Basics of working with the Pandas module.

Download documents to your folder:

deti.csv,

bolezni.xlsx (File -> download as MS Excel),

world.csv.

To work with data in Python, programmers have a tool that will never let you down:pandas. It is a full-featured and intuitive open source library that provides data structures for working with high-dimensional datasets. There are 2 main data structures:

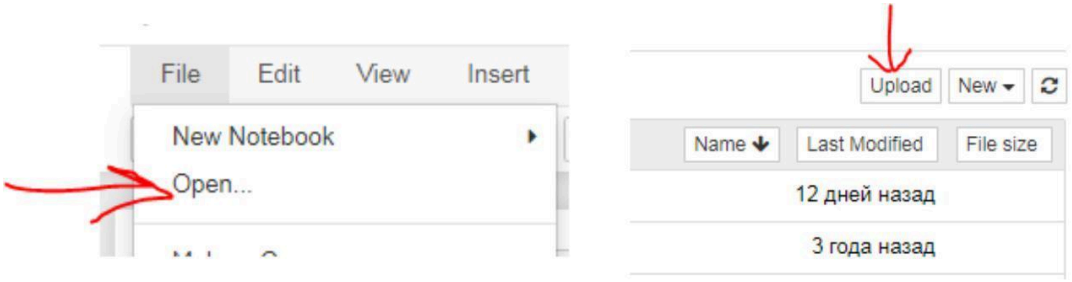
- **Series for one-dimensional arrays;**
- **DataFrame for two-dimensional tables containing rows and columns.**

Importing the module:

```
import pandas as pd
```

If everything is correct, there will be no error.

Task 1. Loading csv files. We consider the desired table into a variable data_st. Step 1. If you do not specify the path to the desired folder, then you must first download the file as follows:



Select the desired file on the computer, and upload to the system:

Step2. In the cell, we write the read command, specifying a semicolon as a separator:

```
data_st=pd.read_csv('deti.csv', sep=';')
```

Step3. We output the table:

```
data_st
```

Jupiter Notebook, allows in some cases to do without the print command.

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
data_st=pd.read_csv('deti.csv', sep=';')
```

```
data_st
```

	disease	2015	2016	2017	2018	2019	2020	2021
0	infectious	9774	8948	8782	10015	10986	9761.0	NaN
1	neoplasms	706	657	655	644	591	429.0	540.0
2	blood	1072	1024	896	883	852	522.0	611.0

Print the table using the print command, see the difference.

```
B [4]: print(data_st)
```

Task 2.Get information about the dataframe using the commands: Shape,

info(), columns, head(), tail(), sample(),isna(), isna().sum(), value_counts().

```
data_st.shape
```

```
(16, 8)
```

```
data_st.info()
```

Task 3. Download Excel files.Step1 is the same as step 1 from task 1.

Step2. In the cell, we write the reading command, indicating, in addition to the file name, the sheet name:

excel_data_df = pd.read_excel('bolezni.xlsx', sheet_name='disease')

Step3. We output the table:

excel_data_df

```
excel_data_df = pd.read_excel('bolezni.xlsx', sheet_name='disease')
```

```
excel_data_df
```

	заболевания	2015г.	2016г.	2017г.	2018г.	2019г.	2020г.	2021г.
0	инфекционные и паразитарные	9774	8948	8782	10015	10986	9761	7366
1	новообразования	706	657	655	644	591	429	540
2	болезни крови	1072	1024	896	883	852	522	611
3	болезни эндокринной системы	1860	2853	2693	3223	2327	1906	1741
4	болезни нервной системы	3979	3081	3307	3576	3282	2765	2929

Task 4.Analyze the data in the tableexcel_data_df using the method describe().

Please note that the data is displayed only for 2020 2021. This is because the COVID-19 row contains non-numeric data. Replace '-' with '0' in the string COVID - 19, using the method replace.

```
excel_data_df=excel_data_df.replace('-',0)
```

excel_data_df=	put the result in the same dataframe
excel_data_df	dataframe name
. replace	replacement method
'-'	what are we changing
0	what we change into

Apply the describe() method to the table again.

Task 5. Add a `metka' column to the dataframe.

