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

`.loc[x]`	Доступ по меткам (индексам)
`.iloc[i]`	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

`.loc[row, col]`	Доступ по меткам (индексам) и именам столбцов
`.iloc[irow, icol]`	Доступ по порядковым номерам строк и столбцов

Чтение данных в DataFrame

`pd.read_csv(path)`	Чтение файла *.csv*
`pd.read_excel(path)`	Чтение файла *.xslx* и *.xls*
`pd.read_json(path)`	Чтение файла *.json*
`pd.read_xml(path)`	Чтение файла *.xml*
`pd.read_sql(query, connection)`	Чтение базы данных *sql*

Запись данных в DataFrame

`df.to_csv(path)`	Запись таблицы в формат *.csv*
`df.to_excel(path)`	Запись таблицы в формат *.xslx* и *.xls*

`df.to_json(path)`	Запись таблицы в формат *.json*
`df.to_xml(path)`	Запись таблицы в формат *.xml*
`df.to_sql(query, connection)`	Запись таблицы в базу данных *sql*

Методы исследования структуры таблицы

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

Методы описательных статистик

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

Изменение типа данных

`df[col].astype('type')`	Преобразовать тип данных столбца
`df[col].dtype`	Получить тип данных столбца

Статистические методы

`df[col].count()`	Количество непустых значений
`df[col].mean()`	Среднее

`df[col].median()`	Медиана
`df[col].std()`	Стандартное отклонение
`df[col].min()`	Минимум
`df[col].max()`	Максимум
`df[col].sum()`	Сумма
`df[col].quantile(x)`	Квантиль уровня x
`df[col].mode()`	Модальное значение (может быть не одно, возвращает объект *Series*)

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

df[(df[col1] < a) & (df[col2] > b) & ... & (df[coln] == z)]
```