количестве кода» (http://mng.bz/7jo7). Библиотека pandas полностью соответствует всем этим характеристикам и представляет собой следующий этап обработки информации, прекрасно подходящий для специалистов по ее анализу, желающих улучшить свои навыки программирования с помощью обладающего большими возможностями современного набора инструментов анализа данных.

1.3. ОБЗОР БИБЛИОТЕКИ PANDAS

Лучше всего открывать для себя возможности библиотеки pandas на практике. Пройдемся по возможностям библиотеки на примере анализа набора данных 700 самых кассовых фильмов всех времен. Мне кажется, вы будете приятно удивлены, насколько интуитивно понятен синтаксис pandas даже для программистов-новичков.

По мере чтения изложенного ниже материала этой главы старайтесь не слишком вникать в примеры кода; вам даже не обязательно их копировать. Наша цель сейчас — взглянуть на укрупненную картину возможностей и функциональности библиотеки pandas. Думайте о том, на *что* способна эта библиотека; позднее мы рассмотрим подробнее, *как это возможно*.

Для написания кода в этой книге в качестве среды программирования используются блокноты Jupyter. Если вам нужна помощь в настройке pandas и блокнотов Jupyter на вашем компьютере — загляните в приложение A. Скачать все наборы данных и готовые блокноты Jupyter можно по адресу https://www.github.com/paskhaver/pandas-in-action.

1.3.1. Импорт набора данных

Приступим! Во-первых, создадим новый блокнот Jupyter в том же каталоге, что и файл `movies.csv`, затем импортируем библиотеку pandas для доступа ко всем ее возможностям:

```
In  [1] import pandas as pd
```

Поле слева от кода (с номером `[1]` в предыдущем примере) отражает порядок выполнения ячеек, отсчитываемый от запуска или перезапуска блокнота Jupyter. Можно выполнять ячейки в произвольном порядке и даже выполнять одну ячейку несколько раз.

Советую вам в ходе чтения данной книги экспериментировать, выполняя различные фрагменты кода в своих Jupyter. Так что не обращайте внимания, если счетчики выполнения у вас будут отличаться от приведенных в тексте.

Наши данные хранятся в одном файле `movies.csv`. Файлы CSV (comma-separated values — значения, отделенные друг от друга запятыми) представляют собой файлы с открытым текстом, в которых строки данных разделяются символом переноса строки, а значения внутри строк — запятыми. Первая строка файла содержит названия столбцов данных. Вот первые три строки нашего файла:

```
Rank,Title,Studio,Gross,Year
1,Avengers: Endgame,Buena Vista,"$2,796.30",2019
2,Avatar,Fox,"$2,789.70",2009
```

Первая строка перечисляет названия пяти столбцов в наборе данных: `Rank` (Позиция), `Title` (Название), `Studio` (Студия), `Gross` (Кассовые сборы) и `Year` (Год). Во второй строке содержится первая запись, то есть данные для первого фильма. Позиция у этого фильма — `1`, название — `"Avengers: Endgame"`, студия — `"Buena Vista"`, кассовые сборы — `"$2,796.30"`, а год — `2019`. В следующей строке содержатся значения для следующего фильма и т. д. во всех 750 с лишним строках файла.

Библиотека pandas может импортировать разнообразные типы файлов, каждому из которых соответствует функция импорта на верхнем уровне библиотеки. *Функция* в библиотеке pandas эквивалентна функции в Excel и представляет собой команду, которую мы отдаем библиотеке или какой-либо сущности из нее. В данном сценарии мы воспользовались для импорта файла `movies.csv` функцией `read_csv`:

```
In  [2] pd.read_csv("movies.csv")

Out [2]
```

	Rank	Title	Studio	Gross	Year
0	1	Avengers: Endgame	Buena Vista	$2,796.30	2019
1	2	Avatar	Fox	$2,789.70	2009
2	3	Titanic	Paramount	$2,187.50	1997
3	4	Star Wars: The Force Awakens	Buena Vista	$2,068.20	2015
4	5	Avengers: Infinity War	Buena Vista	$2,048.40	2018
…	…	…	…	…	…
777	778	Yogi Bear	Warner Brothers	$201.60	2010
778	779	Garfield: The Movie	Fox	$200.80	2004
779	780	Cats & Dogs	Warner Brothers	$200.70	2001
780	781	The Hunt for Red October	Paramount	$200.50	1990
781	782	Valkyrie	MGM	$200.30	2008

```
782 rows × 5 columns
```