```
excel_data_df['metka']=0
excel_data_df
```

9г.	2020г.	2021г.	metka
86	9761	7366	0
91	429	540	0
52	522	611	0

Task 6. Add a % column to 2020 and calculate how many percent is the incidence in 2021 in relation to 2020. Customize the output with two decimal places and a percent sign.

```
B [13]: excel_data_df['% к 2020']=(excel_data_df['2021г.']/excel_data_df['2020г.']*100).round(2)
B [14]: excel_data_df
Out[14]:
```

	заболевания	2015г.	2016г.	2017г.	2018г.	2019г.	2020г.	2021г.	% к 2020
0	инфекционные и паразитарные	9774	8948	8782	10015	10986	9761	7366	75.46
1	новообразования	706	657	655	644	591	429	540	125.87

and a percent sign.

```
B [15]: excel_data_df['% к 2020'] = excel_data_df['% к 2020'].astype(str) + '%'
B [16]: excel_data_df
Out[16]:
```

	заболевания	2015г.	2016г.	2017г.	2018г.	2019г.	2020г.	2021г.	% к 2020
0	инфекционные и паразитарные	9774	8948	8782	10015	10986	9761	7366	75.46%
1	новообразования	706	657	655	644	591	429	540	125.87%
2	болезни крови	1072	1024	896	883	852	522	611	117.05%

Task 7. Add columns `% to 2018, % to 2019 and calculate the percentage of incidence in 2021 in relation to 2018 and 2019. Customize the output with two decimal places after the point and a percent sign.

Task 8.Create a new dataframe from a subset of columns. It may useful if you want to store multiple dataframe columns in a new dataframe, but don't want to write out the names of the columns you want to remove. Specify the table name and the list of columns to be transferred to the new table.

```
B [18]: data_df_1=excel_data_df[['заболевания', '2020г.', '2021г.']]

B [19]: data_df_1
```

Out[19]:

	заболевания	2020г.	2021г.
0	инфекционные и паразитарные	9761	7366

Task 9.Create a new dataframe (data_df_0) from columns 2021, % to 2018 `, % to 2019 `,% to 2020 `.

Task 10.Delete the specified columns. This approach may be useful if only a few columns need to be removed from the dataframe.

```
B [23]: data_df_2=excel_data_df.drop(['2020г.', '2021г.','% к 2020'], axis=1)

B [24]: data_df_2
```

Out[24]:

	заболевания	2015г.	2016г.	2017г.	2018г.	2019г.
0	инфекционные и паразитарные	9774	8948	8782	10015	10986

We use the drop method to remove columns and write the result to a new dataframe. From the method parameters we use labels axis

drop(self,labels=None,axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')

The drop method drops or, in other words, removes the specified labels from rows or columns.labelscan be a single label or a list of labels or columns, to be dropped. axisdetermines if labels are removed from the index/string (0 or index) or column (1 or columns).

Task 11. Delete columns `metka and '% from 2020', '%from 2019', '%from 2018' from the source table

```
excel_data_df
```

	заболевания	2015г.	2016г.	2017г.	2018г.	2019г.	2020г.	2021г.
0	инфекционные и паразитарные	9774	8948	8782	10015	10986	9761	7366
1	новообразования	706	657	655	644	591	429	540
2	болезни крови	1072	1024	896	883	852	522	611

Task 12. Create a list or Series object based on the values of a column. For example, to get a separate list of diseases. The following commands apply here:

list_bolezn=data_df_1['diseases'].tolist()

list_bolezn

Task 13. Create a list based on the values in the 2021 column. From excel_data_df dataframe.

Task 14.In some previous tasks it was necessary to list column headers. Get the list of dataframe column headers from Task 7. Create a list of column headers: dataframe.columns.tolist()

Task 15. Let's sort the dataframe by the values of the column 2̀021y. ,̀use the .sort_values method:excel_data_df.sort_values('2021', ascending=False)

excel_data_df	**dataframe name**
. sort_values	**sorting method**
'2021'	**which column**
ascending=False	**in alphabetical order (reverse alphabetical True)**

Task 16. Sort the dataframe by the values in the diseases column, in reverse alphabetical order.

Task 17. Sort the dataframe by the values of the column 2̀020y. ,Inalphabet order.

From dataframes, you can select rows that meet a given condition, that is, filter. Note that using the method preserves the existing index values. To avoid this use method reset_index()

Task 18. Select rows from the excel_data_df dataframe that have values in the '2021' column Greater than or equal to 5000:excel_data_df[excel_data_df['2021'] >=5000]

Task 19. Select rows from the excel_data_df dataframe that have values in the '2015' column Less than or equal to 1000.

Task 20. Let's add rows with the sum of values from other rows to the dataframe.

