КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра анализа данных и технологий программирования

Р.А. БУРНАШЕВ

АНАЛИЗ ДАННЫХ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON: БИБЛИОТЕКА PANDAS.

Казань – 2022

УДК 025.4.03
ББК 32.971.353

Принято на заседании учебно-методической комиссии КАДиТП
Протокол № 1 от 5 сентября 2022 года

Рецензенты:
кандидат физико-математических наук,
доцент кафедры анализа данных и технологий программирования КФУ
А.И. Еникеев

Бурнашев Р.А.
Анализ данных на языке программирования Python: Библиотека Pandas / Р.А. Бурнашев. – Казань: Казан. ун-т, 2022. – 25 с.

Методическое пособие посвящено программным инструментам и методам, которые позволят начать проводить научные исследования с реальными данными с помощью библиотеки Pandas.
Библиотека Pandas, является одной из часто используемых программных модулей для анализа данных на языке программирования Python. Библиотека является быстрой и удобной в использовании.
Настоящее учебно-методическое пособие адресовано, в первую очередь, студентам таких специальностей, как «Прикладная информатика», «Бизнес-информатика», «Прикладная математика и информатика» и т.д., а также широкому кругу читателей, интересующихся направлением в области анализа данных.

СОДЕРЖАНИЕ

Введение

Python является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, функциональное программирование и др[1-2].

1 Начало работы с Jupyter Notebook: установка и запуск среды разработки, подключение модуля pandas

В теме рассматривается работа с Jupyter notebook - графической веб-оболочкой для IPython, которая расширяет идею консольного подхода к интерактивным вычислениям. Приводится пример запуска и работы, а также основные элементы интерфейса Jupyter notebook.

Запускаем программу Anaconda Navigator (Anaconda3)

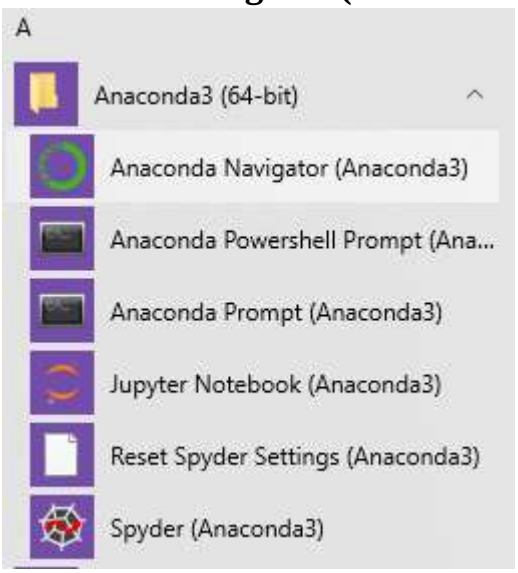


Рис. 1. Запуск программы

Jupyter Notebook входит в состав Anaconda.
Jupyter Notebook – это крайне удобный инструмент для создания красивых аналитических отчетов, так как он позволяет хранить вместе код, изображения, комментарии, формулы и графики.

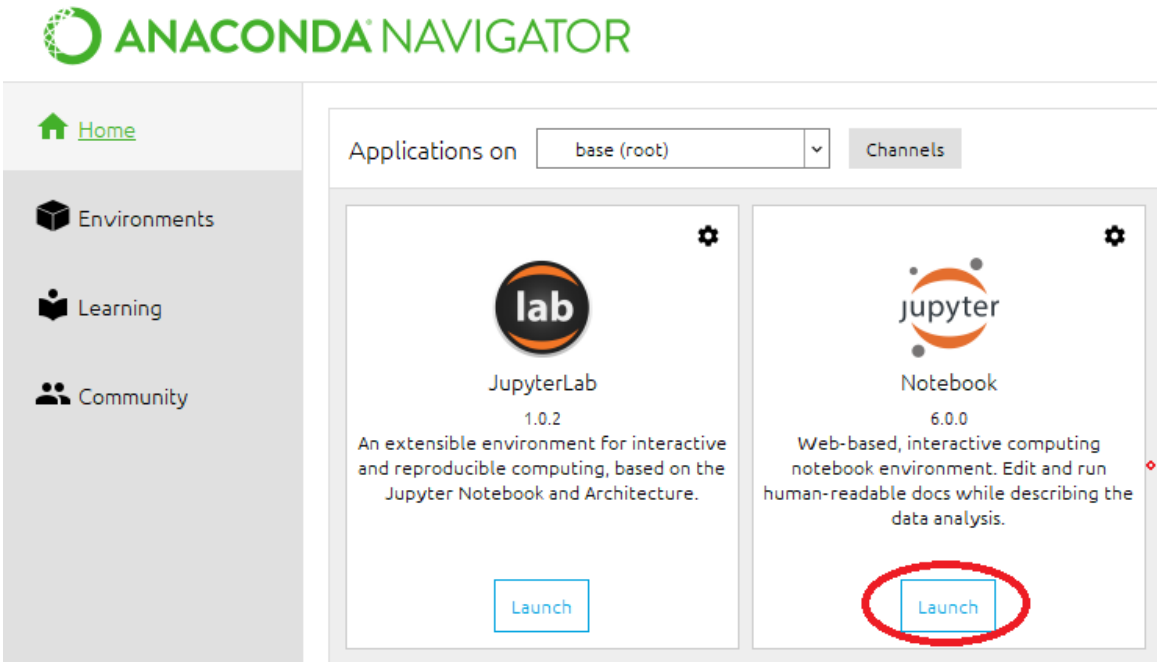


Рис. 2. Запуск среды Jupyter Notebook

Основные элементы интерфейса Jupyter notebook

Из элементов интерфейса можно выделить, панель меню:

Рис. 1. Основные элементы среды

панель инструментов: номера

Рис. 3. Панель инструментов

1 – сохранение;
2 – добавление новой строки кода;
3 – вырезать строку с выбранным кодом;
4 – копировать код;
5 – вставить;
6,7 – передвижение курсора по строкам;
8 – запуск кода программы;
9 – остановка процесса выполнения программы;
10,11 – рестарт строк команды.

и рабочее поле с ячейками:

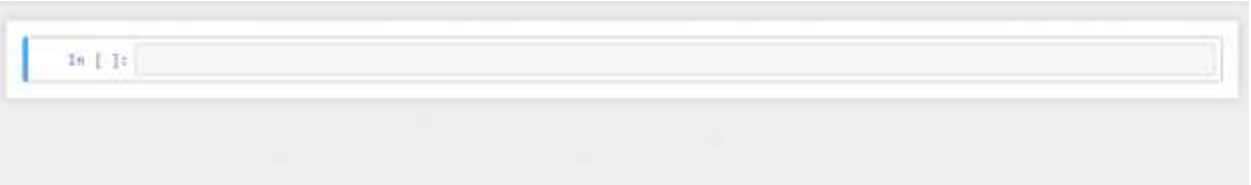


Рис. 4. Поле ввода программного кода

Python-файл может находиться в одном из двух режимов – это режим правки (Edit mode) и командный режим (Command mode). Текущий режим отображается на панели меню в правой части, в режиме правки появляется изображение карандаша, отсутствие этой иконки значит, что python-файл находится в командном режиме.

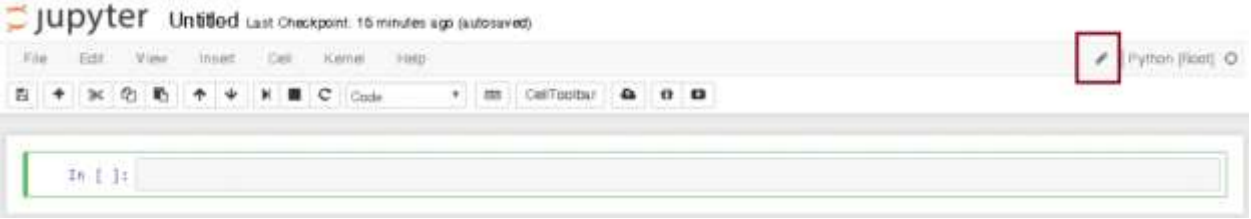


Рис. 5. Командный режим

Запуск и прерывание выполнения кода

Если ваша программа зависла, то можно прервать ее выполнение выбрав на панели меню пункт Kernel -> Interrupt.

Для добавления новой ячейки используйте Insert->Insert Cell Above и Insert->Insert Cell Below.

