**Глава 5**

**Первое знакомство с pandas**

Библиотека pandas будет основным предметом изучения в последующих главах. Она содержит высокоуровневые структуры данных и средства манипуляции ими, спроектированные так, чтобы обеспечить простоту и высокую скорость анализа данных на Python. Эта библиотека построена поверх NumPy, поэтому cii легко пользоваться в приложениях, ориентированных на NumPy.

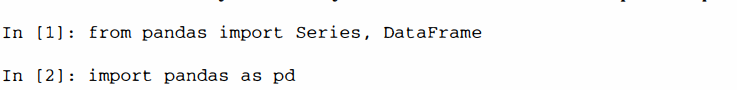
Для сведения - я начал разрабатывать pandas в начале 2008 года, когда работал в компании AQR, занимающейся количественным анализом инвестиционных рисков. Тогда я столкнулся с требованиями, которые не мог полностью удовлетворить ни один из имевшихся в моем распоряжении инструментов:

* Структуры данных с помеченными осями, поддерживающие автоматическое или явное выравнивание. Это помогает избежать типичных ошибок, связанных с невыровненностью данных и работой с данными, поступивши­ ми из разных источников и по-разному проиндексированных.
* Встроенная функциональность для работы с временными рядами.
* Одни и те же структуры должны быть пригодны для обработки как временных рядов, так и данных иного характера.
* Арифметические операции и операции редуцирования (например, суммирование вдоль оси) должны пробрасывать метаданные (метки осей).
* Гибкая обработка отсутствующих данных.
* Объединение и другие реляционные операции, имеющиеся в популярных базах данных (например, на основе SQL).

Я хотел, чтобы все это было сосредоточено в одном месте и предпочтительно написано на языке, подходящем для разработки программ общего назначения. Python казался неплохим кандидатом, но в то время в нем не было встроенных структур данных и инструментов, поддерживающих нужную мне функциональность.

За прошедшие четыре года paшlas превратилась в довольно обширную библиотеку, способную решать куда более широкий круг задач обработки данных, чем я планировал первоначально, но расширялась она, не принося в жертву простоту и удобство использования, к которым я стремился с самого начала. Надеюсь, что, прочитав эту книгу, вы так же, как и я, станете считать ее незаменимым инструментом.

В этой книге используются следующие соглашения об импорте для pandas:



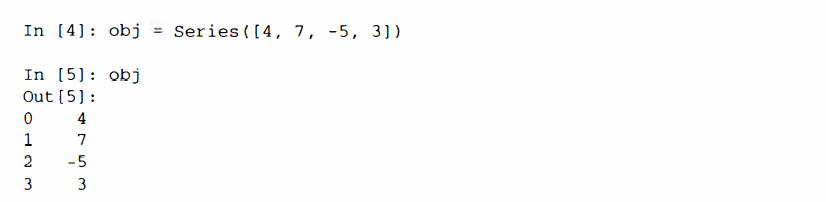
Таким образом, увидев в коде строку pd., знайте, что это ссылка на pandas. Объекты Series и DataFrame используются так часто, что я счел полезным импортировать их в локальное пространство имен.

Введение в структуры данных pandas

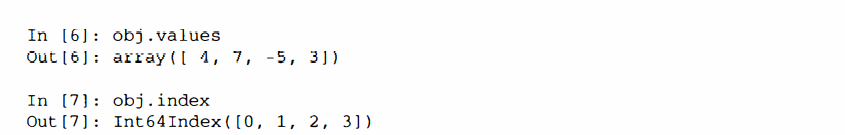
Чтобы начать работу с pandas, вы должны освоить две основные структуры данных: *Saies* и *DataFmme.* Они, конечно, не являются универсальным решением любой задачи, но все же образуют солидную и простую для использования основу большинства приложений.

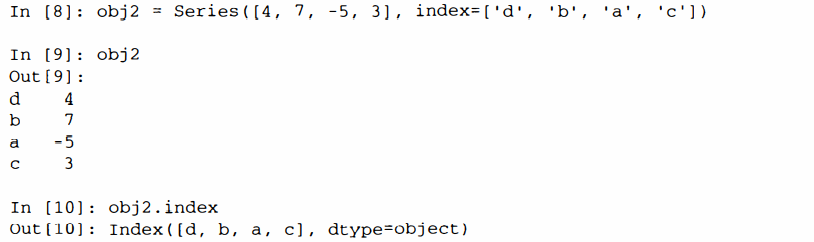
Объект Series

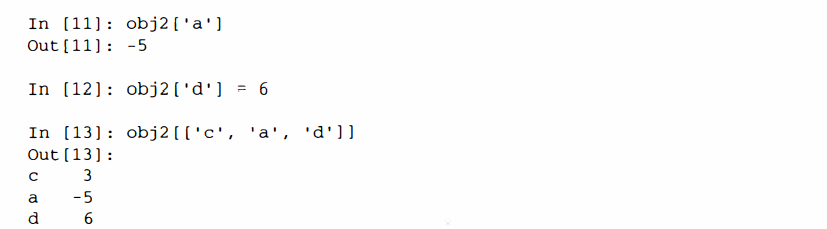
Series - одномерный похожий на массив объект, содержащий массив данных (любого типа, поддерживаемого NumPy) и ассоциированный с ним массив меток, который называется *индексо.м.* Простейший объект Series состоит только из мас­ сива данных:

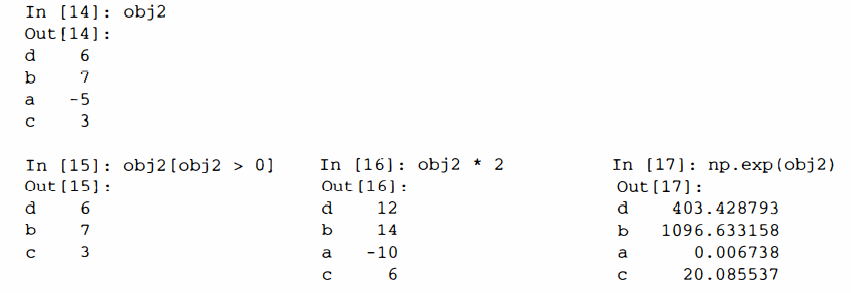


П строковом представлении Series, отображаемом в интерактивном режиме, индекс находится слева, а значения справа. Поскольку мы не задали индекс для данных, то по умолчанию создается индекс, состоящий из целых чисел от О до *N* - 1 (где *N* - длина массива данных). Имея объект Series, получить представление самого массива и его индекса можно с помощью атрибутов values и index соответственно:

Часто желательно создать объект Series с индексом, идентифицирующим каждый элемент данных:

В отличие от обычного массива NumPy, для выборки одного или нескольких элементов из объекта Series можно использовать значения индекса:

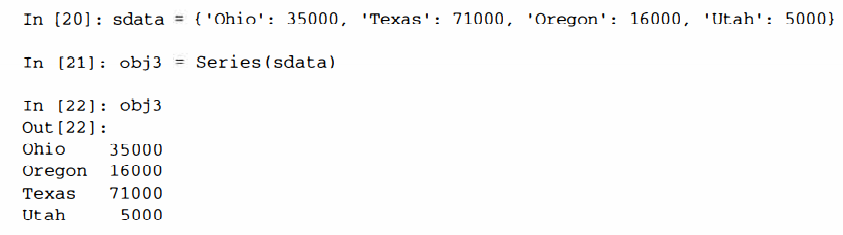
Операции с массивом NuшPy, например фильтрация с помощью булева массива, скалярное умножение или применение математических функций, сохраняют связь между индексом и значением:



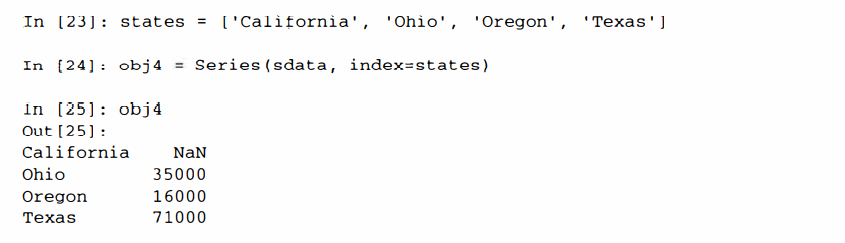
Объект Series можно также представлять себе как упорядоченный словарь фиксированной длины, поскольку он отображает индекс на данные. Его можно передавать многим функциям, ожидающим получить словарь:



