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 13

Выполнил:
Милькевич Александр Александрович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи, направленность (профиль) «Интеллектуальная обработка данных в инфокоммуникационных системах и сетях», 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: изучение структуры Serias и базовых операций

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

Ход выполнения работы:

Ссылка на гит хаб: https://github.com/lolndo/AI_lab5.git

1. Создание Series из списка.

пункт 1

```
data = {
    "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 = pd.DataFrame(data)
display(df)
```

	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 1. Выполнение задания 1

пункт 2

```
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)
display(df)
```

	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 2. пункт 2 задания 1

пункт 3 ¶

```
data_full = {
    "ID": list(range(1, 21)),
    "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_full = pd.DataFrame(data_full)
data_array = np.array(df_full.values)
data_array[:, 2] = np.random.randint(20, 61, size=len(df_full))
df_numpy = pd.DataFrame(data_array, columns=df_full.columns)
display(df_numpy)
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
0	1	Иван	60	Инженер	IT	60000	2
1	2	Ольга	37	Аналитик	Маркетинг	75000	5
2	3	Алексей	59	Менеджер	Продажи	90000	15
3	4	Мария	58	Программист	IT	80000	7
4	5	Сергей	58	Специалист	HR	50000	3
5	6	Анна	47	Разработчик	IT	85000	6
6	7	Дмитрий	49	HR	HR	48000	12
7	8	Елена	33	Маркетолог	Маркетинг	70000	4
8	9	Виктор	43	Юрист	Юридический	95000	10
9	10	Алиса	26	Дизайнер	Дизайн	62000	5
10	11	Павел	26	Администратор	Администрация	55000	7
11	12	Светлана	56	Тестировщик	Тестирование	67000	2
12	13	Роман	41	Финансист	Финансы	105000	20
13	14	Татьяна	53	Редактор	Редакция	72000	9
14	15	Николай	58	Логист	Логистика	75000	11
15	16	Валерия	49	SEO-специалист	SEO	64000	3
16	17	Григорий	35	Бухгалтер	Бухгалтерия	110000	25
17	18	Юлия	27	Директор	Финансы	150000	20
18	19	Степан	37	Экономист	Экономика	98000	14
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8

Рисунок 3. пункт 3 задания 1

пункт 4

```
df_full.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20 entries, 0 to 19
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           20 non-null     int64
 1   Имя          20 non-null     object
 2   Возраст      20 non-null     int64
 3   Должность    20 non-null     object
 4   Отдел        20 non-null     object
 5   Зарплата     20 non-null     int64
 6   Стаж работы  20 non-null     int64
dtypes: int64(4), object(3)
memory usage: 1.2+ KB
```

Рисунок 4. пункт 4 задания 1

2. Получение элемента Series.

```
df_full.to_csv("table1.csv", index=False)
df_csv = pd.read_csv("table1.csv")
```

```
data_full_2 = {
    "ID": list(range(1, 21)),
    "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_full_2)
file_name = 'Клиенты.xlsx'
df.to_excel(file_name, index=False, sheet_name='Клиенты')
```

```
json_file = 'table_1.json'
df_full.to_json(json_file, orient='records', force_ascii=False)
df_from_json = pd.read_json(json_file)
df_from_json.head()
```

	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 5. Выполнение задания 2

3. Фильтрация данных с помощью логической индексации.

```
df_1 = pd.DataFrame(data_full)
print(df_1.loc[df_1["ID"] == 5])

print("\nячейки:")
print(df_1.loc[2]["Возраст"])

print(df_1.at[2, "Отдел"])

print(df_1.iat[3,5])
```

```
   ID     Имя  Возраст   Должность Отдел  Зарплата  Стаж работы
4   5  Сергей       28  Специалист    HR     50000            3

ячейки:
40
Продажи
80000
```

Рисунок 3. Выполнение задания 3

4. Просмотр первых и последних элементов.

