же данных или других данных. В нашем примере мы построим график функции $cos(x)$:
7. **Сохраните или покажите график**:  
   Наконец, вы можете снова сохранить график в файл или показать его на экране:



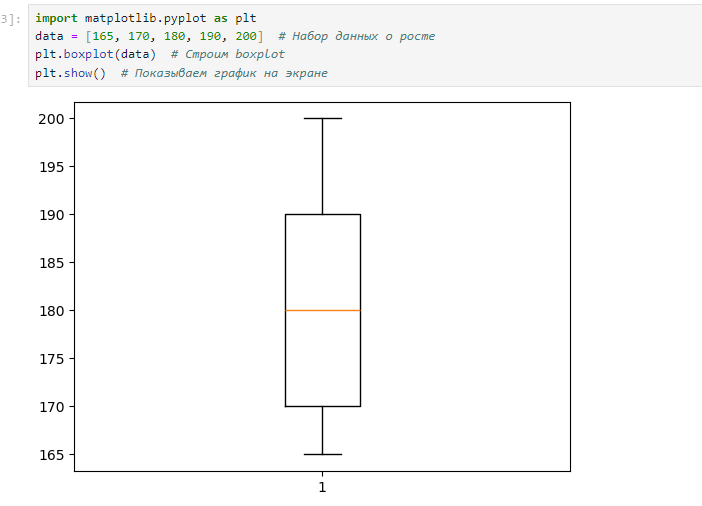
В результате вы получите два графика, расположенных друг над другом. Вы можете настроить их внешний вид, используя различные параметры метода plot.



## 16.6. Boxplot в Matplotlib

**Boxplot** — это график, который используется для визуализации распределения данных. Он показывает медиану, квартили и выбросы в наборе данных. Boxplot также называют «ящик с усами».

Вы можете построить boxplot с помощью метода boxplot из Matplotlib. В этом примере мы построим boxplot для набора данных о росте людей:



Конец формы

Boxplot имеет следующие основные элементы:

* **Медиана**: линия внутри прямоугольника, которая показывает среднее значение данных.
* **Квартили**: верхняя и нижняя границы прямоугольника, которые показывают 25-й и 75-й процентили данных соответственно.
* **Выбросы**: точки за пределами «усов», которые показывают значения, выходящие за пределы 1,5 \* IQR (интерквартильный размах).
* **IQR**: расстояние между верхней и нижней квартилями.

## 16.7. Что такое Seaborn?

**Seaborn** — это библиотека для визуализации данных на Python, которая построена на основе Matplotlib. Она предоставляет простой и удобный интерфейс для создания статистических графиков, таких как гистограммы, диаграммы рассеяния, тепловые карты и другие.

Основные преимущества Seaborn:

* **Простота использования**: Seaborn предоставляет множество готовых функций для построения различных типов графиков. Это позволяет быстро создавать красивые и информативные графики без необходимости писать много кода.
* **Гибкость**: Seaborn позволяет настраивать внешний вид графиков с помощью параметров. Вы можете изменить цвета, стили линий, метки осей и другие элементы графика.
* **Статистическая направленность**: Seaborn предназначен для визуализации статистических данных. Он предоставляет функции для расчёта статистических показателей, таких как среднее значение, стандартное отклонение, корреляция и т. д.
* **Интеграция с Pandas**: Seaborn хорошо интегрируется с библиотекой Pandas для работы с данными. Вы можете легко построить график для столбца DataFrame или для нескольких столбцов одновременно.Начало формы

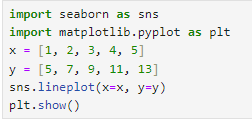
## 16.8. Установка Seaborn.Конец формы

Для использования Seaborn необходимо установить библиотеку с помощью менеджера пакетов, такого как pip или conda. После установки можно импортировать Seaborn и начать создавать графики и диаграммы. А в anaconda она уже встроена и остается только импортировать ее.