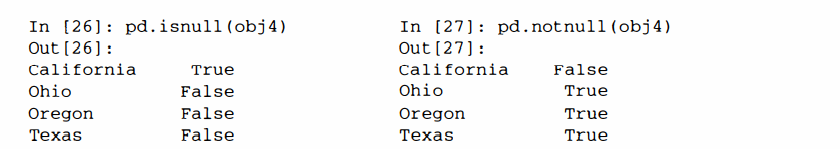
Если имеется словарь Python, содержащий данные, то из него можно создать объект Series:



Если передается только словарь, то в получившемся объекте Series ключи будут храниться в индексе по порядку:



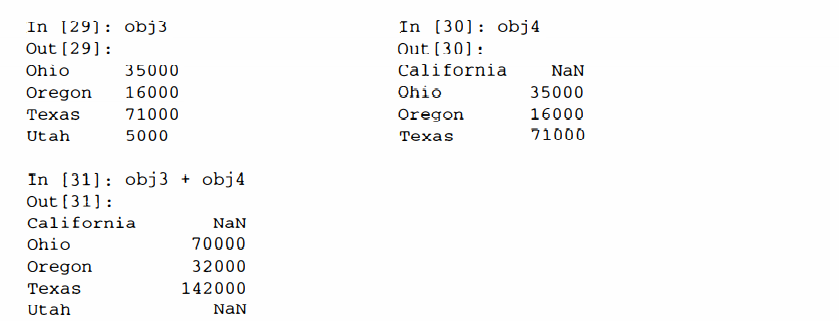
В данном случае 3 значения, найденные в sdata, помещены в соответствующие им позиции, а для метки «California» никакого значения не нашлось, поэтому ей соответствует признак NaN (не число), которым в pandas обозначаются отсутствующие значения. Иногда, говоря об отсутствующих данных, я буду употреблять термин «NA». Для распознавания отсутствующих данных в pandas следует использовать функции isnull и notnull:



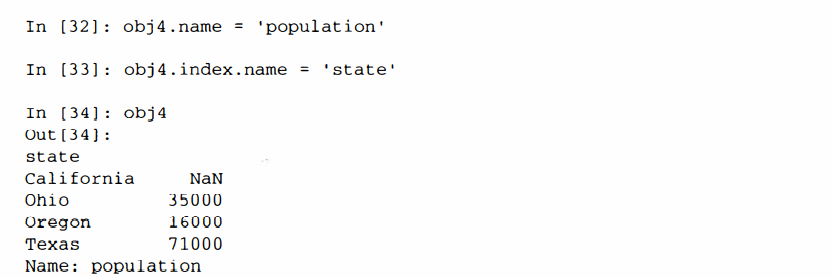
У объекта Series есть также методы экземпляра:



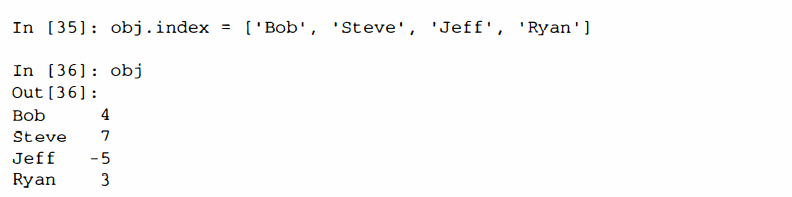
Более подробно работа с отсутствующими данными будет обсуждаться ниже в этой главе. Для многих приложений особенно важно, что при выполнении арифметических операций объект Series автоматически выравнивает данные, которые проиндексированы в разном порядке:



Вопрос о выравнивании данных будет рассмотрен отдельно. И у самого объекта Series, и у его индекса имеется атрибут name, тесно связанный с другими частями функциональности pandas:



Индекс объекта Series можно изменить на месте с помощью присваивания:

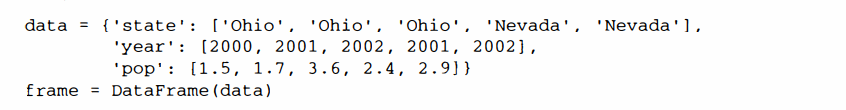


**Объект DataFrame**

Объект DataFrame представляет табличную структуру данных, состоящую из упорядоченной коллекции столбцов, причем типы значений (числовой, строковый, булев и т. д.) в разных столбцах могут различаться. В объекте DataFгame хранятся два индекса: по строкам и по столбцам. Можно считать, что это словар1, объектов Series. По сравнению с другими похожими на DataFrame структурами, которые вам могли встречаться раньше (например, data. frame в языке R), операции со строками и столбцами в DataFrame в первом приближении симметричны. Внутри объекта данные хранятся в виле одного или нескольких двумерных блоков, а не в виде списка, словаря или еще какой-нибудь коллекции одномерных массивов. Технические детали внутреннего устройства DataFrame выходят за рамки этой книги.

Хотя в DataFrame данные хранятся в двумерном формате, в виде таблицы, нетрудно представить и данные более высокой размерности, если воспользоваться иерархическим индексированием. Эту тему мы обсудим в следующем разделе, она лежит в основе многих продвинутых механизмов обработки данных в pandas.

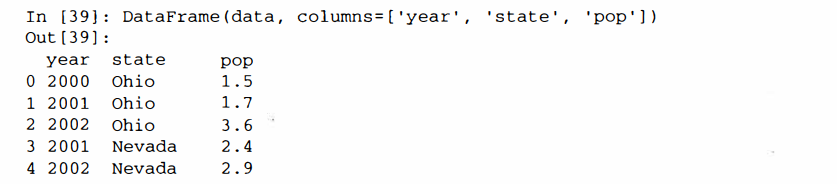
Есть много способов сконструировать объект DataFrame, один из самых распространенных - па основе словаря списков одинаковой длины или массивов NumPy:



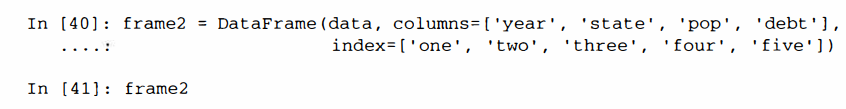
Для получившегося DataFrame автоматически будет построен индекс, как и в случае Series, и столбцы расположатся по порядку:

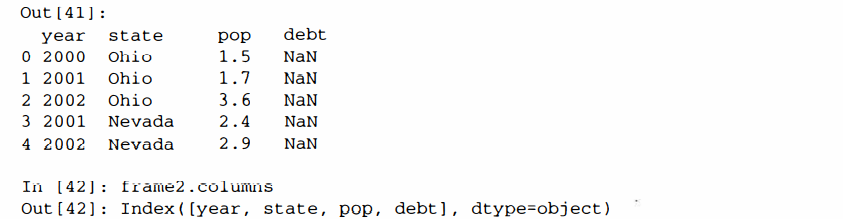


Если задать последовательность столбцов, то столбцы DataFrame расположатся строго в указанном порядке

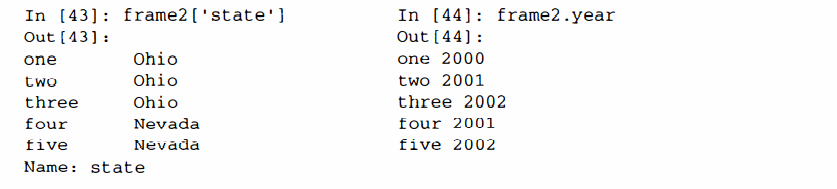


Как и в случае Series, если запросить столбец, которого нет в data, то он будет заполнен значениями NaN:





Столбец DataFrame можно извлечь как объект Series, воспользовавшись нотацией словарей, или с помощью атрибута:

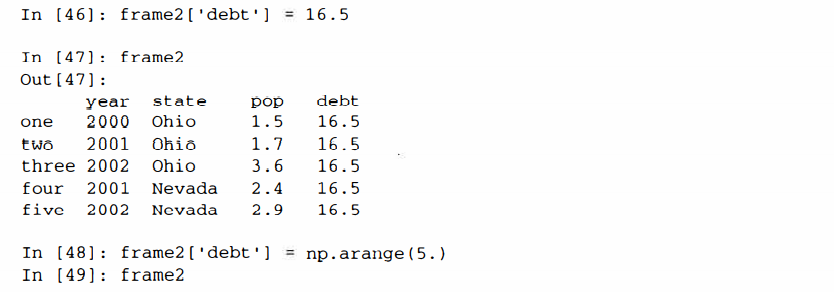


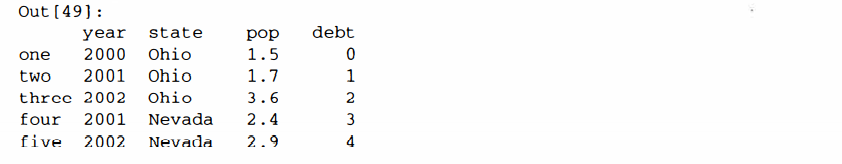
Отметим, что возвращенный объект Series имеет тот же индекс, что и DataFrame, а его атрибут name установлен соответствующим образом.