Библиотека pandas импортирует содержимое CSV-файла в объект `DataFrame`, своего рода контейнер для хранения данных. Различные объекты оптимизируются

для различных типов данных, и взаимодействовать с ними тоже надо по-разному. Библиотека pandas использует один тип объектов (`DataFrame`) для хранения наборов данных с несколькими столбцами и другой тип (`Series`) для хранения наборов данных из одного столбца. `DataFrame` можно сравнить с многостолбцовой таблицей в Excel.

Чтобы не загромождать экран, pandas отображает только первые и последние пять строк `DataFrame`[1]. Пропущенные данные отмечаются строкой с многоточием (`...`).

Наш `DataFrame` состоит из пяти столбцов (`Rank`, `Title`, `Studio`, `Gross`, `Year`) и индекса. Индекс — это столбец слева от `DataFrame`, он содержит числа, расположенные в порядке возрастания. Метки индекса служат идентификаторами строк данных. Индексом `DataFrame` может служить любой столбец. Если не указать библиотеке pandas явным образом, какой столбец использовать, она сгенерирует числовой индекс, начинающийся с нуля.

Какой столбец подходит в качестве индекса? Тот, значения из которого могут выступать в роли первичных идентификаторов строк (ссылок на строки). Из наших пяти столбцов на эту роль лучше всего подходят `Rank` и `Title`. Заменим сгенерированный автоматически числовой индекс на значения из столбца `Title`. Сделать это можно непосредственно во время импорта CSV:

```
In  [3] pd.read_csv("movies.csv", index_col = "Title")

Out [3]
```

Title	Rank	Studio	Gross	Year
Avengers: Endgame	1	Buena Vista	$2,796.30	2019
Avatar	2	Fox	$2,789.70	2009
Titanic	3	Paramount	$2,187.50	1997
Star Wars: The Force Awakens	4	Buena Vista	$2,068.20	2015
Avengers: Infinity War	5	Buena Vista	$2,048.40	2018
…	…	…	…	…
Yogi Bear	778	Warner Brothers	$201.60	2010
Garfield: The Movie	779	Fox	$200.80	2004
Cats & Dogs	780	Warner Brothers	$200.70	2001
The Hunt for Red October	781	Paramount	$200.50	1990
Valkyrie	782	MGM	$200.30	2008

```
782 rows × 4 columns
```

[1] Количество отображаемых библиотекой pandas строк DataFrame можно настраивать при помощи функции set_option(). Например, чтобы отображать десять первых и десять последних строк, выполните команду: pd.set_option('display.max_rows', 20). В книге настройка количества отображаемых строк упоминается и описывается в нескольких разделах. — *Примеч. пер.*

Далее присваиваем этот DataFrame переменной movies, чтобы можно было ссылаться на него в других местах программы. *Переменная* — назначаемое пользователем имя какого-либо объекта в программе:

```
In  [4] movies = pd.read_csv("movies.csv", index_col = "Title")
```

Узнать больше о переменных вы можете из приложения Б.

1.3.2. Операции над объектами DataFrame

Объект DataFrame можно рассматривать с множества различных ракурсов. Можно, например, извлечь несколько строк из его начала:

```
In  [5] movies.head(4)

Out [5]
```

Title	Rank	Studio	Gross	Year
Avengers: Endgame	1	Buena Vista	$2,796.30	2019
Avatar	2	Fox	$2,789.70	2009
Titanic	3	Paramount	$2,187.50	1997
Star Wars: The Force Awakens	4	Buena Vista	$2,068.20	2015

Или, наоборот, заглянуть в конец набора данных:

```
In  [6] movies.tail(6)

Out [6]
```