Для запуска ячейки используете команды из меню Cell, либо следующие сочетания клавиш:

Ctrl+Enter – выполнить содержимое ячейки.

Shift+Enter – выполнить содержимое ячейки и перейти на ячейку ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

Рассмотрим несколько примеров, выполнив которые, вы сразу поймете принцип работы с Jupyter notebook. Для создания папки нажмите на New в правой части экрана и выберите в выпадающем списке Folder.

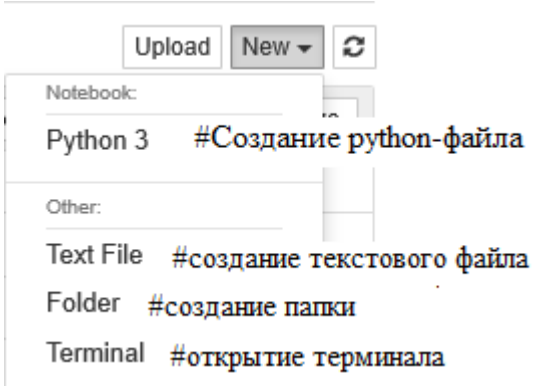


Рис. 6. Создание и запуск программного файла

Создайте новый python-файл, воспользовавшись той же кнопкой New и выбрав “Python 3”.

Рис. 7. Выбор версии ЯП

В результате будет создана строка:

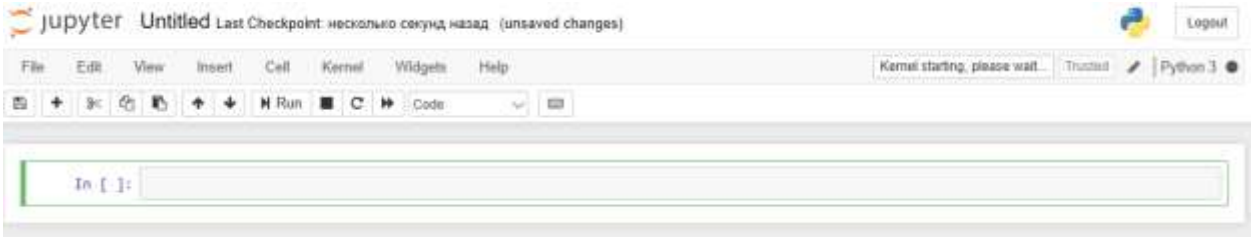
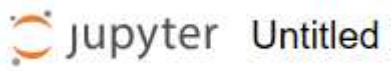


Рис. 8. Создание новой командной строки



Untitled – название вашего файла/папки. Его можно изменить, кликнув по нему и выполнив команду “Изменить” (Rename)

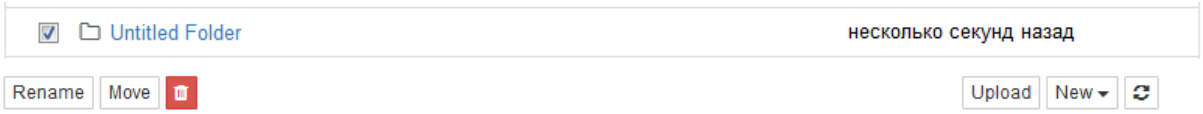
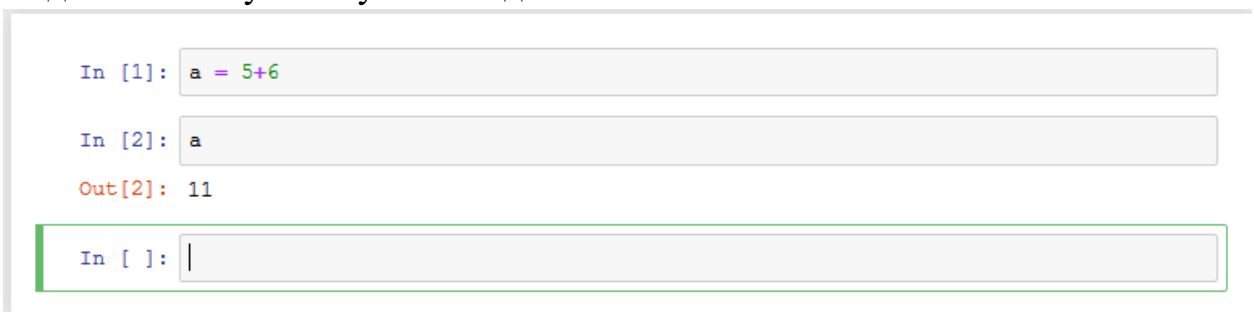


Рис. 9. Наименование файла

Код на языке Python нужно вводить в ячейки.



Для начала решим простую арифметическую задачу: выставите свойство “Code”, введите самостоятельно в ячейке “2 + 3” без кавычек и нажмите Ctrl+Enter или Shift+Enter, в первом случае введенный вами код будет выполнен интерпретатором Python, во втором – будет выполнен код и создана новая ячейка, которая расположится уровнем ниже так, как показано на рисунке.

```
In [1]: 2+3
Out[1]: 5
In [ ]:
```

Рассмотрим пример, ра**боты со списками, а также возвращает список, который состоит из элементов, общих для этих двух списков.**

```
In [16]: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

         b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

In [17]: result = []
         for elem in a:
             if elem in b:
                 result.append(elem)

In [18]: print(result)

         [1, 1, 2, 3, 5, 8, 13]
```

Задания

1. Запустить программу Jupyter Notebook
2. Создать новую папку с названием “First Program”.
3. Создать в папке “First Program” новый python-файл().

Рис. 10. Название программного файла

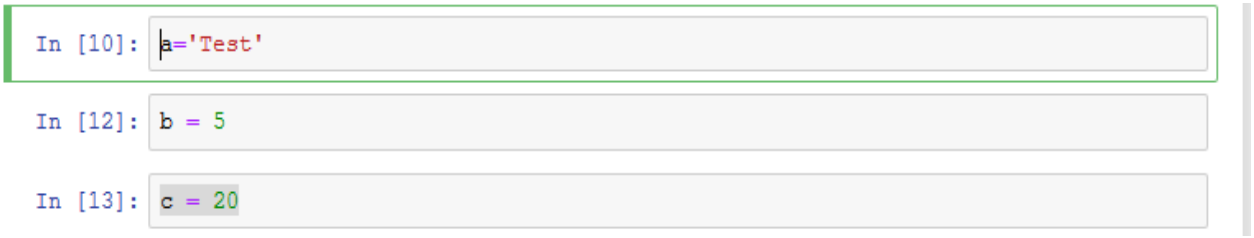
Измените название файла (“Program1”)

4. В этом файле создать две переменные и присвоить им целочисленные значения. Вычислить сумму, разность, произведение. Ответы вывести с помощью print().
5. **Добавьте новую строку кода** и создайте список. Например, создайте список a= [20,30,40,1,6,8,10]. **Запустите строку кода**. С помощью цикла for вывести на экран числа меньше **10.**
6. **Создайте новую строчку кода и создайте новую переменную**

```
In [10]: a='Test'

In [11]: a
Out[11]: 'Test'
```

С помощью инструмента «Вырезать» выполните удаление второй строчки. И вместо нее напишите b = 5 c= 20



Добавьте в новую строку кода следующий фрагмент программы

d=b+c

print(d)

Запустите программу и проверьте результат

7. Создайте два списка. Нужно вернуть список, который состоит из элементов, общих для этих двух списков.

```
spisok = ['Rustam','Petya']

spisok2 = ['Rustam', 'Nikita', 'Nastya']
```

8. Закройте полностью Jupyter Notebook

9. Заново запустите программу - найдите созданную ранее папку - First Program – Запустите файл Program1 – с помощью инструмента «Запуск» запустите программу только с примерами по работе со списками.

2 Введение в модуль pandas, подключение модуля pandas, обзор основных функций

В теме рассматриваются принципы работы с модулем pandas [3]. Приводится примеры установки модуля pandas, рассматривается структура данных

Количество модулей для языка Python огромно, что связано с популярностью языка. Часть модулей собрана в так называемую стандартную библиотеку. Стандартная она потому, что поставляется вместе с установочным пакетом. Однако существуют сторонние библиотеки. Они скачиваются и устанавливаются отдельно.

Для доступа к функционалу модуля, его надо импортировать в программу. После импорта интерпретатор "знает" о существовании дополнительных классов и функций и позволяет ими пользоваться.

В Python импорт осуществляется командой import. При этом существует несколько способов импорта.

