Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**дисциплины**

**«Искусственный интеллект и машинное обучение»**

**Вариант 9**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Кравчук Мирослав Витальевич  2 курс, группа ИТС-б-о-23-1,  11.03.02«Инфокоммуникационные технологии и системы связи», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Проверил:  Доцент департамента цифровых, робототехнических систем и электроники Воронкин Р.А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2025 г.

**Тема работы:** “Введение в pandas: изучение структуры DataFrame и базовых операций”.

**Цель работы:** ознакомить с основами работы с библиотекой pandas, в частности, со структурой данных DataFrame.

Ссылка на git репозиторий: <https://github.com/miron2314/DLab-5.git>

**Порядок выполнения работы:**

1.Создание DataFrame разными способами.

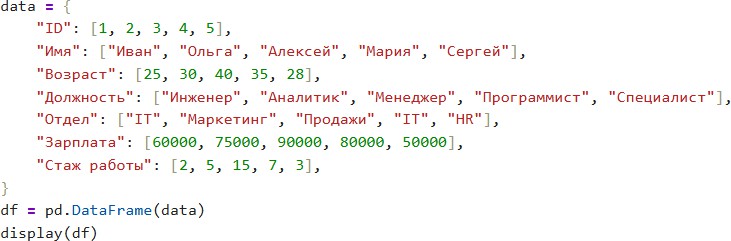


Рисунок 1. Создание df с помощью ditctъ

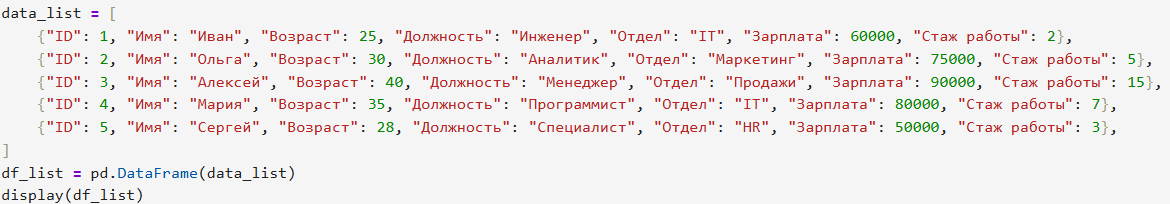


Рисунок 2. Создание df с помощью ditcts list

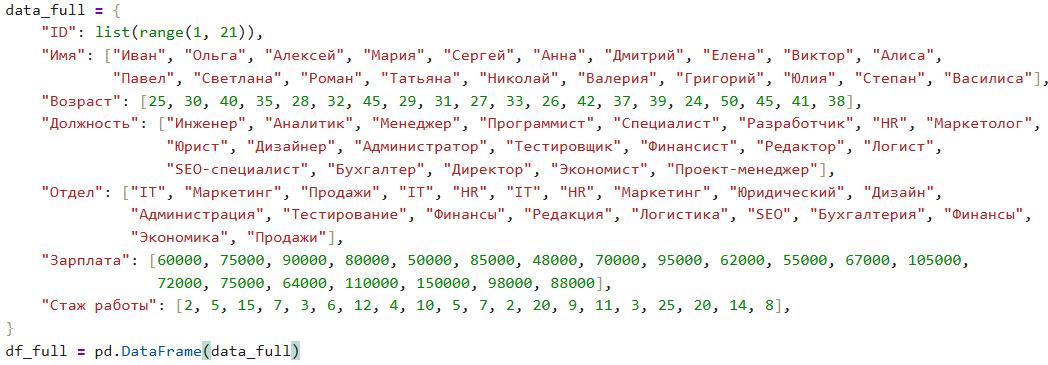


Рисунок 3. Создание df с помощью np массива

2.Чтение данных из файлов (CSV , Excel , JSON ).