Строки также можно извлечь по позиции или по имени, для чего есть два метода, один из них - ix с указанием индексного поля (подробнее об этом ниже):

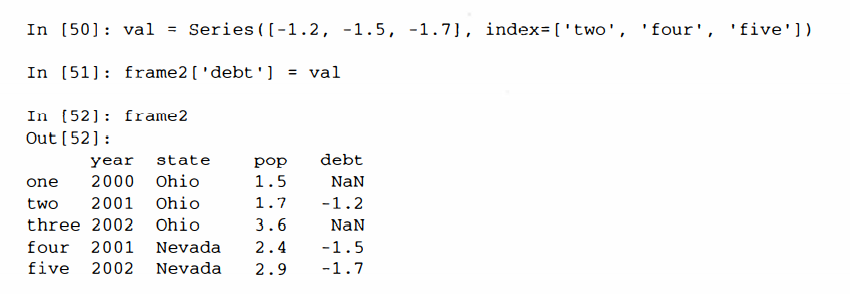


Столбцы можно модифицировать путем присваивания. Например, пустому столбцу «debt» можно было бы присвоить скалярное значение или массив значений:

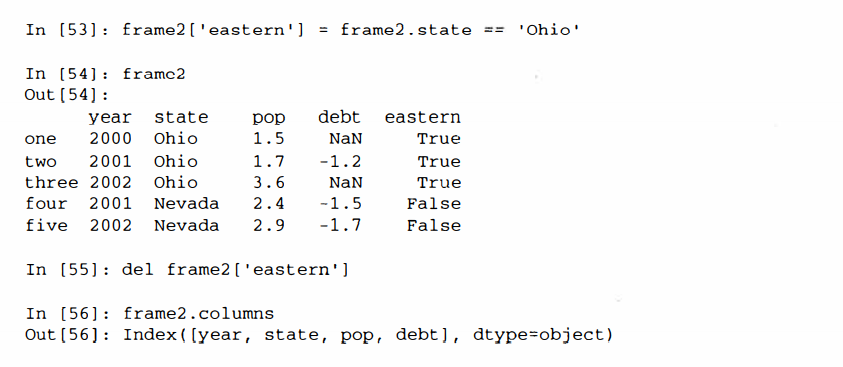


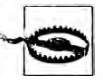


Когда столбцу присваивается список или массив, длина значения должна совпадать с длиной DataFrame. Если же присваивается объект Series, то он будет точно согласован с индексом DataFrame, а в «дырки» будут вставлены значения NA:

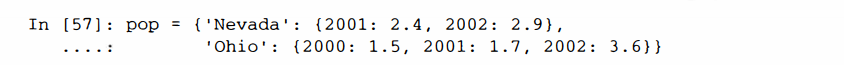


Присваивание несуществующему столбцу приводит к созданию нового столбца. Для удаления столбцов служит ключевое слово del, как и в обычном словаре:

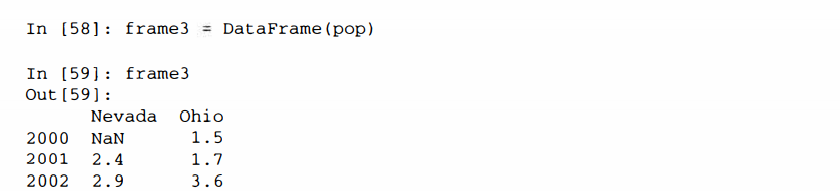


Столбец, возвращенный в ответ на запрос к DataFrame по индексу, является представлением, а не копией данных. Следовательно, любые модификации этого объекта Series, найдут отражение в DataFrame. Чтобы скопировать столбец, нужно вызвать метод сору объекта Serie.

Еще одна распространенная форма данных - словарь словарей:



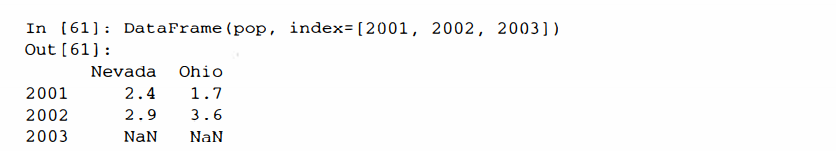
Если передать его конструктору DataFrarne, то ключи внешнего словаря будут интерпретированы как столбцы, а ключи внутреннего словаря - как индексы строк:



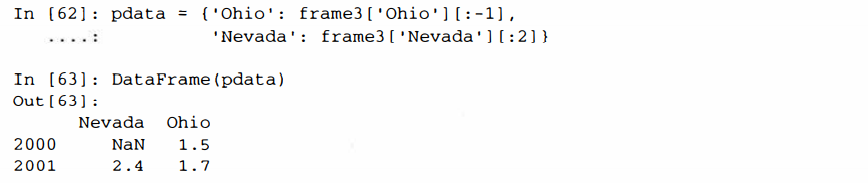
Разумеется, результат можно транспонировать:



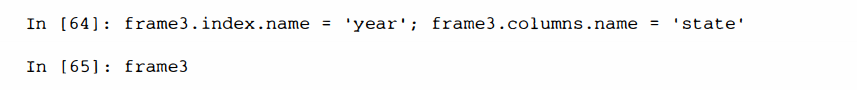
Ключи внутренних словарей объединяются и сортируются для образования индекса результата. Однако этого не происходит, если индекс задан явно:

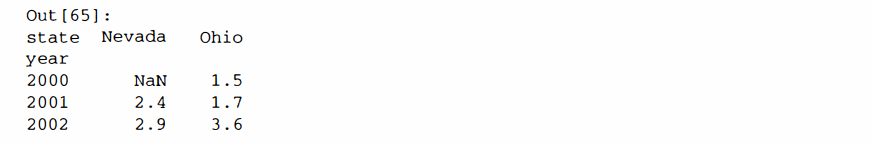


Словари объектов Series интерпретируются очень похоже:



Полный перечень возможных аргументов конструктора DataFrarne приведен в табл. 5.1. Если у объектов, возвращаемых при обращении к атрибутам index и colurnns объекта DataFraшe, установлен атрибут name, то он также выводится:

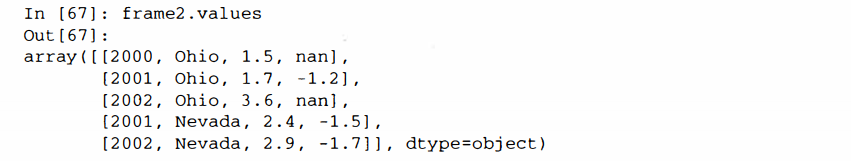




Как и в случае Series, атрибут values возвращает данные, хранящиеся в DataFraшe, в виде двумерного массива ndarray:



Если у столбцов DataFramc разные типы данных, то dtype массива values будет выбран так, чтобы охватить все столбцы:

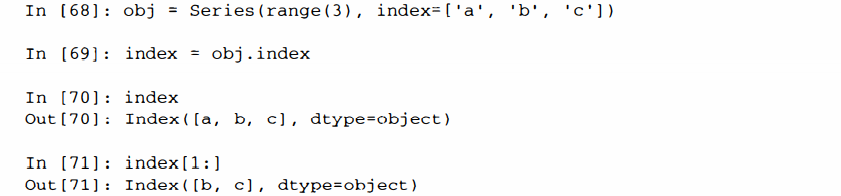


**Таблица 5.1.** Аргументы конструктора DataFrame

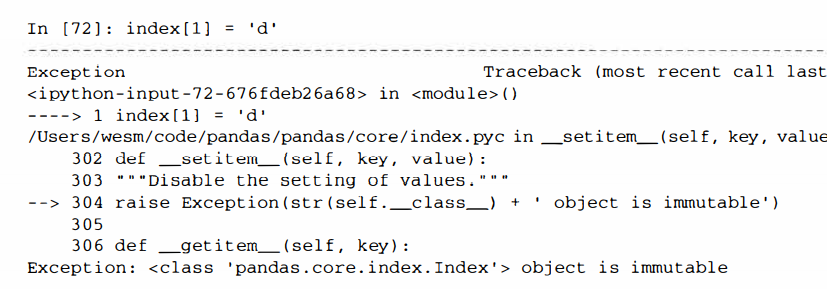
|  |  |
| --- | --- |
| **Тип** | **Примечания** |
| Двумерный ndarray | Матрица данных, дополнительно можно передать метки строк и столбцов |
| Словарь массивов, списков или кортежей | Каждая последовательность становится столбцом объекта DataFrame. Все последовательности должны быть одинаковой длины |
| Структурный массив NumPy | Интерпретируется так же, как «словарь массивов» |
| Словарь объектов Series | Каждое значение становится столбцом. Если индекс явно не задан, то индексы объектов Series объединяются и образуют индекс строк результата |
| Словарь словарей | Каждый внутренний словарь становится столбцом. Ключи объединяются и образуют индекс строк, как в случае «словаря объектов Series» |
| Список словарей или объектов Series | Каждый элемент списка становится строкой объекта DataFrame. Объединение ключей словаря или индексов объектов Series становится множеством меток столбцов DataFrame |
| Список списков или кортежей | Интерпретируется так же, как «двумерный ndarray» |
| Другой объект DataFrame | Используются индексы DataFrame, если явно не заданы другие индексы |
| Объект NumPy MaskedArray | Как «двумерный ndarray» с тем отличием, что замаскированные значения становятся отсутствующими в результирующем объекте DataFrame |