Pandas это высокоуровневая Python библиотека для анализа данных.

В экосистеме Python, pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных. Если в своей работе вы занимаетесь анализом данных или машинным обучением и при этом используете язык Python, то вам необходимо знать и уметь работать с модулем pandas. Модуль pandas присутствует в стандартной поставке Anaconda. Если же его там нет, то его можно установить отдельно.

В случае отсутствия модуля, для его установки выполнить следующую команду:

Перейдите в пуск –> Anaconda3(64-bit) –> **Anaconda Prompt (Anaconda3)**

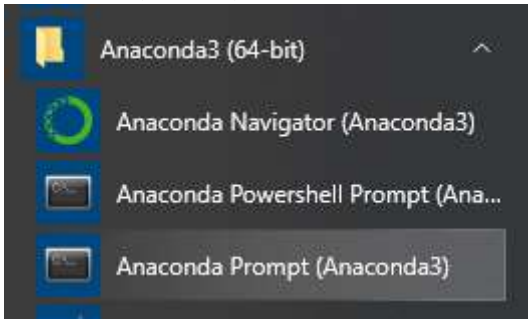


Рис. 11. Запуск командной строки для установки программных модулей

В появившемся окне введите команду **pip install pandas**

Рис. 12. Установка библиотеки pandas

Чтобы эффективно работать с pandas, необходимо освоить самые главные структуры данных библиотеки pandas.

Series

Структура/объект Series представляет из себя объект, похожий на одномерный массив (питоновский список, например), но отличительной его чертой является наличие ассоциированных меток, т.н. индексов, вдоль каждого элемента из списка. Такая особенность превращает его в ассоциативный массив или словарь в Python.

Пример использования Series:

```
import pandas as pd
num = pd.Series([10,53,9,48,31])
```

```
num
```

```
0    10
1    53
2     9
3    48
4    31
dtype: int64
```

где первый столбец – индексы, второй – элементы списка.

Доступ к элементам объекта Series возможны по их индексу:

```
import pandas as pd
num = pd.Series([10,53,9,48,31])
```

```
num[1]
```

```
53
```

Индексы можно задавать явно:

```
import pandas as pd
num = pd.Series([10,53,9,48,31], index=['a','b','c','d','e'])
```

```
num
```

```
a    10
b    53
c     9
d    48
e    31
dtype: int64
```

Имейте в виду, что список с индексами по длине должен совпадать с количеством элементов в Series.

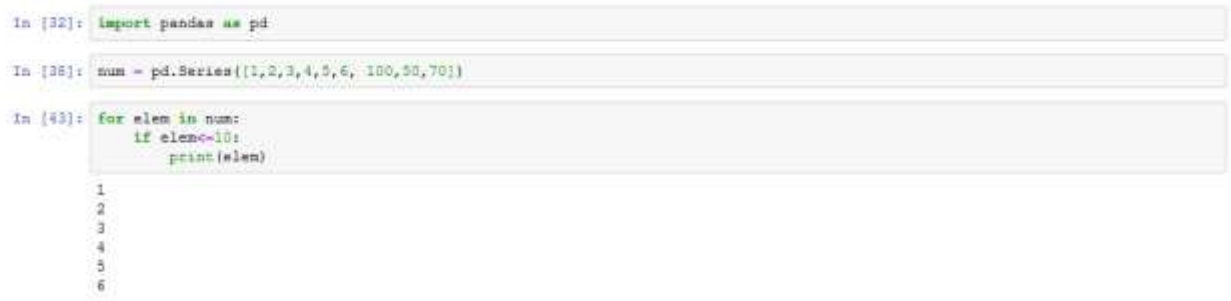
Доступ к строкам по индексу возможен несколькими способами:

- .loc - используется для доступа по строковой метке
- .iloc - используется для доступа по числовому значению (начиная от 0)

Например, выполним запрос по поиску индекса **(a) из списка**

df.loc[‘a’]

Пример, создадим новый список list1 и выведем на экран список элементов меньше 10



Пример, Создадим 2 списка Series и выведем на экран их сумму

Пример, вывести на экран список элементов схожих как первом pd.Series так и во втором pd.Series

```
In [59]: import pandas as pd

In [60]: a = pd.Series([1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89])

In [61]: b = pd.Series([1, 2, 3, 8, 13, 21, 34, 55, 89])

In [62]: result = []

         for elem in a:
             if elem in b:
                 result.append(elem)

In [63]: print(result)
         [1, 1, 2, 3, 5, 8]
```

Пример, с помощью метода **concat** объединим два Series

```
In [124]: import pandas as pd

In [125]: a = pd.Series([1, 1, 2, 3, 5, 8, 13, 21, 34])

In [126]: b = pd.Series([1, 2, 3, 8, 13, 21, 34, 55, 89])

In [127]: c = pd.concat([a,b])

In [128]: c
Out[128]: 0     1
          1     1
          2     2
          3     3
          4     5
          5     8
          6    13
          7    21
          8    34
          0     1
          1     2
          2     3
          3     8
          4    13
          5    21
          6    34
          7    55
          8    89
          dtype: int64
```

DataFrame

Объект DataFrame лучше всего представлять себе в виде обычной таблицы и это правильно, ведь DataFrame является табличной структурой данных. В любой таблице всегда присутствуют строки и столбцы. Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их непосредственными элементами.

```
import pandas as pd
df = pd.DataFrame({'company':['ОАО "Удмуртнефть"','ООО «НОВАТЭК-ЮРХАРОВНЕФТЕГАЗ»','ООО «РН-Юганскнефтегаз»'],
                   'year':['1967','2002','2005']})
```

```
df
```

	company	year
0	ОАО "Удмуртнефть"	1967
1	ООО «НОВАТЭК-ЮРХАРОВНЕФТЕГАЗ»	2002
2	ООО «РН-Юганскнефтегаз»	2005

Объект DataFrame имеет 2 индекса: по строкам и по столбцам. Если индекс по строкам явно не задан, то pandas задаёт целочисленный индекс RangeIndex от 0 до N-1, где N это количество строк в таблице.

Чтобы убедиться, что столбец в DataFrame это Series, извлекаем любой столбец:

```
df['company']
```

```
0                 ОАО "Удмуртнефть"
1    ООО «НОВАТЭК-ЮРХАРОВНЕФТЕГАЗ»
2          ООО «РН-Юганскнефтегаз»
Name: company, dtype: object
```

```
type(df['company'])
```

```
pandas.core.series.Series
```

Доступ к строкам по индексу возможен несколькими способами:

- .loc - используется для доступа по строковой метке
- .iloc - используется для доступа по числовому значению (начиная от 0)

Можно делать выборку по индексу и интересующим колонкам:

```
import pandas as pd
df = pd.DataFrame({'company':['ОАО "Удмуртнефть"','ООО «НОВАТЭК-ЮРХАРОВНЕФТЕГАЗ»','ООО «РН-Юганскнефтегаз»'],
                   'year':['1967','2002','2005']}, index=['3', '20', '20'])
```

```
df.loc['20']
```

	company	year
20	ООО «НОВАТЭК-ЮРХАРОВНЕФТЕГАЗ»	2002
20	ООО «РН-Юганскнефтегаз»	2005

Задания

1. Импортируйте в код модуль pandas (import pandas as pd)
2. Создайте новый список с помощью объекта Series. Элементы списка(не меньше 5): Нефтедобывающие страны, а индекс в таблице место страны во всемирной добыче нефти(1,2,3,4 и тд.)

С помощью метода loc вывести на экран страну, которая занимает первую строчку в рейтинге.

3. **Создайте новый список стран состоящий из 5 элементов(2 страны должны совпадать с первым списком)**, вывести на экран список элементов схожих как первом pd.Series так и во втором pd.Series
4. Создайте таблицу с помощью DataFrame.

Столбцы: Название компании, год, прибыль.

Индексам присвойте названия стран, которым принадлежат эти компании.

5. Выведите компании по индексу одной страны.