Рисунок 4. Сохранение df в csv таблицу и загрузка из csv

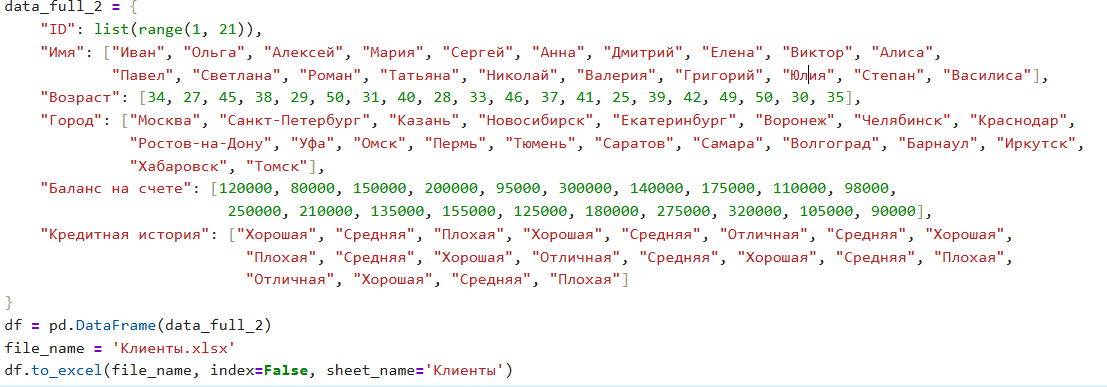


Рисунок 5. Сохранение таблицы 1 df в csv таблицу и загрузка из csv

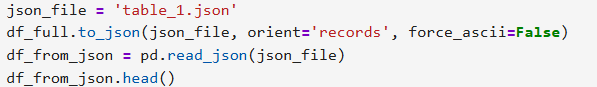


Рисунок 6. Экспорт таблицы 1 в json файл

3.Доступ к данным ( .loc , .iloc , .at , .iat ).



Рисунок 7. Получение информации о сотруднике с ID =5



Рисунок 8. Возраст третьего сотрудника в таблице



Рисунок 9. Название отдела для сотрудника "Мария"



Рисунок 10. Зарплата сотрудника, находящегося в четвертой строке и пятом столбце

4.Добавление новых столбцов и строк.

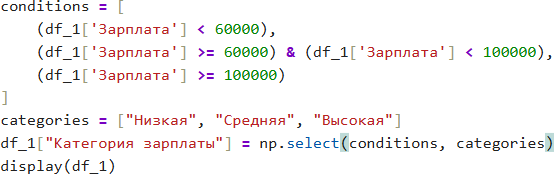


Рисунок 11. Добавление столбца в таблицу с категориями зарплат



Рисунок 12. Добавление сотрудника в таблицу

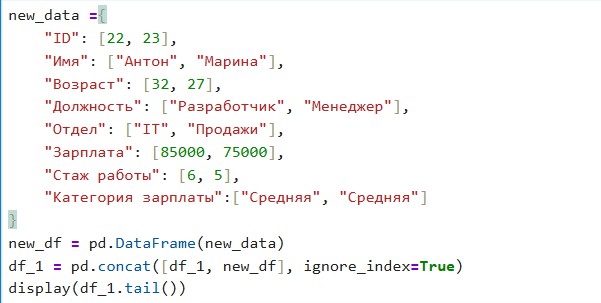


Рисунок 13. Добавление нескольких сотрудника в таблицу

5.Удаление строк и столбцов.



Рисунок 14. Удаление столбца "Категория зарплаты"

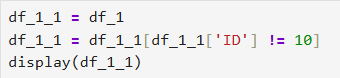


Рисунок 15. Удаление строки с ID =10



Рисунок 16. Удаление строк, где стаж работы <3



Рисунок 17. Удаление всех столбцов кроме Имя, Должность, Зарплата

5.Фильтрация данных (query, isin, between).

