

Создание Series из списков

```
countries = pd.Series(  
    data = ['Англия', 'Канада', 'США', 'Россия', 'Украина', 'Беларусь',  
            'Казахстан'],  
    index = ['UK', 'CA', 'US', 'RU', 'UA', 'BY', 'KZ'],  
    name = 'countries'  
)
```

Создание Series из словаря

```
countries = pd.Series({  
    'UK': 'Англия',  
    'CA': 'Канада',  
    'US': 'США',  
    'RU': 'Россия',  
    'UA': 'Украина',  
    'BY': 'Беларусь',  
    'KZ': 'Казахстан'},  
    name = 'countries'  
)
```

Доступ к данным в Series

<code>.loc[x]</code>	Доступ по меткам (индексам)
<code>.iloc[i]</code>	Доступ по порядковому номеру

Создание DataFrame из словаря

```
countries_df = pd.DataFrame({  
    'country': ['Англия', 'Канада', 'США', 'Россия', 'Украина',  
                'Беларусь', 'Казахстан'],  
    'population': [56.29, 38.05, 322.28, 146.24, 45.5, 9.5, 17.04],  
    'square': [133396, 9984670, 9826630, 17125191, 603628, 207600,  
                2724902]  
},
```

```
index = ['UK', 'CA', 'US', 'RU', 'UA', 'BY', 'KZ']  
)
```

Создание DataFrame из списка

```
countries_df = pd.DataFrame(  
    data = [  
        ['Англия', 56.29, 133396],  
        ['Канада', 38.05, 9984670],  
        ['США', 322.28, 9826630],  
        ['Россия', 146.24, 17125191],  
        ['Украина', 45.5, 603628],  
        ['Беларусь', 9.5, 207600],  
        ['Казахстан', 17.04, 2724902]  
    ],  
    columns= ['country', 'population', 'square'],  
    index = ['UK', 'CA', 'US', 'RU', 'UA', 'BY', 'KZ']  
)
```

Доступ к данным в DataFrame

<code>.loc[row, col]</code>	Доступ по меткам (индексам) и именам столбцов
<code>.loc[irow, icol]</code>	Доступ по порядковым номерам строк и столбцов

Чтение данных в DataFrame

<code>pd.read_csv(path)</code>	Чтение файла .csv
<code>pd.read_excel(path)</code>	Чтение файла .xlsx и .xls
<code>pd.read_json(path)</code>	Чтение файла .json
<code>pd.read_xml(path)</code>	Чтение файла .xml
<code>pd.read_sql(query, connection)</code>	Чтение базы данных sql

Запись данных в DataFrame

<code>df.to_csv(path)</code>	Запись таблицы в формат <code>.csv</code>
<code>df.to_excel(path)</code>	Запись таблицы в формат <code>.xlsx</code> и <code>.xls</code>
<code>df.to_json(path)</code>	Запись таблицы в формат <code>.json</code>
<code>df.to_xml(path)</code>	Запись таблицы в формат <code>.xml</code>
<code>df.to_sql(query, connection)</code>	Запись таблицы в базу данных <code>sql</code>

Методы исследования структуры таблицы

<code>df.shape</code>	Размерность таблицы
<code>df.head(n)</code>	<code>n</code> первых строк таблицы. По умолчанию <code>n = 5</code>
<code>df.tail(n)</code>	<code>n</code> последних строк таблицы. По умолчанию <code>n = 5</code>
<code>df.info()</code>	Вывод информации о столбцах: количество непустых значений, типы данных. Также выводится объём занимаемой таблицей памяти

Методы описательных статистик

<code>df.describe(include=['type'])</code>	Получить статистическую информацию о столбцах указанного типа (по умолчанию типы <code>'int'</code> и <code>'float'</code>)
<code>df[col].value_counts()</code>	Получить частоту каждого уникального значения в столбце <code>col</code>

Изменение типа данных

<code>df[col].astype('type')</code>	Преобразовать тип данных столбца
<code>df[col].dtype</code>	Получить тип данных столбца

Статистические методы

<code>df[col].count()</code>	Количество непустых значений
<code>df[col].mean()</code>	Среднее
<code>df[col].median()</code>	Медиана
<code>df[col].std()</code>	Стандартное отклонение
<code>df[col].min()</code>	Минимум
<code>df[col].max()</code>	Максимум
<code>df[col].sum()</code>	Сумма
<code>df[col].quantile(x)</code>	Квантиль уровня x
<code>df[col].mode()</code>	Модальное значение (может быть не одно, возвращает объект <i>Series</i>)

Фильтрация данных в DataFrame по одному условию

```
mask = df[col] < a
df_filtered = df[mask]

# или

df_filtered=df[df[col]<a]
```

Операторы составных условий

&	Поэлементное И
	Поэлементное ИЛИ

Фильтрация данных по сложным условиям

```
mask1 = df[col1] < a
mask2 = df[col2] >= b
mask3 = df[col3] == c
df_filtered=df[(mask1 | mask2) & mask3]

# или

df_filtered=df[(df[col1] < a | df[col2] >= b) & df[col3] == c ]
```

Фильтрация данных и статистические методы

```
#получение среднего в столбце col2 при условии, что столбец col1 > a

df[df[col1] > a][col2].mean()
```

Best Practice

```
#Хорошо
median = df[col1].median()
df[df[col1] > median][col2].max()

#Плохо
df[df[col1] > df[col1].median()][col2].max()

#Хорошо
mask1 = df[col1] < a
mask2 = df[col2] > b
...
```

```
maskn = df[coln] == z  
  
df[mask1 & mask2 & ... & maskn]  
  
#Плохо (при большом количестве масок такой код становится нечитабельным)  
df[df[col1] < a & df[col2] > b & ... & df[coln] == z]
```