**Индексные объекты**

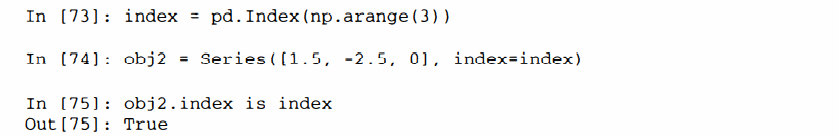
В индексных объектах pandas хранятся метки вдоль осей и прочие метаданные (например, имена осей). Любой массив или иная последовательность меток, указанная при конструировании Series или DataFrame, преобразуется в объект Index:



Индексные объекты неизменяемы, т. е. пользователь не может их модифицировать:



Неизменяемость важна для того, чтобы несколько структур данных могли совместно использовать один и тот же индексный объект, не опасаясь его повредить:



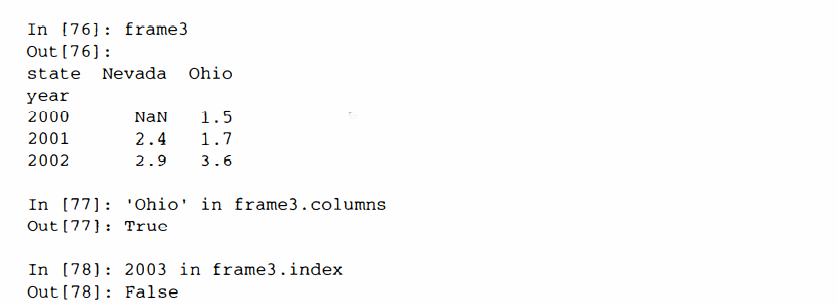
В табл. 5.2 перечислены включенные в библиотеку индексные классы. При некотором усилии можно даже создать подклассы класса Index, если требуется реализовать специализированную функциональность индексирования вдоль оси.

Многим пользователям подробная информация об индексных объектах не нужна, но они, тем не менее, являются важной частью модели данных pandas.

**Таблица 5.2**. Основные индексные объекты в pandas

|  |  |
| --- | --- |
| **Тип** | **Примечания** |
| Index | Наиболее общий индексный объект, представляющий оси в массиве NumPy, состоящем из объектов Python |
| Int64Index | Специализированный индекс для целых значений |
| Multiindex | «Иерархический» индекс, представляющий несколько уровней индексирования по одной оси. Можно считать аналогом массива кортежей |
| Datetimeindex | Хранит временные метки с наносекундной точностью (представлены типом данных NumPy datetime64) |
| Periodindex | Специализированный индекс для данных о периодах (временных промежутках) |

Индексный объект не только похож на массив, но и ведет себя как множество фиксированного размера:



У любого объекта Index есть ряд свойств и методов для ответа на типичные вопросы о хранящихся в нем данных. Они перечислены в табл. 5.3.

**Таблица 5.3**. Методы и свойства объекта lndex

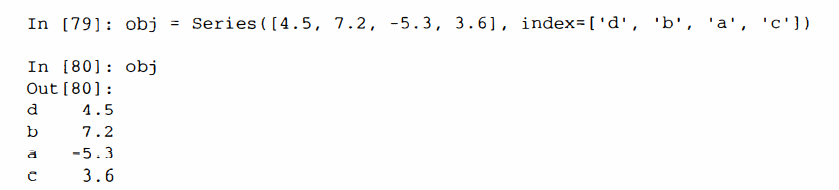
|  |  |
| --- | --- |
| **Метод** | **Описание** |
| append | Конкатенирует с дополнительными индексными объектами, порождая новый объект lndex |
| diff | Вычисляет теоретико-множественную разность, представляя ее в виде индексного объекта |
| intersection | Вычисляет теоретико-множественное пересечение |
| union | Вычисляет теоретико-множественное объединение |
| isin | Вычисляет булев массив, показывающий, содержится ли каждое значение индекса в переданной коллекции |
| delete | Вычисляет новый индексный объект, получающийся после удаления элемента с индексом i |
| drop | Вычисляет новый индексный объект, получающийся после удаления переданных значений |
| insert | Вычисляет новый индексный объект, получающийся после вставки элемента в позицию с индексом i |
| is\_rnonotonic | Возвращает True, если каждый элемент больше или равен предыдущему |
| is\_unique | Возвращает True, если в индексе нет повторяющихся значений |
| unique | Вычисляет массив уникальных значений в индексе |

**Базовая функциональность**

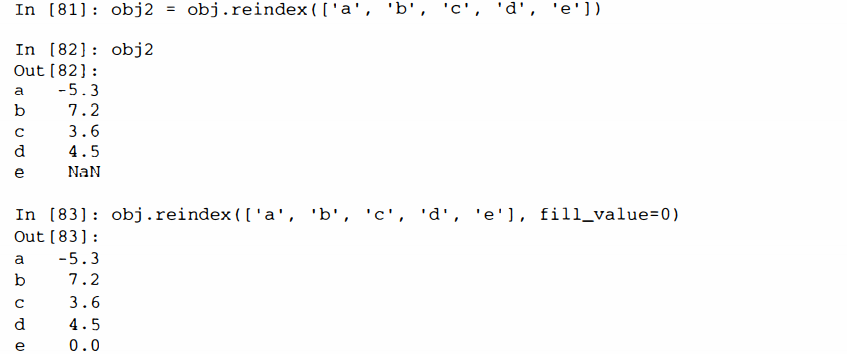
В этом разделе мы рассмотрим фундаментальные основы взаимодействия с данными, хранящимися в объектах Series и DataFraшe. В последующих главах мы более детально обсудим вопросы анализа и манипуляции данными с применением pandas. Эта книга не задумывалась как исчерпывающая документация по библиотеке pandas, я хотел лишь акцентировать внимание на наиболее важных чертах, оставив не столь употребительные (если не сказать эзотерические) вещи для самостоятельного изучения читателю.

**Переиндексация**

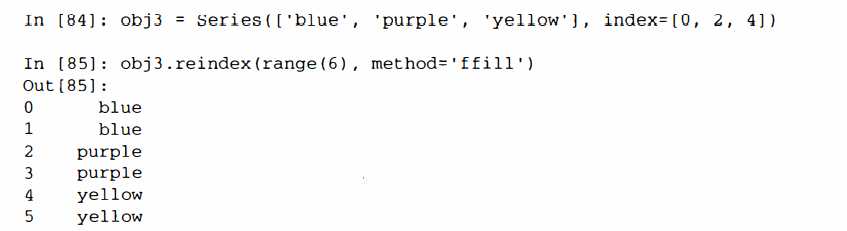
Для объектов pandas критически важен метод reindex, т. е. возможность создания нового объекта, данные в котором согласуются с новым индексом. Рассмотрим простой пример:



Если вызвать reindex для этого объекта Seгies, то данные будут реорганизованы в соответствии с новым индексом, а если каких-то из имеющихся в этом индексе значений раньше не было, то вместо них будут подставлены отсутствующие значения:



Для упорядоченных данных, например временных рядов, иногда желательно произвести интерполяцию, или восполнение отсутствующих значений в процессе переиндексации. Это позволяет сделать параметр method; так, если задать для него значение ffill, то будет произведено прямое восполнение значений:

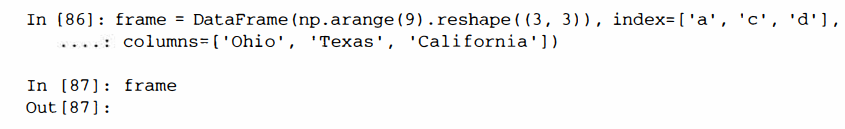


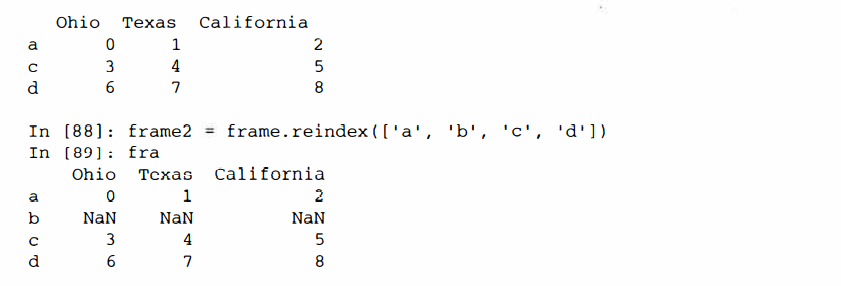
В табл. 5.4 перечислены возможные значения параметра method. В настоящее время интерполяцию, более сложную, чем прямое и обратное восполнение, нужно производить постфактум.