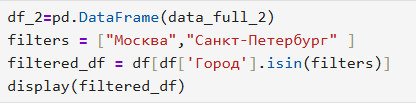


Рисунок 18. Выбор всех клиентов из "Москва" или "Санкт-Петербург"



Рисунок 19. Выбор клиентов, у которых Баланс на счете от 100000 до 250000



Рисунок 20. Фильтр клиентов, у которых "Кредитная история" "Хорошая" и "Баланс на счете»> 150000

6.Подсчет значений (count, value\_counts, nunique).



Рисунок 21. Подсчет количества непустых значений в столбцах



Рисунок 22. Подсчет частоты встречаемых городов

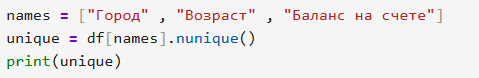


Рисунок 23. Количество уникальных значений в "Город" , "Возраст" , "Баланс на счете"

6.Обнаружение пропусков (isna, notna ).

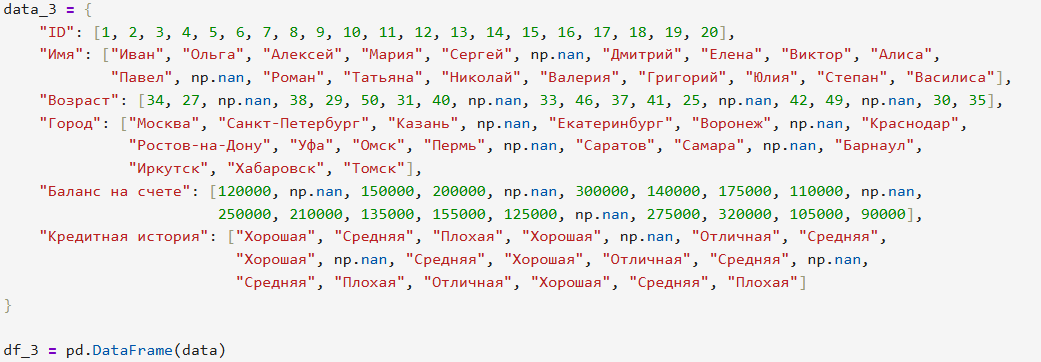


Рисунок 24. df таблицы 3



Рисунок 25. Количество NaN в каждом столбце



Рисунок 26. количество заполненных значений в каждом столбце

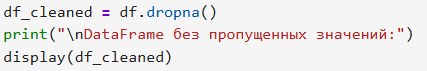


Рисунок 27. Строки, где нет пропущенных значений.

7.Выполнил индивидуальное задание.

**Индивидуальное задание. Вариант 9**

Написать программу на языке программирования Python для решения поставленной задачи (в репозитории должны присутствовать настройки требуемых пакетов для выбранного менеджера пакетов). Приложение должно использовать интерфейс командной строки (модуль argparse ) или графический интерфейс пользователя (модули tkinter , PySide2 , Kivy и т. д.). При работе с датой и временем использовать пакет datetime . Организовать чтение и сохранение данных из/в формат Parquet. Выполнить валидацию сохраненных данных с помощью сторонних приложений для работы с форматом Parquet, например <https://github.com/mukunku/ParquetViewer>. Организовать также удаление данных по одной из колонок DataFrame на выбор обучающихся. Номер варианта определяется по согласованию с преподавателем.Использовать DataFrame , содержащий следующие колонки: название начального пункта маршрута; название конечного пункта маршрута; номер маршрута. Написать программу, выполняющую следующие действия: ввод с клавиатуры данных и добавление строк в DataFrame ; записи должны быть упорядочены по номерам маршрутов; вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, выдать на дисплей соответствующее сообщение.

