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.

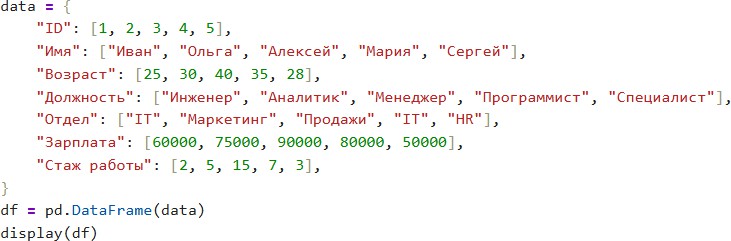


Рисунок 1. Создание df с помощью ditctъ

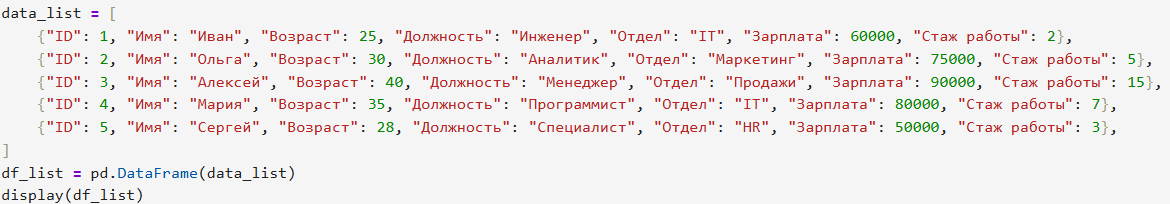


Рисунок 2. Создание df с помощью ditcts list

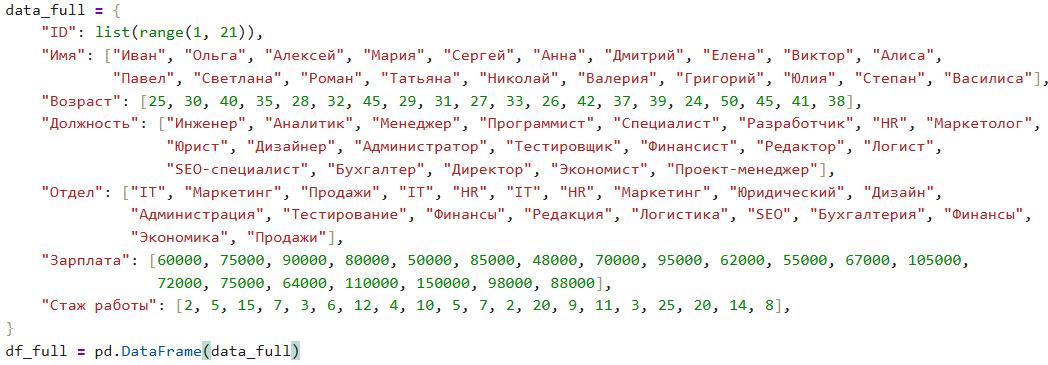


Рисунок 3. Создание df с помощью np массива

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



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

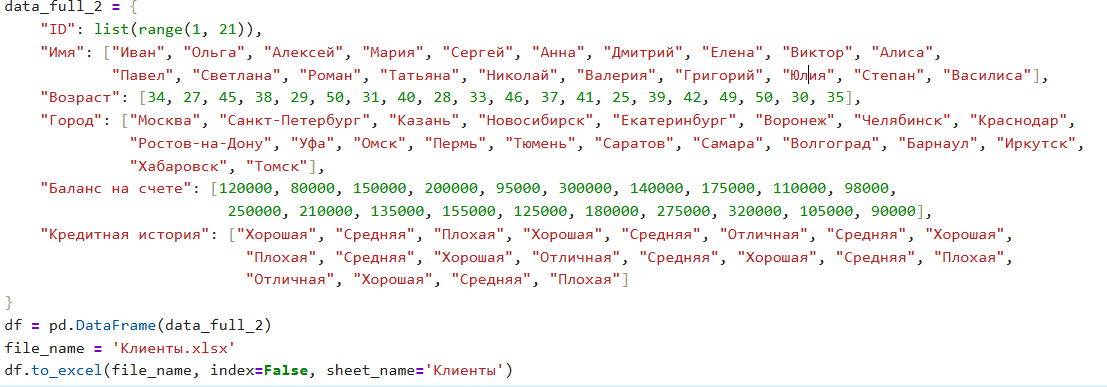


Рисунок 5. Сохранение таблицы 1 df в csv таблицу и загрузка из csv

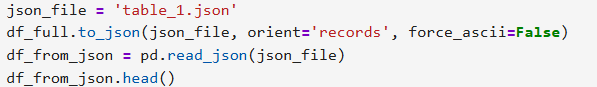


Рисунок 6. Экспорт таблицы 1 в json файл

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.

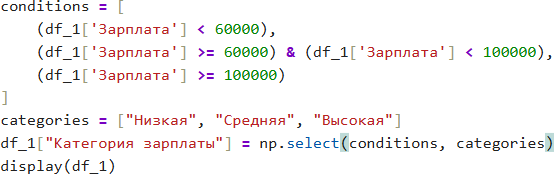


Рисунок 11. Добавление столбца в таблицу с категориями зарплат



Рисунок 12. Добавление сотрудника в таблицу

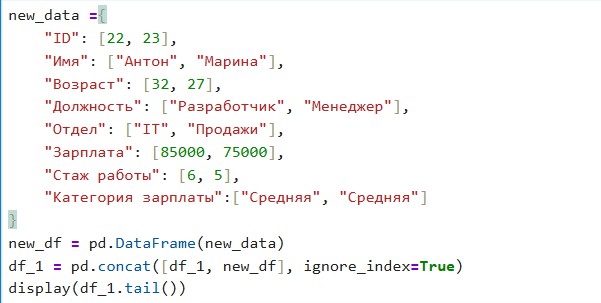


Рисунок 13. Добавление нескольких сотрудника в таблицу

5.Удаление строк и столбцов.



Рисунок 14. Удаление столбца "Категория зарплаты"

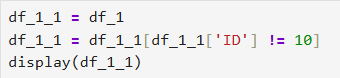


Рисунок 15. Удаление строки с ID =10



Рисунок 16. Удаление строк, где стаж работы <3



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

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

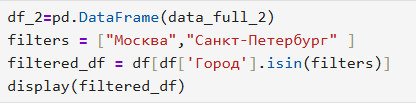


Рисунок 18. Выбор всех клиентов из "Москва" или "Санкт-Петербург"



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

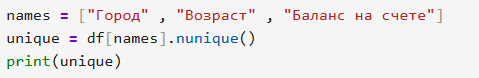


Рисунок 23. Количество уникальных значений в "Город" , "Возраст" , "Баланс на счете"

6.Обнаружение пропусков (isna, notna ).

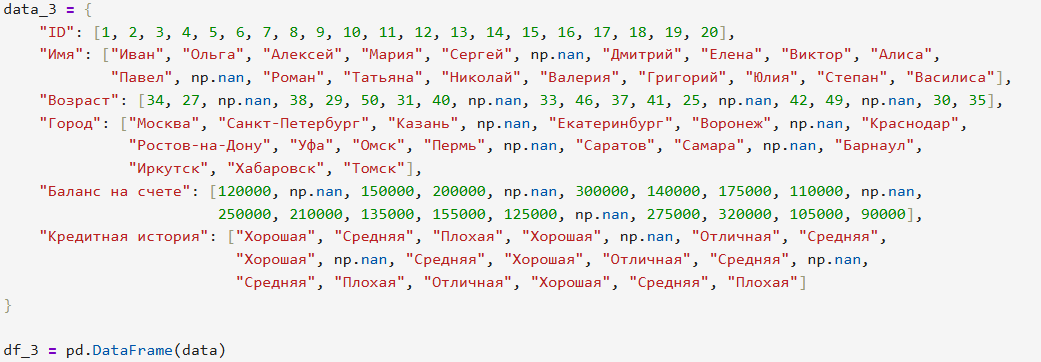


Рисунок 24. df таблицы 3



Рисунок 25. Количество NaN в каждом столбце



Рисунок 26. количество заполненных значений в каждом столбце

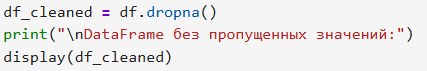


Рисунок 27. Строки, где нет пропущенных значений.

1. Индивидуальное задание.

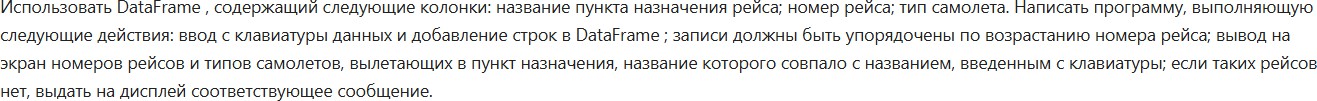


Рисунок 28 . Индивидуальное задание

Код к заданию:

f = pd.DataFrame(columns=["destination", "flight\_number", "aircraft\_type"])

def add\_flight(destination, flight\_number, aircraft\_type): global df

new\_data = pd.DataFrame(

[{"destination": destination, "flight\_number": flight\_number, "aircraft\_type": aircraft\_type}]

)

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

df = df.sort\_values(by="flight\_number").reset\_index(drop=True) def search\_flights(destination):

global df

result = df[df["destination"] == destination] if result.empty:

print(f"Рейсы в пункт {destination} не найдены.") else:

print(result[["flight\_number", "aircraft\_type"]])

def save\_to\_parquet(file\_name="flights.parquet"): global df

df.to\_parquet(file\_name, engine="pyarrow")

def load\_from\_parquet(file\_name="flights.parquet"): global df

df = pd.read\_parquet(file\_name, engine="pyarrow") display(df)

def delete\_flight\_by\_column(column, value): global df

if column not in df.columns:

print("Такой колонки не существует.")

return

df = df[df[column] != value].reset\_index(drop=True) print(f"Удалены записи, где {column} = {value}") display(df)

def input\_add\_flight(b):

destination = destination\_text.value flight\_number = flight\_number\_text.value aircraft\_type = aircraft\_type\_text.value

add\_flight(destination, flight\_number, aircraft\_type)

def input\_search\_flights(b): destination = search\_text.value search\_flights(destination)

def input\_delete\_flight(b): column = column\_text.value value = value\_text.value

delete\_flight\_by\_column(column, value)

def input\_save(b):

file\_name = file\_name\_text.value if file\_name\_text.value else "flights.parquet" save\_to\_parquet(file\_name)

def input\_load(b):

file\_name = load\_file\_name\_text.value if load\_file\_name\_text.value else "flights.parquet"

load\_from\_parquet(file\_name)

destination\_text = widgets.Text(description="Город назначения:") flight\_number\_text = widgets.Text(description="Номер рейса:") aircraft\_type\_text = widgets.Text(description="Тип самолёта:") search\_text = widgets.Text(description="Город для поиска:") column\_text = widgets.Text(description="Колонка для удаления:") value\_text = widgets.Text(description="Значение для удаления:")

file\_name\_text = widgets.Text(description="Имя файла для сохранения:") load\_file\_name\_text = widgets.Text(description="Имя файла для загрузки:")

add\_button = widgets.Button(description="Добавить рейс") search\_button = widgets.Button(description="Поиск рейсов") delete\_button = widgets.Button(description="Удалить рейс") save\_button = widgets.Button(description="Сохранить в файл") load\_button = widgets.Button(description="Загрузить и показать файл")

add\_button.on\_click(input\_add\_flight) search\_button.on\_click(input\_search\_flights) delete\_button.on\_click(input\_delete\_flight) save\_button.on\_click(input\_save) load\_button.on\_click(input\_load)

if name == " main ":

display(destination\_text, flight\_number\_text, aircraft\_type\_text, add\_button) display(search\_text, search\_button)

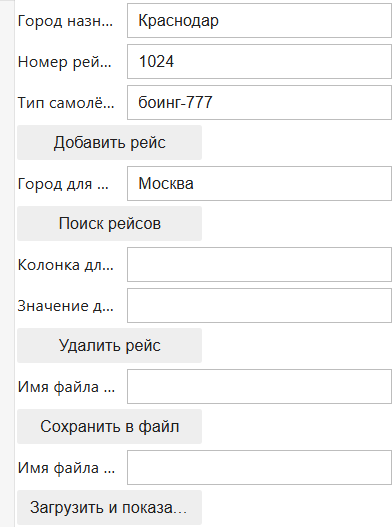
display(column\_text, value\_text, delete\_button) display(file\_name\_text, save\_button) display(load\_file\_name\_text, load\_button)

Рисунок 29 . Интерфейс для программы

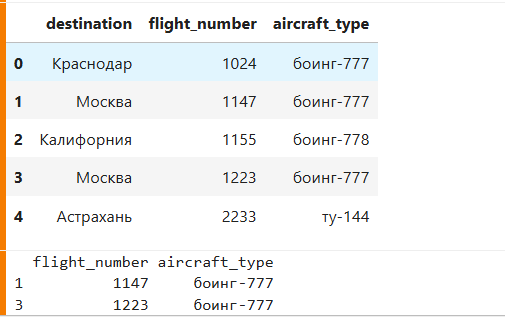
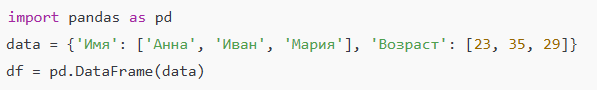


Рисунок 30 . Результат работы программы Ответы на контрольные вопросы:

# Как создать pandas.DataFrame из словаря списков?

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



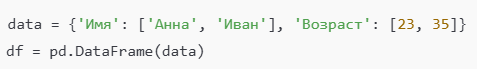
# В чем отличие создания DataFrame из списка словарей и словаря списков?

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

Столбцы будут выведены на основе ключей словаря.

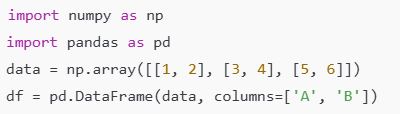


**Словарь списков**: ключи словаря - это названия столбцов, а значения - это списки данных, где каждый элемент в списке соответствует строке.



# 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():



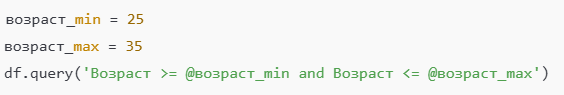
# В чем разница между .query() и .loc[] для фильтрации данных?

**.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