```
data_df_1=data_df_1.append(data_df_1[['2020г.', '2021г.']].sum(axis=0), ignore_index=True)
```

data_df_1
. append
(data_df_1[['2020', '2021']] .
sum(axis=0),
ignore_index=True)

dataframe name
the method will add a string to the end of the dataframe specify
which columns of the dataframe we are working with the method
applied to the data in the columns
indexes are ignored, i.e. do not touch line numbers

COVID-19	1663.0	3512.0
NaN	199304.0	221298.0

You can see that the title row is NaN because we didn't add the values of the disease name column. Let's add the title of this line:

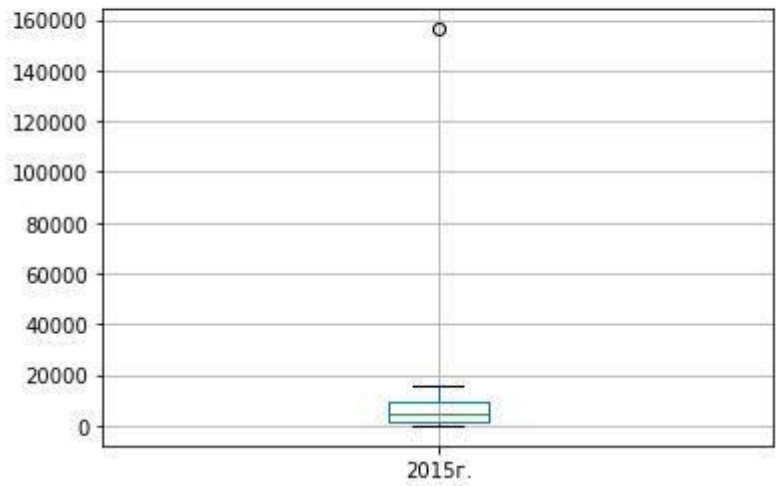
```
data_df_1.loc[16, 'заболевания']='Сумма'
```

Task 21. Add rows to the dataframe with the average of other rows, use the method mean(axis=0).

Task 22. Build a "Box with a mustache" on the excel_data_df dataframe, on the column '2015'