**Листинг программы:**

import pandas as pd

import os

from datetime import datetime

from IPython.display import display, clear\_output

import ipywidgets as widgets

class RouteManager:

    def \_\_init\_\_(self):

        self.df = pd.DataFrame(columns=['start\_point', 'end\_point', 'route\_number', 'created\_at'])

    def add\_route(self, start: str, end: str, number: int):

        """Добавление нового маршрута с автоматической сортировкой"""

        new\_route = pd.DataFrame({

            'start\_point': [start],

            'end\_point': [end],

            'route\_number': [number],

            'created\_at': [datetime.now()]

        })

        self.df = pd.concat([self.df, new\_route], ignore\_index=True)

        self.df = self.df.sort\_values(by='route\_number').reset\_index(drop=True)

        return f"✅ Маршрут №{number} добавлен: {start} → {end}"

    def find\_route(self, number: int):

        """Поиск маршрута по номеру"""

        result = self.df[self.df['route\_number'] == number]

        return result

    def delete\_by\_column(self, column: str, value: str):

        """Удаление записей по значению в колонке"""

        if column in self.df.columns:

            before = len(self.df)

            self.df = self.df[self.df[column] != value]

            return f"🗑️ Удалено {before - len(self.df)} записей где {column} = {value}"

        return "❌ Ошибка: неверное название колонки"

    def save\_to\_parquet(self, filename: str):

        """Сохранение данных в Parquet-формат"""

        self.df.to\_parquet(filename, engine='pyarrow')

        return f"💾 Данные сохранены в {filename}"

    def load\_from\_parquet(self, filename: str):

        """Загрузка данных из Parquet-файла"""

        if os.path.exists(filename):

            self.df = pd.read\_parquet(filename, engine='pyarrow')

            self.df = self.df.sort\_values(by='route\_number').reset\_index(drop=True)

            return f"📂 Данные загружены из {filename}\nВсего маршрутов: {len(self.df)}"

        return f"❌ Файл {filename} не найден"

    def show\_all\_routes(self):

        """Показать все маршруты"""

        if self.df.empty:

            return "🚫 Нет данных о маршрутах"

        return self.df[['start\_point', 'end\_point', 'route\_number']]

def create\_ui():

    """Создание пользовательского интерфейса"""

    manager = RouteManager()

    output = widgets.Output()

    # Виджеты для добавления маршрута

    start\_input = widgets.Text(placeholder='Москва', description='Откуда:')

    end\_input = widgets.Text(placeholder='Санкт-Петербург', description='Куда:')

    number\_input = widgets.IntText(description='Номер:', value=1)

    add\_button = widgets.Button(description='Добавить', button\_style='success')

    # Виджеты для поиска маршрута

    search\_input = widgets.IntText(description='Поиск:', value=1)

    search\_button = widgets.Button(description='Найти', button\_style='info')

    # Виджеты для удаления маршрутов

    column\_dropdown = widgets.Dropdown(

        options=['Начальный пункт', 'Конечный пункт', 'Номер маршрута'],

        value='Номер маршрута',

        description='Удалить по:'

    )

    value\_input = widgets.Text(description='Значение:', value='')

    delete\_button = widgets.Button(description='Удалить', button\_style='danger')

    # Виджеты для работы с файлами

    filename\_input = widgets.Text(value='routes.parquet', description='Файл:')

    save\_button = widgets.Button(description='Сохранить', button\_style='primary')

    load\_button = widgets.Button(description='Загрузить', button\_style='primary')

    # Обработчики событий

    def on\_add\_button\_clicked(b):

        with output:

            clear\_output()

            try:

                result = manager.add\_route(

                    start\_input.value,

                    end\_input.value,

                    number\_input.value

                )

                print(result)

                display(manager.show\_all\_routes())

            except Exception as e:

                print(f"⛔ Ошибка: {str(e)}")

    def on\_search\_button\_clicked(b):

        with output:

            clear\_output()

            try:

                routes = manager.find\_route(search\_input.value)

                if not routes.empty:

                    print(f"🔍 Найденные маршруты №{search\_input.value}:")

                    display(routes[['start\_point', 'end\_point', 'route\_number']])

                else:

                    print(f"🚫 Маршрутов с номером {search\_input.value} не найдено")

            except Exception as e:

                print(f"⛔ Ошибка: {str(e)}")

    def on\_delete\_button\_clicked(b):

        with output:

            clear\_output()

            try:

                # Преобразование русских названий в английские

                column\_map = {

                    'Начальный пункт': 'start\_point',

                    'Конечный пункт': 'end\_point',

                    'Номер маршрута': 'route\_number'

                }

                column = column\_map[column\_dropdown.value]

                # Преобразование значения для номера маршрута

                if column == 'route\_number':

                    value = int(value\_input.value)

                else:

                    value = value\_input.value

                result = manager.delete\_by\_column(column, value)

                print(result)

                display(manager.show\_all\_routes())

            except Exception as e:

                print(f"⛔ Ошибка: {str(e)}")

    def on\_save\_button\_clicked(b):

        with output:

            clear\_output()

            try:

                print(manager.save\_to\_parquet(filename\_input.value))

            except Exception as e:

                print(f"⛔ Ошибка: {str(e)}")

    def on\_load\_button\_clicked(b):

        with output:

            clear\_output()

            try:

                result = manager.load\_from\_parquet(filename\_input.value)

                print(result)

                display(manager.show\_all\_routes())

            except Exception as e:

                print(f"⛔ Ошибка: {str(e)}")

    # Привязка обработчиков

    add\_button.on\_click(on\_add\_button\_clicked)

    search\_button.on\_click(on\_search\_button\_clicked)

    delete\_button.on\_click(on\_delete\_button\_clicked)

    save\_button.on\_click(on\_save\_button\_clicked)

    load\_button.on\_click(on\_load\_button\_clicked)

    # Группировка виджетов

    add\_box = widgets.VBox([

        widgets.HTML("<h3>Добавить маршрут</h3>"),

        widgets.HBox([start\_input, end\_input, number\_input, add\_button])

    ])

    search\_box = widgets.VBox([

        widgets.HTML("<h3>Поиск маршрута</h3>"),

        widgets.HBox([search\_input, search\_button])

    ])

    delete\_box = widgets.VBox([

        widgets.HTML("<h3>Удалить маршруты</h3>"),

        widgets.HBox([column\_dropdown, value\_input, delete\_button])

    ])

    file\_box = widgets.VBox([

        widgets.HTML("<h3>Работа с файлами</h3>"),

        widgets.HBox([filename\_input, save\_button, load\_button])

    ])

    # Отображение интерфейса

    display(widgets.VBox([

        widgets.HTML("<h2 style='color: #1a73e8;'>🚌 Управление маршрутами</h2>"),

        add\_box,

        search\_box,

        delete\_box,

        file\_box,

        widgets.HTML("<h3>Результат:</h3>"),

        output

    ]))

# Запуск интерфейса

create\_ui()