## 16.9. Базовые графики Seaborn.

Вот некоторые из наиболее популярных графиков в Seaborn:

1. **Линейный график (lineplot)** — отображает зависимость между двумя переменными в виде линии. Линейные графики используются для отображения тенденций, изменений во времени и т.д.
2. **Столбчатый график (barplot)** — представляет данные в виде столбцов. Столбчатые графики используются для сравнения значений, распределения и т.п.
3. **Круговая диаграмма (piechart)** — показывает долю каждого значения в общем объёме. Круговые диаграммы используются для демонстрации пропорций, соотношения и т.д.
4. **Диаграмма рассеяния (scatterplot)** — показывает связь между двумя переменными. Диаграммы рассеяния используются для выявления корреляции, зависимости и т.д.
5. **Тепловая карта (heatmap)** — представляет двумерные данные в виде цветовой карты. Тепловые карты используются для визуализации матриц, таблиц и т.д.
6. **Boxplot (ящик с усами)** — показывает распределение данных с помощью квартилей. Boxplots используются для анализа выбросов, медианы и т.п.
7. **Гистограмма (histogram)** — отображает распределение значений в виде гистограммы. Гистограммы используются для анализа частот, плотности и т. п.
8. **График плотности (density plot)** — похож на гистограмму, но показывает плотность распределения значений. Графики плотности используются для оценки распределения, вероятности



пример построения базового графика с использованием библиотеки Seaborn

## 16.10. Ядерная оценка плотности.

**Ядерная оценка плотности (Kernel Density Estimation, KDE)** — это метод непараметрического оценивания функции плотности вероятности случайной величины. Он используется для оценки плотности распределения вероятностей по набору данных.

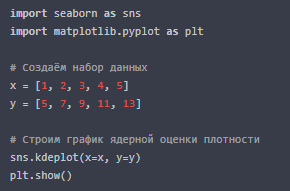
Идея ядерной оценки плотности заключается в том, что плотность вероятности в каждой точке пространства оценивается как взвешенная сумма значений ядра (функции, которая определяет форму оценки) в этой точке. Веса определяются расстоянием от точки до каждого из элементов выборки.

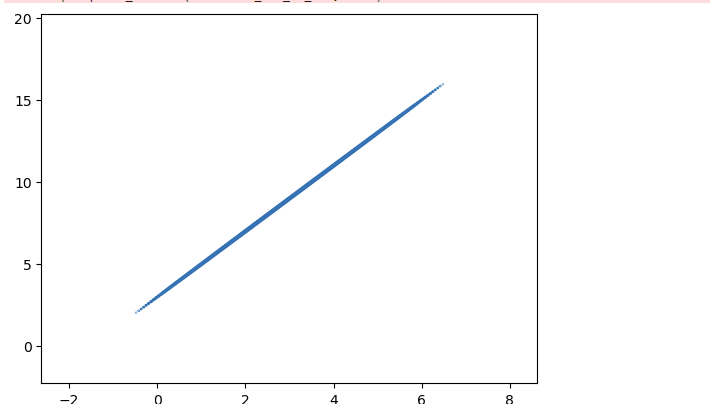
**Основные шаги ядерной оценки плотности:**

1. Выбор ядра и определение его параметров (ширина или масштаб).
2. Вычисление весов для каждого элемента выборки на основе расстояния до текущей точки.
3. Суммирование весов с использованием выбранного ядра для получения оценки плотности в текущей точке.
4. Повторение шагов 2 и 3 для всех точек пространства, где требуется оценить плотность.
5. Визуализация полученной оценки плотности с помощью графиков или других методов.

В Seaborn есть функция sns.kdeplot, которая позволяет построить график ядерной оценки плотности для одномерного или двумерного набора данных. Эта функция может быть полезна для визуализации распределения данных и выявления потенциальных выбросов.

Вот пример использования sns.kdeplot для построения графика ядерной оценки плотности:





Этот код создаст график ядерной оценки плотности для набора данных (x, y). Вы можете настроить этот пример, изменив данные или добавив дополнительные параметры в функцию kdeplot.

Начало формы

Конец формы