Таблица 5.4. Значение параметра method функции reindex (интерполяция)

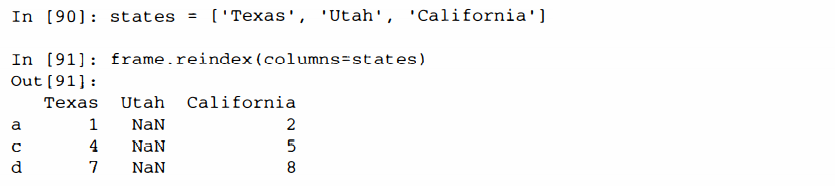
|  |  |
| --- | --- |
| **Значение** | **Описание** |
| ffill или pad | Восполнить (или перенести) значения в прямом направлении |
| Ьfill или backfill | Восполнить (или перенести) значения в обратном направлении |

В случае объекта DataFrame функция reindex может изменять строки, столбцы или то и другое. Если ей передать просто последовательность, то в результирующем объекте переиндексируются строки:

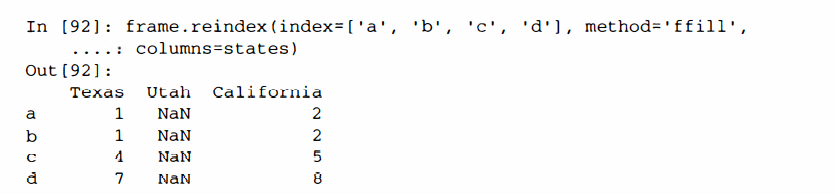




Столбцы можно переиндексировать, задав ключевое слово columns:



Строки и столбцы можно переиндексировать за одну операцию, хотя интерполяция будет применена только к строкам (к оси О):

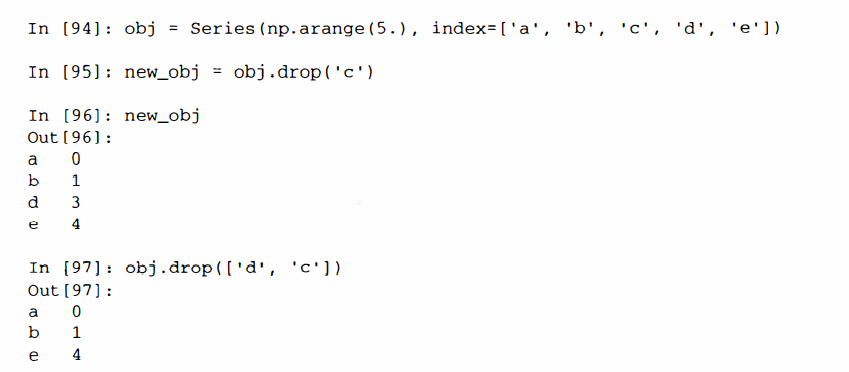


**Таблица 5.5**. Аргументы функции reindex

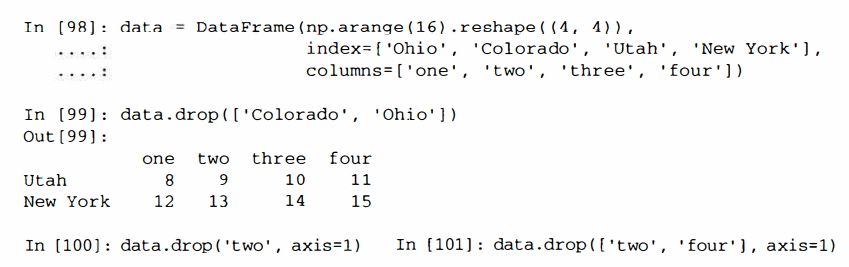
|  |  |
| --- | --- |
| **Аргумент** | **Описание** |
| index | Последовательность, которая должна стать новым индексом. Может быть экземпляром Index или любой другой структурой данных Python, похожей на последовательность. Экземпляр Index будет использован «как есть», без копирования |
| method | Метод интерполяции (восполнения), возможные значения приведены в табл. 5.4 |
| fill\_value | Значение, которой должно подставляться вместо отсутствующих значений, появляющихся в результате переиндексации |
| limit | При прямом или обратном восполнении максимальная длина восполняемой лакуны |
| level | Сопоставить с простым объектом Index на указанном уровне Multiindex, иначе выбрать подмножество |
| сору | Не копировать данные, если новый индекс эквивалентен старому. По умолчанию True (т. е. всегда копировать данные) |

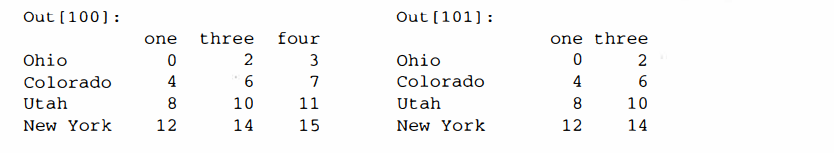
**Удаление элементов из оси**

Удалить один или несколько элементов из оси просто, если имеется индексный массив или список, не содержащий этих значений. Метод drop возвращает новый объект, в котором указанные значения удалены из оси:



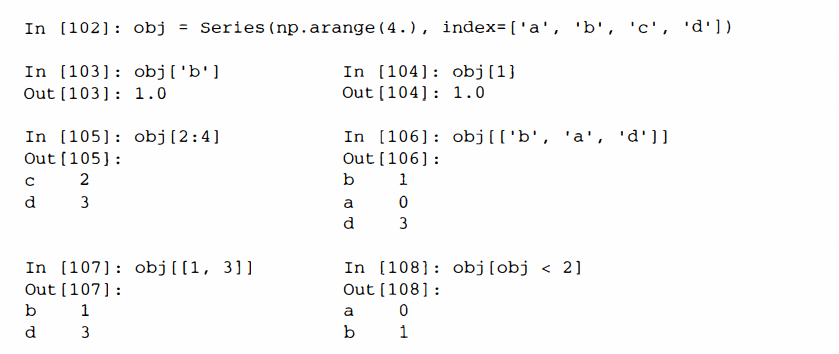
В случае DataFrame указанные в индексе значения можно удалить из любой оси:





**Доступ по индексу, выборка и фильтрация**

Доступ по индексу к объекту Series ( obj [ ... J) работает так же, как для массивов NшnPy с тем отличием, что индексами могут быть не только целые, но любые значения из индекса объекта Series. Вот несколько примеров:



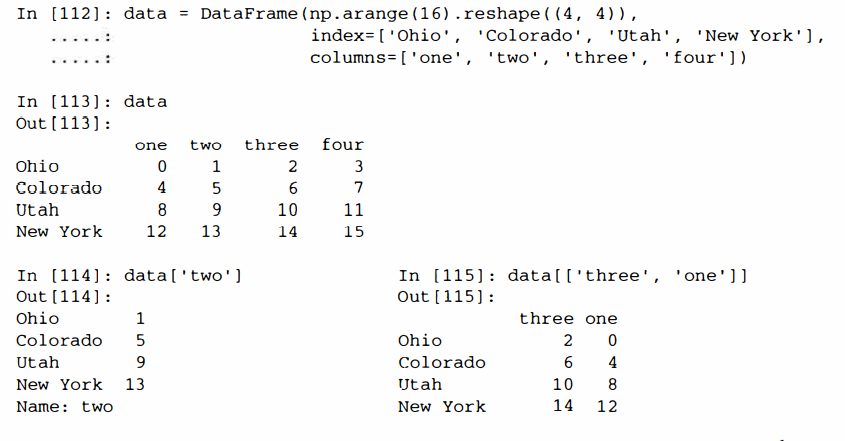
Вырезание с помощью меток отличается от обычного вырезания в Python тем, что конечная точка включается:



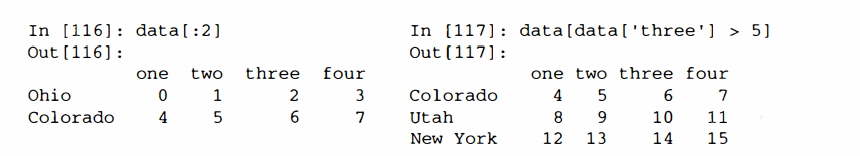
Установка с помощью этих методов работает ожидаемым образом:



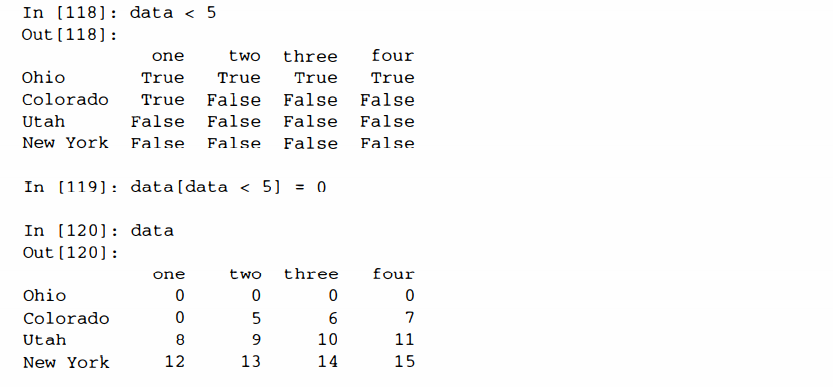
Как мы уже видели, доступ по индексу к DataFrame применяется для извлечения одного или нескольких столбцов путем задания единственного значения или последовательности:



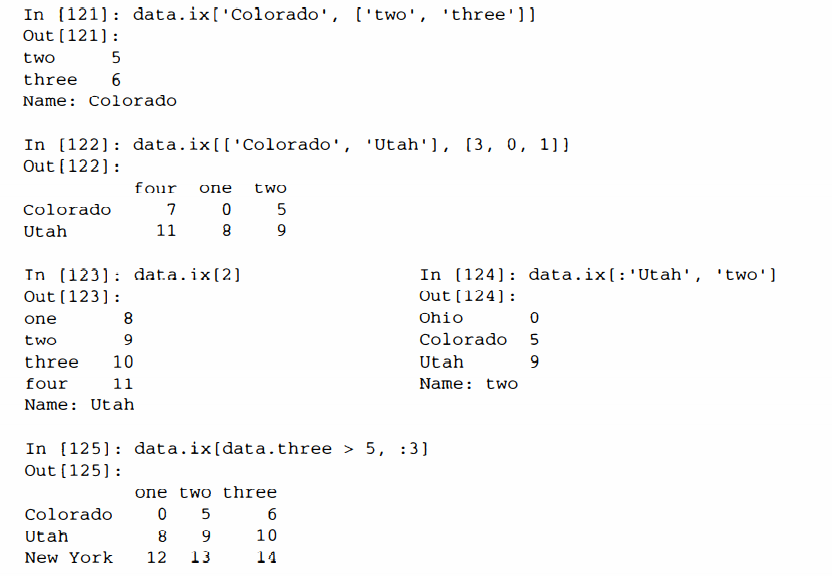
У доступа по индексу есть несколько частных случаев. Во-первых, выборка строк с помощью вырезания или булева массива:



Некоторым читателям такой синтаксис может показаться непоследовательным, по он был выбран исключительно из практических соображений. Еще одна возможность - доступ по индексу с помощью булева DataFramc, например порожденного в результате скалярного сравнения:



Идея в том, чтобы сделать DataFraшe синтаксически более похожим на ndarray в данном частном случае. Для доступа к строкам по индексу с помощью меток я ввел специальное индексное поле ix. Оно позволяет выбрать подмножество строк и столбцов DataFrame с применение нотации NumPy, дополненной метками осей. Как я уже говорил, это еще и более лаконичный способ выполнить переиндексацию:



Таким образом, существует много способов выборки и реорганизации данных, содержащихся в объекте pandas. Для DataFrame краткая сводка многих их них приведена в табл. 5.6. Позже мы увидим, что при работе с иерархическими индексами есть ряд дополнительных возможностей

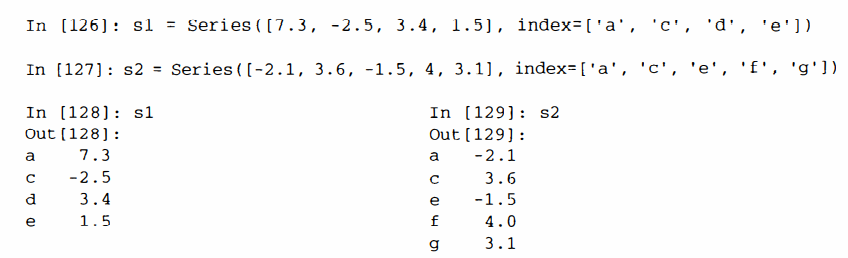
**Таблица 5.6**. Варианты доступа по индексу для объекта DataFrame

|  |  |
| --- | --- |
| **Вариант** | **Примечание** |
| obj [val] | Выбрать один столбец или последовательность столбцов из DataFrame. Частные случаи: булев массив (фильтрация строк), срез (вырезание строк) или булев DataFrame (установка значений в позициях, удовлетворяющих некоторому критерию) |
| obj.ix (val] | Выбрать одну строку или подмножество строк из DataFrame |
| obj.ix[:, val] | Выбрать один столбец или подмножество столбцов |
| obj.ix[vall, val2] | Выбрать строки и столбцы |
| метод reindex | Привести одну или несколько осей в соответствие с новыми индексами |
| Метод хs | Выбрать одну строку или столбец по метке и вернуть объект Series |
| методы icol, irow | Выбрать одну строку или столбец соответственно по целочисленному номеру и вернуть объект Series |
| методы get\_value, set\_value | Выбрать одно значение по меткам строки и столбца |

Проектируя pandas, я подспудно ощущал, что нотация frarile [: , col] для выборки столбца слишком громоздкая (и провоцирующая ошибки), поскольку выборка столбца - одна из самых часто встречающихся операций. Поэтому я пошел на компромисс и перенес все более выразительные операции доступа с помощью индексов-меток в ix.

**Арифметические операции и выравнивание данных**

Одна из самых важных черт pandas - поведение арифметических операций для объектов с разными индексами. Если при сложении двух объектов обнаруживаются различные пары индексов, то результирующий индекс будет объединением индексов. Рассмотрим простой пример:

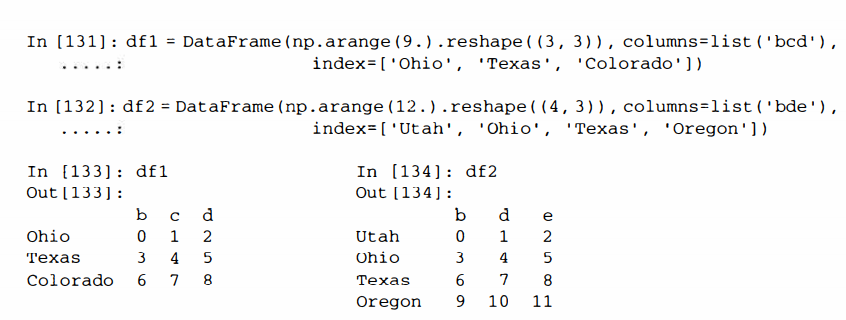


Сложение этих объектов дает:

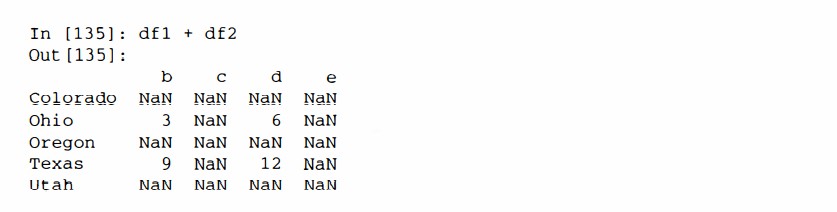


Вследствие внутреннего выравнивания данных образуются отсутствующие значения в позициях, для которых не нашлось соответственной пары. Отсутствующие значения распространяются на последующие операции.