Title	Rank	Studio	Gross	Year
21 Jump Street	777	Sony	$201.60	2012
Yogi Bear	778	Warner Brothers	$201.60	2010
Garfield: The Movie	779	Fox	$200.80	2004
Cats & Dogs	780	Warner Brothers	$200.70	2001
The Hunt for Red October	781	Paramount	$200.50	1990
Valkyrie	782	MGM	$200.30	2008

Можно узнать, сколько строк содержит объект DataFrame:

```
In  [7] len(movies)

Out [7] 782
```

Можно также запросить у pandas число строк и столбцов в DataFrame. Наш текущий набор данных содержит 782 строки и 4 столбца.

```
In  [8] movies.shape

Out [8] (782, 4)
```

Можно выяснить общее количество ячеек:

```
In  [9] movies.size

Out [9] 3128
```

Можно запросить типы данных наших четырех столбцов. В следующих ниже выведенных результатах `int64` означает целочисленный столбец, а `object` — текстовый столбец:

```
In  [10] movies.dtypes

Out [10]

Rank       int64
Studio    object
Gross     object
Year       int64
dtype: object
```

Можно извлечь строку из набора данных по ее порядковому номеру, то есть индексу. В большинстве языков программирования отсчет индексов начинается с 0. Следовательно, если нужно извлечь 500-й фильм в наборе данных, надо выбирать строку с индексом 499:

```
In  [11] movies.iloc[499]

Out [11] Rank           500
         Studio         Fox
         Gross      $288.30
         Year          2018
         Name: Maze Runner: The Death Cure, dtype: object
```

Pandas вернула тут новый объект `Series` — одномерный маркированный массив значений, нечто вроде столбца данных с идентификаторами для каждой строки. Обратите внимание, что метки объекта `Series` (`Rank`, `Studio`, `Gross`, `Year`) соответствуют четырем столбцам из объекта `DataFrame` `movies`. Таким образом, pandas изменила представление значений исходной строки.

Можно также воспользоваться меткой индекса для обращения к строке объекта `DataFrame`. Напомним, что индекс нашего `DataFrame` содержит названия фильмов. Извлечем значения для строки любимой всеми мелодрамы *Forrest Gump*. В следующем примере строка извлекается по метке индекса, а не числовой позиции:

```
In  [12] movies.loc["Forrest Gump"]

Out [12] Rank            119
         Studio    Paramount
```

```
Gross       $677.90
Year           1994
Name: Forrest Gump, dtype: object
```

Метки индекса могут повторяться. Например, два фильма из нашего `DataFrame` называются `"101 Dalmatians"` (оригинальный, 1961 года, и ремейк 1996-го):

```
In  [13] movies.loc["101 Dalmatians"]

Out [13]
```

Title	Rank	Studio	Gross	Year
101 Dalmatians	425	Buena Vista	$320.70	1996
101 Dalmatians	708	Buena Vista	$215.90	1961

И хотя библиотека pandas допускает повторы, я рекомендую стремиться к уникальным меткам индекса. Уникальный набор меток позволяет pandas быстрее находить и извлекать конкретные строки.

Фильмы в рассматриваемом CSV-файле отсортированы по столбцу `Rank`. Как быть, если необходимо получить пять наиболее свежих фильмов? Можно отсортировать `DataFrame` по значениям другого столбца, в данном случае — `Year`:

```
In  [14] movies.sort_values(by = "Year", ascending = False).head()

Out [14]
```

Title	Rank	Studio	Gross	Year
Avengers: Endgame	1	Buena Vista	2796.3	2019
John Wick: Chapter 3 - Parab...	458	Lionsgate	304.7	2019
The Wandering Earth	114	China Film Corporation	699.8	2019
Toy Story 4	198	Buena Vista	519.8	2019
How to Train Your Dragon: Th...	199	Universal	519.8	2019

Можно также сортировать объекты `DataFrame` по значениям из нескольких столбцов. Отсортируем `movies` сначала по столбцу `Studio`, а затем — по столбцу `Year`. И получим фильмы, отсортированные в алфавитном порядке по студии и году выхода:

```
In  [15] movies.sort_values(by = ["Studio", "Year"]).head()

Out [15]
```

Title	Rank	Studio	Gross	Year
The Blair Witch Project	588	Artisan	$248.60	1999
101 Dalmatians	708	Buena Vista	$215.90	1961
The Jungle Book	755	Buena Vista	$205.80	1967
Who Framed Roger Rabbit	410	Buena Vista	$329.80	1988
Dead Poets Society	636	Buena Vista	$235.90	1989

Можно также отсортировать индекс для отображения фильмов в алфавитном порядке названий:

```
In  [16] movies.sort_index().head()

Out [16]

Title             Rank           Studio    Gross  Year
-----------------------------------------------------
10,000 B.C.        536  Warner Brothers  $269.80  2008
101 Dalmatians     708      Buena Vista  $215.90  1961
101 Dalmatians     425      Buena Vista  $320.70  1996
2 Fast 2 Furious   632        Universal  $236.40  2003
2012                93             Sony  $769.70  2009
```

Все выполненные нами операции возвращали *новые*, созданные ими самими объекты `DataFrame`. Библиотека pandas не меняла исходный объект `movies` из CSV-файла. То, что эти операции не разрушают данные, очень удобно и служит стимулом для экспериментов. Всегда можно убедиться, что результат правильный, прежде чем его фиксировать в качестве окончательного.

1.3.3. Подсчет значений в Series

Попробуем более сложный вариант анализа. Пусть необходимо выяснить, у какой киностудии больше всего кассовых фильмов. Для решения этой задачи нужно подсчитать, сколько раз каждая студия встречается в столбце `Studio`.

Можно извлечь отдельный столбец данных из `DataFrame` в виде `Series`. Обратите внимание, что библиотека pandas сохраняет индекс `DataFrame`, то есть названия фильмов, в полученном объекте `Series`:

```
In  [17] movies["Studio"]

Out [17] Title
         Avengers: Endgame                  Buena Vista
         Avatar                                 Fox
         Titanic                              Paramount
         Star Wars: The Force Awakens       Buena Vista
         Avengers: Infinity War             Buena Vista
                                              ...
         Yogi Bear                       Warner Brothers
         Garfield: The Movie                    Fox
         Cats & Dogs                     Warner Brothers
         The Hunt for Red October             Paramount
         Valkyrie                               MGM
         Name: Studio, Length: 782, dtype: object
```

При большом количестве строк в `Series` pandas сокращает набор данных и отображает только пять первых и пять последних строк.

Мы отделили столбец `Studio` и можем теперь подсчитать число вхождений каждого из уникальных значений кинокомпаний. Ограничим выборку десятью наиболее успешными киностудиями:

```
In  [18] movies["Studio"].value_counts().head(10)

Out [18] Warner Brothers    132
         Buena Vista        125
         Fox                117
         Universal          109
         Sony                86
         Paramount           76
         Dreamworks          27
         Lionsgate           21
         New Line            16
         MGM                 11
         Name: Studio, dtype: int64
```

Приведенное выше возвращаемое значение — еще один объект `Series`! В нем библиотека pandas использует студии из столбца `Studio` в качестве меток индекса, а соответствующие им значения количества фильмов — как значения `Series`.

1.3.4. Фильтрация столбца по одному или нескольким критериям

Нередко бывает необходимо извлечь подмножество строк на основе одного или нескольких критериев. Для этой цели в Excel служит инструмент Фильтр.

Допустим, что нам нужно найти фильмы, выпущенные Universal Studios. Решить эту задачу в pandas можно с помощью одной строки кода:

```
In  [19] movies[movies["Studio"] == "Universal"]

Out [19]
```

Title	Rank	Studio	Gross	Year
Jurassic World	6	Universal	$1,671.70	2015
Furious 7	8	Universal	$1,516.00	2015
Jurassic World: Fallen Kingdom	13	Universal	$1,309.50	2018
The Fate of the Furious	17	Universal	$1,236.00	2017
Minions	19	Universal	$1,159.40	2015
…	…	…	…	…
The Break-Up	763	Universal	$205.00	2006
Everest	766	Universal	$203.40	2015
Patch Adams	772	Universal	$202.30	1998
Kindergarten Cop	775	Universal	$202.00	1990
Straight Outta Compton	776	Universal	$201.60	2015

```
109 rows × 4 columns
```

Можно также присвоить условие фильтрации переменной, чтобы читатели кода понимали контекст:

```
In  [20] released_by_universal = (movies["Studio"] == "Universal")
         movies[released_by_universal].head()

Out [20]

Title                            Rank     Studio      Gross  Year
-------------------------------------------------------------------
Jurassic World                      6  Universal  $1,671.70  2015
Furious 7                           8  Universal  $1,516.00  2015
Jurassic World: Fallen Kingdom     13  Universal  $1,309.50  2018
The Fate of the Furious            17  Universal  $1,236.00  2017
Minions                            19  Universal  $1,159.40  2015
```

Есть возможность фильтровать строки `DataFrame` и по нескольким критериям. В примере ниже мы найдем все фильмы, выпущенные Universal Studios в 2015 году:

```
In  [21] released_by_universal = movies["Studio"] == "Universal"
         released_in_2015 = movies["Year"] == 2015
         movies[released_by_universal & released_in_2015]

Out [21]
Title                   Rank     Studio       Gross  Year
-----------------------------------------------------------
Jurassic World             6  Universal  $1,671.70  2015
Furious 7                  8  Universal  $1,516.00  2015
Minions                   19  Universal  $1,159.40  2015
Fifty Shades of Grey     165  Universal    $571.00  2015
Pitch Perfect 2          504  Universal    $287.50  2015
Ted 2                    702  Universal    $216.70  2015
Everest                  766  Universal    $203.40  2015
Straight Outta Compton   776  Universal    $201.60  2015
```

Рассмотренный пример включает строки, удовлетворяющие обоим условиям. Мы также можем осуществить выборку фильмов, удовлетворяющих хотя бы одному из этих условий: выпущенные Universal Studios *или* выпущенные в 2015 году. В итоге получается более длинный объект `DataFrame`, поскольку вероятность удовлетворить только одному условию выше, чем сразу двум:

```
In  [22] released_by_universal = movies["Studio"] == "Universal"
         released_in_2015 = movies["Year"] == 2015
         movies[released_by_universal | released_in_2015]

Out [22]

Title                            Rank       Studio      Gross  Year
-------------------------------------------------------------------
Star Wars: The Force Awakens        4  Buena Vista  $2,068.20  2015
Jurassic World                      6    Universal  $1,671.70  2015
```

```
Furious 7                            8    Universal  $1,516.00  2015
Avengers: Age of Ultron              9  Buena Vista  $1,405.40  2015
Jurassic World: Fallen Kingdom      13    Universal  $1,309.50  2018
                                  …     …            …          …     …
The Break-Up                       763    Universal    $205.00  2006
Everest                            766    Universal    $203.40  2015
Patch Adams                        772    Universal    $202.30  1998
Kindergarten Cop                   775    Universal    $202.00  1990
Straight Outta Compton             776    Universal    $201.60  2015

140 rows × 4 columns
```

Библиотека pandas позволяет фильтровать `DataFrame` и другими способами. Можно выбрать, скажем, значения столбцов больше или меньше конкретного значения. В следующем примере мы выбираем фильмы, выпущенные до 1975 года:

```
In  [23] before_1975 = movies["Year"] < 1975
         movies[before_1975]

Out [23]

Title               Rank           Studio    Gross   Year
-----------------------------------------------------------
The Exorcist         252  Warner Brothers  $441.30   1973
Gone with the Wind   288              MGM  $402.40   1939
Bambi                540              RKO  $267.40   1942
The Godfather        604        Paramount  $245.10   1972
101 Dalmatians       708      Buena Vista  $215.90   1961
The Jungle Book      755      Buena Vista  $205.80   1967
```

Можно также задать диапазон для значений. Например, извлечь фильмы, выпущенные с 1983 по 1986 год:

```
In  [24] mid_80s = movies["Year"].between(1983, 1986)
         movies[mid_80s]

Out [24]

Title                                 Rank     Studio     Gross   Year
-------------------------------------------------------------------------
Return of the Jedi                     222        Fox  $475.10   1983
Back to the Future                     311  Universal  $381.10   1985
Top Gun                                357  Paramount  $356.80   1986
Indiana Jones and the Temple of Doom   403  Paramount  $333.10   1984
Crocodile Dundee                       413  Paramount  $328.20   1986
Beverly Hills Cop                      432  Paramount  $316.40   1984
Rocky IV                               467        MGM  $300.50   1985
Rambo: First Blood Part II             469    TriStar  $300.40   1985
Ghostbusters                           485   Columbia  $295.20   1984
Out of Africa                          662  Universal  $227.50   1985
```

Можно воспользоваться для фильтрации строк индексом `DataFrame`. Например, код ниже преобразует все названия фильмов в индексе в нижний регистр и находит фильмы, в названии которых содержится слово `dark`:

```
In  [25] has_dark_in_title = movies.index.str.lower().str.contains("dark")
         movies[has_dark_in_title]

Out [25]
```

Title	Rank	Studio	Gross	Year
transformers: dark of the moon	23	Paramount	$1,123.80	2011
the dark knight rises	27	Warner Brothers	$1,084.90	2012
the dark knight	39	Warner Brothers	$1,004.90	2008
thor: the dark world	132	Buena Vista	$644.60	2013
star trek into darkness	232	Paramount	$467.40	2013
fifty shades darker	309	Universal	$381.50	2017
dark shadows	600	Warner Brothers	$245.50	2012
dark phoenix	603	Fox	$245.10	2019

Обратите внимание, что библиотека pandas находит все фильмы, в названиях которых содержится слово `dark`, вне зависимости от того, в каком именно месте (позиции) в названии оно встречается.

1.3.5. Группировка данных

Следующая задача — самая сложная из встретившихся нам к данному моменту. Нас интересует, у какой студии самые высокие кассовые сборы из всех. Давайте агрегируем значения из столбца `Gross` по студии.

Первая проблема: значения в столбце `Gross` хранятся в виде текста, а не чисел. Библиотека pandas импортирует значения столбца как текст, чтобы сохранить символы доллара и запятые из исходного CSV-файла. Можно преобразовать значения столбца в десятичные числа, но для этого необходимо удалить оба упомянутых символа. В примере ниже мы заменяем все вхождения `"$"` и `","` пустыми строками. Данная операция аналогична функции Найти и заменить в Excel:

```
In  [26] movies["Gross"].str.replace(
             "$", "", regex = False
         ).str.replace(",", "", regex = False)

Out [26] Title
         Avengers: Endgame               2796.30
         Avatar                          2789.70
         Titanic                         2187.50
         Star Wars: The Force Awakens    2068.20
         Avengers: Infinity War          2048.40
                                            ...
         Yogi Bear                        201.60
```

```
         Garfield: The Movie              200.80
         Cats & Dogs                      200.70
         The Hunt for Red October         200.50
         Valkyrie                         200.30
         Name: Gross, Length: 782, dtype: object
```

Удалив символы, мы можем преобразовать значения столбца `Gross` из текста в числа с плавающей точкой:

```
In  [27] (
             movies["Gross"]
            .str.replace("$", "", regex = False)
            .str.replace(",", "", regex = False)
            .astype(float)
         )

Out [27] Title
         Avengers: Endgame               2796.3
         Avatar                          2789.7
         Titanic                         2187.5
         Star Wars: The Force Awakens    2068.2
         Avengers: Infinity War          2048.4
                                          ...
         Yogi Bear                        201.6
         Garfield: The Movie              200.8
         Cats & Dogs                      200.7
         The Hunt for Red October         200.5
         Valkyrie                         200.3
         Name: Gross, Length: 782, dtype: float64
```

Опять же эти операции временные и не модифицируют исходного `Series` `Gross`. Во всех предыдущих примерах библиотека pandas создавала копию исходной структуры данных, выполняла операцию и возвращала новый объект. В следующем примере мы явным образом заменяем столбец `Gross` в `movies` новым столбцом, содержащим числа с плавающей точкой. Теперь преобразование зафиксировано в наборе как результат всех выполненных операций:

```
In  [28] movies["Gross"] = (
             movies["Gross"]
             .str.replace("$", "", regex = False)
             .str.replace(",", "", regex = False)
             .astype(float)
         )
```

Наше преобразование типа данных открывает возможности для других вычислений и операций. В следующем примере вычисляются средние кассовые сборы по всем фильмам:

```
In  [29] movies["Gross"].mean()

Out [29] 439.0308184143222
```

Возвращаемся к изначальной задаче — вычислению кассовых сборов, агрегированных по студиям. В первую очередь необходимо идентифицировать студии и разбить на подмножества фильмы (строки), относящиеся к каждой из них. Этот процесс называется *группировкой*. Приведенный ниже код служит для группирования строк объекта `DataFrame` по значениям из столбца `Studio`:

```
In  [30] studios = movies.groupby("Studio")
```

Можно попросить pandas подсчитать количество фильмов каждой из студий:

```
In  [31] studios["Gross"].count().head()

Out [31] Studio
         Artisan                     1
         Buena Vista               125
         CL                          1
         China Film Corporation      1
         Columbia                    5
         Name: Gross, dtype: int64
```

Приведенные выше результаты отсортированы по названию студии. Можно вместо этого отсортировать `Series` по количеству фильмов в порядке убывания:

```
In  [32] studios["Gross"].count().sort_values(ascending = False).head()

Out [32] Studio
         Warner Brothers    132
         Buena Vista        125
         Fox                117
         Universal          109
         Sony                86
         Name: Gross, dtype: int64
```

Далее просуммируем значения из столбца `Gross` по студиям. Pandas распознает подмножество фильмов, относящихся к каждой из студий, извлекает из строки соответствующие значения столбца `Gross` и суммирует их:

```
In  [33] studios["Gross"].sum().head()

Out [33] Studio
         Artisan                     248.6
         Buena Vista               73585.0
         CL                          228.1
         China Film Corporation      699.8
         Columbia                   1276.6
         Name: Gross, dtype: float64
```

Опять же библиотека pandas сортирует результаты по названию студии. Мы хотим найти студии с самыми высокими кассовыми сборами, так что отсортируем

значения `Series` в порядке убывания сборов. Вот пять студий с наибольшими кассовыми сборами:

```
In  [34] studios["Gross"].sum().sort_values(ascending = False).head()

Out [34] Studio
         Buena Vista        73585.0
         Warner Brothers    58643.8
         Fox                50420.8
         Universal          44302.3
         Sony               32822.5
         Name: Gross, dtype: float64
```

С помощью всего нескольких строк кода мы смогли извлечь из этого непростого набора данных немало интересной информации. Например, у студии Warner Brothers в этом списке больше фильмов, чем у Buena Vista, зато совокупные кассовые сборы у Buena Vista выше. А значит, средние кассовые сборы фильмов студии Buena Vista выше, чем у фильмов студии Warner Brothers.

Мы затронули лишь верхушку айсберга возможностей библиотеки pandas. Надеюсь, эти примеры показали все множество разнообразных способов преобразования данных и операций над ними в этой замечательной библиотеке. Нам предстоит обсудить все коды из этой главы намного подробнее в последующих главах книги. А в главе 2 займемся одним из основных «кирпичиков» библиотеки pandas: объектом `Series`.

РЕЗЮМЕ

- Pandas — библиотека для анализа данных, основанная на языке программирования Python.
- Pandas превосходно справляется со сложными операциями над большими наборами данных и отличается сжатым синтаксисом.
- В качестве альтернатив использованию библиотеки pandas можно рассматривать визуальное приложение электронных таблиц Excel, статистический язык программирования R и комплект программ SAS.
- Программирование требует от специалиста иного набора навыков, чем работа с Excel или «Google Таблицами».
- Pandas способна импортировать множество разнообразных форматов файлов. Один из популярных форматов — CSV, в нем строки разделяются разрывами строк, а значения внутри строк — запятыми.
- `DataFrame` — основная структура данных библиотеки pandas. По существу, она представляет собой таблицу данных с несколькими столбцами.