Библиотека Pandas - это мощный инструмент для обработки и анализа данных в языке программирования Python. Основными структурами данных в Pandas являются Series и DataFrame.

1. **Series:**
   * **Описание:** **Series** - это одномерная структура данных, аналогичная массиву или столбцу в таблице. Она состоит из двух основных компонентов: данных и индекса. Индекс помогает идентифицировать каждый элемент в серии.
   * **Создание:** Можно создать Series из списка, массива NumPy или словаря.
   * **Пример создания:**

|  |
| --- |
| **import pandas as pd**  **import numpy as np**  **data = np.array([10, 20, 30, 40, 50])**  **series\_object = pd.Series(data)** |

1. **DataFrame:**

* **Описание:** DataFrame - это двумерная табличная структура данных, аналогичная таблице в базе данных или электронной таблице Excel. Он состоит из строк и столбцов, где каждый столбец может быть разного типа данных.
* **Создание:** Можно создать DataFrame из массива NumPy, списка списков, словаря или других DataFrame.
* **Пример создания:**

|  |
| --- |
| **import pandas as pd**  **import numpy as np**  **data = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])**  **columns = ['Column1', 'Column2', 'Column3']**  **dataframe\_object = pd.DataFrame(data, columns=columns)** |

1. **Методы:**
   * **Операции с элементами:**
     1. Доступ к элементам: **df['Column1']** или **df.loc[row\_index, 'Column1']**.
     2. Изменение значений: **df.at[row\_index, 'Column1'] = new\_value**.
   * **Добавление/удаление данных:**
     1. Добавление столбца: **df['NewColumn'] = values**.
     2. Удаление столбца: **df = df.drop('Column2', axis=1)**.
     3. Добавление строки: **df = df.append({'Column1': 10, 'Column2': 20}, ignore\_index=True)**.
     4. Удаление строки: **df = df.drop(1)**.
   * **Обработка пропущенных значений:**
     1. Проверка на пропущенные значения: **df.isnull()** или **df.notnull()**.
     2. Удаление пропущенных значений: **df = df.dropna()**.
     3. Заполнение пропущенных значений: **df = df.fillna(value)**.
   * **Группировка и агрегация:**
     1. **df.groupby('Column1').mean()** - среднее значение по группам.
     2. **df.groupby('Column1').sum()** - сумма по группам.
   * **Фильтрация данных:**
     1. **df[df['Column1'] > 5]** - фильтрация по условию.

Эти методы и операции позволяют удобно и эффективно работать с данными в Pandas.

NumPy - это библиотека для работы с массивами и матрицами в Python. Основными структурами данных в NumPy являются массивы (numpy arrays). Давайте рассмотрим основные аспекты использования NumPy:

1. **Массивы (Arrays):**
   * **Описание:** Массив в NumPy представляет собой N-мерную структуру данных, где N может быть любым числом. Он предоставляет эффективные операции для выполнения математических операций над массивами.
   * **Создание:** Массивы могут быть созданы из списков, кортежей или других массивов.
   * **Пример создания:**

|  |
| --- |
| **import numpy as np**  **array\_1d = np.array([1, 2, 3, 4, 5])**  **array\_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])** |

1. **Математические операции:**

* **Операции с массивами:** NumPy предоставляет множество математических операций, которые можно выполнять с массивами.
* **Пример операций:**

|  |
| --- |
| **import numpy as np**  **array = np.array([1, 2, 3, 4, 5])**  **array = array + 5**  **array = np.sqrt(array)** |

1. **Удаление элементов:**

* **Удаление элементов из массива:** NumPy позволяет удалять элементы по индексу.
* **Пример удаления элемента:**

|  |
| --- |
| **import numpy as np**  **array = np.array([1, 2, 3, 4, 5])**  **array = np.delete(array, 2) # Удаление элемента по индексу 2** |

1. **Поиск элементов:**

* **Поиск элементов в массиве:** Можно использовать условные операторы или функции NumPy для поиска элементов.
* **Пример поиска элементов:**

|  |
| --- |
| **import numpy as np**  **array = np.array([1, 2, 3, 4, 5])**  **matching\_elements = np.where(array == 3)** |

1. **Размер массива:**

* **Получение размера массива:** Можно использовать атрибут **shape** для получения размера массива.
* **Пример получения размера:**

|  |
| --- |
| **import numpy as np**  **array\_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])**  **array\_shape = array\_2d.shape** |

Эти возможности NumPy позволяют эффективно работать с массивами и выполнять разнообразные математические операции.