3. Основные объекты в pandas, простейшие операции над данными

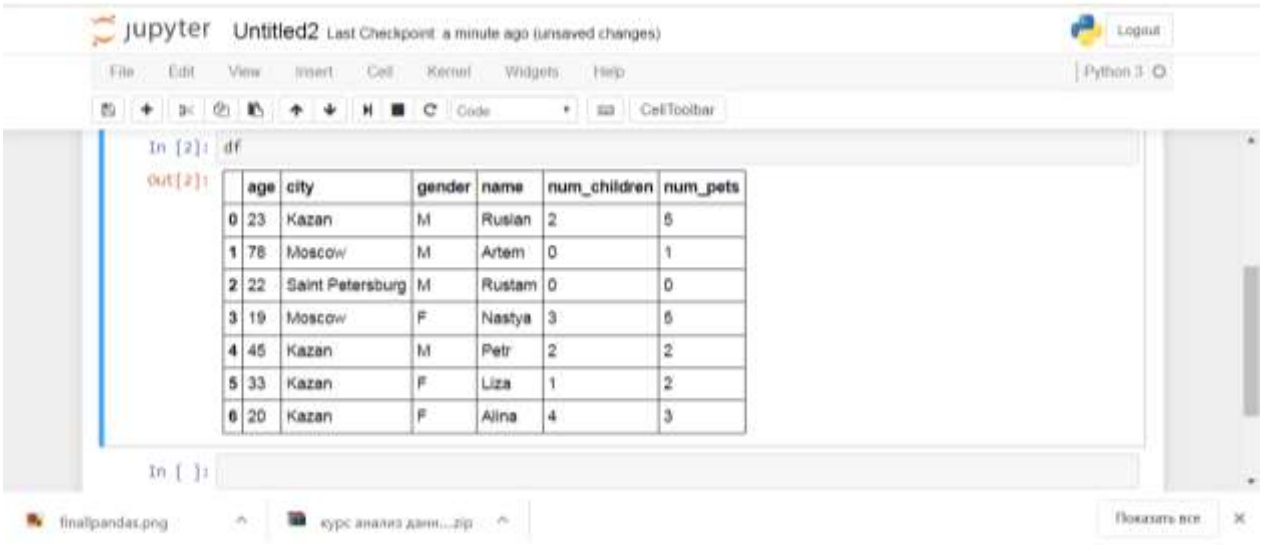
В теме рассматривается объект DataFrame. Приводятся примеры загрузки и чтения данных, а также создания DataFrame и выполнение операций над данными.

Объект DataFrame

В экосистеме Python, pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных. Одна из главных структур данных библиотеки pandas – это DataFrame.

Объект DataFrame является табличной структурой данных. В любой таблице всегда присутствуют строки и столбцы. Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их непосредственными элементами.

Pandas DataFrame состоит из трех основных компонентов: **данных**, **строк и столбцов**.



	age	city	gender	name	num_children	num_pets
0	23	Kazan	M	Ruslan	2	5
1	78	Moscow	M	Artem	0	1
2	22	Saint Petersburg	M	Rustam	0	0
3	19	Moscow	F	Nastya	3	5
4	45	Kazan	M	Petr	2	2
5	33	Kazan	F	Liza	1	2
6	20	Kazan	F	Alina	4	3

Рис. 13. Объект DataFrame

Создание DataFrame
Работа со строками и столбцами
Индексирование и выбор данных
Работа с отсутствующими данными
Перебор строк и столбцов

Пример, создания нового объекта DataFrame

#импорт модуля pandas

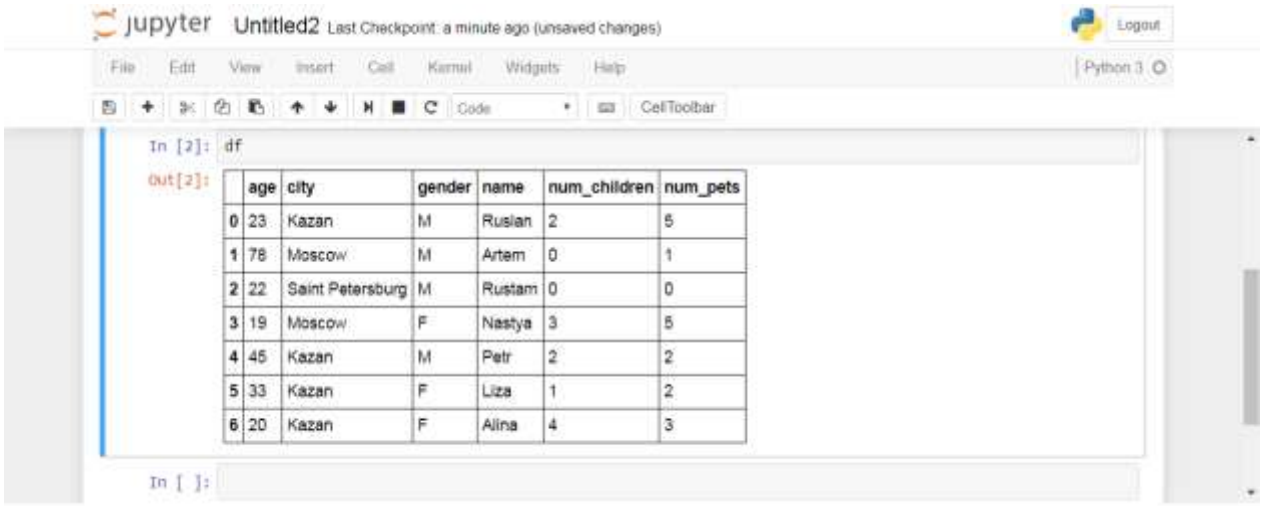
import pandas as pd

#создание объекта DataFrame (где - name,age,gender,city, num_children, num_pets название столбцов, а данные строк в [‘ …. .’])

 # строковые значения списка в кавычках, а числовые без кавычек

```
df = pd.DataFrame({
   'name':['Ruslan','Artem','Rustam','Nastya','Petr','Liza','Alina'],
   'age':[23,78,22,19,45,33,20],
   'gender':['M','M','M','F','M','F','F'],
   'city':['Kazan','Moscow','Saint Petersburg','Moscow','Kazan','Kazan','Kazan'],
   'num_children':[2,0,0,3,2,1,4],
   'num_pets':[5,1,0,5,2,2,3]
})
```

df

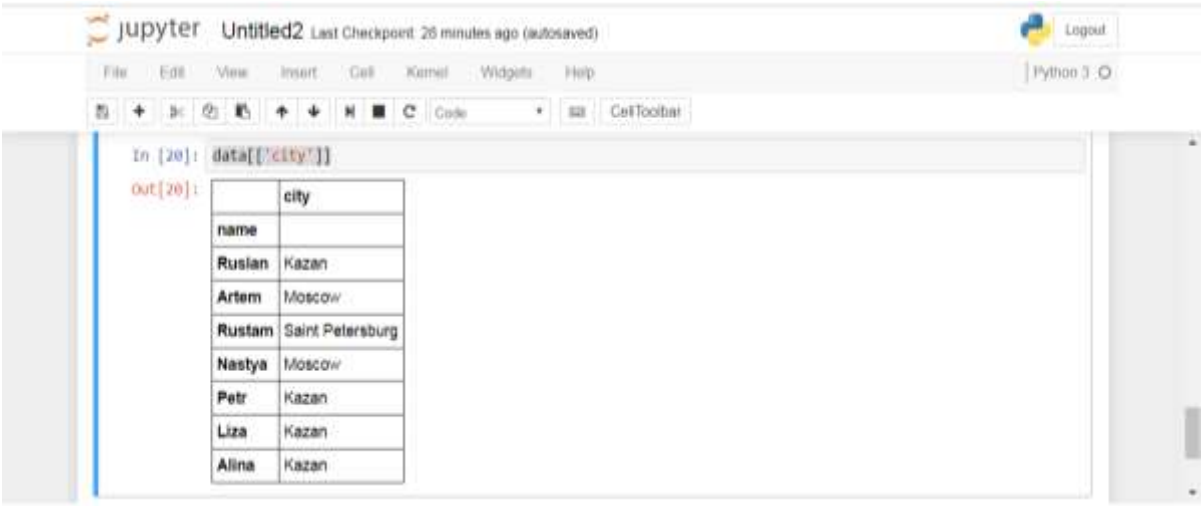


	age	city	gender	name	num_children	num_pets
0	23	Kazan	M	Ruslan	2	5
1	78	Moscow	M	Artem	0	1
2	22	Saint Petersburg	M	Rustam	0	0
3	19	Moscow	F	Nastya	3	5
4	45	Kazan	M	Petr	2	2
5	33	Kazan	F	Liza	1	2
6	20	Kazan	F	Alina	4	3

