

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»**

Выполнил:  
Хохлачев Вадим Александрович  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи», очная  
форма обучения

---

(подпись)

Проверил:  
Ассистент департамента цифровых,  
робототехнических систем и  
электроники Хацукова А.И.

---

(подпись)

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

Ставрополь, 2025 г.

## Тема: Введение в pandas: изучение структуры DataFrame и базовых операций

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

### Порядок выполнения работы:

1. Ознакомились с теоретическим материалом
2. Создание репозитория

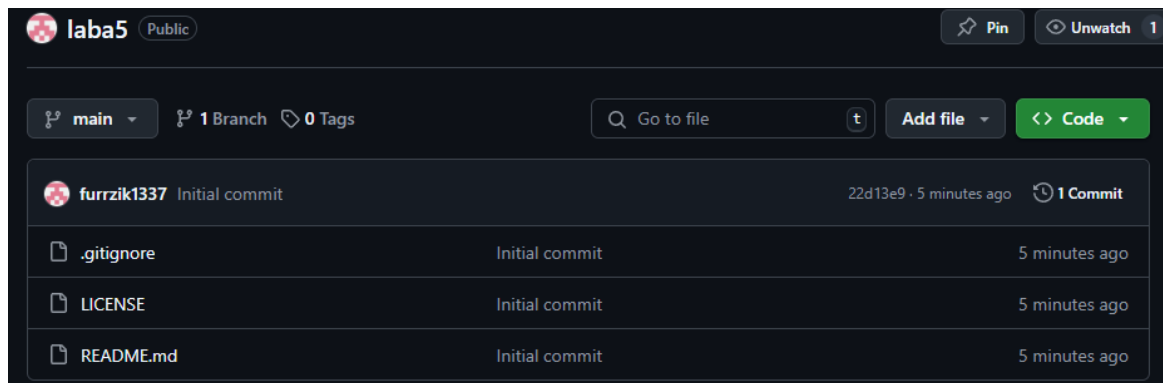


Рисунок 1. Репозиторий

3. Выполнено клонирование репозитория

```
C:\Users\furrzik>git clone https://github.com/furrzik1337/laba5.git
Cloning into 'laba5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование

4. Выполнено практическое задание 1.

```

import pandas as pd

# Данные из таблицы (первые 5 сотрудников)
data_dict = {
    'ID': [1, 2, 3, 4, 5],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей'],
    'Возраст': [25, 30, 40, 35, 28],
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000],
    'Стаж работы': [2, 5, 15, 7, 3]
}

df_dict = pd.DataFrame(data_dict)
print("DataFrame из словаря списков:")
print(df_dict)
print("\nИнформация о типах данных:")
print(df_dict.info())

```

DataFrame из словаря списков:

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3

Информация о типах данных:

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ID           5 non-null      int64
1   Имя          5 non-null      object
2   Возраст      5 non-null      int64
3   Должность    5 non-null      object
4   Отдел        5 non-null      object
5   Зарплата     5 non-null      int64
6   Стаж работы  5 non-null      int64
dtypes: int64(4), object(3)
memory usage: 412.0+ bytes
None

```

Рисунок 1. DataFrame из словаря списков

```

data_list = [
    {'ID': 1, 'Имя': 'Иван', 'Возраст': 25, 'Должность': 'Инженер',
     'Отдел': 'IT', 'Зарплата': 60000, 'Стаж работы': 2},
    {'ID': 2, 'Имя': 'Ольга', 'Возраст': 30, 'Должность': 'Аналитик',
     'Отдел': 'Маркетинг', 'Зарплата': 75000, 'Стаж работы': 5},
    {'ID': 3, 'Имя': 'Алексей', 'Возраст': 40, 'Должность': 'Менеджер',
     'Отдел': 'Продажи', 'Зарплата': 90000, 'Стаж работы': 15},
    {'ID': 4, 'Имя': 'Мария', 'Возраст': 35, 'Должность': 'Программист',
     'Отдел': 'IT', 'Зарплата': 80000, 'Стаж работы': 7},
    {'ID': 5, 'Имя': 'Сергей', 'Возраст': 28, 'Должность': 'Специалист',
     'Отдел': 'HR', 'Зарплата': 50000, 'Стаж работы': 3}
]

df_list = pd.DataFrame(data_list)
print("\nDataFrame из списка словарей:")
print(df_list)
print("\nИнформация о типах данных:")
print(df_list.info())

```

DataFrame из списка словарей:

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3

Информация о типах данных:

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   ID           5 non-null      int64
1   Имя          5 non-null      object
2   Возраст      5 non-null      int64
3   Должность    5 non-null      object
4   Отдел        5 non-null      object
5   Зарплата     5 non-null      int64
6   Стаж работы  5 non-null      int64
dtypes: int64(4), object(3)
memory usage: 412.0+ bytes
None

```

Рисунок 2. DataFrame из списка словарей

```
import numpy as np

# Генерируем случайные возраста от 20 до 60
np.random.seed(42) # Для воспроизводимости
random_ages = np.random.randint(20, 61, size=5)

# Используем те же данные, но с случайными возрастами
df_numpy = pd.DataFrame({
    'ID': [1, 2, 3, 4, 5],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей'],
    'Возраст': random_ages,
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000],
    'Стаж работы': [2, 5, 15, 7, 3]
})

print("\nDataFrame с случайными возрастами из NumPy:")
print(df_numpy)
print("\nИнформация о типах данных:")
print(df_numpy.info())
```

DataFrame с случайными возрастами из NumPy:

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	58	Инженер	IT	60000	2
1	2	Ольга	48	Аналитик	Маркетинг	75000	5
2	3	Алексей	34	Менеджер	Продажи	90000	15
3	4	Мария	27	Программист	IT	80000	7
4	5	Сергей	40	Специалист	HR	50000	3

Информация о типах данных:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    ID          5 non-null      int64
1    Имя         5 non-null      object
2    Возраст     5 non-null      int32
3    Должность   5 non-null      object
4    Отдел       5 non-null      object
5    Зарплата    5 non-null      int64
6    Стаж работы 5 non-null      int64
dtypes: int32(1), int64(3), object(3)
memory usage: 392.0+ bytes
None
```

Рисунок 3. DataFrame из массива NumPy

5. Выполнено практическое задание 2.

```

import pandas as pd

# Создаем DataFrame из Таблицы 1 (сотрудники)
data_employees = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [25, 30, 40, 35, 28, 32, 45, 29, 31, 27, 33, 26, 42, 37, 39, 24, 50, 45, 41, 38],
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист', 'Разработчик', 'HR',
                  'Маркетолог', 'Юрист', 'Дизайнер', 'Администратор', 'Тестировщик', 'Финансист', 'Редактор',
                  'Логист', 'SEO-специалист', 'Бухгалтер', 'Директор', 'Экономист', 'Проект-менеджер'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR', 'IT', 'HR', 'Маркетинг', 'Юридический', 'Дизайн',
              'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Бухгалтерия',
              'Финансы', 'Экономика', 'Продажи'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000, 85000, 48000, 70000, 95000, 62000, 55000, 67000,
                  105000, 72000, 75000, 64000, 110000, 150000, 98000, 88000],
    'Стаж работы': [2, 5, 15, 7, 3, 6, 12, 4, 10, 5, 7, 2, 20, 9, 11, 3, 25, 20, 14, 8]
}

df_employees = pd.DataFrame(data_employees)

# Сохраняем в CSV
df_employees.to_csv('tablica.csv', index=False, encoding='utf-8')

# Читаем из CSV
df_from_csv = pd.read_csv('tablica.csv', encoding='utf-8')
print("Данные из CSV файла:")
print(df_from_csv.head())

```

Данные из CSV файла:

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3

Рисунок 4. Сохранение в CSV файл

```
# Создаем DataFrame из Таблицы 2 (клиенты)
data_clients = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [34, 27, 45, 38, 29, 50, 31, 40, 28, 33, 46, 37, 41, 25, 39, 42, 49, 50, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань', 'Новосибирск', 'Екатеринбург', 'Воронеж', 'Челябинск',
              'Краснодар', 'Ростов-на-Дону', 'Уфа', 'Омск', 'Пермь', 'Тюмень', 'Саратов', 'Самара',
              'Волгоград', 'Барнаул', 'Иркутск', 'Хабаровск', 'Томск'],
    'Баланс на счете': [120000, 80000, 150000, 200000, 95000, 300000, 140000, 175000, 110000, 98000,
                        250000, 210000, 135000, 155000, 125000, 180000, 275000, 320000, 105000, 90000],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Отличная', 'Средняя',
                           'Хорошая', 'Плохая', 'Средняя', 'Хорошая', 'Отличная', 'Средняя', 'Хорошая',
                           'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df_clients = pd.DataFrame(data_clients)

# Сохраняем в Excel
with pd.ExcelWriter('data.xlsx') as writer:
    df_clients.to_excel(writer, sheet_name='Клиенты', index=False)

# Читаем из Excel
df_from_excel = pd.read_excel('data.xlsx', sheet_name='Клиенты')
print("\nДанные из Excel файла:")
print(df_from_excel.head())
```

Данные из Excel файла:

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34	Москва	120000	Хорошая
1	2	Ольга	27	Санкт-Петербург	80000	Средняя
2	3	Алексей	45	Казань	150000	Плохая
3	4	Мария	38	Новосибирск	200000	Хорошая
4	5	Сергей	29	Екатеринбург	95000	Средняя

Рисунок 5. Сохранение в Excel (data.xlsx)

```
# Экспортируем DataFrame сотрудников в JSON
df_employees.to_json('tablica.json', orient='records', force_ascii=False)

# Читаем из JSON
df_from_json = pd.read_json('tablica.json')
print("\nДанные из JSON файла:")
print(df_from_json.head())
```

Данные из JSON файла:

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	25	Инженер	IT	60000	2
1	2	Ольга	30	Аналитик	Маркетинг	75000	5
2	3	Алексей	40	Менеджер	Продажи	90000	15
3	4	Мария	35	Программист	IT	80000	7
4	5	Сергей	28	Специалист	HR	50000	3

Рисунок 6. Экспорт в формат JSON

6. Выполнено практическое задание 3.

```

import pandas as pd

# Создаем DataFrame из Таблицы 1
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [25, 30, 40, 35, 28, 32, 45, 29, 31, 27, 33, 26, 42, 37, 39, 24, 50, 45, 41, 38],
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист', 'Разработчик', 'HR',
                  'Маркетолог', 'Юрист', 'Дизайнер', 'Администратор', 'Тестировщик', 'Финансист', 'Редактор',
                  'Логист', 'SEO-специалист', 'Бухгалтер', 'Директор', 'Экономист', 'Проект-менеджер'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR', 'IT', 'HR', 'Маркетинг', 'Юридический', 'Дизайн',
              'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Бухгалтерия',
              'Финансы', 'Экономика', 'Продажи'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000, 85000, 48000, 70000, 95000, 62000, 55000, 67000,
                  105000, 72000, 75000, 64000, 110000, 150000, 98000, 88000],
    'Стаж работы': [2, 5, 15, 7, 3, 6, 12, 4, 10, 5, 7, 2, 20, 9, 11, 3, 25, 20, 14, 8]
}

df = pd.DataFrame(data)

# 1. Получение информации о сотруднике с ID = 5 с помощью .loc[]
employee_5 = df.loc[df['ID'] == 5]
print("1. Сотрудник с ID = 5:")
print(employee_5)

# 2. Возраст третьего сотрудника в таблице с .iloc[]
# (индексация начинается с 0, поэтому третий сотрудник имеет индекс 2)
age_3rd = df.iloc[2]['Возраст']
print("\n2. Возраст третьего сотрудника:", age_3rd)

# 3. Название отдела для сотрудника "Мария" с .at[]
# Сначала находим индекс строки с именем "Мария"
maria_index = df[df['Имя'] == 'Мария'].index[0]
department_maria = df.at[maria_index, 'Отдел']
print("\n3. Отдел сотрудника Мария:", department_maria)

# 4. Зарплата сотрудника в четвертой строке и пятом столбце с .iat[]
# (индексация начинается с 0: строка 3, столбец 4)
salary = df.iat[3, 4]
print("\n4. Зарплата сотрудника в 4 строке и 5 столбце:", salary)

# Объяснение разницы между методами:
print("\nРазница между методами доступа:")
print("""
.loc[] - доступ по меткам (названиям строк и столбцов)
.iloc[] - доступ по числовым индексам (позициям)
.at[] - быстрый доступ к скалярному значению по метке (аналог .loc для одного значения)
.iat[] - быстрый доступ к скалярному значению по индексу (аналог .iloc для одного значения)
""")

```

Рисунок 7. Доступ к данным с помощью loc,iloc,at,iat

7. Выполнено практическое задание 4.



```

import pandas as pd

# Создаем DataFrame из Таблицы 1
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [25, 30, 40, 35, 28, 32, 45, 29, 31, 27, 33, 26, 42, 37, 39, 24, 50, 45, 41, 38],
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист', 'Разработчик', 'HR',
                  'Маркетолог', 'Юрист', 'Дизайнер', 'Администратор', 'Тестировщик', 'Финансист', 'Редактор',
                  'Логист', 'SEO-специалист', 'Бухгалтер', 'Директор', 'Экономист', 'Проект-менеджер'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR', 'IT', 'HR', 'Маркетинг', 'Юридический', 'Дизайн',
              'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Бухгалтерия',
              'Финансы', 'Экономика', 'Продажи'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000, 85000, 48000, 70000, 95000, 62000, 55000, 67000,
                  105000, 72000, 75000, 64000, 110000, 150000, 98000, 88000],
    'Стаж работы': [2, 5, 15, 7, 3, 6, 12, 4, 10, 5, 7, 2, 20, 9, 11, 3, 25, 20, 14, 8]
}

df = pd.DataFrame(data)

# 1. Добавляем столбец "Категория зарплаты"
conditions = [
    (df['Зарплата'] < 60000),
    (df['Зарплата'] >= 60000 & (df['Зарплата'] < 100000)),
    (df['Зарплата'] >= 100000)
]
categories = ['Низкая', 'Средняя', 'Высокая']
df['Категория зарплаты'] = pd.cut(df['Зарплата'],
                                   bins=[0, 59999, 99999, float('inf')],
                                   labels=categories)

# 2. Добавляем нового сотрудника
df.loc[21] = [21, 'Антон', 32, "Разработчик", "IT", 85000, 6, 'Средняя']

# 3. Добавляем двух новых сотрудников через pd.concat()
new_employees = pd.DataFrame({
    'ID': [22, 23],
    'Имя': ['Михаил', 'Екатерина'],
    'Возраст': [29, 35],
    'Должность': ['Аналитик', 'Менеджер'],
    'Отдел': ['Маркетинг', 'Продажи'],
    'Зарплата': [72000, 95000],
    'Стаж работы': [4, 8],
    'Категория зарплаты': ['Средняя', 'Средняя']
})

df = pd.concat([df, new_employees], ignore_index=True)

# Выводим обновленный DataFrame
print("Обновленный DataFrame:")
print(df)

# Проверяем итоговое количество строк и столбцов
print("\nИтоговые размеры таблицы:", df.shape)

```

Рисунок 8. Добавление новых столбцов и строк

8. Выполнено практическое задание 5.

```

import pandas as pd

# Создаем DataFrame из предоставленных данных
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [25, 30, 40, 35, 28, 32, 45, 29, 31, 27, 33, 26, 42, 37, 39, 24, 50, 45, 41, 38],
    'Должность': ['Инженер', 'Аналитик', 'Менеджер', 'Программист', 'Специалист', 'Разработчик', 'HR', 'Маркетолог',
                  'Юрист', 'Дизайнер', 'Администратор', 'Тестировщик', 'Финансист', 'Редактор', 'Логист', 'SEO-специалист',
                  'Бухгалтер', 'Директор', 'Экономист', 'Проект-менеджер'],
    'Отдел': ['IT', 'Маркетинг', 'Продажи', 'IT', 'HR', 'IT', 'HR', 'Маркетинг', 'Юридический', 'Дизайн',
              'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Бухгалтерия', 'Финансы',
              'Экономика', 'Продажи'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000, 85000, 48000, 70000, 95000, 62000,
                  55000, 67000, 105000, 72000, 75000, 64000, 110000, 150000, 98000, 88000],
    'Стаж работы': [2, 5, 15, 7, 3, 6, 12, 4, 10, 5, 7, 2, 20, 9, 11, 3, 25, 20, 14, 8]
}

df = pd.DataFrame(data)

# 1. Удаление столбца "Категория зарплаты" - такого столбца нет в таблице, пропускаем
print("Исходная таблица:")
print(df)

# 2. Удаление строки с ID = 10
df_step2 = df.drop(df[df['ID'] == 10].index)
print("\nПосле удаления строки с ID = 10:")
print(df_step2)

# 3. Удаление всех строк, где Стаж работы < 3 лет
df_step3 = df_step2[df_step2['Стаж работы'] >= 3]
print("\nПосле удаления строк с опытом работы < 3 лет:")
print(df_step3)

# 4. Удаление всех столбцов, кроме Имя, Должность, Зарплата
df_step4 = df_step3[['Имя', 'Должность', 'Зарплата']]
print("\nПосле оставления только столбцов Имя, Должность, Зарплата:")
print(df_step4)

```

Рисунок 9. Удаление строк и столбцов

9. Выполнено практическое задание 6.

```

import pandas as pd

# Создаем DataFrame из предоставленных данных (объединяем две части таблицы)
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [34, 27, 45, 38, 29, 50, 31, 40, 28, 33, 46, 37, 41, 25, 39, 42, 49, 50, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань', 'Новосибирск', 'Екатеринбург', 'Воронеж', 'Челябинск',
              'Краснодар', 'Ростов-на-Дону', 'Уфа', 'Омск', 'Пермь', 'Тюмень', 'Саратов', 'Самара',
              'Волгоград', 'Барнаул', 'Иркутск', 'Хабаровск', 'Томск'],
    'Баланс на счете': [120000, 80000, 150000, 200000, 95000, 300000, 140000, 175000, 110000, 98000,
                        250000, 210000, 135000, 155000, 125000, 180000, 275000, 320000, 105000, 90000],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Отличная', 'Средняя',
                           'Хорошая', 'Плохая', 'Средняя', 'Хорошая', 'Отличная', 'Средняя', 'Хорошая',
                           'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

# 1. Клиенты из Москвы или Санкт-Петербурга (используем isin)
filter1 = df[df['Город'].isin(['Москва', 'Санкт-Петербург'])]
print("\n1. Клиенты из Москвы или Санкт-Петербурга:")
print(filter1)

# 2. Клиенты с балансом от 100000 до 250000 (используем between)
filter2 = df[df['Баланс на счете'].between(100000, 250000)]
print("\n2. Клиенты с балансом от 100000 до 250000:")
print(filter2)

# 3. Клиенты с хорошей кредитной историей и балансом > 150000 (используем query)
filter3 = df.query("`Кредитная история` == 'Хорошая' and `Баланс на счете` > 150000")
print("\n3. Клиенты с хорошей кредитной историей и балансом > 150000:")
print(filter3)

```

Рисунок 10. Фильтрация данных

10. Выполнено практическое задание 7.

```

import pandas as pd

# Создаем DataFrame из предоставленных данных (объединяем две части таблицы)
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Анна', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Светлана', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [34, 27, 45, 38, 29, 50, 31, 40, 28, 33, 46, 37, 41, 25, 39, 42, 49, 50, 30, 35],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань', 'Новосибирск', 'Екатеринбург', 'Воронеж', 'Челябинск',
              'Краснодар', 'Ростов-на-Дону', 'Уфа', 'Омск', 'Пермь', 'Тюмень', 'Саратов', 'Самара',
              'Волгоград', 'Барнаул', 'Иркутск', 'Хабаровск', 'Томск'],
    'Баланс на счете': [120000, 80000, 150000, 200000, 95000, 300000, 140000, 175000, 110000, 98000,
                        250000, 210000, 135000, 155000, 125000, 180000, 275000, 320000, 105000, 90000],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Отличная', 'Средняя',
                           'Хорошая', 'Плохая', 'Средняя', 'Хорошая', 'Отличная', 'Средняя', 'Хорошая',
                           'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

# 1. Количество непустых значений в каждом столбце
count_non_empty = df.count()
print("1. Количество непустых значений в каждом столбце:")
print(count_non_empty)

# 2. Частота встречаемости значений в столбце "Город"
city_frequency = df['Город'].value_counts()
print("\n2. Частота встречаемости городов:")
print(city_frequency)

# 3. Количество уникальных значений в указанных столбцах
unique_values = {
    'Город': df['Город'].nunique(),
    'Возраст': df['Возраст'].nunique(),
    'Баланс на счете': df['Баланс на счете'].nunique()
}
print("\n3. Количество уникальных значений:")
for column, count in unique_values.items():
    print(f"{column}: {count}")

```

Рисунок 11. Подсчет значений

11. Выполнено практическое задание 8.

```

import pandas as pd
import numpy as np

# Создаем DataFrame из предоставленных данных
data = {
    'ID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17],
    'Имя': ['Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Дмитрий', 'Елена', 'Виктор', 'Алиса',
            'Павел', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Юлия', 'Степан', 'Василиса'],
    'Возраст': [34.0, 27.0, np.nan, 38.0, 29.0, 50.0, 40.0, np.nan, 33.0, 46.0, 37.0, 25.0, np.nan, 42.0, np.nan, 30.0, 35.0],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань', np.nan, 'Екатеринбург', 'Воронеж', 'Краснодар',
              'Ростов-на-Дону', 'Уфа', 'Омск', 'Пермь', 'Саратов', 'Самара', 'Барнаул', 'Иркутск', 'Хабаровск', 'Томск'],
    'Баланс на счете': [120000.0, np.nan, 150000.0, 200000.0, np.nan, 300000.0, 175000.0, 110000.0,
                       np.nan, 250000.0, 210000.0, 155000.0, 125000.0, 275000.0, 320000.0, 105000.0, 90000.0],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', np.nan, 'Отличная', 'Хорошая',
                           np.nan, 'Средняя', 'Хорошая', 'Отличная', np.nan, 'Средняя', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

# 1. Подсчет количества NaN в каждом столбце
nan_count = df.isna().sum()
print("1. Количество пропущенных значений (NaN) в каждом столбце:")
print(nan_count)

# 2. Подсчет количества заполненных значений в каждом столбце
notna_count = df.notna().sum()
print("\n2. Количество заполненных значений в каждом столбце:")
print(notna_count)

# 3. DataFrame только с полными строками (без пропусков)
complete_rows = df.dropna()
print("\n3. DataFrame без строк с пропущенными значениями:")
print(complete_rows)

# Объяснение разницы между isna() и notna()
print("\nОбъяснение:")
print("Метод .isna() возвращает True для пропущенных значений (NaN) и False для заполненных.")
print("Метод .notna() возвращает False для пропущенных значений (NaN) и True для заполненных.")
print("Таким образом, .notna() является противоположностью .isna()")

```

Рисунок 12. Обнаружение пропусков

## 12. Выполнено индивидуальное задание

```

[7]: import pandas as pd
     from IPython.display import display

•[5]: #Установка пакетов и настройка окружения
      !pip install pandas black flake8 isort pre-commit pytest
      !python -m venv .venv

```

Рисунок 13. Импорт библиотек и добавление пакетов

```

# import pandas as pd
from IPython.display import display

# Создаем DataFrame для хранения расписания
columns = ["Пункт назначения", "Номер поезда", "Время отправления"]
schedule = pd.DataFrame(columns=columns)

def add_train()::
    """Функция для добавления нового поезда"""
    global schedule

    print("\nДобавление нового поезда:")
    destination = input("Введите пункт назначения: ")
    train_num = input("Введите номер поезда: ")
    departure_time = input("Введите время отправления (Часы:Минуты): ")

    # Добавляем новую запись
    new_row = pd.DataFrame([[destination, train_num, departure_time]], columns=columns)
    schedule = pd.concat([schedule, new_row], ignore_index=True)

    # Сортируем по времени отправления
    schedule['Время отправления'] = pd.to_datetime(schedule['Время отправления'], format='%H:%M')
    schedule = schedule.sort_values('Время отправления')
    schedule['Время отправления'] = schedule['Время отправления'].dt.strftime('%H:%M')

    print("\nПоезд успешно добавлен!")
    display(schedule)

def search_trains()::
    """Функция для поиска поездов по пункту назначения"""
    print("\nПоиск поездов по пункту назначения")
    destination = input("Введите пункт назначения для поиска: ")

    # Ищем поезда (без учета регистра)
    found_trains = schedule[schedule['Пункт назначения'].str.lower() == destination.lower()]

    if not found_trains.empty:
        print(f"\nНайдено {len(found_trains)} поездов в {destination}:")
        display(found_trains)
    else:
        print(f"\nПоездов в пункт '{destination}' не найдено.")

def main_menu()::

```

Рисунок 14. 1-я часть основного кода

```

def main_menu():
    """Главное меню программы"""
    while True:
        print("\n" + "="*50)
        print("1. Добавить поезд")
        print("2. Найти поезда по пункту назначения")
        print("3. Показать все поезда")
        print("4. Выход")

        choice = input("Выберите действие (1-4): ")

        if choice == "1":
            add_train()
        elif choice == "2":
            search_trains()
        elif choice == "3":
            print("\nТекущее расписание поездов:")
            display(schedule)
        elif choice == "4":
            print("Выход из программы.")
            break
        else:
            print("Неверный ввод. Пожалуйста, выберите от 1 до 4.")

# Запускаем программу
print("="*50)
print("Программа для работы с расписанием поездов")
print("="*50)
main_menu()

```

Рисунок 15. 2-я часть основного кода

```

=====
Программа для работы с расписанием поездов
=====

=====
1. Добавить поезд
2. Найти поезда по пункту назначения
3. Показать все поезда
4. Выход
Выберите действие (1-4): 1

Добавление нового поезда:
Введите пункт назначения: Москва
Введите номер поезда: 7
Введите время отправления (Часы:Минуты): 15:20

Поезд успешно добавлен!

  Пункт назначения  Номер поезда  Время отправления
0             Москва             7             15:20

=====
1. Добавить поезд
2. Найти поезда по пункту назначения
3. Показать все поезда
4. Выход
Выберите действие (1-4): 4
Выход из программы.

```

Рисунок 16. Результат кода

```

#Тест
def test_add_train():
    test_df = pd.DataFrame(columns=columns)
    test_row = pd.DataFrame([["Москва", "001", "10:00"]], columns=columns)
    test_df = pd.concat([test_df, test_row])
    assert len(test_df) == 1
    print("Тест пройден: поезд добавляется корректно")

test_add_train()

Тест пройден: поезд добавляется корректно

```

Рисунок 17. Проверка результата кода

```

#проверки стиля
!pip install nbqa flake8 --quiet
!nbqa flake8 individ.ipynb --ignore=E501

```

Рисунок 18. Проверка стиля



13. Зафиксировал изменения и отправил изменения на репозиторий

### **Ответы на контрольные вопросы:**

#### **1. Как создать pandas.DataFrame из словаря списков?**

```
import pandas as pd  
  
data = {'Колонка1': [1, 2, 3], 'Колонка2': ['a', 'b', 'c']}  
  
df = pd.DataFrame(data)
```

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

- Список словарей — каждый словарь соответствует одной строке DataFrame, ключи — имена столбцов. - Словарь списков — ключи — имена столбцов, а списки — значения для этих столбцов (каждый элемент списка — значение в строке).

#### **3. Как создать pandas.DataFrame из массива NumPy ?**

Для создания pandas.DataFrame из массива NumPy достаточно передать массив в конструктор pd.DataFrame(). Например:

```
df = pd.DataFrame(numpy_array)
```

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

Используйте функцию pd.read\_csv() с параметром sep=';', например:

```
df = pd.read_csv('filename.csv', sep=';')
```

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

Используйте функцию pd.read\_excel() с параметром sheet\_name, например:

```
df = pd.read_excel('file.xlsx', sheet_name='Sheet1')
```

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

Чтение из JSON (pd.read\_json) загружает данные из текстового формата с иерархической структурой, часто медленнее и с большим размером файла; чтение из Parquet (pd.read\_parquet) — из бинарного колоночного формата, который быстрее и эффективнее хранит большие объёмы данных.

#### **7. Как проверить типы данных в DataFrame после загрузки?**

`df.dtypes` - выводит типы данных для каждого столбца.

**8. Как определить размер DataFrame (количество строк и столбцов)?**

`df.shape` возвращает кортеж (количество строк, количество столбцов).

**9. В чем разница между `.loc[]` и `.iloc[]` ?**

`.loc[]` работает с метками (именами строк/столбцов), `.iloc[]` - с числовыми индексами.

**10. Как получить данные третьей строки и второго столбца с `.iloc[]` ?**

`df.iloc[2, 1]` (индексация с 0).

**11. Как получить строку с индексом "Мария" из DataFrame ?**

`df.loc['Мария']`.

**12. Чем `.at[]` отличается от `.loc[]` ?**

`.at[]` быстрее для доступа к одному элементу, `.loc[]` более универсален.

**13. В каких случаях `.iat[]` работает быстрее, чем `.iloc[]` ?**

`.iat[]` работает быстрее `.iloc[]` при доступе к одиночным значениям, так как:

- Оптимизирован специально для скалярного доступа
- Не поддерживает срезы и булеву индексацию

Полезен в циклах при обработке отдельных ячеек

Пример: `df.iat[2, 3]` быстрее чем `df.iloc[2, 3]` для одного значения

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

`df[df['Город'].isin(['Москва', 'СПб'])]`

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

`df[df['Возраст'].between(25, 35)]`

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

`.query()` использует строковые выражения, `.loc[]` - булевы массивы.

**17. Как использовать переменные Python внутри `.query()` ?**

Используйте @: df.query('Возраст > @min\_age').

**18. Как узнать, сколько пропущенных значений в каждом столбце DataFrame ?**

df.isna().sum()

**19. В чем разница между .isna() и .notna() ?**

.isna() находит пропуски, .notna() - не пропуски.

**20. Как вывести только строки, где нет пропущенных значений?**

df.dropna()

**21. Как добавить новый столбец "Категория" в DataFrame , заполнив его фиксированным значением "Неизвестно" ?**

df['Категория'] = 'Неизвестно'

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

df.loc[новый\_индекс] = [значения].

**23. Как удалить столбец "Возраст" из DataFrame ?**

df.drop('Возраст', axis=1, inplace=True)

**24. Как удалить все строки, содержащие хотя бы один NaN , из DataFrame ?**

df.dropna()

**25. Как удалить столбцы, содержащие хотя бы один NaN , из DataFrame ?**

df.dropna(axis=1)

**26. Как посчитать количество непустых значений в каждом столбце DataFrame ?**

df.count().

**27. Чем .value\_counts() отличается от .nunique() ?**

.value\_counts() - частоты значений, .nunique() - количество уникальных.

**28. Как определить сколько раз встречается каждое значение в столбце "Город" ?**

df['Город'].value\_counts().

**29. Почему display(df) лучше, чем print(df) , в Jupyter Notebook?**

`display()` форматирует вывод в Jupyter.

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

```
pd.set_option('display.max_rows', N)
```

**Вывод:** познакомились с основами работы с библиотекой pandas, в частности, со структурой данных DataFrame

**Ссылка на GitHub:** <https://github.com/furrzik1337/laba5>