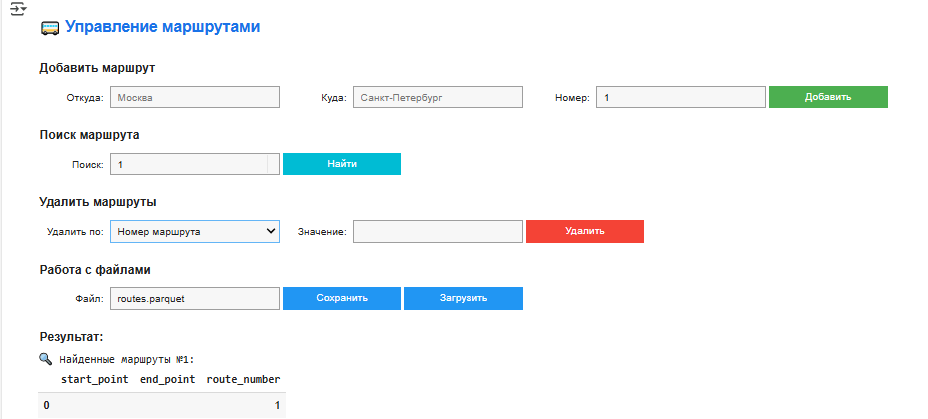


Рисунок 28. Интерфейс для программы

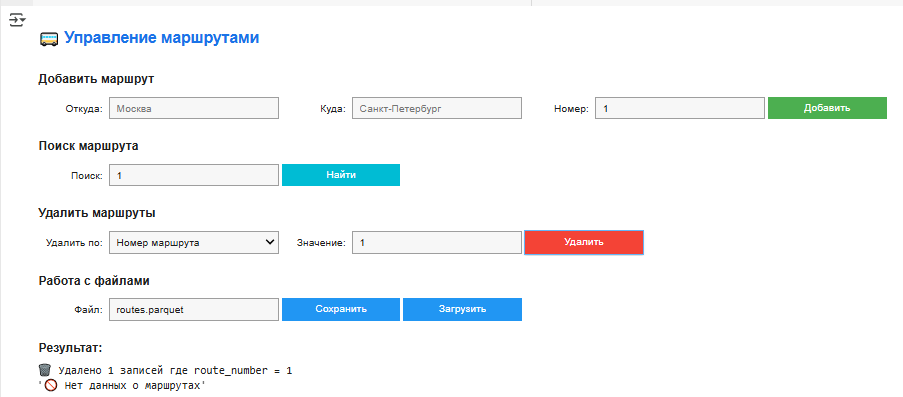


Рисунок 29. Результат работы программы

8. Сохранил изменения в репозитории

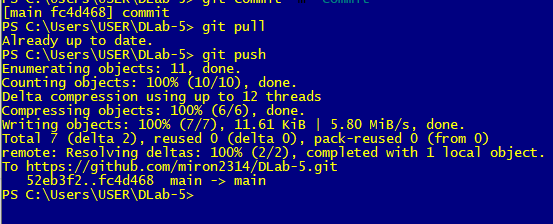
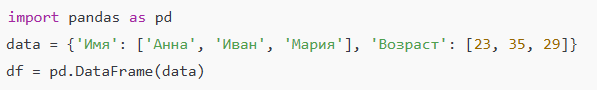


Рисунок 30. Добавление изменений в репозиторий

**Ответы на контрольные вопросы:**

# Как создать pandas.DataFrame из словаря списков?

Чтобы создать DataFrame из словаря списков, используйте конструкцию pd.DataFrame():



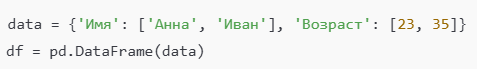
# В чем отличие создания DataFrame из списка словарей и словаря списков?

**Список словарей**: каждый словарь представляет строку данных.

Столбцы будут выведены на основе ключей словаря.

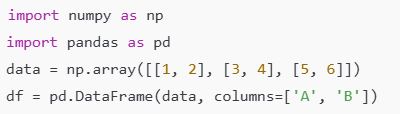


**Словарь списков**: ключи словаря — это названия столбцов, а значения — это списки данных, где каждый элемент в списке соответствует строке.



# Как создать pandas.DataFrame из массива NumPy?

Для создания DataFrame из массива NumPy, передайте его в pd.DataFrame():



# Как загрузить DataFrame из CSV-файла, указав разделитель ;?

Для загрузки данных из CSV с разделителем ; используйте параметр sep:



# Как загрузить данные из Excel в pandas.DataFrame и выбрать конкретный лист?

Для загрузки данных из Excel и указания конкретного листа:



# Чем отличается чтение данных из JSON и Parquet в pandas?

**JSON**: используется для работы с текстовыми данными в формате JSON.



**Parquet**: это бинарный формат, который более эффективен по скорости и использует схемы данных. Он поддерживает типизацию и сжимаемые данные.