Рис. 14. Вывод данных на экран

Примеры по работе с DataFrame

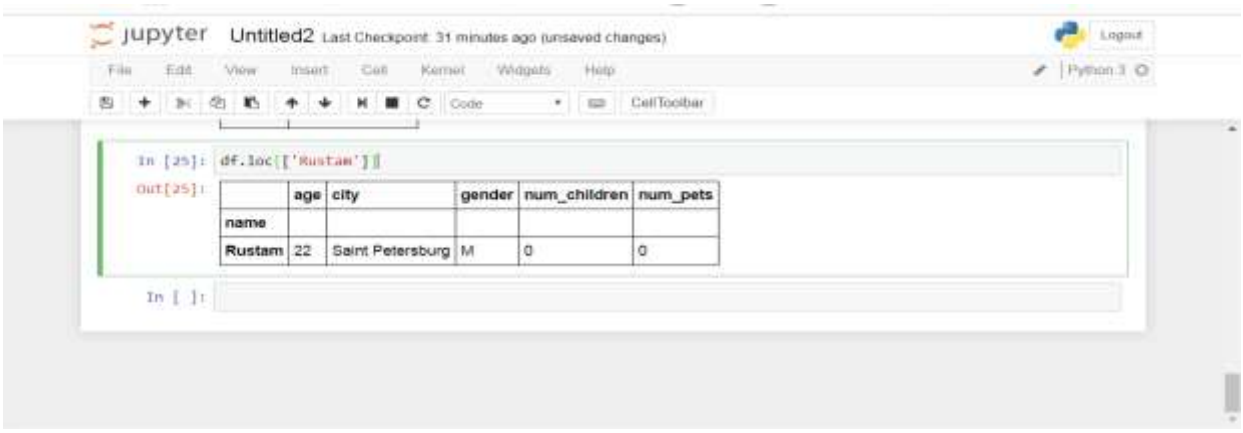
```
#запись данных
df.to_csv('my.csv')
 # чтение данных из файла и установка индекса (name)
data = pd.read_csv("my.csv", index_col ="name")
#удаление столбца
data.drop(['Unnamed: 0'],axis=1)
#вывод отдельного столбца (Например – города)
data[['city']]
```



	city
name	
Ruslan	Kazan
Artem	Moscow
Rustam	Saint Petersburg
Nastya	Moscow
Petr	Kazan
Liza	Kazan
Alina	Kazan

Рис. 15. Вывод столбцов

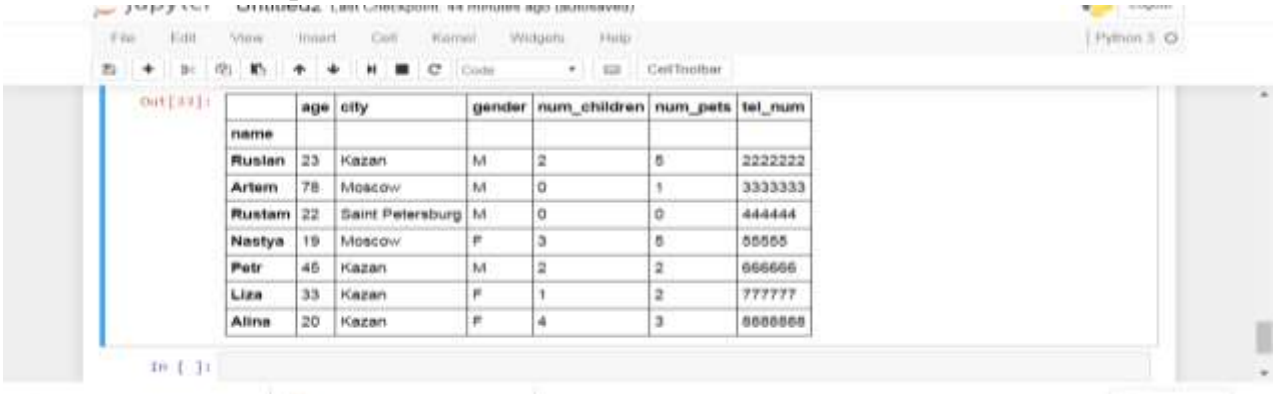
#вывод данных по индексу (столбец name)
df.loc[['Rustam']]



	age	city	gender	num_children	num_pets
name					
Rustam	22	Saint Petersburg	M	0	0

Рис. 16. Поиск по индексу

Пример, создадим новый столбец под название «tel_number»
#создадим список из телефонных номеров и добавим его в нашу таблицу
telephone = ['2222222', '3333333','444444','55555', '666666', '777777' , '8888888']
df['tel_num'] = telephone



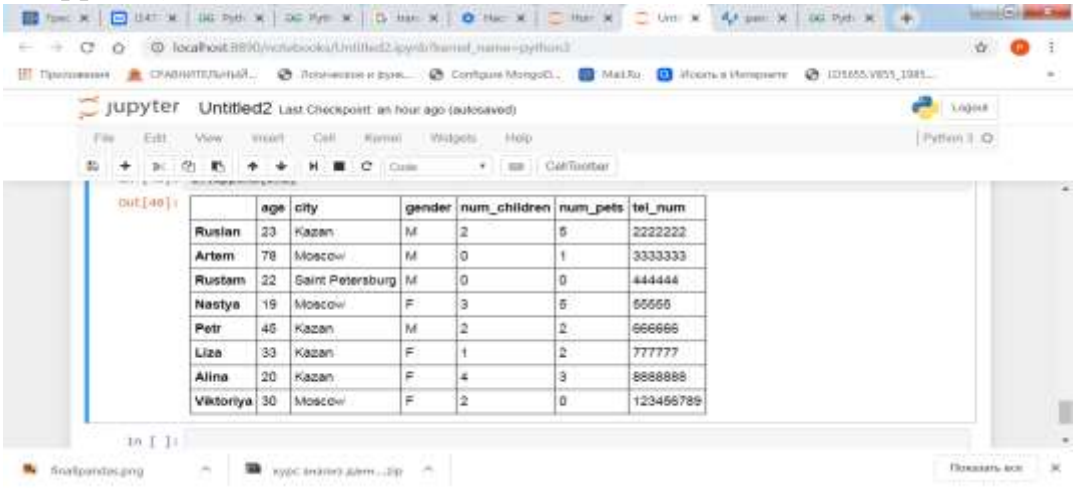
	age	city	gender	num_children	num_pets	tel_num
name						
Ruslan	23	Kazan	M	2	5	2222222
Artem	78	Moscow	M	0	1	3333333
Rustam	22	Saint Petersburg	M	0	0	444444
Nastya	19	Moscow	F	3	5	55555
Petr	45	Kazan	M	2	2	666666
Liza	33	Kazan	F	1	2	777777
Alina	20	Kazan	F	4	3	8888888

#Добавим новую строку в наш DataFrame
df2 = pd.DataFrame({

```
    'age':[30],
    'gender':['F'],
    'city':['Moscow'],
    'num_children':[2],
    'num_pets':[0],
    'tel_num':['123456789']
}, index = ['Viktoriya'])
```

#объединим первый df – DataFrame с вновь созданным df2

```
df.append(df2)
```



	age	city	gender	num_children	num_pets	tel_num
Ruslan	23	Kazan	M	2	5	2222222
Artem	78	Moscow	M	0	1	3333333
Rustam	22	Saint Petersburg	M	0	0	444444
Nastya	19	Moscow	F	3	5	55555
Petr	46	Kazan	M	2	2	666666
Liza	33	Kazan	F	1	2	777777
Alina	20	Kazan	F	4	3	8888888
Viktoriya	30	Moscow	F	2	0	123456789