```
excel_data_df.boxplot(['2015г.'])
```

```
<AxesSubplot:>
```

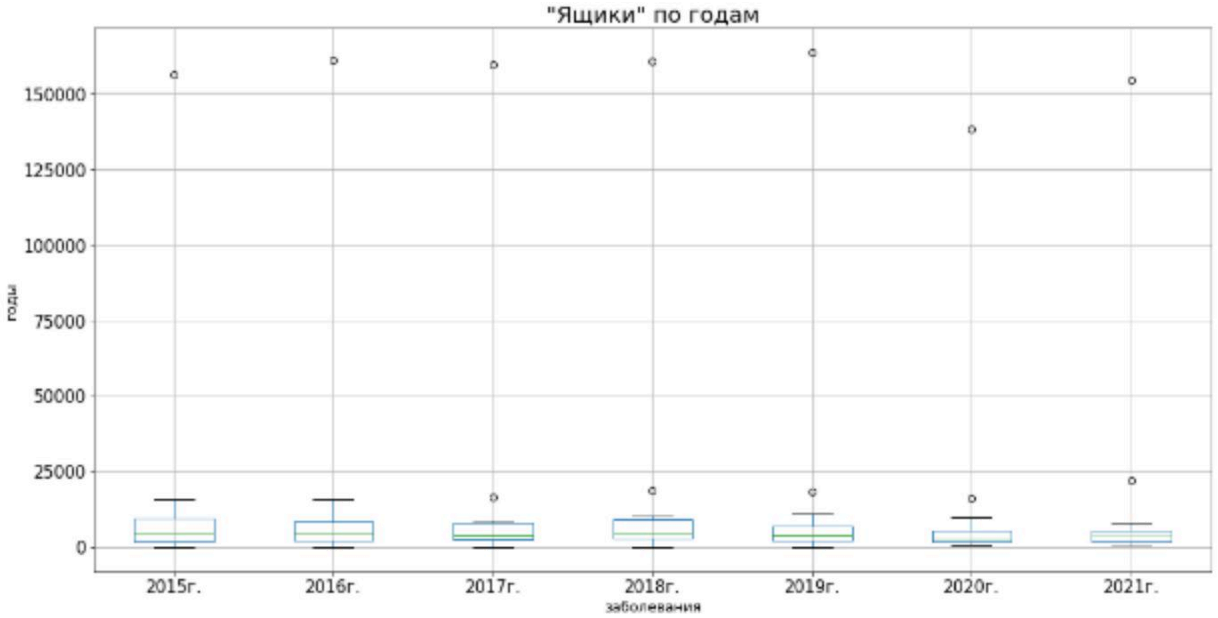


Task 23. Build a "Box with a mustache" on the excel_data_df dataframe, on all columns. Import the Library matplotlib:

```
import matplotlib.pyplot as plt
```

This can be done immediately before building, however, it is common to specify loadable modules (libraries) in the first lines of the program (code). Therefore, it is good practice to load the library in the first cell and re-execute all commands: Cell -> Run All or Kernel -> Restart & Run All

```
excel_data_df.plot(kind = 'box', grid=True, figsize=(18,9), fontsize=(15))
plt.title('"Ящики" по годам', fontsize=20)
plt.xlabel('заболевания', fontsize=13)
plt.ylabel('годы', fontsize=13)
plt.show()
```



excel_data_df.plot(kind = 'box', grid=True, figsize=(18,9), fontsize=(15))	dataframe method building(type diagrams = box `, mesh=Yes, size construction area, font size)
plt.title('"Boxes" by years', fontsize=20)	title, title font size
plt.xlabel('diseases', fontsize=13)	x-axis label, label font size
plt.ylabel('years', fontsize=13)	y-axis label, label font size
plt.show()	output for demonstration

Task 24. Load the date frameworld.csv.The dataframe contains the following data.

columns	Description
CCA3	3 digit country/area code
Name	Name of country/territories
Part_world	The part of the world in which the country/territory is located
2022	Population of the country/territories in 2022.
2020	Population of the country/territories in 2020.
2015	Population of the country/territories in 2015.
2010	Population of the country/territories in 2010.
2000	Population of the country/territories in 2000.
1990	Population of the country/territories in 1990.
1980	Population of the country/territories in 1980.
1970	Population of the country/territories in 1970.
area	Area of the country/territories in square kilometers.
Density	Population density per square kilometer.

Percentage	Percentage of population for each country/territory.

Study the table. Be sure to use the isna()+sum() and info() methods.

One of the basic functions of data analysis is grouping and aggregation. In pandas functiongroup bycan be combined with one or more functions aggregation to quickly and easily summarize data.*aggregation function*is a function that takes multiple individual values and returns a summary. In most cases, the returned data is a single value. The most common aggregation functions are*simple average*(simple average) or*summation*(summation) values.

Task 25. Simple aggregation example:world_df['Area'].agg(['sum', 'mean']):

```
world_df['Area'].agg(['sum', 'mean'])
```

```
sum     1.309061e+08
mean    7.000324e+05
Name: Area, dtype: float64
```

There are several ways to call an aggregation function, such as using a dictionary: df.agg({'fare': ['sum', 'mean'],'sex' : ['count']}).

```
world_df.agg({'Density': ['sum', 'mean'],'Name' : ['count']})
```

	Density	Name
count	NaN	187.0
mean	307.394866	NaN
sum	57482.840000	NaN

The most common built-in aggregation functions are basic math functions, including*amount*(sum),*mean*(means) *median value*(median),*minimum*(minimum), *maximum*(maximum),*standard deviation*(standard deviation)*dispersion*(variance), *mean absolute deviation* (mean absolute deviation) and*work*(product).

Our dataframe contains 187 rows, not very much, however, this number can be optimized for further analysis, for example, by applying grouping: world_df.groupby('Part_world') on the 'Part_world' column.

Task 26. To “see” which rows belong to which group, use the methodgroups, and also give the grouped table a new name.

```
world_gr_p=world_df.groupby('Part_world').groups
world_gr_p
```

```
{'Africa': [2, 4, 18, 22, 26, 27, 29, 32, 33, 37, 44
9, 60, 63, 67, 68, 79, 84, 90, 91, 92, 96, 97, 100,
0, 121, 136, 139, 144, 146, 148, 149, 154, 155, 157,
```

Task 27. Summarize the data in the grouped table and place it in a new one:

```
world_gr=world_df.groupby(['Part_world']).sum()
world_gr
```

Part_world	2022	2020	2015	2010	2000	
Africa	1424840	1358828	1199377	1053606	817544	63
Asia	4316009	4264954	4082729	3868779	3431490	295
Australia						

Task 28. Let's group the data by the "Part_world" column and find various characteristics by columns:

```
world_df.groupby(["Part_world"]).agg({"Name": "count", "Area":["sum","mean"],
                                      "Density": "mean", "Percent_num": "mean"}).reset_index()
```

And reset the old indexes.

Task 29. It can be seen that more than 50% of the world's population lives in Asian countries. Let's form a dataframe containing only Asian countries. Let's reset the old indexes:

```
world_df_a=                                                    .reset_index()
world_df_a
```

	index	CCA3	Name	Part_world	2022	2020	2015	2010
0	0	AF	Afghanistan	Asia	41129	38972	33753	28190
1	7	AM	Armenia	Asia	2780	2806	2879	2946
2	10	AZ	Azerbaijan	Asia	10358	10285	9863	9237
3	12	BH	Bahrain	Asia	1472	1477	1362	1214
4	13	BD	Bangladesh	Asia	171186	167421	157830	148391

You can see that in addition to the new ones, the old indexes are preserved, let's get rid of them:

```
world_df_a=                                                    ].reset_index(drop=True)
world_df_a
```

	CCA3	Name	Part_world	2022	2020	2015	2010	2000	1990
0	AF	Afghanistan	Asia	41129	38972	33753	28190	19543	10695
1	AM	Armenia	Asia	2780	2806	2879	2946	3169	3557
2	AZ	Azerbaijan	Asia	10358	10285	9863	9237	8190	7428
3	BH	Bahrain	Asia	1472	1477	1362	1214	711	517
4	BD	Bangladesh	Asia	171186	167421	157830	148391	129193	107148

Task 30. Sort the dataframe by the '2022' column. Place in the same dataframe.

Task 31. Let's build a graph.

```
world_df_a.plot(x='Name', y='2022', kind='bar', grid=True, figsize=(18,14))
plt.title('График населения стран Азии')
plt.xlabel('Страны')
plt.ylabel('Население')
plt.show()
```

Task 32. Let's build a graph using another library. Import the Library seaborn:

```
import seaborn as sns
```

This can be done immediately before building, however, it is common to specify loadable modules (libraries) in the first lines of the program (code). Therefore, it is good practice to load the library in the first cell and re-execute all commands: Cell -> Run All or Kernel -> Restart & Run All.

```
plt.figure(figsize=(18,20))
plt.grid()
sns.barplot(x='2022', y='Name', data=world_df_a)
plt.title('График населения азиатских стран', fontsize=15)
plt.xlabel('Население')
plt.ylabel('Страны')
plt.show()
```