В случае DataFrame выравнивание производится как для строк, так и для столбцов:

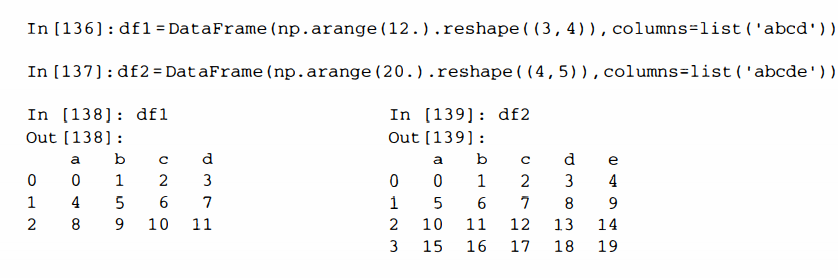


При сложении этих объектов получается DataFrame, индекс и столбцы которого являются объединениями индексов и столбцов слагаемых:



**Восполнение значений в арифметических методах**

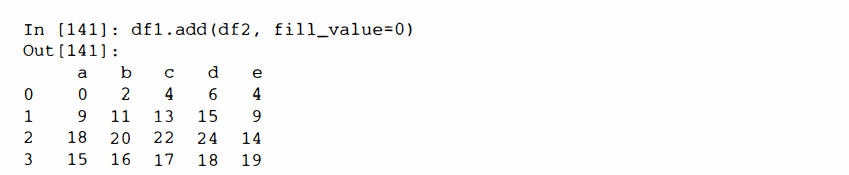
При выполнении арифметических операций с объектами, проиндексированными по-разному, иногда желательно поместить специальное значение, например 0, в позиции операнда, которым в другом операнде соответствует отсутствующая позиция:



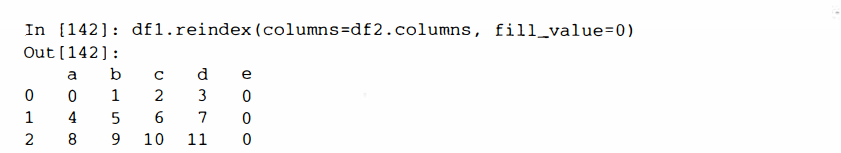
Сложение этих объектов порождает отсутствующие значения в позициях, которые имеются не в обоих операндах:



Теперь я вызову метод add объекта dfl и передам ему объект df2 и значение параметра fill\_value:



Точно так же, выполняя переиндексацию объекта Serics или DataFrame, можно указать восполняемое значение:

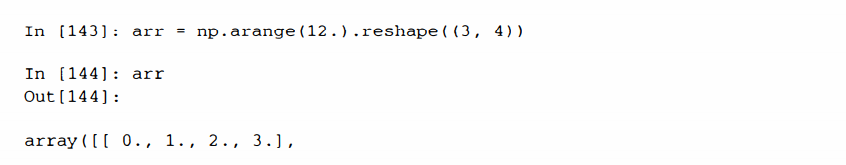


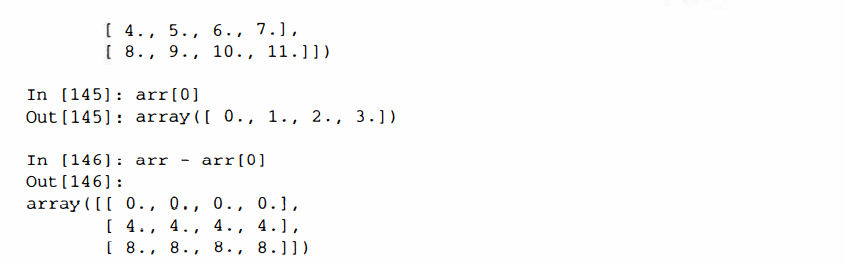
**Таблица 5.7.** Гибкие арифметические методы

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| add | Сложение (+) |
| sub | Вычитание (-) |
| div | Деление (/) |
| mul | Умножение (\*) |

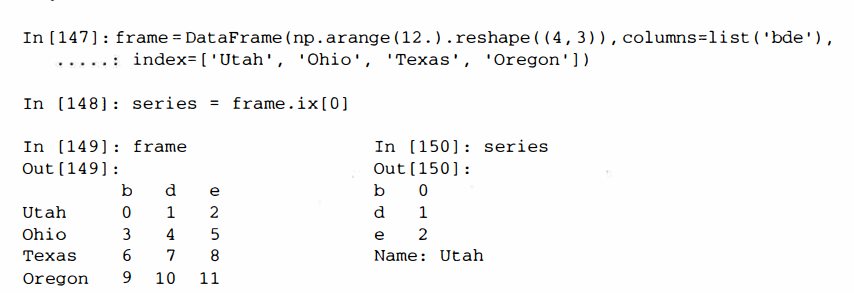
**Операции между DataFrame и Series**

Как и в случае массивов NumPy, арифметические операции между DataFrame и Series корректно определены. В качестве пояснительного примера рассмотрим вычисление разности между двумерным массивом и одной из его строк:





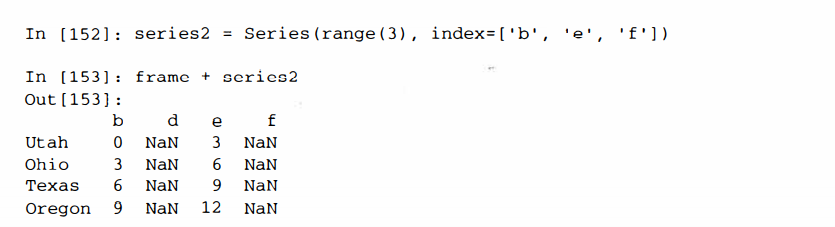
Это называется укладыванием и подробно объясняется в главе 12. Операции между DataFrarne и Series аналогичны:



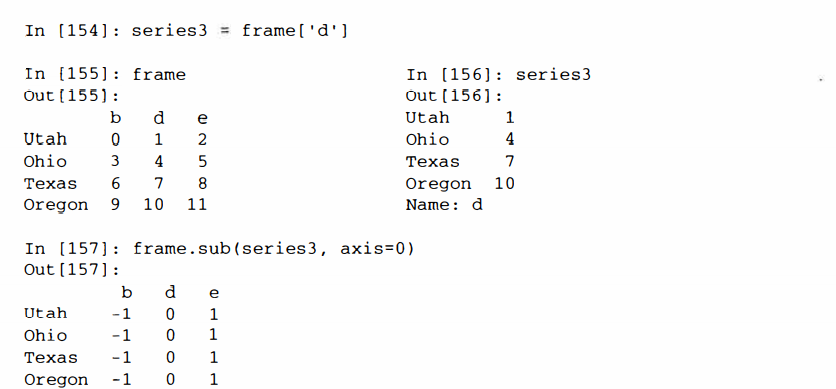
По умолчанию при выполнении арифметических операций между DataFrarne и Series индекс Series сопоставляется со столбцами DataFrarne, а укладываются строки:



Если какой-нибудь индекс не найден либо в столбцах DataFraшe, либо в индексе Series, то объекты переиндексируются для образования объединения:



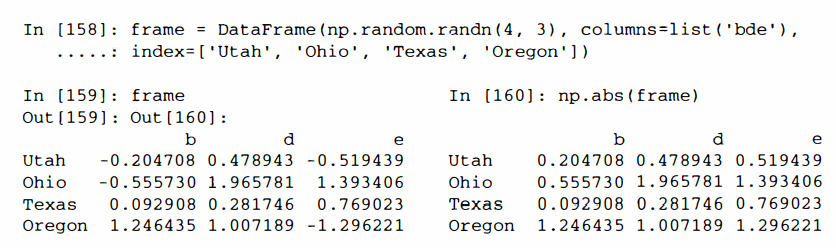
Если вы хотите вместо этого сопоставлять строки, а укладывать столбцы, то должны будете воспользоваться каким-нибудь арифметическим методом. Например:



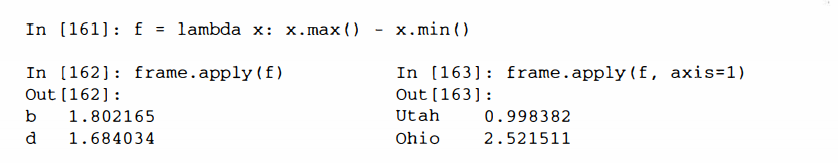
Передаваемый номер оси — это ось, вдоль которой производится сопоставление. В данном случае мы хотим сопоставлять с индексом строк DataFrame и укладывать поперек.