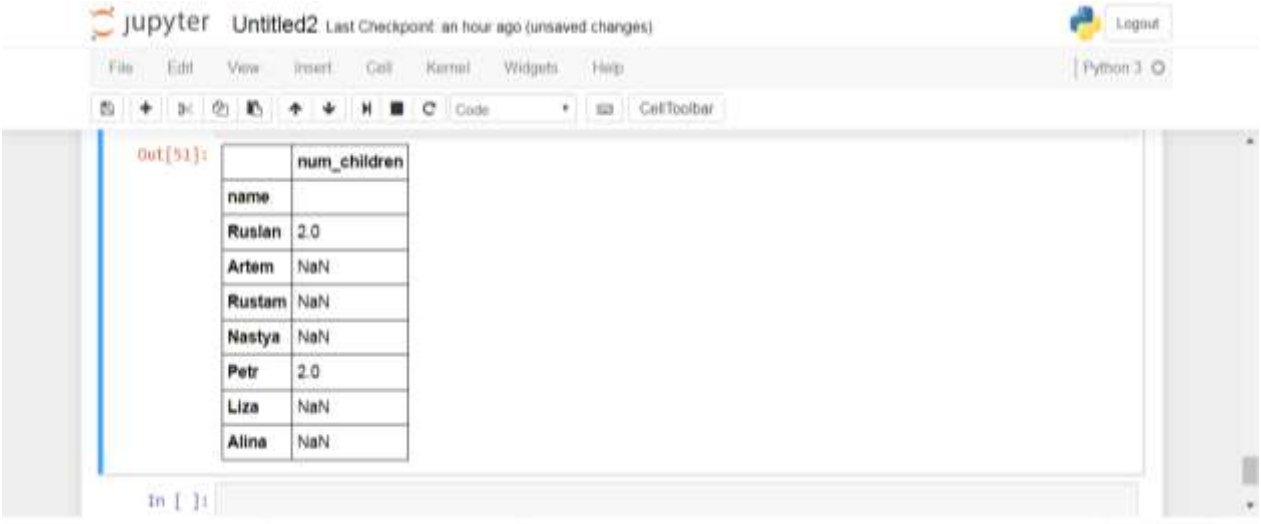
Рис. 17. Объединение двух объектов

 # Создадим новый столбец, который примет в себя сумму элементов двух столбцов

```
df['Summa'] = df['первый столбец'] + df['второй столбец']
```

#Выполним поиск данных по столбцу (Например у кого в семье 2 ребёнка)

```
df[['num_children']].where(df['num_children'] == 2)
```



	num_children
name	
Ruslan	2.0
Artem	NaN
Rustam	NaN
Nastya	NaN
Petr	2.0
Liza	NaN
Alina	NaN

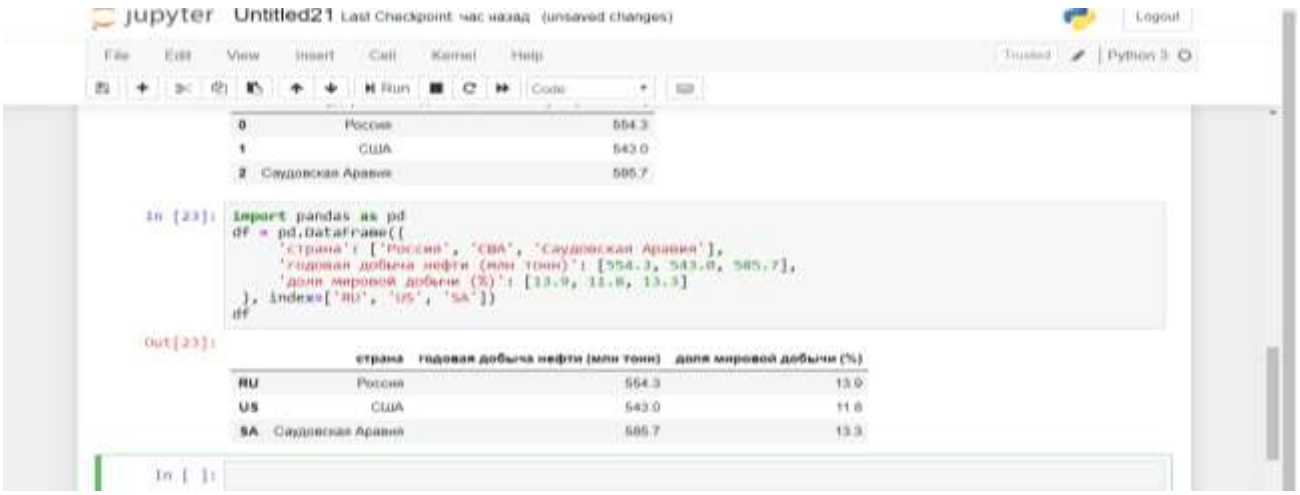
Рис. 18. Поиск данных

Чтение и запись данных

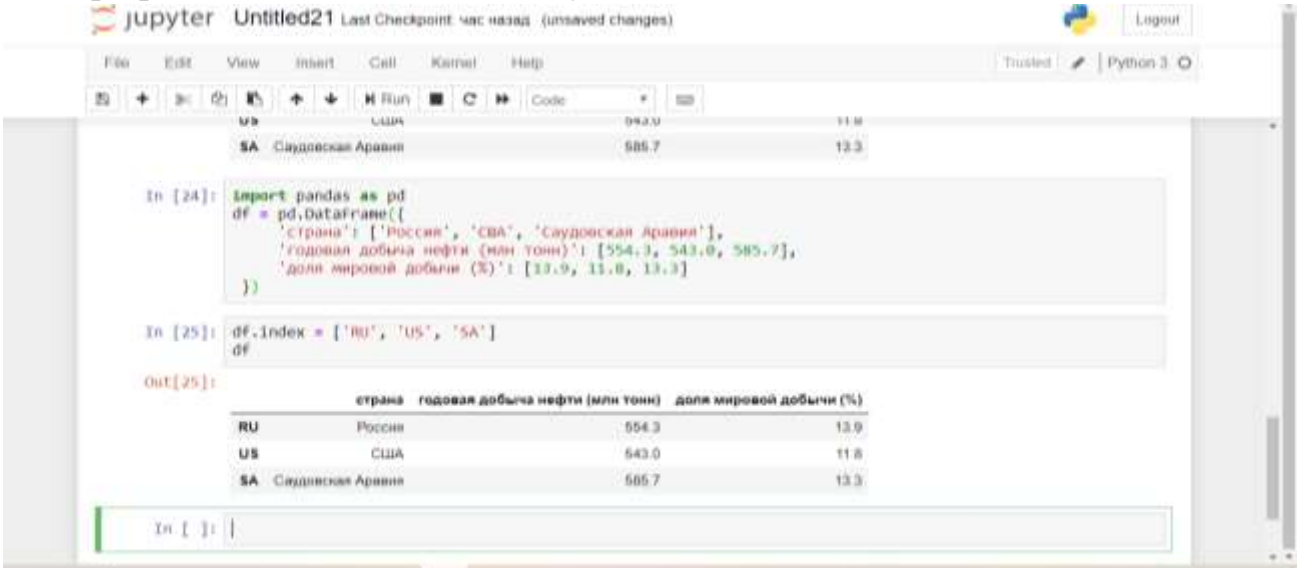
pandas поддерживает все самые популярные форматы хранения данных: csv, excel, sql, буфер обмена, html и многое другое. Чаще всего приходится работать с csv-файлами. Например, чтобы сохранить наш DataFrame со странами, достаточно написать: ***df.to_csv('имя_файла.csv')***. Считать данные из csv-файла и превратить в DataFrame можно функцией ***read_csv***.

Задание

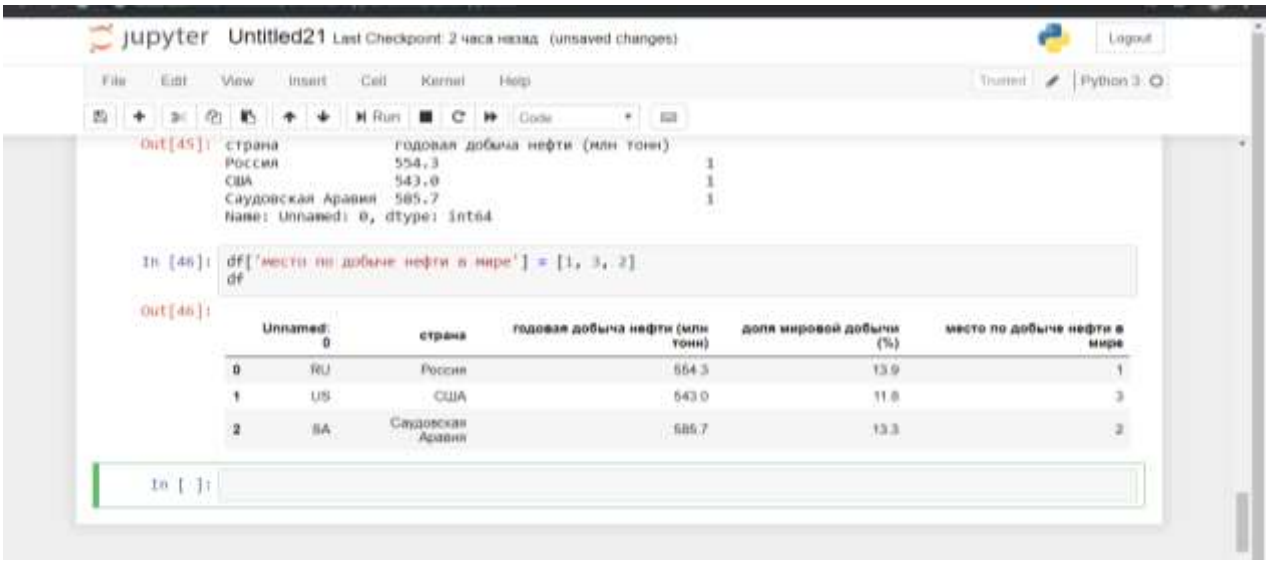
1. Создайте таблицу с помощью объекта DataFrame (столбцы: Страна, Добыча нефти в год, Доля мировой добычи нефти)
2. Вывод отдельного столбца (Например – страна)
3. Задайте новый индекс по строкам , при формировании самого объекта DataFrame:



