**Работа с Pandas**

| Создание объекта Series | a = [1, 3, 5, 7, 2]  b = **pd.Series**(a) |
| --- | --- |
| Можно задать индексы явно | b = pd.Series(a,**index**= [0, 1, 2, 3, 4])  b = pd.Series(a, index = ['a', 'b', 'c', 'd', 'e']) |
| Получение индексов | b.index |
| Поменять индексы | b.index = [10, 11, 12, 13, 14] |
| Задать тип данных | b = pd.Series(a,**dtype**= np.float64) |
| Поменять тип данных позже | b = b.**astype**(np.float64) |
| Создать Series из словаря (ключи становятся индексами, значения - данными) | d = {'1st':'a', '2nd':'b', '3rd':'c', '4th':'d', '5th':'e'}  b = pd.Series(d) |
| Просмотр значений массива Series | b.**values** |
| Доступ к данным по индексу  Доступ по списку индексов | b[0]  b[[0, 3]] |
| **Выбор данных из массива Series** |  |
| первые несколько элементов (по умолчанию - 5)  последние несколько элементов | b.**head**()  b.head(3)  b.**tail**() |
| Выборка по условию:   * по логическому "и" * по логическому "или" | b[b > 5]  b[(b > 5) **&** (b % 2 == 0)]  b[(b > 5) **|** (b % 3 == 0)] |
| Изменение элементов |  |
| Элементу с индексом 0 присваивается  значение 4  Все числе в Series, которые меньше 5,  приравниваются к 0  Элементам под индексами 0, 1, 2  присваивается значение 1 | b[0] = 4  b[b < 5] = 0  b[[0, 1, 2]] = 1 |
| Добавление и удаление данных Добавление данных  Удаление данных по индексу | b = b.**append**(pd.Series({6:10, 7:11, 8:12, 9:13}))  b = b.drop([0, 1, 2]) |
| Запись и чтение файла Запись Series в файл с форматом pickle  Чтение файла | b.**to\_pickle**('b.pkl')  b2 = pd.**read\_pickle**('b.pkl') |
| **Структура данных DataFrame** |  |
| Создание DataFrame | df = pd.DataFrame({'col1': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'], 'col2': [1, 3, 5, 7, 9, 11, 13, 15]}) |
| **Просмотр информации о DataFrame**  Просмотреть форму массива  Список столбцов  Список индексов  Покажет кол-во строк в столбце, тип  данных  Статистические хар-ки | df.shape  df.columns  df.index  df.info()  df.describe() |
| **Выбор данных из массива** |  |
| Просмотр нескольких первых строк  DataFrame  Просмотр последних 5-ти строк  Вывод первых строк для отдельного  столбца | df.head()  df.head(3)  df.tail()  df['col1'].head()  df.col1.head() |
| Изменение индекса | df.index = [2, 4, 6, 8, 10, 12, 14, 16] |
| **Выбор данных по индексу**  Указывается индекс строки и имя столбца  Если второй аргумент не указан,  выбираются все столбцы  Срез по строкам, верхняя граница включена | df.loc[2, 'col1']  df.loc[4]  df.loc[2:4] |
| **Выбор данных по позиции**  Просмотр первой строки во всех  столбцах  Просмотр среза первых двух строк, верхняя граница не включена  Для просмотра данных конкретного столбца указывается его индекс | df.iloc[0]  df.iloc[0:2]  df.iloc[2:6, 0] |
| **Выбор по условию**  Выбираем те строки, в которых значение  col1 = b  Результат такой же, но выводятся не все  столбцы, а только col2  Просмотр тех строк, в которых значение  col2 больше 10, при этом выводятся  только значения col1  Пример логического "и"  Пример логического "или"  Указание отрезка, которому должен  соответствовать столбец  Можно указать конкретные элементы для значений столбца  Указание строк, которые не соответствуют значениям в методе isin  Запрос с помощью метода .query (более короткий) | df.loc[df['col1'] == 'b']  df.loc[df['col1'] == 'b', 'col2']  df.loc[df['col2'] > 10, 'col1']  df.loc[(df['col2'] > 10) & (df['col1'] != 'g')]  df.loc[(df['col2'] > 10) | (df['col2'] % 9 == 0)]  df.loc[df['col2'].**between**(11,13)]  df.loc[df['col1'].**isin**(['a', 'b', 'c', 'd', 'e'])]  df.loc[**~**df['col1'].isin(['a', 'b', 'c', 'd', 'e'])]  df.query('col1 == "b"') название столбца можно писать без кавычек  df.**query**('col2 > 10') |
| Получение DataFrame из Series | s = df['col1']  df2 = pd.**DataFrame**(s) |
| Копирование DataFrame | df\_copy = df.**copy**() |
| Случайный выбор и перемешивание |  |
| Случайный выбор n-го количества строк | df.**sample**(n = 2) |
| Случайный выбор доли от исходного DataFrame | df.sample(**frac** = 0.5) выборка половины (0,5) исходного DataFrame |
| Случайный выбор с возвращением (строки могут повторяться) | df.sample(frac = 0.5, **replace**=True) |
| Случайное перемешивание для повторения одной и той же случайной выборки (в скобках к-л число) | df.sample(frac = 1, **random\_state** = 42) |
| Запись и чтение DataFrame из файлов |  |
| Запись в файл разделитель по умолчанию ",")  Запись в excel и pickle - с помощью методов to\_excel, to\_pickle. | df.to\_csv('Test.csv', sep = ';', index = False) |
| Чтение из файла  Чтение excel и pickle - read\_excel, read\_pickle | df\_new = pd.read\_csv('Test.csv', sep = ';')  df\_new |
| **Работа с данными DataFrame** |  |
| Слияние данных | df1 = pd.**merge**(authors, books, **on** = 'author\_id', **how** = 'inner') # ‘left’, ‘right’, ‘outer’ |
| Работа с пропущенными данными Посмотрим строки df4, в которых название книги пропущено  Альтернативный способ  Просмотр строк, в которых не пропущено имя автора  Заполнение пропущенных значений специальным текстом | df4.loc[df4['book\_title'].**isnull**()]  df4[df4['book\_title'].isnull()]  df4[df4['author\_name'].**notnull**()]  df4['author\_name'] = df4['author\_name'].**fillna**('unknown') |
| Добавление столбцов | **df4[‘quantity’]**  df4.loc[(df4['author\_name'] != 'unknown') & (df4['book\_title'] != 'unknown'), 'quantity'] = 1 |
| Заполнение пропусков в столбце нулями | df4['quantity'].**fillna**(0, inplace = True) |
| Приведение данных столбца к типу int (невозможно, если есть пропуски) | df4['quantity'] = df4['quantity'].**astype**(int) |
| Сделаем столбец author\_id индексом | df4.**set\_index**('author\_id', inplace = True) |
| Сброс индекса | df4.**reset\_index**(inplace = True) |
| Удаление данных (axis=0 - удаление строки, axis=1- удаление столбца) удалим строку (по значению индекса) | df4.**drop**('price', **axis = 1**, inplace = True)  df4 = df4.**drop**(1, **axis = 0**) |
| Сортировка |  |
| Вернем удаленную строку  ignore\_index = True - чтобы индексы соединяемых таблиц не учитывались | df4 = df4.**append**(  {  'author\_id': 2,  'author\_name': 'Tolstoy',  'book\_title': 'War and Peace',  'quantity': 1,  },  ignore\_index = True,  ) |
| Отсортируем по столбцу author\_id: | df4.**sort\_values(by=**'author\_id', inplace=True) |
| Переустановим индекс:  drop=True - текущий столбец из индексов не нужно сохранять в таблице, а можно удалить | df4.**reset\_index(drop=True**, inplace=True) |
| Соединение датафреймов |  |
| Конкатенация по оси 0 (добавление новых строк): | df6 = pd.**concat**([df4, df5], **axis=0**, ignore\_index=True) |
| Конкатенация по оси 1 (добавление новых столбцов): | df8 = pd.**concat**([df6, df7], **axis=1)** |
| Применение функций и методов |  |
| Значения столбцов можно поэлементно перемножать и складывать | df8['total'] = df8['quantity'] \* df8['price'] |
| **Статистические показатели**  Минимум  Максимум  Среднее  Медиана  Среднее квадратическое значение  Дисперсия | df8['price'].**max**()  df8['price'].**min**()  df8['price'].**mean**()  df8['price'].**median**()  df8['price'].**std**()  df8['price'].**var**() |
| Вывод нескольких наибольших значений   * nsmallest для наибольших | df8.**nlargest**(3, 'price') |
| Уникальные значения заданного столбца | df8['author\_name'].**unique**() |
| Количество уникальных значений | df8['author\_name'].**nunique**() |
| Сколько раз появляется уникальное значение в данном столбце | df8['author\_name'].**value\_counts**() |
| Применение функций, которых нет в numpy и pandas | df8['book\_title'].apply(lambda x: x.**upper**()) |
| Группировка данных |  |
| 1-й способ: | df8.**groupby**('author\_name')['price'].max() |
| 2-й способ: | price\_agg = df8.groupby('author\_name').**agg**({'price': 'max', ‘total’: ‘count’}) |