# Как проверить типы данных в DataFrame после загрузки?

Для проверки типов данных столбцов используйте атрибут .dtypes:



# Как определить размер DataFrame (количество строк и столбцов)?

Для получения размера используйте атрибут .shape:



# В чем разница между .loc[] и .iloc[]?

**loc[]** используется для выбора по меткам (индексам).

**.iloc[]** используется для выбора по позициям (индексам по порядку).

# Как получить данные третьей строки и второго столбца с .iloc[]?

Используйте .iloc[] с позициями:



# Как получить строку с индексом "Мария" из DataFrame?

Используйте .loc[]:



# Чем .at[] отличается от .loc[]?

**.at[]** используется для получения одного значения по меткам, быстрее, чем .loc[], но работает только для одного значения (не для целых строк или столбцов).

**.loc[]** позволяет выбрать более сложные срезы данных.

# В каких случаях .iat[] работает быстрее, чем .iloc[]?

.iat[] работает быстрее, чем .iloc[], если нужно получить одно значение по позиции, так как .iat[] более специализирован для этого.

# Как выбрать все строки, где "Город" равен "Москва" или "СПб", используя .isin()?

Для выбора строк с .isin():



# Как отфильтровать DataFrame, оставив только строки, где "Возраст" от 25 до 35 лет, используя .between()?

Для фильтрации с .between():



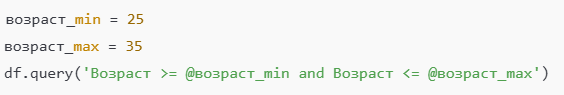
# В чем разница между .query() и .loc[] для фильтрации данных?

**.query()** позволяет использовать строковые выражения для фильтрации, что делает код более читаемым и удобным.

**.loc[]** требует использования явных условий, часто более гибкое, но менее удобное для простых фильтров.

# Как использовать переменные Python внутри .query()?

Для использования переменных в .query() передайте их через параметр local\_dict:



# Как узнать, сколько пропущенных значений в каждом столбце DataFrame?

Используйте .isna() и .sum():



# В чем разница между .isna() и .notna()?

**.isna()** возвращает True для пропущенных значений (NaN).

**.notna()** возвращает True для непустых значений.

# Как вывести только строки, где нет пропущенных значений?

Для вывода строк без пропущенных значений:



# Как добавить новый столбец "Категория" в DataFrame, заполнив его фиксированным значением "Неизвестно"?

Добавьте новый столбец:



# Как добавить новую строку в DataFrame, используя .loc[]?

Используйте .loc[] для добавления строки по новому индексу:



# Как удалить столбец "Возраст" из DataFrame?

Для удаления столбца используйте drop():



# Как удалить все строки, содержащие хотя бы один NaN, из DataFrame?

Для удаления строк с NaN:



# Как удалить столбцы, содержащие хотя бы один NaN, из DataFrame?

Для удаления столбцов с NaN:



# Как посчитать количество непустых значений в каждом столбце DataFrame?

Для подсчета непустых значений используйте .count():



# Чем .value\_counts() отличается от .nunique()?

**.value\_counts()** возвращает количество уникальных значений в столбце.

**.nunique()** возвращает количество уникальных значений, но не учитывает их частоту.

# Как определить сколько раз встречается каждое значение в столбце "Город"?

Используйте .value\_counts():



# Почему display(df) лучше, чем print(df), в Jupyter Notebook?

display(df) в Jupyter отображает DataFrame в виде таблицы с форматированием, что делает данные более читаемыми, в отличие от print(df), который просто выводит их в виде текста.

# Как изменить максимальное количество строк, отображаемых в DataFrame в Jupyter Notebook?

Чтобы изменить максимальное количество строк, отображаемых в DataFrame в Jupyter Notebook, можно использовать параметр pd.set\_option().

Вывод: ознакомился с основами работы с библиотекой pandas, в частности, со структурой данных DataFrame.

**Вывод:** в ходе работы были получены навыки работы с библиотекой pandas, в частности, со структурой данных DataFrame