или при работе с DataFrame “на лету’

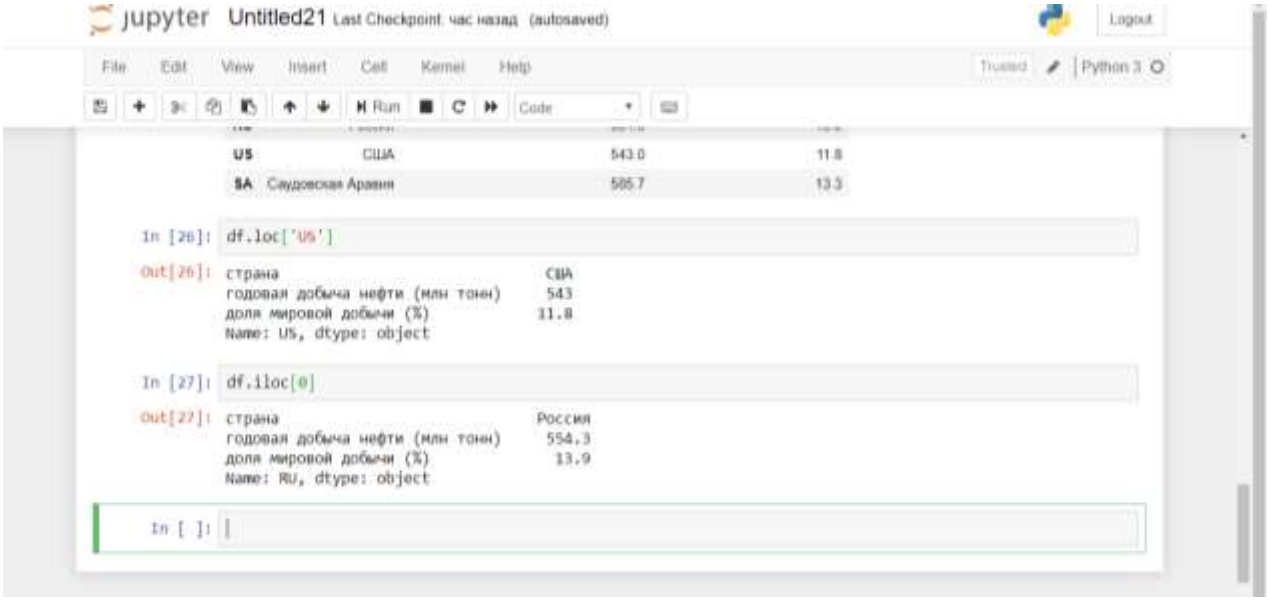


4. Создайте новый столбец с данными:



5. Выполните доступ к строкам по индексу, воспользовавшись следующими способами:

- .loc - используется для доступа по строковой метке
- .iloc - используется для доступа по числовому значению (начиная от 0)



6. Выполните поиск по индексу и интересующим колонкам:
7. Сохраните созданную Таблицу в csv-файл.
8. Выполните чтение данных из файла и установите новый индекс (например - страна)
9. Выполните удаление ненужных столбцов

4. Группировка и агрегирование в pandas

В теме рассматриваются вопросы группировки и агрегации данных. Приводятся примеры агрегации (статистика по группе) и группировке данных с помощью groupby()

Агрегация — это процесс превращения значений набора данных в одно значение. Например, у нас есть следующий набор данных

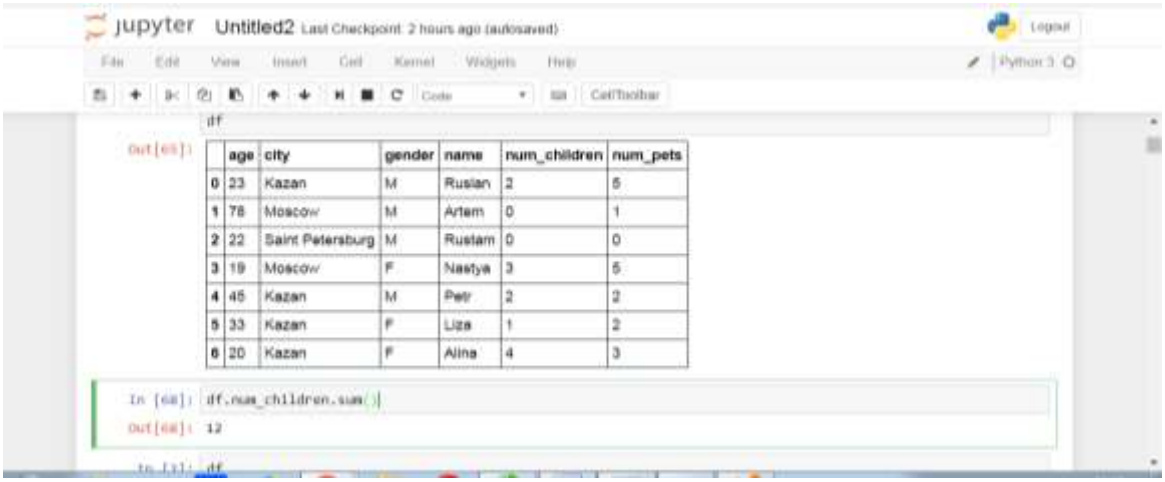
```
import pandas as pd
df = pd.DataFrame({
   'name':['Ruslan','Artem','Rustam','Nastya','Petr','Liza','Alina'],
   'age':[23,78,22,19,45,33,20],
   'gender':['M','M','M','F','M','F','F'],
   'city':['Kazan','Moscow','Saint Petersburg','Moscow','Kazan','Kazan','Kazan'],
   'num_children':[2,0,0,3,2,1,4],
   'num_pets':[5,1,0,5,2,2,3]
})
```

После создания или загрузки DataFrame полезно посмотреть что же там за данные, для этого можно воспользоваться следующими операциями:

- **df.head()** *# первые строки*
- **df.tail()** *# последние строки*
- **df.sample(5)** *# случайно выбранное кол-ва строк, полезно использовать для уменьшения матрицы для прогонки тестов*
- **df.shape** *# по аналогии с питру - размерность матрицы*

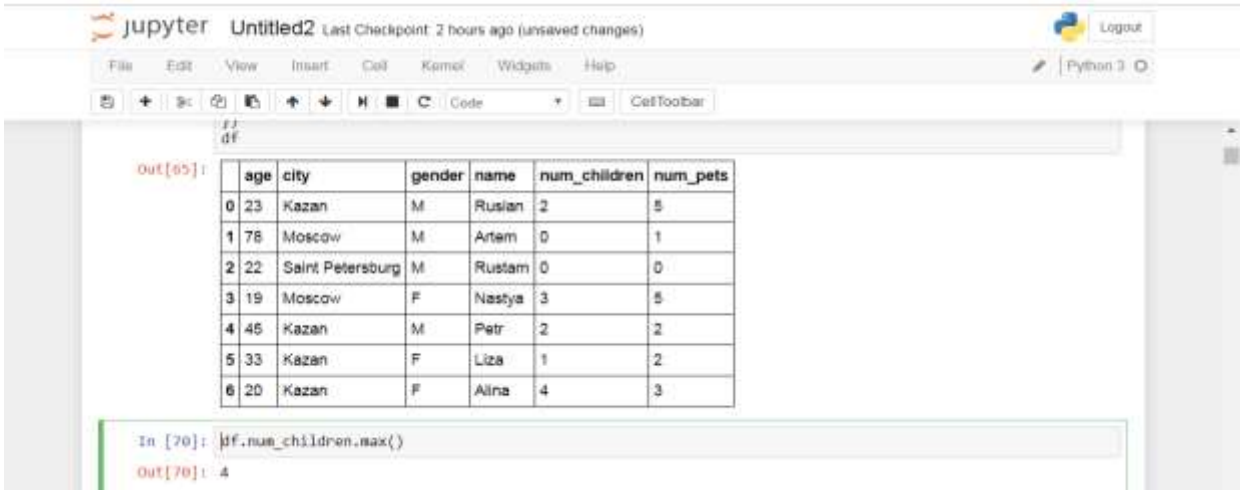
Пример, проверим сколько всего детей у сотрудников

```
df.num_children.sum()
```



	age	city	gender	name	num_children	num_pets
0	23	Kazan	M	Ruslan	2	5
1	78	Moscow	M	Artem	0	1
2	22	Saint Petersburg	M	Rustam	0	0
3	19	Moscow	F	Nastya	3	5
4	45	Kazan	M	Petr	2	2
5	33	Kazan	F	Liza	1	2
6	20	Kazan	F	Alina	4	3

С помощью min() и max() мы можем узнать максимальное и минимальное значение в таблице



	age	city	gender	name	num_children	num_pets
0	23	Kazan	M	Ruslan	2	5
1	78	Moscow	M	Artem	0	1
2	22	Saint Petersburg	M	Rustam	0	0
3	19	Moscow	F	Nastya	3	5
4	46	Kazan	M	Petr	2	2
5	33	Kazan	F	Liza	1	2
6	20	Kazan	F	Alina	4	3

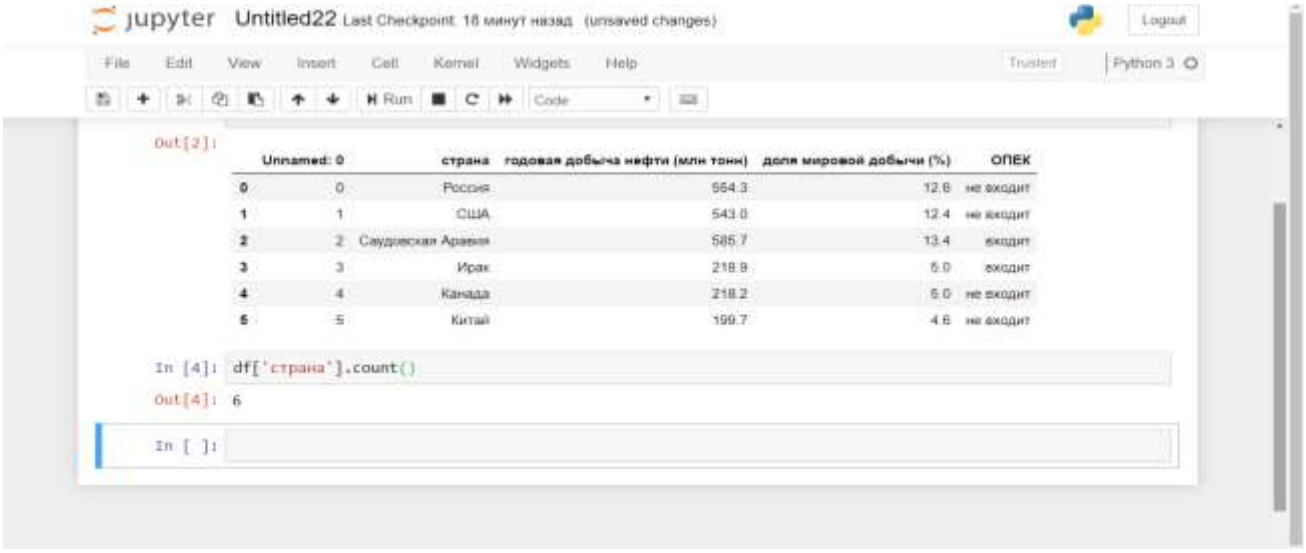
Задание

1. Посчитать количество строк в df.
2. Посчитать общее значение столбца .[column_name].
3. Найти наименьшее значение столбца .[column_name].
4. Найти среднестатистические показатели.[column_name].

Решение:

1. Посчитать количество строк в df

df.count() или df[‘страна’].count()



	Unnamed: 0	страна	годовая добыча нефти (млн тонн)	доля мировой добычи (%)	ОПЕК
0	0	Россия	564.3	12.8	не входит
1	1	США	543.0	12.4	не входит
2	2	Саудовская Аравия	585.7	13.4	входит
3	3	Ирак	218.9	5.0	входит
4	4	Канада	218.2	5.0	не входит
5	5	Китай	199.7	4.6	не входит

2. Посчитать общее значение столбца

df[‘годовая добыча нефти (млн тонн)’].sum()

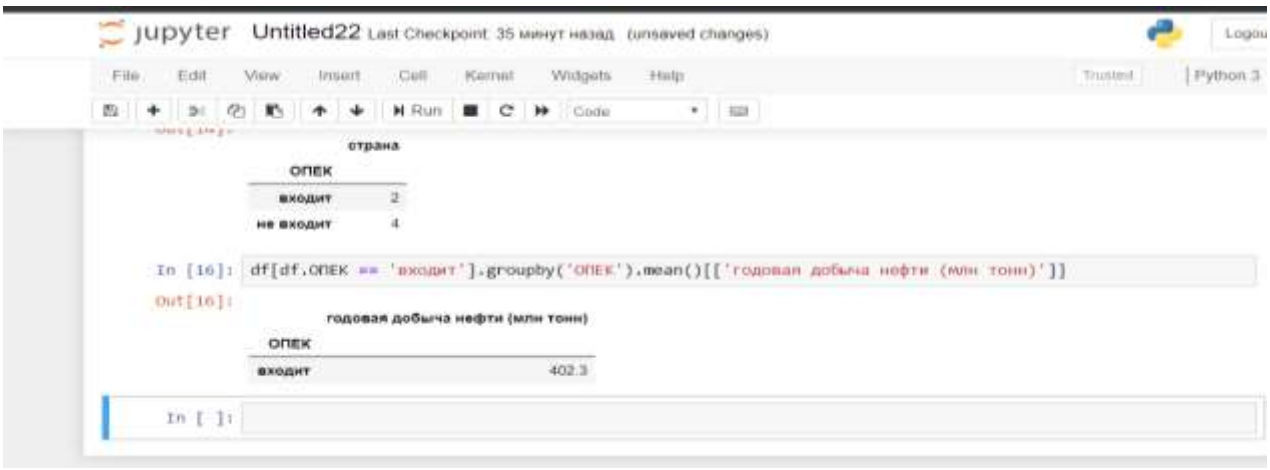
3. Найти наименьшее значение столбца, агрегация данных min() и max()
df[‘годовая добыча нефти (млн тонн)’].min() – *вывод минимального значения*
df[‘годовая добыча нефти (млн тонн)’].min() – *вывод максимальное значение*
4. Наконец, стоит посчитать среднестатистические показатели, например среднее и медиану, агрегация данных mean() и .median():
df[‘годовая добыча нефти (млн тонн)’].mean() – *вывод среднего значения*
df[‘годовая добыча нефти (млн тонн)’].median () – *вывод медианы*

Группировка данных

Группировка данных один из самых часто используемых методов при анализе данных. В pandas за группировку отвечает метод *.groupby.*

Функция groupby в действии:

Пример.
Вывести на экран среднюю добычу нефти среди стран, входящих в ОПЕК:


jupyter Untitled22 Last Checkpoint: 35 минут назад (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
страна
ОПЕК
входит 2
не входит 4
In [16]: df[df.ОПЕК == 'входит'].groupby('ОПЕК').mean()[['годовая добыча нефти (млн тонн)']]
Out[16]:
годовая добыча нефти (млн тонн)
ОПЕК
входит 402.3
In [ ]:

Библиографический список

1. Pilgrim M. , Willison S. . Dive Into Python 3 : т. T. 2. Springer, 2009. с.
2. Van Rossum G. , Drake F. L. . Python 3 Reference Manual : т. Scotts Valley, CA: CreateSpace, 2009. с.
3. Il'ichev V.Ju., Jurik E.A. Analiz massivov dannyh s ispol'zovaniem biblioteki Pandas dlja Python // Nauchnoe obozrenie. Tehnicheskie nauki. – 2020. – № 4. – S. 41-45