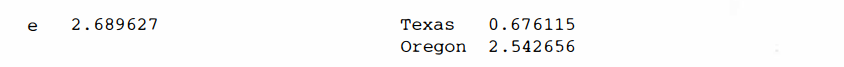
**Применение функций и отображение**

Универсальные функции NumPy (поэлементные методы массивов) отлично работают и с объектами pandas:

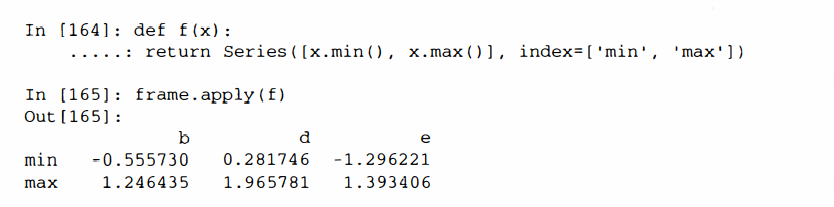


Еще одна часто встречающаяся операция - применение функции, определенной для одномерных массивов, к каждому столбцу или строке. Именно это и делает метод apply объекта DataFrame:

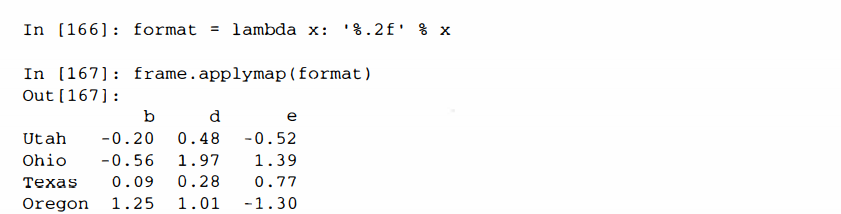




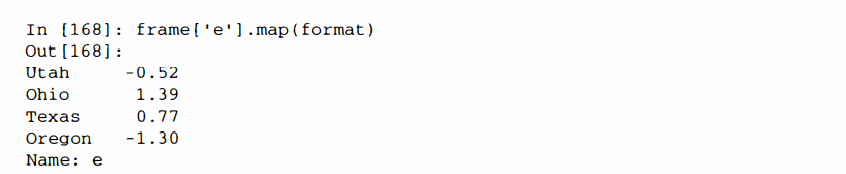
Многие из наиболее распространенных статистик массивов (например, surn и rnean) - методы DataFrame, поэтому применять apply в этом случае необязательно. Функция, передаваемая методу apply, не обязана возвращать скалярное значение, она может вернуть и объект Series, содержащий несколько значений:



Можно использовать и поэлементные функции Python. Допустим, требуется вычислить форматированную строку для каждого элемента frame с плавающей точкой. Это позволяет сделать метод applymap:

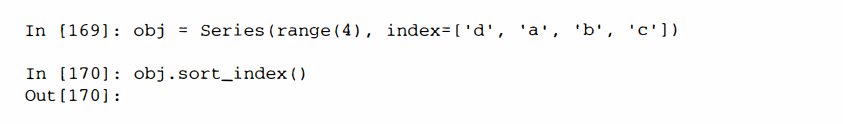


Этот метод называется applymap, потому что в классе Se1·ies есть метод map для применения функции к каждому элементу:



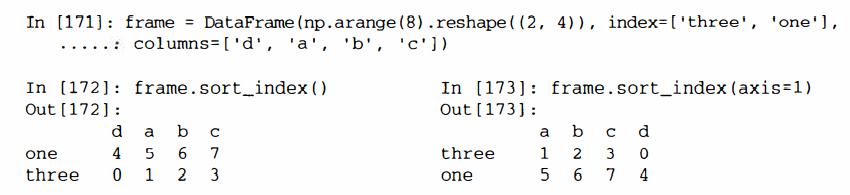
**Сортировка и ранжирование**

Сортировка набора данных по некоторому критерию - еще одна важная встроенная операция. Для лексикографической сортировки по индексу служит метод sort\_index, который возвращает новый отсортированный объект:

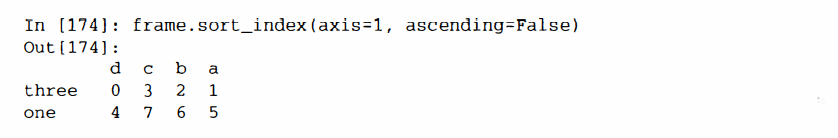




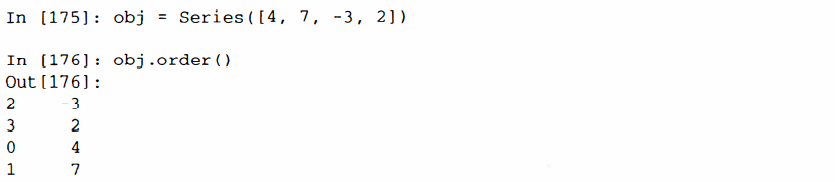
В случае DataFrame можно сортировать по индексу, ассоциированному с любой осью:



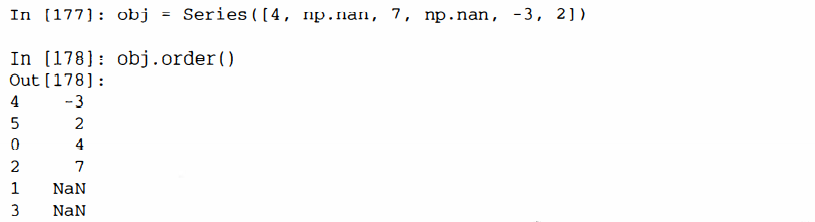
По умолчанию данные сортируются в порядке возрастания, но можно отсортировать их и в порядке убывания:



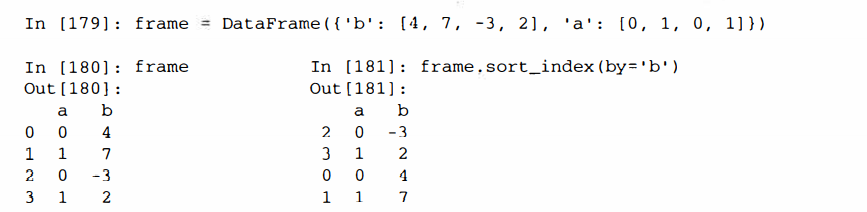
Для сортировки Series по значениям служит метод order:



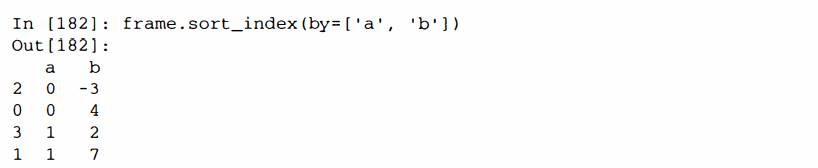
Отсутствующие значения по умолчанию оказываются в конце Series:



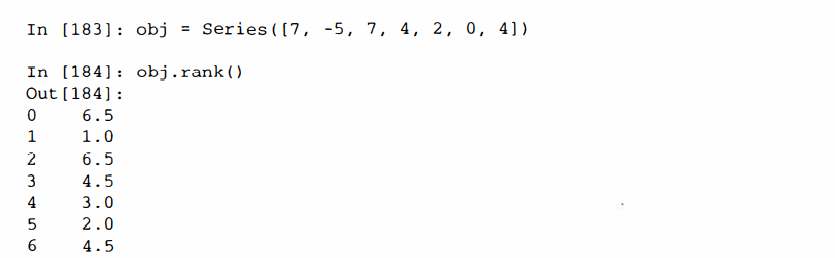
Объект Dataframe можно сортировать по значениям в одном или нескольких столбцах. Для этого имена столбцов следует передать в качестве значения параметра Bу:



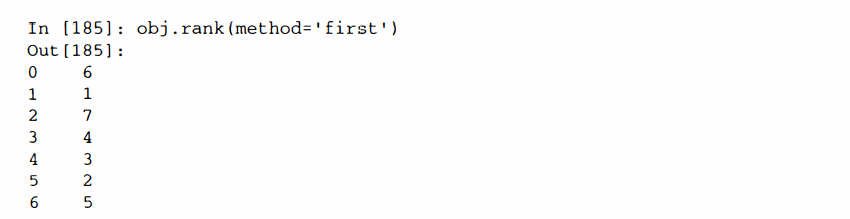
Для сортировки по нескольким столбцам следует передать список имен:



Ранжирование тесно связано с сортировкой, заключается оно в присваивании рангов - от единицы до числа присутствующих в массиве элементов. Это аналогично косвенным индексам, порождаемым методом numpy. argsort, с тем отличием, что существует правило обработки связанных рангов. Для ранжирования применяется метод rank объектов Series и DataFrame; по умолчанию rank обрабатывает связанные ранги, присваивая каждой группе средний ранг:



Ранги можно также присваивать u соответствии с порядком появления в данных:



Естественно, можно ранжировать и в порядке убывания: