Содержание

Предисловие

> В отличие от стереотипного подвенечного стиля, оно было – если использовать технический термин – элегантным, как компьютерный алгоритм, который всего несколькими строками исходного кода достигает впечатляющего результата.
>
> *– Грэм Симсион, «Проект "Рози"»*

Добро пожаловать в книгу «*Элегантный SciPy*». Мы собираемся провести довольно много времени, сосредоточившись на той части заголовка книги, которая относится к «SciPy», поэтому давайте воспользуемся моментом, чтобы поразмышлять над словом «элегантный». Существует масса руководств, учебных пособий и веб-сайтов документации, которые дают всестороннее описание пакета SciPy. Книга «*Элегантный SciPy*» идет дальше. Она представляет собой нечто большее, чем просто обучение приемам написания по-настоящему рабочего программного кода. Мы вдохновим вас на написание программного кода, который будет по-настоящему потрясающим!

В романе «*Проект "Рози"*» (между прочим, уморительная книга; когда закончите читать «*Элегантный SciPy*», обязательно прочтите в Википедии[1] материал, описывающий предысторию ее написания) Грэм Симсион переиначивает общепринятые нормы, связанные со словом «элегантный». Большинство людей использует это слово для описания визуальной простоты, стиля и изящества, например, первого мобильника iPhone. Вместо этого герой Грэма Симсиона, Дон Тиллман, дает *определение* элегантности, используя термин «компьютерный алгоритм». Мы надеемся, что после прочтения этой книги вы получите ясное понимание того, что он имеет в виду, и что впредь, занимаясь чтением или написанием куска элегантного программного кода, вы будете ощущать умиротворение в лучах его красоты и изящества. (Возьмите на заметку: авторы могут быть подвержены гиперболе.)

Хороший фрагмент программного кода вызывает ощущение удовлетворенности. Когда вы на него смотрите, его замысел *ясен*, он нередко *краток* (но не настолько краток, чтобы быть туманным), и он *эффективен* при выполнении практической задачи. Для авторов удовольствие от анализа элегантного программного кода лежит в скрытых внутри него уроках и в том, как он вдохновляет нас быть *творческими* в подходах к новым алгоритмическим задачам.

Как ни странно, креативность, помимо всего прочего, может также заставлять нас умничать за счет читателя и писать маловразумительный про-

---

[1] См. https://en.wikipedia.org/wiki/The_Rosie_Project и https://ru.wikipedia.org/wiki/Проект_«Рози».

граммный код, который очень трудно понять. PEP8[1] (руководство по стилю программирования на Python) и PEP20[2] (дзэн языка Python) нам напоминают, что «программный код читается намного больше раз, чем пишется» и поэтому «читаемость имеет значение».

Краткость элегантного кода обеспечивается *не* за счет упаковки кучи вложенных вызовов функций, а за счет абстракции и рационального использования функций. Она требует одной-двух минут на то, чтобы вникнуть, но в конечном счете обеспечивает вам четкий момент понимания. Как только вы начнете разбираться в различных компонентах программного кода, его правильность должна стать очевидной. Этому способствуют ясные имена переменных и функций и тщательно продуманные комментарии, которые программный код *объясняют*, а не просто его *описывают*.

Инженер-программист Дж. Брэдфорд Хиппс (J. Bradford Hipps) недавно в газете «*Нью-Йорк таймс*»[3] заявил: «для того чтобы писать хороший программный код, следует почитать Вирджинию Вульф»:

> *На практике разработка программного обеспечения имеет гораздо более творческий характер, чем алгоритмический.*
> *Разработчик обращается к своему редактору исходного кода таким же образом, как писатель к чистому листку бумаги. […] Разработчика и писателя также могут отличать общее здоровое нетерпение по отношению к тому, как все «делалось всегда», и воспроизводящееся стремление нарушать общепринятые правила. Когда модуль закончен или страницы завершены, их качество оценивается в сопоставлении со многими из тех же самых стандартов: элегантностью, краткостью, целостностью; обнаружением симметрий там, где их существование ни разу не было замечено. И даже красотой.*

Это именно та позиция, которую мы примем в этой книге.

Теперь, когда мы рассмотрели «элегантную» часть заголовка, давайте возвратимся к «SciPy».

В зависимости от контекста «SciPy» может означать пакет программного обеспечения, экосистему или сообщество разработчиков. Отчасти великолепие SciPy обусловлено тем, что он имеет превосходную онлайновую документацию[4] и учебные пособия[5], поэтому предлагать читателю очередной справочник было бы бессмысленно. Вместо этого в книге «*Элегантный SciPy*» будет представлен самый лучший программный код, который был создан при помощи SciPy.

---

[1] См. https://www.python.org/dev/peps/pep-0008/.

[2] См. https://www.python.org/dev/peps/pep-0020/.

[3] См. https://www.nytimes.com/2016/05/22/opinion/sunday/to-write-software-read-novels.html.

[4] См. https://docs.scipy.org/.

[5] См. http://www.scipy-lectures.org/.

Отобранный нами программный код подчеркивает умное, элегантное использование расширенного функционала NumPy, SciPy и других связанных с ними библиотек. Начинающий читатель научится применять эти библиотеки к реальным задачам, используя красивый программный код. При этом в обоснование наших примеров мы используем реальные научные данные.

Как и сам SciPy, мы хотели построить книгу «*Элегантный SciPy*» на основе сообщества разработчиков. Многие приводимые в книге примеры были взяты из рабочего программного кода, обнаруженного в обширной экосистеме научного программирования на Python, и отобраны для демонстрации кратко очерченных выше принципов элегантного программного кода.

Для кого эта книга предназначена?

Цель книги «*Элегантный SciPy*» – побудить вас поднять свои навыки программирования на языке Python на более высокий уровень. Вы изучите пакет SciPy на примере самого лучшего программного кода.

Прежде чем приступить к работе, вы должны иметь представление о языке Python и знать, что такое переменные, функции, циклы, и, возможно, немного разбираться в библиотеке NumPy. Хотя вполне может быть, что вы даже отточили свои навыки программирования на Python на основе такого продвинутого материала, как, например, «Python. К вершинам мастерства» (Fluent Python)[1]. Если это к вам не относится, то, прежде чем продолжить чтение этой книги, вам следует начать с каких-нибудь учебных пособий по Python для начинающих, таких как Software Carpentry[2].

Но, возможно, для вас «стек SciPy» – все равно, что пункт меню сети экспресс-блинных IHOP, и вы чувствуете себя не в своей тарелке, когда речь идет о его применении на практике. Или, возможно, вы являетесь ученым, который прочитал в Сети несколько учебных пособий по Python и скачал несколько аналитических сценариев из другой лаборатории или у другого сотрудника вашей собственной лаборатории и повозился с ними некоторое время. И, возможно, думаете, что в той или иной степени одиноки в своей попытке научиться программировать SciPy. Вы ошибаетесь.

По ходу изложения мы научим вас использовать Интернет в качестве своего справочника. И мы будем направлять вас к спискам рассылок, хранилищам и конференциям, где вы встретите аналогично мыслящих ученых, которые в своем опыте работы находятся немного дальше, чем вы.

Это такая книга, которую вы прочтете один раз, но будете к ней возвращаться в дальнейшем в поисках вдохновения (и, возможно, чтобы еще раз восхититься некоторыми элегантными фрагментами программного кода!).

---

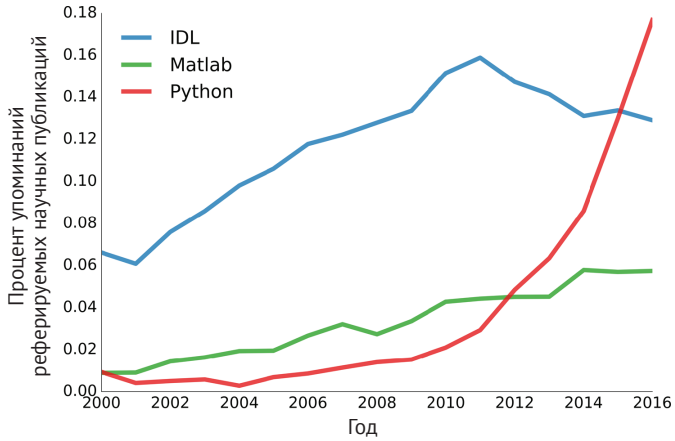
[1] См. http://shop.oreilly.com/product/0636920032519.do.

[2] См. http://software-carpentry.org/.

Почему именно SciPy?

Библиотеки NumPy и SciPy составляют ядро научной экосистемы языка Python. В программной библиотеке SciPy реализован набор функций для обработки научных данных из таких областей, как статистика, обработка сигналов, обработка изображений и математическая оптимизация. Библиотека SciPy надстроена поверх библиотеки NumP, которая предназначена для вычислительной обработки числовых массивов. За последние несколько лет вся экосистема приложений и библиотек продемонстрировала существенный рост, опираясь как раз на NumPy и SciPy, с охватом широкого спектра дисциплин, который среди прочих включает астрономию, биологию, метеорологию, климатологию и материаловедение.

И этот рост не проявляет никаких признаков к ослаблению. В 2014 г. Томас Робитэйл и Крис Бомон (Thomas Robitaille и Chris Beaumont) задокументировали[1] рост применения языка Python в астрономии. Вот то, что мы обнаружили, когда обновили[2] их график во второй половине 2016 г.:



Совершенно очевидно, что в течение многих последующих лет SciPy и связанные с ней библиотеки будут основными в подавляющей части аналитической обработки научных данных.

Еще одним убедительным примером служит тот факт, что организация Software Carpentry, которая обучает ученых вычислительным навыкам, используя для этого чаще всего язык Python, сегодня едва справляется со спросом на свои учебные курсы.

---

[1] См. https://nbviewer.jupyter.org/github/ChrisBeaumont/adass_proceedings/blob/master/Mining%20acknowledgments%20in%20ADS.ipynb.

[2] См. https://gist.github.com/jni/3339985a016572f178d3c2f18e27ec0d.

Что такое экосистема SciPy?

> SciPy (произносится как «Сай Пи») – это экосистема программного обеспечения с открытым исходным кодом на основе Python для математики, науки и техники.
>
> – http://www.scipy.org

Экосистема SciPy представляет собой нестрого определенную коллекцию пакетов Python. В книге «Элегантный SciPy» мы встретимся с большинством ее главных компонентов:

- **NumPy** – фундамент научных вычислений на Python. Эта библиотека обеспечивает эффективные числовые массивы и широкую поддержку численных вычислений, включая линейную алгебру, случайные числа и разложение в ряды Фурье. Уникальная особенность NumPy заключена в ее «N-мерных массивах», или массивах ndarray. Эти структуры данных эффективным образом хранят числовые величины и задают решетку в любом количестве размерностей (подробнее об этом чуть позже)[1];
- **SciPy**, как библиотека, является коллекцией эффективных численных алгоритмов для таких областей, как обработка сигналов, интеграция, оптимизация и статистика. Эти алгоритмы обернуты в легкие для использования интерфейсы[2];
- **Matplotlib** – мощный пакет для построения графиков в двух измерениях (и элементарных 3D-графиков). Он берет свое название от навеявшего его разработку синтаксиса Matlab[3];
- **IPython** – интерактивный интерфейс для языка Python, который позволяет оперативно взаимодействовать с вашими данными и тестировать свои идеи[4];
- блокнот **Jupyter** работает в вашем браузере и позволяет составлять документы с широкими функциональными возможностями, которые объединяют в себе программный код, текст, математические выражения и интерактивные элементы интерфейса[5]. На самом деле при подготовке настоящей книги текст программ был преобразован в блокноты Jupyter и выполнялся там (благодаря этому мы знаем, что все примеры выполняются правильно). Проект Jupyter, начавшийся как расширение IPython,

---

[1] См. http://www.numpy.org/.

[2] См. http://www.scipy.org/scipylib/index.html.

[3] См. http://matplotlib.org/.

[4] См. https://ipython.org/.

[5] См.: *Перес Ф.* «Грамотные вычисления» и вычислительная воспроизводимость: IPython в эпоху журналистики, ориентированной на данные (публикация в блоге). 19 апреля 2013 г. (*Fernando Perez*. 'Literate computing' and computational reproducibility: IPython in the age of data-driven journalism. http://blog.fperez.org/2013/04/literate-computing-and-computational.html).

теперь поддерживает многочисленные языки, включая Cython, Julia, R, Octave, Bash, Perl и Ruby[1];

- **pandas** обеспечивает быстрые столбчатые структуры данных в простом для применения пакете. Он в особенности подходит для работы с помеченными наборами данных, такими как таблицы или реляционные базы данных, и для управления данными временных рядов и скользящими окнами. Кроме того, пакет pandas располагает несколькими удобными инструментами, предназначенными для анализа данных, в т. ч. разбором, очисткой и агрегированием данных, а также построением графиков[2];
- **scikit-learn** предоставляет унифицированный интерфейс к алгоритмам машинного обучения[3];
- **scikit-image** обеспечивает инструменты анализа изображений, которые напрямую интегрируются в остальную часть экосистемы SciPy[4].

Помимо перечисленных выше библиотек, существует целый ряд других пакетов Python, которые являются частью экосистемы SciPy, и некоторые из них мы также увидим. Хотя в центре внимания настоящей книги будут библиотеки NumPy и SciPy, именно большое количество окружающих их пакетов делает Python движущей силой научных вычислений.

Великий катаклизм: Python 2 против Python 3

В ваших путешествиях по Python вы, вероятно, уже слышали пересуды о том, какая версия Python лучше. Возможно, вы задавались вопросом, а разве нельзя просто взять последнюю версию. (Сразу отвечаем: можно.)

В конце 2008 г. разработчики ядра Python выпустили Python 3, который среди прочих улучшений стал основным обновлением языка с более оптимальной обработкой текста (на многочисленных естественных языках) в кодировке Юникод, согласованностью типов и обработкой потоковых данных. Как язвительно заметил Дуглас Адамс[5] по поводу создания Вселенной, «это рассердило многих людей и повсеместно признавалось плохим ходом». И все потому, что программный код на Python 2.6 или 2.7 обычно не может исполняться интерпретатором Python 3, по крайней мере без небольшой модификации (хотя изменения, как правило, не слишком обременительны).

Всегда существует трение между неумолимой силой прогресса и обратной совместимостью. В этом случае команда разработчиков ядра языка Python решила, чтобы устранить некоторые несоответствия, в особенности в лежащем в основе программном интерфейсе C, требуется полный разрыв, и продвинула

---

1 См. http://jupyter.org/.

2 См. http://pandas.pydata.org/.

3 См. http://scikit-learn.org/.

4 См. http://scikit-image.org/.

5 См.: *Адамс Д.* Путеводитель автостопщика по Галактике (*Douglas Adams*. The Hitchhiker's Guide to the Galaxy. London: Pan Books, 1979).

язык в XXI век (Python 1.0 появился в 1994 г., более 20 лет назад, что в технологическом мире представляет собой целую жизнь).

Вот один из тех приемов, благодаря которым при переходе к версии 3 язык Python стал лучше:

```
print "Привет, Мир!"   # инструкция print в Python 2
print("Привет, Мир!")  # функция print в Python 3
```

Зачем создавать столько шума, чтобы добавить несколько круглых скобок! Все верно. Но если вместо этого вы хотите направить печать в другой *поток*, такой, например, как *стандартная ошибка*, т. е. в обычное место для отладочной информации?

```
print >>sys.stderr, "фатальная ошибка"       # Python 2
print("фатальная ошибка", file=sys.stderr)   # Python 3
```

Такое изменение, несомненно, выглядит целесообразнее. А что же происходит в Python версии 2? Авторы не знают, с полным на то правом.

Еще одно изменение состоит в том, как в Python 3 рассматривается деление целых чисел. Это делается точно так же, как большинство людей выполняет данную операцию. (Обратите внимание, что серия символов >>> говорит о том, что мы набираем программный код в интерактивной оболочке Python.)

```
# Python 2
>>> 5 / 2 2
# Python 3
>>> 5 / 2 2.5
```

Кроме того, нас также сильно порадовал новый оператор *умножения матриц* `@`, введенный в Python 3.5 в 2015 г. Обратитесь к главам 5 и 6, чтобы увидеть несколько практических примеров применения этого оператора!

Возможно, самым большим улучшением в Python 3 является поддержка Юникода, т. е. приема кодирования текста, который позволяет использовать не только английский алфавит, но и любой другой существующий в мире алфавит. Python 2 позволял вам определять строковое значение в кодировке Юникода следующим образом:

```
beta = u"β"
```

Но в Python 3 абсолютно все является Юникодом:

```
β = 0.5
print(2 * β)
```

```
1.0
```

Команда разработчиков ядра языка Python приняла абсолютно правильное решение, что в программном коде на Python имеет смысл в качестве объектов первого класса поддерживать символы всех естественных языков. Это в особенности актуально теперь, когда большинство новых программистов проживают не в англоязычных странах. Ради функциональной совместимости мы рекомендуем в большей части исходного кода по-прежнему использовать ан-

глийские символы. Эта возможность может пригодиться, например, в блокнотах Jupyter, утяжеленных математическими формулами.

Наберите в терминале IPython или в блокноте Jupyter LaTeX-е имя символа, после чего нажмите клавишу Tab. В результате это имя будет расширено в символ Юникода. Например, `\beta<TAB>` станет β.

Обновление до Python 3 также разрушает большую часть существующего программного кода в версии 2.x и в некоторых случаях Python 3 исполняется медленнее, чем прежде. Несмотря на эти недостатки, мы рекомендуем всем пользователям как можно скорее обновиться до третьей версии (до 2020 г. Python 2.x теперь находится только в режиме обслуживания), поскольку большинство существовавших проблем будет решено вместе с совершенствованием версий 3.x. И действительно, в этой книге мы используем многие новые возможности Python 3.

В данной книге мы используем **Python 3.6**.

Дополнительные материалы для чтения, касающиеся перехода на версию 3, вы сможете найти на ресурсе Эда Шофилда Python-Future[1] и в справочнике книжного формата[2] Ника Коглана.

Экосистема и сообщество SciPy

SciPy – это главная библиотека с довольно-таки большой функциональностью. Вместе с NumPy она является одним из уникальных приложений Python. Она положила начало огромному количеству связанных с ней библиотек, которые опираются на ее функционал. Со многими этими библиотеками вы будете работать на протяжении всей книги.

Создатели данных библиотек и многие их пользователи встречаются на многочисленных мероприятиях и конференциях по всему миру. Это такие мероприятия, как ежегодная конференция SciPy в Остин (США), EuroSciPy, SciPy Индия, PyData, и другие. Мы настоятельно вам рекомендуем посетить одну из них и встретиться с авторами лучшего научного программного обеспечения в мире Python. Если же у вас нет возможности попасть на эти конференции или же вы просто хотите прочувствовать их характер, воспользуйтесь Сетью[3], где участники этих конференций публикуют свои дискуссии.

Бесплатное программное обеспечение и программное обеспечение с открытым исходным кодом (FOSS)

Сообщество SciPy приветствует разработку программного обеспечения с открытым исходным кодом. Исходный код почти всех библиотек SciPy находится

---

[1] См. http://python-future.org/.

[2] См. http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html.

[3] См. https://www.youtube.com/user/EnthoughtMedia/playlists.

в свободном доступе для чтения, правки и повторного использования компонентов любым, кто в этом заинтересован.

Если вы хотите, чтобы другие разработчики использовали ваш программный код, то один из лучших способов этого добиться состоит в том, чтобы сделать его свободным и открытым. Если вы используете программное обеспечение с закрытым исходным кодом и получаете не тот результат, который вы хотели достигнуть, то вам не повезло. Вы можете послать разработчику электронное сообщение и попросить, чтобы он добавил новый функционал (что часто не срабатывает!), либо написать новое программное обеспечение самостоятельно. Если исходный код находится в открытом доступе, то вы с легкостью можете добавить или изменить его функциональность, используя приемы, которые вы узнаете из этой книги.

Таким же образом, если вы находите в компоненте программного обеспечения системную ошибку, то наличие доступа к исходному коду может в значительной степени облегчить работу как пользователя, так и разработчика. Даже если вы не вполне разбираетесь в исходном коде, вы, как правило, продвинетесь в диагностике возникшей проблемы гораздо дальше и поможете разработчику с ее исправлением. Обычно это полезный познавательный опыт для всех!

Открытый исходный код, открытая наука

В научном программировании все вышеупомянутые сценарии чрезвычайно распространены и важны: научное программное обеспечение часто строится на предыдущей работе либо ее видоизменяет самым интересным образом. А вследствие высокого темпа научных публикаций и прогресса большое количество программного кода остается без тщательного тестирования перед его выпуском, что приводит к незначительным или системным ошибкам.

Еще одна веская причина, чтобы сделать исходный код открытым, состоит в том, чтобы способствовать развитию производимых исследований. Многие из нас по своему опыту знают, когда, читая действительно актуальную исследовательскую работу и затем скачивая исходный код, чтобы проверить его на своих собственных данных, мы обнаруживаем: исполняемый файл не скомпилирован для вашей операционной системы, невозможно разобраться, как его запустить, код имеет ошибки, отсутствуют важные компоненты. Или мы вообще получаем неожиданные результаты. Делая научное программное обеспечение открытым, мы не только улучшаем качество этого программного обеспечения, но и позволяем ясно увидеть, каким образом был написан код, какие допущения были приняты и жестко запрограммированы. Открытый исходный код помогает решать многие из этих проблем. Он также позволяет другим ученым опираться на исходный код своих коллег, способствуя новому сотрудничеству и ускоряя научный прогресс.

Лицензии на программное обеспечение с открытым исходным кодом

Если вы хотите, чтобы ваш программный код использовали другие, то вы *должны* его лицензировать. Если вы его не лицензируете, то по умолчанию он будет

закрытым. Даже если вы свой код опубликуете (например, разместив его в публичном хранилище GitHub), то без лицензии на программное обеспечение ваш программный код никто не имеет права использовать, править или распространять.

Выбирая среди многих вариантов лицензирования, сначала необходимо решить, что именно вы хотите позволить людям делать с вашим исходным кодом. Предоставить людям право продавать ваш исходный код для получения прибыли? Или право продавать программное обеспечение, в котором используется ваш исходный код? Или же вы хотите ограничить использование своего исходного кода только бесплатным программным обеспечением?

Есть две широкие категории FOSS-лицензий (лицензий на свободное и открытое программное обеспечение, Free and Open Source Software):

- разрешительная лицензия;
- свободная лицензия (Copy-left).

Разрешительная лицензия означает, что вы предоставляете любому пользователю право использовать, править и распространять ваш исходный код любым способом, который ему нравится, включая применение вашего исходного кода в качестве коммерческого программного обеспечения. В этой категории популярными вариантами являются лицензии MIT (программная лицензия Массачусетского технологического института) и BSD (программная лицензия университета Беркли). Сообщество SciPy приняло новую лицензию BSD (так называемую «Модифицированную BSD», или «3-пунктовую лицензию BSD»). Использование такой лицензии подразумевает получение помощи относительно исходного кода от огромного количества людей, включая тех, кто трудится в информационной индустрии и стартапах.

Свободные лицензии также позволяют другим разработчикам использовать, править и распространять ваш исходный код. Вместе с тем эти лицензии еще предписывают, что производный исходный код должен распространяться в соответствии со свободной лицензией. Таким образом свободные лицензии ограничивают то, что именно пользователи могут делать с этим исходным кодом.

Самой популярной свободной лицензией является Публичная лицензия GNU или GPL. Главный ее недостаток связан с использованием свободной лицензии и состоит в том, что часто ваш исходный код становится недоступным для любых потенциальных пользователей или участников из частного сектора. И среди них в будущем можете оказаться вы сами! Как результат этот факт может существенно уменьшить вашу пользовательскую базу и, следовательно, успех вашего программного обеспечения. В науке это может означать меньшее количество цитирования.

Для получения более подробной справки относительно выбора лицензии обратитесь на веб-сайт, посвященный выбору лицензии Choose a License[1]. От-

[1] См. http://choosealicense.com/.

носительно лицензирования в научном контексте мы рекомендуем публикацию в блоге «The Whys and Hows of Licensing Scientific Code» (Вопросы «почему» и «как» относительно лицензирования научного программного кода) под авторством Джейка Вандерпласа (Jake VanderPlas), директора по исследованиям в области естествознания Вашингтонского университета и разносторонней суперзвезды SciPy. Собственно, здесь мы процитируем Джейка, чтобы убедительно довести до вас ключевые моменты лицензирования программного обеспечения:

> ...если вы извлечете из статьи всего три порции информации, то пусть они будут следующими:
> 1. Всегда лицензируйте свой код. Нелицензированный код является закрытым, поэтому любая открытая лицензия лучше, чем ничего (но см. п. 2).
> 2. Всегда используйте лицензию, совместимую с общей публичной лицензией (GPL). GPL-совместимые лицензии гарантируют вашему исходному коду широкую совместимость и включают GPL, новую BSD, MIT и другие (но см. п. 3).
> 3. Всегда используйте разрешительную лицензию в стиле BSD. Разрешительная лицензия, такая как новая лицензия BSD или лицензия MIT, более предпочтительна, по сравнению со свободной лицензией, такой как GPL или LGPL.

Весь исходный код в этой книге доступен в соответствии с 3-пунктовой лицензией BSD (3-clause BSD license). Там, где мы разместили фрагменты исходного кода других авторов. Этот исходный код обычно действует в соответствии с разрешительной открытой лицензией в той или иной форме (хотя не обязательно в соответствии с лицензией BSD).

Что касается вашего собственного исходного кода, то мы рекомендуем вам применять практику своего сообщества. В научном Python это означает 3-пунктовую лицензию BSD, в то время как в сообществе разработчиков на языке R, например, принята лицензия GPL.

GitHub: исходный код в социальном пространстве

Мы немного поговорили о распространении своего исходного кода в соответствии с лицензией на открытое программное обеспечение. И будем надеяться, что огромное число людей будут скачивать ваш исходный код, использовать его, исправлять ошибки и добавлять новые свойства. В связи с этим возникает вопрос: где разместить свой исходный код, чтобы люди смогли его найти? Как эти исправления ошибок и новые свойства вернутся в ваш исходный код? Каким образом отслеживать все вопросы и изменения? Можно представить, как все это совсем скоро выйдет из-под контроля.

Присоединяйтесь к GitHub.

GitHub[1] – это веб-сайт для размещения, совместного использования и разработки исходного кода. В его основе лежит программная система управления

[1] См. https://github.com/.

версиями Git[1]. Обучению работе с GitHub посвящено несколько замечательных ресурсов, таких как «*Введение в GitHub*»[2] Питера Белла и Брента Бира. Подавляющее большинство проектов в экосистеме SciPy размещено на GitHub, поэтому, безусловно, стоит научиться его использовать!

Веб-сайт GitHub оказал значительное влияние на участие разработчиков и внесение ими открытого исходного кода. Это было достигнуто за счет предоставления пользователям возможности публиковать исходный код и свободно сотрудничать. Любой участник может зайти и создать копию (так называемое ответвление *fork*) исходного кода и отредактировать, как его душе будет угодно. Он в конечном счете может внести эти изменения назад в оригинальный исходный код, создав *запрос на включение внесенных изменений*. Существует целый ряд хороших возможностей, таких как управление текущими вопросами и запросами на изменение, а также возможность определять, кто непосредственно может редактировать ваш исходный код. Вы даже можете отслеживать правки участников и получать другую детальную статистику. Помимо вышеперечисленного, существует целый ряд других замечательных особенностей GitHub. Однако мы предоставим вам самим возможность обнаружить многие из них самостоятельно. С некоторыми из них вы познакомитесь, когда будете читать последующие главы. В сущности, GitHub демократизировал разработку программного обеспечения (рис. П.1) и существенно снизил барьер доступа к нему.

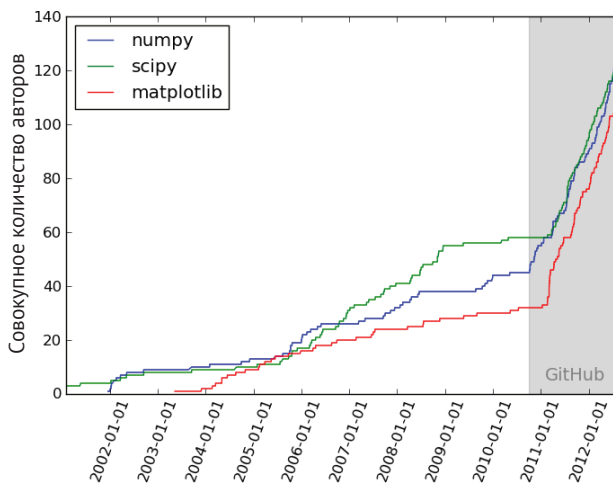


Рис. П.1 ❖ Влияние GitHub
(используется с разрешения автора, Джейка Вандерпласа)

[1] См. http://git-scm.com/.

[2] См. http://shop.oreilly.com/product/0636920033059.do.

Оставьте свой след в экосистеме SciPy

По мере накопления опыта работы с SciPy и после того, как вы начнете использовать эту библиотеку в вашей исследовательской работе, может оказаться, что тому или иному пакету не хватает функционала, в котором вы нуждаетесь. Либо вы посчитаете, что можете сделать что-то эффективнее. А возможно, найдете ошибку. Когда вы дойдете до этого, значит, пришла пора начать принимать участие в экосистеме SciPy.

Мы настоятельно рекомендуем вам попробовать. Сообщество существует, потому что люди готовы делиться своим исходным кодом и совершенствовать существующий исходный код. И если каждый из нас внесет свой небольшой вклад, то вместе мы добьемся многого. Но помимо любых альтруистических причин содействия сообществу, существует несколько вполне практических личных выгод. Сотрудничая с сообществом, вы станете более профессиональным программистом. Любой вносимый вами исходный код будет рассмотрен другими участниками, и вы получите отзыв в виде замечаний и комментариев. В качестве побочного эффекта вы научитесь использовать Git и GitHub, которые сами по себе являются очень полезными инструментами для технической поддержки и совместного использования вашего собственного исходного кода. Вы даже можете обнаружить, что взаимодействие с сообществом SciPy предоставляет вам более широкую научную сеть и удивительные возможности карьерного роста.

Мы хотим, чтобы вы задумались над тем, чтобы стать больше, чем просто пользователем SciPy. Присоединившись к сообществу, ваша работа сделает его лучше для всех научных программистов.

Капля эксцентричности Python

Если вы обеспокоены тем, что сообщество SciPy может стать для вновь прибывшего участника вызывающим благоговение местом, то следует напомнить, что это сообщество состоит из таких же людей, как и вы, ученых, которые, как правило, имеют прекрасное чувство юмора.

В стране Python вы неизбежно найдете ссылки на шоу «Монти Пайтон»[1]. Пакет Airspeed Velocity[2] измеряет быстродействие вашего программного обеспечения (подробнее о нем позднее) и ссылается на строчку из «Монти Пайтона и Священного Грааля» «какова воздушная скорость полета необремененной ласточки?».

А вот еще один забавно названный пакет – «Sux». Он позволяет использовать пакеты Python 2 из Python 3. В имени пакета обыгрывается слово «six» (шесть),

---

[1] Монти Пайтон (англ. Monty Python) – комик-группа из Великобритании, состоявшая из шести человек. Благодаря своему новаторскому, абсурдистскому юмору участники «Монти Пайтон» находятся в числе самых влиятельных комиков всех времен. Группа известна во многом благодаря юмористическому телешоу «Летающий цирк Монти Пайтона». – *Прим. перев.*

[2] См. http://spacetelescope.github.io/asv/using.html.

а сам пакет позволяет использовать синтаксис Python 3 в Python 2 с новозеландским акцентом. Синтаксис Sux уменьшает разочарование от использования пакетов, предназначенных только для Python 2, после того как вы перешли на Python 3:

```
import sux
p = sux.to_use('my_py2_package')
```

В целом имена библиотек Python могут вызывать бурное веселье, и мы надеемся, что вы хорошо проведете время, когда будете придумывать свое собственное!

Получение помощи

Когда мы попадаем в тупик, наш первый шаг – поискать в Интернете подсказку, которая поможет найти решение текущей задачи, либо информацию об ошибке, сообщение о которой мы получили. Эти поиски обычно приводят нас на Stack Overflow[1], превосходный вопросно-ответный сайт для программистов. Если вы с первого раза не нашли того, что ищете, то попробуйте обобщить критерии поиска, чтобы найти кого-то, кто имеет схожие проблемы.

Иногда вы можете оказаться фактически первым человеком, который задаст этот конкретный вопрос (это наиболее вероятно в тех случаях, когда вы используете совершенно новый пакет). Но не отчаивайтесь! Не все потеряно! Как было отмечено выше, сообщество SciPy дружелюбно и разбросано в различных частях межсетей. Ваш следующий шаг состоит в том, чтобы обратиться в поисковик с запросом «<имя библиотеки> список рассылки» и найти список рассылки с адресами, куда можно обратиться за помощью. Авторы библиотек и компетентные пользователи регулярно их читают и очень радушны ко вновь прибывшим участникам. Обратите внимание, что общепринятое правило поведения в сообществе требует, чтобы, перед тем как отправлять свои вопросы, вы *подписались* на список. Если вы не подпишетесь, это будет означать, что перед регистрацией вашего электронного адреса в списке кто-то должен будет вручную проверить, что данный адрес не является спамным. Присоединение к очередному списку рассылки может показаться ненужным, но мы настоятельно рекомендуем поступать именно так: это как раз то место, где надо учиться!

Инсталляция языка Python

В настоящей книге мы исходим из того, что у вас уже установлен Python 3.6 (либо его более поздняя версия) и все необходимые пакеты SciPy. Мы перечислим все использованные нами необходимые библиотеки и их версии в файле *environment.yml*, прилагаемом вместе с файлами данных и исходными кодами

[1] См. http://stackoverflow.com/.

к этой книге. Самый легкий способ получить все необходимые компоненты – установить conda[1], инструмент для управления средой разработки на Python. Затем вы можете передать файл *environment.yml* в conda, чтобы за один шаг установить правильные версии всех необходимых пакетов.

```
conda env create --name elegant-scipy -f путь/к/environment.yml
source activate elegant-scipy
```

За дополнительной информацией обратитесь к хранилищу книги на GitHub[2].

Доступ к книжным материалам

Весь исходный код и данные, приводимые в этой книге, доступны в нашем хранилище на GitHub. В файле README хранилища вы найдете инструкции по созданию блокнотов Jupyter из специальных исходных файлов с облегченной разметкой markdown. После этого блокноты Jupyter можно запускать в интерактивном режиме, используя данные, которые включены в хранилище.

Начинаем

Мы подобрали самый элегантный исходный код, который предлагает сообщество SciPy. Мы также займемся исследованием нескольких реальных научных задач, решаемых при помощи SciPy. Эта книга еще дает возможность мимолетно взглянуть на радушное и готовое к сотрудничеству научное программистское сообщество, которое надеется, что вы присоединитесь.

Добро пожаловать в элегантный SciPy.

Условные обозначения, принятые в книге

В книге используются следующие типографские условные обозначения.

Курсивный шрифт
: Указывает новые термины, URL-адреса, адреса электронной почты, имена и расширения файлов.

`Моноширинный шрифт`
: Используется для листингов программ, а также внутри абзацев для отсылки на элементы программ, такие как переменные или имена функций, базы данных, типы данных, переменные окружающей среды, операторы и ключевые слова.

`Жирный моноширинный шрифт`
: Показывает команды либо другой текст, который должен быть напечатан самим пользователем.

---

[1] См. http://conda.pydata.org/miniconda.html.

[2] См. https://github.com/elegant-scipy/elegant-scipy.

Курсивный моноширинный шрифт

Показывает текст, который должен быть заменен на предоставленные пользователем значения либо на значения, определяемые контекстом.

Данный элемент обозначает общее замечание.

Данный элемент обозначает подсказку или совет.

Данный элемент обозначает предупреждение или предостережение.

Использование цвета

В некоторых приводимых в книге примерах используются различные цвета, которые не видны в печатной версии этой книги. Читателям печатной версии книги рекомендуется обратиться к исходным блокнотам на http://elegant-scipy.org/.

Использование примеров программ

Дополнительный материал (примеры программного кода, упражнения и т. д.) доступен для скачивания с https://github.com/elegant-scipy/elegant-scipy.

Эта книга предназначена, чтобы помочь вам в решении своих задач. В целом, если код примеров предлагается вместе с книгой, вы можете использовать его в своих программах и документации. Вам не нужно связываться с нами с просьбой о разрешении, если вы не воспроизводите значительную часть кода. Например, написание программы, которая использует несколько фрагментов кода из данной книги, официального разрешения не требует. Продажа либо распространение компакт-диска с примерами из книг издательства O'Reilly требует официального разрешения. Ответ на вопрос цитированием данной книги и приведение выдержек кода примеров в качестве цитат разрешения не требует. Включение значительного количества кода примеров из данной книги в документацию на ваш продукт требует официального разрешения.

Мы ценим, но не требуем атрибуции. Атрибуция обычно включает в себя титул, автора, издателя и ISBN. Например: «*Нуньес-Иглесиас Х., Уолт ван дер Ш., Дэшноу Х.* Элегантный SciPy. O'Reilly. ISBN 978-1-491-92287-3».

Если вы считаете, что применяете примеры кода, выходя за рамки их справедливого использования или разрешения, выданного выше, то обращайтесь к нам по адресу permissions@oreilly.com.

Благодарности

Выражаем признательность многим и многим людям, которые внесли существенный вклад в эту книгу. Она бы не появилась без вашей помощи.

Прежде всего мы хотим поблагодарить многих участников, внесших свой вклад в разработку NumPy, SciPy и связанных с ними библиотек. Мы надеемся, что в этой книге мы отдали должное вашей удивительной работе.

Также благодарим многих участников, внесших свой вклад в развитие более широкой научной экосистемы Python, включая тех, кто обеспечил фундамент для нескольких наших глав: Винеша Бирокдара, Мэтта Роклин и Уоррена Векессер. Мы также должны поблагодарить тех, чей вклад мы не смогли включить к моменту публикации. Ваша работа нас вдохновила, и мы надеемся включить ее в будущие версии книги. Мы также благодарим Николаса Ругира за многие из его предположений, которые мы включили в качестве примеров и упражнений.

Другие предоставили нам данные и исходный код, который сэкономил нам часы поиска и изысканий. Мы благодарим Лэв Варшни за исходный программный код на MATLAB для макета спектрального графа мозга червя (главы 3 и 6) и Стефано Аллезина за данные о пищевой сети заповедника Сент-Марка (глава 6).

Мы обязаны всем, кто внес исправления и предложения во время предварительного показа книги, включая Билла Каца, Мэттиаса Бассоннира и Марка Хюн-ки Кима.

Мы благодарим наших технических рецензентов, Томаса Кэзуэлла, Нелли Вароко, Лэва Варшни и Грега Уилсон, которые великодушно нашли время в своих плотных графиках, чтобы причесать наши заключительные черновики и поделиться с нами своими экспертными оценками.

Хотя мы продолжим совершенствовать эту книгу на основе полученных от вас, наши читатели, комментариев, должны отдать должное нашим друзьям и семьям, которые корректировали самые ранние версии и предоставили ценные отзывы, предложения и поддержку. Малкольм Горман, Алисия Ошэк, ПВ Ван дер Уолт, Саймон Кокбек, Нелли Вароко и Ариэль Рокем: спасибо вам.

И конечно, мы благодарим наших редакторов издательства О'Рейли, Мэг Бланшетт, Брайана Макдональда и Нэн Барбер. Мы особенно благодарны Мэг, которая первая сделала нам предложение по поводу книги и предоставила бесценные указания на раннем этапе работы, когда у нас не было ни малейшего понятия, что мы должны делать.

Комментарии переводчика

Весь материал настоящей книги протестирован в среде Windows 10. При тестировании исходного кода за основу взят Python версии 3.6.4 (время перевода книги – март 2018 г.).

Прилагаемый к книге адаптированный и скорректированный исходный код примеров лучше всего разместить в подпапке домашней папки пользователя (`/home/python_projects` или `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\python_projects`). Ниже приведена структура папки с прилагаемыми примерами:

data	Файлы данных, используемые во всех главах и упражнениях
figures	Иллюстрации из книги
images	Фотографии, используемые в главах и упражнениях
markdown	Файлы с упрощенной разметкой глав книги на английском языке для генерирования блокнотов Jupyter
notes	Дополнительные примечания авторов с примерами исходного кода
pics	Изображения и графики, полученные в результате применения примеров программного кода
script	Дополнительные сценарии Python по темам книги
style	Стилевые файлы, используемые в примерах программного кода
tools	Дополнительные вспомогательные сценарии для работы с LaTex, HTML и Jupyter
Глава 1 – глава 8, решения задач	Исходный код примеров в виде сценариев Python, блокнотов Jupyter и файлов HTML

Базовый набор библиотек для разработчика

В обычных условиях библиотеки Python можно скачать и установить из каталога библиотек Python PyPi (https://pypi.python.org/). Однако следует учесть, что для работы некоторых библиотек, в частности SciPy и Scikit-learn, в ОС Windows требуется, чтобы в системе была установлена библиотека Numpy+MKL. Библиотека **Numpy+MKL** привязана к библиотеке Intel® Math Kernel Library и включает в свой состав необходимые динамические библиотеки (DLL), расположенные в каталоге numpy.core. Библиотеку Numpy+MKL следует скачать с хранилища whl-файлов на веб-странице Кристофа Голька из Лаборатории динамики флуоресценции Калифорнийского университета в г. Ирвайн (http://www.lfd.uci.edu/~gohlke/pythonlibs/) и установить при помощи менеджера пакетов pip как whl (соответствующая процедура установки пакетов в формате whl описана ниже). Например, для 64-разрядной операционной системы Windows и среды Python 3.6 команда будет такой:

```
pip install numpy-1.14.2+mkl-cp36-cp36m-win_amd64.whl
```

Подобный режим установки также касается библиотек scipy, scikit-image и scikit-learn. Стоит также отметить, что эти особенности установки не относятся к ОС Linux и Mac. Далее приводятся сведения об основополагающих библиотеках.

- **NumPy**, основополагающая библиотека, необходимая для научных вычислений на Python.
- **SciPy**, библиотека, используемая в математике, естественных науках и инженерном деле. Требует наличия numpy+mkl.
- **Matplotlib**, библиотека для работы с двумерными графиками.
- **Pandas**, инструмент для анализа структурных данных и временных рядов. Требует наличия numpy и некоторых других. Для чтения файлов Excel требует установки библиотеки xlrd.
- **Scikit-learn**, интегратор классических алгоритмов машинного обучения. Требует наличия numpy+mkl.

- **Jupyter**, интерактивная онлайновая вычислительная среда.
- **PyQt5**, библиотека инструментов для программирования графического интерфейса пользователя, требуется для работы инструментальной среды программирования `Spyder`.
- **Spyder**, инструментальная среда программирования на Python.

Набор библиотек, используемых в книге

Ниже перечислены так называемые зависимости, т. е. библиотеки и их версии, от которых зависит приводимый в книге исходный код. Все они устанавливаются при помощи менеджера пакетов `pip` стандартным образом, за исключением первых четырех библиотек, особенности установки которых в ОС Windows были упомянуты выше:

- numpy=1.12*
- scipy=0.19*
- scikit-learn=0.18*
- scikit-image=0.13*
- pandas=0.19*
- matplotlib=2.0*
- xlrd=1.0*
- jupyter=1.0*
- sympy=1.*
- toolz=0.8*
- six=1.10*
- beautifulsoup4=4.5*
- pillow
- lxml
- networkx
- notedown>=1.5
- jupytercontrib>=0.0.5
- bs4

Протокол установки библиотек

Если вы хотите устанавливать более свежие версии библиотек и держать весь процесс установки под своим контролем, то ниже предлагается список команд установки библиотек. В случае нескольких библиотек может потребоваться установка из файлов `whl`, которые можно взять из хранилища whl-файлов (http://www.lfd.uci.edu/~gohlke/pythonlibs/).

```
pip install --upgrade pip
pip install numpy
```

либо как whl: `pip install numpy-1.14.2+mkl-cp36-cp36m-win_amd64.whl`

```
pip install scipy
```

либо как whl: `pip install scipy-1.0.1-cp36-cp36m-win_amd64.whl`

```
pip install Scikit-learn
```

либо как whl: `pip install scikit_learn-0.19.1-cp36-cp36m-win_amd64.whl`

```
pip install scikit-image
```

либо как whl: `pip install scikit_image-0.13.1-cp36-cp36m-win_amd64.whl`

Все остальные библиотеки устанавливаются стандартным образом:

```
pip install pandas
pip install matplotlib
pip install jupyter
...
pip install pyqt5
pip install spyder
```

В зависимости от базовой ОС, версий языка Python и версий программных библиотек устанавливаемые вами версии whl-файлов могут отличаться от приведенных выше, где показаны последние на апрель 2018 г. версии для 64-разрядной ОС Windows и Python 3.6.4.

Установка библиотек Python из whl-файлов

Библиотеки для Python можно разрабатывать не только на чистом Python. Довольно часто библиотеки пишутся на C (динамические библиотеки), и для них создается обертка Python, или же библиотека пишется на Python, но для оптимизации узких мест часть кода пишется на C. Такие библиотеки получаются очень быстрыми, однако библиотеки с вкраплениями кода на C программисту на Python тяжелее установить ввиду банального отсутствия соответствующих знаний либо необходимых компонентов и настроек в рабочей среде (в особенности в Windows). Для решения описанных проблем разработан специальный формат (файлы с расширением `.whl`) для распространения библиотек, который содержит заранее скомпилированную версию библиотеки со всеми ее зависимостями. Формат whl поддерживается всеми основными платформами (Mac OS X, Linux, Windows).

Установка производится с помощью менеджера библиотек pip. В отличие от обычной установки командой `pip install <имя_библиотеки>`, вместо имени библиотеки указывается путь к whl-файлу `pip install <путь/к/whl_файлу>`. Например:

```
pip install C:\temp\scipy-1.0.1-cp36-cp36m-win_amd64.whl
```

Откройте окно командой строки и при помощи команды cd перейдите в каталог, где размещен ваш whl-файл. Скопируйте в этот каталог имя вашего whl-файла. В этом случае полный путь указывать не понадобится. Например:

```
pip install scipy-1.0.1-cp36-cp36m-win_amd64.whl
```

При выборе библиотеки важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать whl-файлы с веб-сайта Кристофа Голька, где библиотеки постоянно обновляются. В архиве содержатся все библиотеки, какие только будут нужны.

Установка и настройка инструментальной среды Spyder

Spyder – это инструментальная среда для научных вычислений для языка Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это простая, легковесная и бесплатная интерактивная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию элементы интерфейса `PyQt5` и `PySide`: редактор исходного кода, редактор массивов данных `NumPy`, редактор словарей, а также консоли Python и IPython и многое другое.

Чтобы установить среду Spyder в Ubuntu Linux, используя официальный менеджер библиотек, нужна всего одна команда:

```
sudo apt-get install spyder
```

Чтобы установить с использованием менеджера библиотек `pip`:

```
sudo apt-get install python-qt5 python-sphinx
sudo pip install spyder
```

Для обновления:

```
sudo pip install -U spyder
```

Установка среды Spyder в Fedora 27:

```
dnf install python-spyder
```

Установка среды Spyder в Windows:

```
pip install spyder
```

Среда Spyder требует обязательной установки библиотеки PyQt5 (`pip install pyqt5`).

Блокноты Jupyter

Среди файлов с исходным кодом Python можно найти файлы с расширением `.html` и `.ipynb`. Последние – это файлы блокнотов интерактивной среды программирования Jupyter (http://jupyter.org/). Блокноты Jupyter позволяют иметь в одном месте исходный код, результаты выполнения исходного кода, графики данных и документацию, которая поддерживает синтаксис упрощенной разметки Markdown и мощный синтаксис LaTeX.

Интерактивная среда программирования Jupyter – это зонтичный проект, который наряду с Python предназначен для выполнения в обычном веб-браузере

небольших программ и фрагментов программного кода на других языках программирования, в том числе Julia, R и многих других (уже более 40 языков).

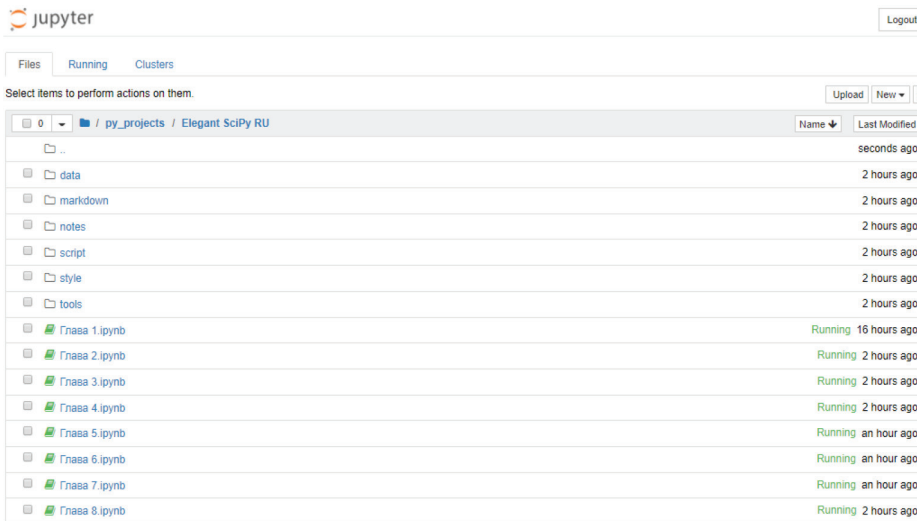
Интерактивная среда программирования Jupyter устанавливается, как обычно, при помощи менеджера пакетов `pip`.

```
pip install jupyter
```

Чтобы запустить интерактивную среду Jupyter, следует в командной оболочке или окне терминала набрать и исполнить приведенную ниже команду:

```
jupyter notebook
```

Локальный сервер интерактивной среды Jupyter запустится в браузере, заданном по умолчанию (как правило, по адресу http://localhost:8888/). После этого нужный блокнот Jupyter можно просто выбрать из меню Jupyter.



