

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-3**

Выполнил:  
Пугачев Кирилл Дмитриевич  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 Инфокоммуникационные  
технологии и системы связи,  
очная форма обучения

---

(подпись)

Проверила:  
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 SERIES И БАЗОВЫХ ОПЕРАЦИЙ

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

**Ссылка на репозиторий:** <https://github.com/chillkirill/LABA5AI>

**Ход работы:**

## 1. Выполнение практических заданий.

```
Администратор: Anaconda Prompt
vs2015_runtime pkgs/main/win-64::vs2015_runtime-14.42.34433-hbfb602d_5
wheel pkgs/main/win-64::wheel-0.45.1-py312haa95532_0
xz pkgs/main/win-64::xz-5.6.4-h4754444_1
zlib pkgs/main/win-64::zlib-1.2.13-h8cc25b3_1

Proceed ([y]/n)? y

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate myenv
#
# To deactivate an active environment, use
#
#   $ conda deactivate

(base) C:\Users\user>conda activate myenv
(myenv) C:\Users\user>
```

Рисунок 1. Установка окружения

```
Администратор: Anaconda Prompt
css2-1.4.0 tornado-6.5.1 traitlets-5.14.3 txt2tags-3.9 types-python-dateutil-2
-template-1.3.0 urllib3-2.4.0 wcwidth-0.2.13 webcolors-24.11.1 webencodings-0.
ion-4.0.14

(myenv) C:\Users\user>pip install numpy
Requirement already satisfied: numpy in e:\anaconda\envs\myenv\lib\site-packag

(myenv) C:\Users\user>pip install pandas
Collecting pandas
  Downloading pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.26.0 in e:\anaconda\envs\myenv\lib\sit
Requirement already satisfied: python-dateutil>=2.8.2 in e:\anaconda\envs\myen
ost0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in e:\anaconda\envs\myenv\lib\site-pac
) (1.17.0)
Downloading pandas-2.2.3-cp312-cp312-win_amd64.whl (11.5 MB)
----- 11.5/11.5 MB 10.6 MB/s eta 0:00:00
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: pytz, tzdata, pandas
Successfully installed pandas-2.2.3 pytz-2025.2 tzdata-2025.2

(myenv) C:\Users\user>
```

Рисунок 2. Установка библиотек

```
Администратор: Anaconda Prompt - conda activate myenv - conda install ipykernel

six                pkgs/main/win-64::six-1.17.0-py312haa95532_0
stack_data         pkgs/main/noarch::stack_data-0.2.0-pyhd3eb1b0_0
tornado            pkgs/main/win-64::tornado-6.5-py312h827c3e9_0
traitlets          pkgs/main/win-64::traitlets-5.14.3-py312haa95532_0
wcwidth            pkgs/main/noarch::wcwidth-0.2.5-pyhd3eb1b0_0
zeromq             pkgs/main/win-64::zeromq-4.3.5-hd77b12b_0

Proceed ([y]/n)? python -m ipykernel install --user --name=myenv --display-name="Python (myenv)"
Invalid choice: python -m ipykernel install --user --name=myenv --display-name="python (myenv)"
Proceed ([y]/n)? y

Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(myenv) C:\Users\user>
```

Рисунок 3. Активация myenv



Рисунок 4. Результат

```
(myenv) C:\Users\user>conda install -c conda-forge black
Channels:
 - conda-forge
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda\envs\myenv

  added / updated specs:
    - black

The following packages will be downloaded:



| package                   | build        | size   | channel     |
|---------------------------|--------------|--------|-------------|
| black-25.1.0              | pyh866005b_0 | 169 KB | conda-forge |
| ca-certificates-2025.4.26 | h4c7d964_0   | 149 KB | conda-forge |


```

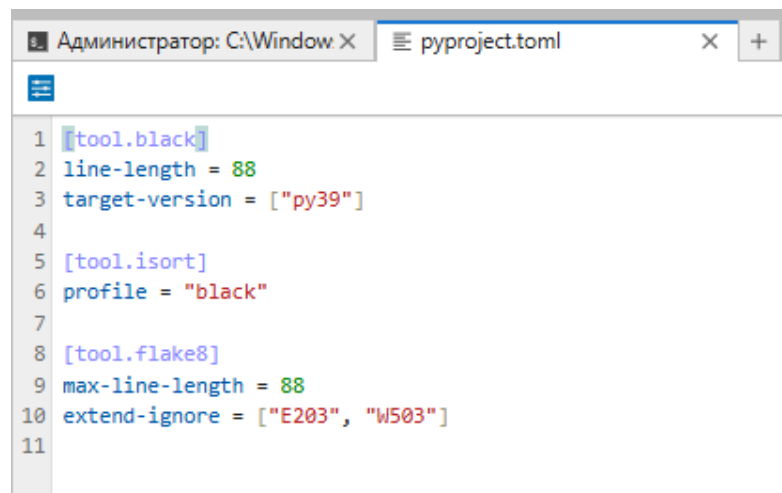
Рисунок 5. Установка утилиты black

```

PS E:\Anaconda\Scripts\llaba\5 laba> isort . --check-only
Skipped 2 files
PS E:\Anaconda\Scripts\llaba\5 laba> black .
Skipping ip n i p p n n i n in
n i i nning pip in k p
n 🍌🍌🍌
12 left unchanged.
PS E:\Anaconda\Scripts\llaba\5 laba> isort .
Skipped 2 files
PS E:\Anaconda\Scripts\llaba\5 laba> flake8 .
.\ipynb_checkpoints\1ind-checkpoint.py:5:23: F821 undefined name 'null'
.\ipynb_checkpoints\1ind-checkpoint.py:14:89: E501 line too long (91 > 88 characters)
.\ipynb_checkpoints\1ind-checkpoint.py:32:89: E501 line too long (101 > 88 characters)
.\ipynb_checkpoints\1ind-checkpoint.py:52:89: E501 line too long (121 > 88 characters)
.\ipynb_checkpoints\1ind-checkpoint.py:61:89: E501 line too long (101 > 88 characters)
.\ipynb_checkpoints\1ind-checkpoint.py:65:89: E501 line too long (102 > 88 characters)
.\ipynb_checkpoints\1ind-checkpoint.py:72:23: F821 undefined name 'null'
.\ipynb_checkpoints\1pr-checkpoint.py:66:89: E501 line too long (91 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:75:89: E501 line too long (129 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:76:89: E501 line too long (138 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:77:89: E501 line too long (139 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:78:89: E501 line too long (134 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:79:89: E501 line too long (133 > 88 characters)
.\ipynb_checkpoints\1pr-checkpoint.py:101:23: F821 undefined name 'null'
.\ipynb_checkpoints\2.1pr-checkpoint.py:64:89: E501 line too long (91 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:65:89: E501 line too long (114 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:66:89: E501 line too long (126 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:67:89: E501 line too long (105 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:68:89: E501 line too long (146 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:69:89: E501 line too long (169 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:70:89: E501 line too long (235 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:71:89: E501 line too long (169 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:72:89: E501 line too long (96 > 88 characters)
.\ipynb_checkpoints\2.1pr-checkpoint.py:89:23: F821 undefined name 'null'
.\ipynb_checkpoints\2.2pr-checkpoint.py:42:89: E501 line too long (91 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:43:89: E501 line too long (114 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:44:89: E501 line too long (126 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:45:89: E501 line too long (105 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:46:89: E501 line too long (154 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:47:89: E501 line too long (128 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:48:89: E501 line too long (109 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:49:89: E501 line too long (112 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:50:89: E501 line too long (143 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:51:89: E501 line too long (144 > 88 characters)
.\ipynb_checkpoints\2.2pr-checkpoint.py:69:23: F821 undefined name 'null'
.\ipynb_checkpoints\2.3pr-checkpoint.py:26:89: E501 line too long (91 > 88 characters)
.\ipynb_checkpoints\2.3pr-checkpoint.py:27:89: E501 line too long (114 > 88 characters)

```

Рисунок 6. Результат утилитов



```

1 [tool.black]
2 line-length = 88
3 target-version = ["py39"]
4
5 [tool.isort]
6 profile = "black"
7
8 [tool.flake8]
9 max-line-length = 88
10 extend-ignore = ["E203", "W503"]
11

```

Рисунок 7. Создание конфигурационного файла

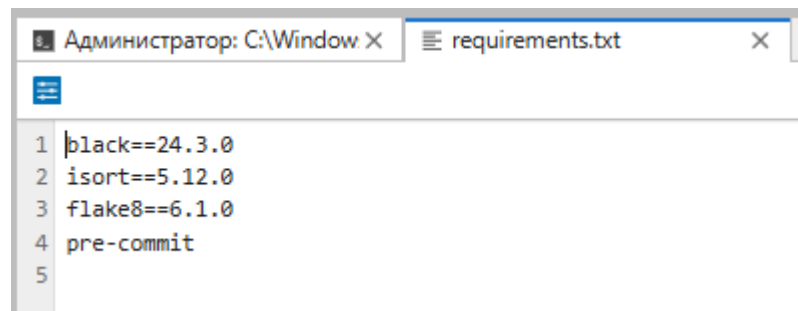


Рисунок 8. Создание конфигурационного файла.

```
import pandas as pd
import numpy as np

# DataFrame из словаря списков
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': ['ИТ', 'Маркетинг', 'Продажи', 'ИТ', 'HR'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000],
    'Стаж работы': [2, 5, 15, 7, 3]
}
df_from_dict = pd.DataFrame(data_dict)

# DataFrame из списка словарей
list_of_dicts = [
    {'ID': 1, 'Имя': 'Иван', 'Возраст': 25, 'Должность': 'Инженер', 'Отдел': 'ИТ', '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': 'ИТ', 'Зарплата': 80000, 'Стаж работы': 7},
    {'ID': 5, 'Имя': 'Сергей', 'Возраст': 28, 'Должность': 'Специалист', 'Отдел': 'HR', 'Зарплата': 50000, 'Стаж работы': 3}
]
df_from_list_of_dicts = pd.DataFrame(list_of_dicts)

# DataFrame из массива NumPy со случайными числами (от 20 до 60) для возраста
np.random.seed(0)
random_ages = np.random.randint(20, 61, size=5)
df_from_np = pd.DataFrame({'Возраст': random_ages})

# Проверка типов данных с помощью .info()
print("DataFrame из словаря списков:")
df_from_dict.info()

print("\nDataFrame из списка словарей:")
df_from_list_of_dicts.info()

print("\nDataFrame из массива NumPy:")
df_from_np.info()
```

Рисунок 9. Практическое задание 1

```
DataFrame из словаря списков:
<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
```

```
DataFrame из списка словарей:
<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
```

```
DataFrame из массива NumPy:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 1 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Возраст      5 non-null      int32
dtypes: int32(1)
memory usage: 152.0 bytes
```

Рисунок 10. Результат первого задания

```
[12]: import pandas as pd

# DataFrame из 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', 'Илья', 'Степан', 'Василис'],
    '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': ['ИТ', 'Маркетинг', 'Продажи', 'ИТ', 'ИТ', 'ИТ', 'ИТ', 'Маркетинг', 'Юридический', 'Дизайн', 'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Финансы'],
    'Зарплата': [60000, 75000, 90000, 80000, 50000, 85000, 45000, 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)

# DataFrame в CSV-файл
csv_filename = 'tab1.csv'
df.to_csv(csv_filename, index=False, encoding='utf-8')

# таблицу обратно из CSV в DataFrame
loaded_df = pd.read_csv(csv_filename)

print(loaded_df)
```

ID	Имя	Возраст	Должность	Отдел	Зарплата
0	Иван	25	Инженер	ИТ	60000
1	Ольга	30	Аналитик	Маркетинг	75000
2	Алексей	40	Менеджер	Продажи	90000
3	Мария	35	Программист	ИТ	80000
4	Сергей	28	Специалист	HR	50000
5	Анна	32	Разработчик	ИТ	85000
6	Дмитрий	45	HR	ИТ	45000
7	Елена	29	Маркетолог	Маркетинг	70000
8	Виктор	31	Юрист	Юридический	95000
9	Алиса	27	Дизайнер	Дизайн	62000
10	Павел	33	Администратор	Администрация	55000
11	Светлана	26	Тестировщик	Тестирование	67000
12	Роман	42	Финансист	Финансы	105000
13	Татьяна	37	Редактор	Редакция	72000
14	Николай	39	Логист	Логистика	75000
15	Валерия	24	SEO-специалист	SEO	64000
16	Григорий	50	Бухгалтер	Финансы	110000
17	Илья	45	Директор	Финансы	150000
18	Степан	41	Экономист	Продажи	98000
19	Василис	38	Проект-менеджер	Продажи	88000

Рисунок 11. Практическое задание 2.1

```

import pandas as pd

# DataFrame из таблицы2
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', 'Василисa'],
    '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',
                           'Средняя', 'Отличная', 'Средняя', 'Хорошая', 'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df_clients = pd.DataFrame(data_clients)

# DataFrame в Excel-файл
excel_filename = 'tab2.xlsx'
sheet_name = 'Клиенты'
df_clients.to_excel(excel_filename, sheet_name=sheet_name, index=False)

# таблица из файла обратно в DataFrame
loaded_df = pd.read_excel(excel_filename, sheet_name=sheet_name)

print(loaded_df)

```

	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
5	6	Анна	50	Воронеж	300000	Отличная
6	7	Дмитрий	31	Челябинск	140000	Средняя
7	8	Елена	40	Краснодар	175000	Хорошая
8	9	Виктор	28	Ростов-на-Дону	110000	Плохая
9	10	Алиса	33	Уфа	98000	Средняя
10	11	Павел	46	Омск	250000	Средняя
11	12	Светлана	37	Пермь	210000	Отличная
12	13	Роман	41	Тюмень	135000	Средняя
13	14	Татьяна	25	Саратов	155000	Хорошая
14	15	Николай	39	Самара	125000	Средняя
15	16	Валерия	42	Волгоград	180000	Плохая
16	17	Григорий	49	Барнаул	275000	Отличная
17	18	Юлия	50	Иркутск	320000	Хорошая
18	19	Степан	30	Хабаровск	105000	Средняя
19	20	Василисa	35	Томск	90000	Плохая

Рисунок 12. Практическое задание 2.2

```

import pandas as pd

df_from_dict = pd.DataFrame({
    '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', 'Василисa'],
    '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': ['ИТ', 'Маркетинг', 'Продажи', 'ИТ', 'HR', 'ИТ', 'HR', 'Маркетинг', 'Юридический', 'Дизайн', 'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Финансы'],
    '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]
})

# Экспорт DataFrame в JSON файл
json_filename = 'employees.json'
df_from_dict.to_json(json_filename, orient='records', force_ascii=False)

# JSON файл обратно в DataFrame
loaded_json_df = pd.read_json(json_filename, orient='records')

# первые 5 строк
print(loaded_json_df.head())

```

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

Рисунок 13. Практическое задание 2.3



```
[2]: import pandas as pd

# DataFrame из 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': [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': ['ИТ', 'Маркетинг', 'Продажи', 'ИТ', 'HR', 'ИТ', 'HR', 'Маркетинг', 'Юридический', 'Дизайн', 'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Финансы'],
    '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)

# информация о сотруднике с ID = 5 с помощью .loc[]
df.set_index('ID', inplace=True)
info_ID_5 = df.loc[5]
print("Информация о сотруднике с ID=5:\n", info_ID_5)

# возраст третьего сотрудника в таблице с помощью .iloc[]
age_third_employee = df.iloc[2]['Возраст']
print("\nВозраст третьего сотрудника:", age_third_employee)

# название отдела для сотрудника "Мария" с помощью .at[]
department_maria = df.at[df.index[df['Имя'] == 'Мария'][0], 'Отдел']
print("\nОтдел сотрудницы 'Мария':", department_maria)

# зарплата сотрудника, находящегося в четвертой строке и пятом столбце, используя .iat[]
salary_value = df.iat[3, 4]
print("\nЗарплата сотрудника из 4-й строки и 5-го столбца:", salary_value)
```

Информация о сотруднике с ID=5:

Имя	Сергей
Возраст	28
Должность	Специалист
Отдел	HR
Зарплата	85000
Стаж работы	3

Name: 5, dtype: object

Возраст третьего сотрудника: 40

Отдел сотрудницы 'Мария': ИТ

Зарплата сотрудника из 4-й строки и 5-го столбца: 80000

Рисунок 14. Практическое задание 3

```
import pandas as pd

# Создаем DataFrame из 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': [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': ['ИТ', 'Маркетинг', 'Продажи', 'ИТ', 'HR', 'ИТ', 'HR', 'Маркетинг', 'Юридический', 'Дизайн', 'Администрация', 'Тестирование', 'Финансы', 'Редакция', 'Логистика', 'SEO', 'Финансы'],
    '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)

# новые строки как DataFrame
new_employees = pd.DataFrame([
    ['Антон', 32, 'Разработчик', 'ИТ', 85000, 6],
    ['Ольга', 29, 'Аналитик', 'Маркетинг', 95000, 4]
], columns=['Имя', 'Возраст', 'Должность', 'Отдел', 'Зарплата', 'Стаж работы'])

# Добавляем в DataFrame
df = pd.concat([df, new_employees], ignore_index=True)

print(df)
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата
0	1.0	Иван	25	Инженер	ИТ	60000
1	2.0	Ольга	30	Аналитик	Маркетинг	75000
2	3.0	Алексей	40	Менеджер	Продажи	90000
3	4.0	Мария	35	Программист	ИТ	80000
4	5.0	Сергей	28	Специалист	HR	85000
5	6.0	Анна	32	Разработчик	ИТ	85000
6	7.0	Дмитрий	45	HR	HR	48000
7	8.0	Елена	29	Маркетолог	Маркетинг	70000
8	9.0	Виктор	31	Юрист	Юридический	95000
9	10.0	Алиса	27	Дизайнер	Дизайн	62000
10	11.0	Павел	33	Администратор	Администрация	55000
11	12.0	Светлана	26	Тестировщик	Тестирование	67000
12	13.0	Роман	42	Финансист	Финансы	105000
13	14.0	Татьяна	37	Редактор	Редакция	72000
14	15.0	Николай	39	Логист	Логистика	75000
15	16.0	Валерия	24	SEO-специалист	SEO	64000
16	17.0	Григорий	50	Бухгалтер	Финансы	110000
17	18.0	Юлия	45	Директор	Финансы	150000
18	19.0	Степан	41	Экономист	Продажи	98000
19	20.0	Василиса	38	Проект-менеджер	Продажи	88000
20	NaN	Антон	32	Разработчик	ИТ	85000
21	NaN	Ольга	29	Аналитик	Маркетинг	95000

Рисунок 15. Практическое задание 4



```

1]: import pandas as pd

# DataFrame из 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': [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', 'Финансы'],
    '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)

df['Категория зарплаты'] = pd.cut(df['Зарплата'], bins=[0, 60000, 100000, float('inf')], labels=['Низкая', 'Средняя', 'Высокая'], right=False)

print("До удаления столбца 'Категория зарплаты':")
print(df)
# Удалил
df = df.drop(columns=['Категория зарплаты'])
print("\nПосле удаления столбца 'Категория зарплаты':")
print(df)

# Удалить строку с ID = 10
df = df[df['ID'] != 10]
print("\nПосле удаления строки с ID=10:")
print(df)

# Удалить все строки, где Стаж работы < 3 лет
df = df[df['Стаж работы'] >= 3]
print("\nПосле удаления строк, где стаж работы < 3 лет:")
print(df)

# Удалить все столбцы, кроме Имя, Должность, Зарплата
df = df[['Имя', 'Должность', 'Зарплата']]
print("\nПосле удаления всех столбцов, кроме 'Имя', 'Должность', 'Зарплата':")
print(df)

```

ID	Имя	Возраст	Должность	Отдел	Зарплата	
0	1	Иван	25	Инженер	IT	60000
1	2	Ольга	30	Аналитик	Маркетинг	75000
2	3	Алексей	40	Менеджер	Продажи	90000
3	4	Мария	35	Программист	IT	80000
4	5	Сергей	28	Специалист	HR	50000
5	6	Анна	32	Разработчик	IT	85000

Рисунок 16. Практическое задание 5

```

import pandas as pd

# DataFrame из 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, 50, 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, 150000, 200000, 95000, 300000, 140000, 175000, 110000, 98000,
                       250000, 21000, 135000, 155000, 125000, 180000, 275000, 320000, 105000, 90000],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Отличная', 'Средняя', 'Хорошая', 'Плохая', 'Средняя',
                           'Средняя', 'Отличная', 'Средняя', 'Хорошая', 'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

# Выберите всех клиентов из "Москва" или "Санкт-Петербург" с помощью .isin()
cities = ['Москва', 'Санкт-Петербург']
clients_in_cities = df[df['Город'].isin(cities)]
print("Клиенты из Москвы или Санкт-Петербурга:")
print(clients_in_cities)

# Выберите клиентов, у которых Баланс на счете от 100000 до 250000 с помощью .between()
clients_balance_between = df[df['Баланс на счете'].between(100000, 250000)]
print("\nКлиенты с балансом от 100000 до 250000:")
print(clients_balance_between)

# Отфильтруйте клиентов, у которых "Кредитная история" "Хорошая" и "Баланс на счете" > 150000 с помощью .query()
clients_good_credit_high_balance = df.query('`Кредитная история` == "Хорошая" and `Баланс на счете` > 150000')
print("\nКлиенты с хорошей кредитной историей и балансом > 150000:")
print(clients_good_credit_high_balance)

```

ID	Имя	Возраст	Город	Баланс на счете	Кредитная история	
0	1	Иван	34	Москва	120000	Хорошая
1	2	Ольга	27	Санкт-Петербург	80000	Средняя

ID	Имя	Возраст	Город	Баланс на счете	Кредитная история	
0	1	Иван	34	Москва	120000	Хорошая
2	3	Алексей	45	Казань	150000	Плохая
3	4	Мария	38	Новосибирск	200000	Хорошая
6	7	Дмитрий	31	Челябинск	140000	Средняя
7	8	Елена	40	Краснодар	175000	Хорошая
8	9	Виктор	28	Ростов-на-Дону	110000	Плохая
10	11	Павел	46	Омск	250000	Средняя
12	13	Роман	41	Тюмень	135000	Средняя
13	14	Татьяна	25	Саратов	155000	Хорошая

Рисунок 17. Практическое задание 6

```
import pandas as pd

# DataFrame из 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, 50, 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, 320000, 110000, 98000,
                       250000, 21000, 135000, 155000, 125000, 180000, 275000, 320000, 105000, 90000],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Отличная', 'Средняя', 'Хорошая', 'Плохая', 'Средняя',
                           'Средняя', 'Отличная', 'Средняя', 'Хорошая', 'Средняя', 'Плохая', 'Отличная', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

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

# частота встречаемости значений в "Город" (.value_counts())
city_counts = df['Город'].value_counts()
print("\nЧастота встречаемости значений в 'Город':")
print(city_counts)

# количество уникальных значений в "Город", "Возраст" и "Баланс на счете" (.nunique())
unique_counts = {
    'Город': df['Город'].nunique(),
    'Возраст': df['Возраст'].nunique(),
    'Баланс на счете': df['Баланс на счете'].nunique()
}
print("\nКоличество уникальных значений:")
for key, value in unique_counts.items():
    print(f"{key}: {value}")
```

Количество непустых значений в каждом столбце:

ID	20
Имя	20
Возраст	20
Город	20
Баланс на счете	20
Кредитная история	20

dtype: int64

Частота встречаемости значений в 'Город':

Город	
Москва	1

Рисунок 18. Практическое задание 7

```
import pandas as pd

# DataFrame из 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', 'Дмитрий', 'Елена', 'Виктор', 'Алиса', 'Павел', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия',
            'Иван', 'Ольга', 'Алексей', 'Мария', 'Сергей', 'Дмитрий', 'Елена', 'Виктор', 'Алиса', 'Павел', 'Роман', 'Татьяна', 'Николай', 'Валерия', 'Григорий', 'Юлия'],
    'Возраст': [34.0, 27.0, None, 38.0, 29.0, 50.0, 31.0, 40.0, 33.0, 46.0, 37.0, 41.0, 35.0, 42.0, 49.0, None, 30.0, 35.0, None, None],
    'Город': ['Москва', 'Санкт-Петербург', 'Казань', None, 'Екатеринбург', 'Воронеж', None, 'Краснодар', 'Уфа', 'Омск', 'Пермь', 'Саратов', 'Самара', None, 'Барнаул', 'И',
            'Москва', 'Санкт-Петербург', 'Казань', None, 'Екатеринбург', 'Воронеж', None, 'Краснодар', 'Уфа', 'Омск', 'Пермь', 'Саратов', 'Самара', None, 'Барнаул', 'И'],
    'Баланс на счете': [120000.0, 0, 150000.0, 200000.0, 300000.0, 140000.0, 175000.0, 320000.0, 110000.0, None, 250000.0, 210000.0, 135000.0, 155000.0, None, 275000.0, 320000.0,
                    120000.0, 80000.0, 150000.0, 200000.0, 95000.0, 300000.0, 140000.0, 175000.0, 320000.0, 110000.0, 98000.0, 250000.0, 21000.0, 135000.0, 155000.0, 125000.0, 180000.0, 275000.0, 320000.0, 105000.0, 90000.0],
    'Кредитная история': ['Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Хорошая', 'Отличная', 'Средняя', 'Хорошая', 'Средняя', 'Хорошая', 'Средняя', 'Хорошая', 'Средняя', 'Хорошая', 'Отличная', 'Средняя', 'Плохая',
                           'Хорошая', 'Средняя', 'Плохая', 'Хорошая', 'Хорошая', 'Отличная', 'Средняя', 'Хорошая', 'Средняя', 'Хорошая', 'Отличная', 'Средняя', 'Плохая', 'Хорошая', 'Средняя', 'Плохая']
}

df = pd.DataFrame(data)

# Количество NaN
nan_counts = df.isna().sum()
print("Количество NaN в каждом столбце:\n", nan_counts)

# Количество заполненных значений
filled_counts = df.notna().sum()
print("\nКоличество заполненных значений в каждом столбце:\n", filled_counts)

# Строки без пропущенных значений
df_no_nan = df.dropna()
print("\nDataFrame без пропущенных значений:\n", df_no_nan)
```

Количество NaN в каждом столбце:

ID	0
Имя	2
Возраст	4
Город	5
Баланс на счете	4
Кредитная история	2

dtype: int64

Количество заполненных значений в каждом столбце:

ID	20
Имя	18
Возраст	16
Город	15
Баланс на счете	16
Кредитная история	18

dtype: int64

DataFrame без пропущенных значений:

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
0	1	Иван	34.0	Москва	120000.0	Хорошая
1	2	Ольга	27.0	Санкт-Петербург	0.0	Средняя
4	5	Сергей	29.0	Екатеринбург	300000.0	Хорошая
5	6	Дмитрий	50.0	Воронеж	140000.0	Отличная

Рисунок 19. Практическое задание 8

3. Использовать `DataFrame`, содержащий следующие колонки: фамилия и инициалы; номер группы; успеваемость (дополнительные колонки по предметам). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных и добавление строк в `DataFrame`; записи должны быть упорядочены по алфавиту; вывод на дисплей фамилий и номеров групп для всех студентов, имеющих хотя бы одну оценку 2; если таких студентов нет, вывести соответствующее сообщение.

Рисунок 20. Задание к 3 варианту

```
import pandas as pd

# Создаем начальный DataFrame с примерными данными
data = {
    "Фамилия и инициалы": ["Пугачев К.Д.", "Якушенко А.А.", "Мидов А.И."],
    "Номер группы": [101, 102, 103],
    "А1": [5, 2, 4],
    "ИКТ": [3, 2, 5],
    "ТЭРЦ": [4, 5, 3]
}

df = pd.DataFrame(data)

# Функция для добавления новой строки
def add_student():
    surname = input("Введите фамилию и инициалы: ")
    group = int(input("Введите номер группы: "))

    # Ввод оценок по предметам
    subjects = ["А1", "ИКТ", "ТЭРЦ"]
    grades = {}
    for subj in subjects:
        grade_input = input(f"Введите оценку по {subj} (или оставьте пустым, если нет): ")
        if grade_input.strip() == "":
            grades[subj] = None
        else:
            grades[subj] = int(grade_input)

    # Новая строка
    new_row = {
        "Фамилия и инициалы": surname,
        "Номер группы": group,
        **grades
    }

    # Добавление в DataFrame
    global df
    df = df.append(new_row, ignore_index=True)
    df = df.sort_values(by="Фамилия и инициалы").reset_index(drop=True)

# Основной цикл для добавления студентов
while True:
    command = input("Введите 'добавить' для добавления студента или 'выход' для завершения: ").strip().lower()
    if command == "добавить":
        add_student()
    elif command == "выход":
        break
    else:
        print("Некорректная команда. Попробуйте снова.")

# Поиск студентов с хотя бы одной оценкой 2
students_with_two = df[df[["А1", "ИКТ", "ТЭРЦ"]].apply(lambda row: 2 in row.values, axis=1)]

if not students_with_two.empty:
    for index, row in students_with_two.iterrows():
        print(f"Фамилия: {row['Фамилия и инициалы']}, Номер группы: {row['Номер группы']}")
else:
    print("Нет студентов с оценкой 2.")
```

Введите 'добавить' для добавления студента или 'выход' для завершения: 11 for history. Search history with c-1/c-4

Рисунок 21. Индивидуально задание 1

## Ответы на контрольные вопросы:

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

Можно передать словарь, где ключи — названия столбцов, а значения — списки данных. `pandas` автоматически создаст таблицу по этим данным.

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

Отличия:

- Список словарей: лучше, когда у данных разные наборы ключей, и некоторые строки могут иметь пропуски.
- Словарь списков: удобно, когда все столбцы и строки есть заранее, и структура однородна.

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

Можно передать массив NumPy, указав имена столбцов через columns.

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

```
df = pd.read_csv('файл.csv', delimiter=';')
```

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

```
df = pd.read_excel('файл.xlsx', sheet_name='Лист1')
```

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

- JSON: читается функцией `pd.read_json()`. Формат текстовый, удобен для обмена данными.
- Parquet: читается `pd.read_parquet()`. Это бинарный формат, более эффективный для больших данных и поддерживает схему.

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

```
print(df.dtypes)
```

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

```
rows, cols = df.shape
```

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

- `.loc[]` использует метки индексов (имена строк),
- `.iloc[]` использует целочисленные позиции (индексы по порядку).

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

```
value = df.iloc[2, 1]
```

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

```
row = df.loc['Мария']
```

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

- .at[] — быстрый доступ к одному элементу по метке (строка и столбец).
- .loc[] — для доступа к диапазонам или множественным значениям.

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

- .iat[] — для получения одного элемента по позиций, быстрее при одиночных обращениях.
- .iloc[] — для срезов и более сложных операций.

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

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

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

```
df_filtered = df[df['Возраст'].between(25, 35)]
```

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

- .query() принимает строку с условием, что позволяет писать компактные выражения.
- .loc[] использует Python-выражения, обращение по именам столбцов.

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

```
min_age = 25
```

```
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() — возвращает True для пропущенных значений.
- .notna() — возвращает True для непустых значений.

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

```
df_clean = 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[] ?**

```
new_index = 'Новая'
```

```
df.loc[new_index] = ['Иван', 28, 'Москва', 'Неизвестно']
```

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

```
df = df.drop('Возраст', axis=1)
```

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

```
df_clean = df.dropna()
```

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

```
df_clean = 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() — считает количество каждого уникального значения в столбце.

- .nunique() — возвращает число уникальных значений.

**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() показывает таблицу в более читаемом виде с форматированием, а print() выводит в виде текста, что менее удобно.

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

```
import pandas as pd
```

```
pd.set_option('display.max_rows', 100)
```

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