```
conditions = [
    (df_1['Зарплата'] < 60000),
    (df_1['Зарплата'] >= 60000) & (df_1['Зарплата'] < 100000),
    (df_1['Зарплата'] >= 100000)
]
categories = ["Низкая", "Средняя", "Высокая"]
df_1["Категория зарплаты"] = np.select(conditions, categories)
display(df_1)
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы	Категория зарплаты
0	1	Иван	25	Инженер	IT	60000	2	Средняя
1	2	Ольга	30	Аналитик	Маркетинг	75000	5	Средняя
2	3	Алексей	40	Менеджер	Продажи	90000	15	Средняя
3	4	Мария	35	Программист	IT	80000	7	Средняя
4	5	Сергей	28	Специалист	HR	50000	3	Низкая
5	6	Анна	32	Разработчик	IT	85000	6	Средняя
6	7	Дмитрий	45	HR	HR	48000	12	Низкая
7	8	Елена	29	Маркетолог	Маркетинг	70000	4	Средняя
8	9	Виктор	31	Юрист	Юридический	95000	10	Средняя
9	10	Алиса	27	Дизайнер	Дизайн	62000	5	Средняя
10	11	Павел	33	Администратор	Администрация	55000	7	Низкая
11	12	Светлана	26	Тестировщик	Тестирование	67000	2	Средняя
12	13	Роман	42	Финансист	Финансы	105000	20	Высокая
13	14	Татьяна	37	Редактор	Редакция	72000	9	Средняя
14	15	Николай	39	Логист	Логистика	75000	11	Средняя
15	16	Валерия	24	SEO-специалист	SEO	64000	3	Средняя
16	17	Григорий	50	Бухгалтер	Бухгалтерия	110000	25	Высокая
17	18	Юлия	45	Директор	Финансы	150000	20	Высокая
18	19	Степан	41	Экономист	Экономика	98000	14	Средняя
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8	Средняя

Рисунок 6. Выполнение задания 4

```
df_1.loc[20] = [21,"Антон", 32, "Разработчик", "IT", 85000, 6,"Средняя"]
display(df_1.tail())
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы	Категория зарплаты
16	17	Григорий	50	Бухгалтер	Бухгалтерия	110000	25	Высокая
17	18	Юлия	45	Директор	Финансы	150000	20	Высокая
18	19	Степан	41	Экономист	Экономика	98000	14	Средняя
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8	Средняя
20	21	Антон	32	Разработчик	IT	85000	6	Средняя

Рисунок 7. пункт 2 задания 4

```
new_data ={
    "ID": [22, 23],
    "Имя": ["Антон", "Марина"],
    "Возраст": [32, 27],
    "Должность": ["Разработчик", "Менеджер"],
    "Отдел": ["IT", "Продажи"],
    "Зарплата": [85000, 75000],
    "Стаж работы": [6, 5],
    "Категория зарплаты":["Средняя", "Средняя"]
}
new_df = pd.DataFrame(new_data)
df_1 = pd.concat([df_1, new_df], ignore_index=True)
display(df_1.tail())
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы	Категория зарплаты
18	19	Степан	41	Экономист	Экономика	98000	14	Средняя
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8	Средняя
20	21	Антон	32	Разработчик	IT	85000	6	Средняя
21	22	Антон	32	Разработчик	IT	85000	6	Средняя
22	23	Марина	27	Менеджер	Продажи	75000	5	Средняя

Рисунок 8. пункт 3 задания 4

5. Просмотр первых и последних элементов.

```
df_1=df_1.drop(columns=['Категория зарплаты'])
display(df_1.head())
```

	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

```
df_1_1 = df_1
df_1_1 = df_1_1[df_1_1['ID'] != 10]
display(df_1_1)
```

Рисунок 9. задание 5

```
df_1_1 = df_1
df_1_1 = df_1_1[df_1_1['ID'] != 10]
display(df_1_1)
```

	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
5	6	Анна	32	Разработчик	IT	85000	6
6	7	Дмитрий	45	HR	HR	48000	12
7	8	Елена	29	Маркетолог	Маркетинг	70000	4
8	9	Виктор	31	Юрист	Юридический	95000	10
10	11	Павел	33	Администратор	Администрация	55000	7
11	12	Светлана	26	Тестировщик	Тестирование	67000	2
12	13	Роман	42	Финансист	Финансы	105000	20
13	14	Татьяна	37	Редактор	Редакция	72000	9
14	15	Николай	39	Логист	Логистика	75000	11
15	16	Валерия	24	SEO-специалист	SEO	64000	3
16	17	Григорий	50	Бухгалтер	Бухгалтерия	110000	25
17	18	Юлия	45	Директор	Финансы	150000	20
18	19	Степан	41	Экономист	Экономика	98000	14
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8
20	21	Антон	32	Разработчик	IT	85000	6
21	22	Антон	32	Разработчик	IT	85000	6
22	23	Марина	27	Менеджер	Продажи	75000	5

Рисунок 10. пункт 2 задания 5

```
df_1_1 = df_1_1[df_1_1['Стаж работы'] >= 3]
display(df_1_1)
```

	ID	Имя	Возраст	Должность	Отдел	Зарплата	Стаж работы
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
5	6	Анна	32	Разработчик	IT	85000	6
6	7	Дмитрий	45	HR	HR	48000	12
7	8	Елена	29	Маркетолог	Маркетинг	70000	4
8	9	Виктор	31	Юрист	Юридический	95000	10
10	11	Павел	33	Администратор	Администрация	55000	7
12	13	Роман	42	Финансист	Финансы	105000	20
13	14	Татьяна	37	Редактор	Редакция	72000	9
14	15	Николай	39	Логист	Логистика	75000	11
15	16	Валерия	24	SEO-специалист	SEO	64000	3
16	17	Григорий	50	Бухгалтер	Бухгалтерия	110000	25
17	18	Юлия	45	Директор	Финансы	150000	20
18	19	Степан	41	Экономист	Экономика	98000	14
19	20	Василиса	38	Проект-менеджер	Продажи	88000	8
20	21	Антон	32	Разработчик	IT	85000	6
21	22	Антон	32	Разработчик	IT	85000	6
22	23	Марина	27	Менеджер	Продажи	75000	5

Рисунок 11. пункт 3 задания 5

```
df_1_1 = df_1_1[['Имя', 'Должность', 'Зарплата']]
display(df_1_1)
```

	Имя	Должность	Зарплата
1	Ольга	Аналитик	75000
2	Алексей	Менеджер	90000
3	Мария	Программист	80000
4	Сергей	Специалист	50000
5	Анна	Разработчик	85000
6	Дмитрий	HR	48000
7	Елена	Маркетолог	70000
8	Виктор	Юрист	95000
10	Павел	Администратор	55000
12	Роман	Финансист	105000
13	Татьяна	Редактор	72000
14	Николай	Логист	75000
15	Валерия	SEO-специалист	64000
16	Григорий	Бухгалтер	110000
17	Юлия	Директор	150000
18	Степан	Экономист	98000
19	Василиса	Проект-менеджер	88000
20	Антон	Разработчик	85000
21	Антон	Разработчик	85000
22	Марина	Менеджер	75000

Рисунок 12. пункт 4 задания 5

6. Проверка пропущенных значений.

```
df_2=pd.DataFrame(data_full_2)
filters = ["Москва","Санкт-Петербург" ]
filtered_df = df[df['Город'].isin(filters)]
display(filtered_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

```
filtered_df=df_2[df_2['Баланс на счете'].between(100000, 250000)]
display(filtered_df)
```

	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	Хорошая
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
18	19	Степан	30	Хабаровск	105000	Средняя

```
filtered_df_2 =df.query('`Кредитная история` == "Хорошая" and `Баланс на счете` > 150000')
display(filtered_df_2)
```

	ID	Имя	Возраст	Город	Баланс на счете	Кредитная история
3	4	Мария	38	Новосибирск	200000	Хорошая
7	8	Елена	40	Краснодар	175000	Хорошая
10	11	Павел	46	Омск	250000	Хорошая
13	14	Татьяна	25	Саратов	155000	Хорошая
17	18	Юлия	50	Иркутск	320000	Хорошая

Рисунок 13. Выполнение задания 6

7. Заполнение пропущенных значений.

```
count_per_column = df_2.count()
print(count_per_column)
```

```
ID                  20
Имя                 20
Возраст             20
Город               20
Баланс на счете     20
Кредитная история   20
dtype: int64
```

```
count_cities = df["Город"].value_counts()
print(count_cities)
```

```
Город
Москва             1
Санкт-Петербург    1
Хабаровск          1
Иркутск            1
Барнаул            1
Волгоград          1
Самара             1
Саратов            1
Тюмень             1
Пермь              1
Омск               1
Уфа                1
Ростов-на-Дону     1
Краснодар          1
Челябинск          1
Воронеж            1
Екатеринбург       1
Новосибирск        1
Казань             1
Томск              1
Name: count, dtype: int64
```

```
names = ["Город" , "Возраст" , "Баланс на счете"]
unique = df[names].nunique()
print(unique)
```

```
Город             20
Возраст           19
Баланс на счете   20
dtype: int64
```

Рисунок 14. Выполнение задания 7

8. Арифметические операции с series.

```
data_3 = {
    "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", np.nan, "Дмитрий", "Елена", "Виктор", "Алиса",
            "Павел", np.nan, "Роман", "Татьяна", "Николай", "Валерия", "Григорий", "Юлия", "Степан", "Василиса"],
    "Возраст": [34, 27, np.nan, 38, 29, 50, 31, 40, np.nan, 33, 46, 37, 41, 25, np.nan, 42, 49, np.nan, 30, 35],
    "Город": ["Москва", "Санкт-Петербург", "Казань", np.nan, "Екатеринбург", "Воронеж", np.nan, "Краснодар",
              "Ростов-на-Дону", "Уфа", "Омск", "Пермь", np.nan, "Саратов", "Самара", np.nan, "Барнаул",
              "Иркутск", "Хабаровск", "Томск"],
    "Баланс на счете": [120000, np.nan, 150000, 200000, np.nan, 300000, 140000, 175000, 110000, np.nan,
                        250000, 210000, 135000, 155000, 125000, np.nan, 275000, 320000, 105000, 90000],
    "Кредитная история": ["Хорошая", "Средняя", "Плохая", "Хорошая", np.nan, "Отличная", "Средняя",
                          "Хорошая", np.nan, "Средняя", "Хорошая", "Отличная", "Средняя", np.nan,
                          "Средняя", "Плохая", "Отличная", "Хорошая", "Средняя", "Плохая"]
}

df_3 = pd.DataFrame(data)
```

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

```
Количество NaN в каждом столбце:
 ID                   0
Имя                   0
Возраст               0
Город                 0
Баланс на счете       0
Кредитная история     0
dtype: int64
```

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

```
Количество заполненных значений в каждом столбце:
 ID                   20
Имя                   20
Возраст               20
Город                 20
Баланс на счете       20
Кредитная история     20
dtype: int64
```

Рисунок 15. Выполнение задания 8

```
df_cleaned = df.dropna()
print("\nDataFrame без пропущенных значений:")
display(df_cleaned)
```

```
DataFrame без пропущенных значений:
```

	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	Хорошая
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	Василиса	35	Томск	90000	Плохая

Рисунок 16.2. Выполнение задания 8

9. Использовать DataFrame , содержащий следующие колонки: фамилия, имя; номер телефона; дата рождения (список из трех чисел). Написать программу, выполняющую следующие действия: ввод с клавиатуры данных и добавление строк в DataFrame ; записи должны быть упорядочены по трем первым цифрам номера телефона; вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, выдать на дисплей соответствующее сообщение.

```
import pandas as pd
from IPython.display import display
```

```
!python -m venv .venv•••
```

```
df = pd.DataFrame(columns=["Last Name", "First Name", "Phone Number", "Birth Date"])


def input_data():
    last_name = input("Фамилия: ").strip()
    first_name = input("Имя: ").strip()
    phone = input("номер телефона: ").strip()
    if not phone.isdigit() or len(phone) < 3:
        print(
            "Ошибка: номер телефона должен содержать минимум 3 цифры и состоять только из цифр."
        )
        return None

    print("Введите дату рождения (дд.мм.гг):")
    try:
        day = int(input("День (число): "))
        month = int(input("Месяц (число): "))
        year = int(input("Год (число): "))
    except ValueError:
        print("Ошибка: дата рождения должна состоять из чисел.")
        return None

    birth_date = [day, month, year]
    return {
        "Last Name": last_name,
        "First Name": first_name,
        "Phone Number": phone,
        "Birth Date": birth_date,
    }


def add_to_df(data, df):

    new_row = pd.DataFrame([data])
    df = pd.concat([df, new_row], ignore_index=True)

    df["Sort Key"] = df["Phone Number"].str[:3]
    df = df.sort_values(by="Sort Key").reset_index(drop=True)
    df = df.drop(columns=["Sort Key"])

    return df


def search_by_last_name(last_name, df):

    results = df[df["Last Name"].str.lower() == last_name.lower()]
    if results.empty:
        print(f"Person with last name '{last_name}' not found.")
    else:
        print(f"Records found for last name '{last_name}':")
        display(results)
```

Рисунок 17.1. Листинг индивидуального задания

```python
def search_by_last_name(last_name, df):

    results = df[df["Last Name"].str.lower() == last_name.lower()]
    if results.empty:
        print(f"Person with last name '{last_name}' not found.")
    else:
        print(f"Records found for last name '{last_name}':")
        display(results)


if __name__ == "__main__":
    while True:
        print("1 - Добавить запись")
        print("2 - Поиск по фамилии")
        print("0 - Закрыть справочник")
        if df.empty:
            print("Список пока пуст :(")
        else:
            display(df)

        choice = input("Выберите функцию: ").strip()

        if choice == "1":
            data = input_data()
            if data:
                df = add_to_df(data, df)
                print("Добавить запись")

        elif choice == "2":
            last = input("Напишите Фамилию: ").strip()
            search_by_last_name(last, df)

        elif choice == "3":
            if df.empty:
                print("Список пуст :(")
            else:
                display(df)

        elif choice == "0":
            print("Закрытие")
            break

        else:
            print("Попробуйте еще раз")
```

```
1 - Добавить запись
2 - Поиск по фамилии
0 - Закрыть справочник
Список пока пуст :(
Выберите функцию:  0
Закрытие
```

```
!pip install nbqa black --quiet
!nbqa black indi.ipynb
```

```
All done! \u2728 \U0001f370 \u2728

1 file left unchanged.
```

Рисунок 17.2. Листинг индивидуального задания

Вывод: в ходе лабораторной работы были изучены структуры DataFrame, базовые операции по работе с ними, а также использовали утилиты для проверки кода.

Контрольные вопросы:

1. Как создать pandas.DataFrame из словаря списков

Используйте `pd.DataFrame(data_dict)`, где `data_dict` - это словарь, где ключи - имена столбцов, а значения - списки.

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

Из списка словарей DataFrame строится по ключам словарей, а из словаря списков - по ключам словаря как названиям столбцов.

3. Как создать pandas.DataFrame из массива NumPy

Используйте `pd.DataFrame(numpy_array)`, где `numpy_array` - это массив NumPy.

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

Используйте `pd.read_csv('file.csv', delimiter=';')`.

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

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

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

JSON - это текстовый формат, удобный для хранения и обмена данными. Parquet - бинарный формат, оптимизированный для хранения и обработки больших объемов данных.

7. Как проверить типы данных в DataFrame после загрузки

Используйте метод `.dtypes`.

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

Используйте атрибут `.shape`, который возвращает кортеж (число строк, число столбцов).

9. В чем разница между .loc[] и .iloc[]

`.loc[]` используется для индексации по названиям индексов и столбцов, а `.iloc[]` - по позициям (целым числам).

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

Используйте `df.iloc[2, 1]`.

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

Используйте `df.loc['Мария']`.

12. Чем .at[] отличается от .loc[]

`.at[]` используется для получения одного значения по индексу и имени столбца и быстрее, чем `.loc[]`.

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

`.iat[]` быстрее при доступе к одному элементу по позиции, так как не поддерживает метаданные.

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

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

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

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

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

`.query()` использует строку запроса и позволяет писать более читаемые выражения, в то время как `.loc[]` использует булевый индекс.

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

Используйте `@` перед именем переменной, например: `df.query('Возраст > @age')`.

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

Используйте `df.isnull().sum()`.

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

`.isna()` возвращает True для пропущенных значений, а `.notna()` - для непустых значений.

20. Как вывести только строки, где нет пропущенных значений

Используйте `df.dropna()`.

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

Используйте `df['Категория'] = 'Неизвестно'`.

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

Используйте `df.loc[new_index] = new_value`, где `new_index` - это индекс новой строки, а `new_value` - значения для этой строки.

23. Как удалить столбец "Возраст" из DataFrame

Используйте `df.drop(columns='Возраст', inplace=True)`.

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

Используйте `df.dropna()`.

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

Используйте `df.dropna(axis=1)`.

26. Как посчитать количество непустых значений в каждом столбце DataFrame

Используйте `df.count()`.

27. Чем .value_counts() отличается от .nunique()?

`.value_counts()` возвращает количество уникальных значений с частотой, а `.nunique()` - просто количество уникальных значений.

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

Используйте `df['Город'].value_counts()`.

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

`display()` выводит DataFrame с форматированием, поддерживает стили и является более удобным для визуализации.

30. Как изменить максимальное количество строк, отображаемых в DataFrame в Jupyter Notebook. Используйте pd.set_option('display.max_rows', n), где n - желаемое количество строк.