Файлы HTML представляют собой простые копии блокнотов Jupyter для просмотра в веб-браузере.

Представление чисел

Используемая в языке Python и в настоящей книге форма записи вещественных чисел и больших чисел с несколькими группами разрядов отличается от принятой в России. Ниже в таблице показаны эти отличия:

Разделитель групп разрядов	
США	Россия
запятая (,)	пробел
65,535	65 535

Десятичный разделитель	
США	Россия
точка (.)	запятая (,)
3.14159	3,14159

Во время работы с Python вы часто будете встречать это разночтение – программируя на Python, вам придется иметь дело с отличающимся форматом записи чисел и сталкиваться с задачей форматирования чисел в локальное для России представление во время вывода результатов вычислений.

Для решения этой задачи, в качестве одного из вариантов, в Python имеется встроенная библиотека `locale`, которая позволяет настраивать форму представления чисел и соответствующим образом их форматировать. В общем случае, когда в программе используется форматирование чисел, следует начинать программу с приведенных ниже строк кода:

```
import locale            # импортировать библиотеку по работе с локалями
loc = locale.getlocale() # запомнить текущую локаль
locale.setlocale(locale.LC_ALL, "Russian_Russia.1251") # изменить локаль
```

В последней строке задается форма написания чисел, принятая у нас в стране. Далее вы размещаете свой собственный программный код, к примеру:

```
cost = 50000.0          # стоимость в рублях
print('Стоимость изделия составляет ₽',
      locale.format('%.2f', cost, grouping=True),
      sep='')
```

И в конце программы следует вернуть локаль, которая была вначале:

```
locale.setlocale(locale.LC_ALL, loc)  # вернуть локаль назад
```

Если выполнить этот фрагмент кода, то вы получите результат, в котором представление чисел будет соответствовать принятому у нас стандарту, т. е. с разделением групп разрядов пробелом и использованием в качестве десятичного разделителя запятой:

```
Стоимость изделия составляет ₽50 000,00
```

Глава 1

Элегантный NumPy: фундамент научного программирования на Python

> [Библиотека NumPy] повсюду. Она окружает нас. Даже сейчас она с нами рядом. Ты видишь ее, когда смотришь в окно или включаешь телевизор. Ты ощущаешь ее, когда работаешь, идешь в церковь, когда платишь налоги.
>
> *– Морфеус, к/ф «Матрица»*

Эта глава затрагивает некоторые статистические функции SciPy, однако особое внимание в ней уделено исследованию массива NumPy, структуры данных, которая лежит в основе почти всех численных научных вычислений в Python. Мы увидим, как операции с массивами NumPy позволяют создавать краткий и эффективный исходный код для управления числовыми данными.

В нашем случае мы будем использовать данные экспрессии генов из проекта «Атлас ракового генома» (The Cancer Genome Atlas, TCGA) для предсказания смертности среди больных раком кожи. Мы будем работать в направлении этой цели на протяжении всей этой главы и главы 2, по ходу знакомясь с некоторыми ключевыми понятиями SciPy. Прежде чем мы сможем предсказать смертность, мы должны нормализовать данные экспрессии, используя метод под названием «*нормализация RPKM*». Он позволяет сравнивать результаты измерений между различными образцами и генами. (Мы раскроем смысл понятия «экспрессия гена» буквально через мгновение.)

Чтобы вас заинтриговать и познакомить с идеями этой главы, давайте начнем с фрагмента кода. Так же, как и в других главах, мы начинаем с примера исходного кода, который, по нашему мнению, воплощает элегантность и мощь той или иной функции экосистемы SciPy. В данном случае мы хотим подчеркнуть правила векторизации и транслирования библиотеки NumPy, ко-

торые позволяют нам очень эффективно управлять массивами данных и делать о них выводы.

```
def rpkm(counts, lengths):
    """Вычислить прочтения на тысячу оснований экзона на миллион
    картированных прочтений (reads per kilobase transcript per million reads).

    RPKM = (10^9 * C) / (N * L)

    где:
    C = количество прочтений, картированных на ген
    N = суммы количеств картированных (выровненных) прочтений в эксперименте
    L = длина экзона в парах оснований для гена

    Параметры
    ---------
    counts: массив, форма (N_genes, N_samples)
        РНК-сек (или подобные) количественные данные, где столбцы являются
        отдельными образцами и строки – генами.
    lengths: массив, форма (N_genes,)
        Длины генов в парах оснований в том же порядке, что и
        строки в counts.

    Возвращает
    ----------
    normed: массив, форма (N_genes, N_samples)
        Матрица количеств counts, нормализованная согласно RPKM.
    """
    N = np.sum(counts, axis=0) # просуммировать каждый столбец, чтобы
                               # получить суммы количеств прочтений на образец
    L = lengths
    C = counts

    normed = 1e9 * C / (N[np.newaxis, :] * L[:, np.newaxis])

    return(normed)
```

Этот пример иллюстрирует несколько приемов, благодаря которым массивы NumPy могут сделать ваш код элегантнее:

- массивы могут быть одномерными, как списки, но могут быть и двумерными, как матрицы, и даже более высокой размерности. Это позволяет представлять множество различных видов числовых данных. В нашем случае мы манипулируем двумерной матрицей;
- с массивами можно выполнять операции вдоль осей. В первой строке мы вычисляем сумму каждого столбца, задав ось `axis=0`;
- массивы позволяют выражать сразу несколько числовых операций. Например, ближе к концу функции мы делим двумерный массив количеств (*C*) на одномерный массив постолбцовых сумм (*N*). Такая операция называется транслированием. Дополнительная информация о том, как она работает, буквально через мгновение!

Прежде чем мы начнем вникать в мощные возможности NumPy, давайте потратим немного времени, чтобы разобраться в биологических данных, с которыми мы будем работать.

Введение в данные: что такое экспрессия гена?

Мы построим свою работу, опираясь на *анализ экспрессии генов*, который позволит продемонстрировать силу библиотек NumPy и SciPy в решении реальной биологической задачи. Мы воспользуемся библиотекой pandas, которая опирается на NumPy, чтобы прочитать и преобразовать наши файлы данных, и затем мы будем эффективно манипулировать нашими данными в массивах NumPy.

Так называемая *центральная догма* молекулярной биологии[1] постулирует, что вся информация, необходимая для анализа клетки (или в данном случае организма), хранится в молекуле, называемой дезоксирибонуклеиновой кислотой, или ДНК. Эта молекула имеет периодически повторяющийся остов, на котором лежат химические группы, именуемые *основаниями*, в последовательности (рис. 1.1). Имеется четыре вида оснований, сокращенно A, C, G и T, составляющих алфавит, посредством которого сохраняется информация.

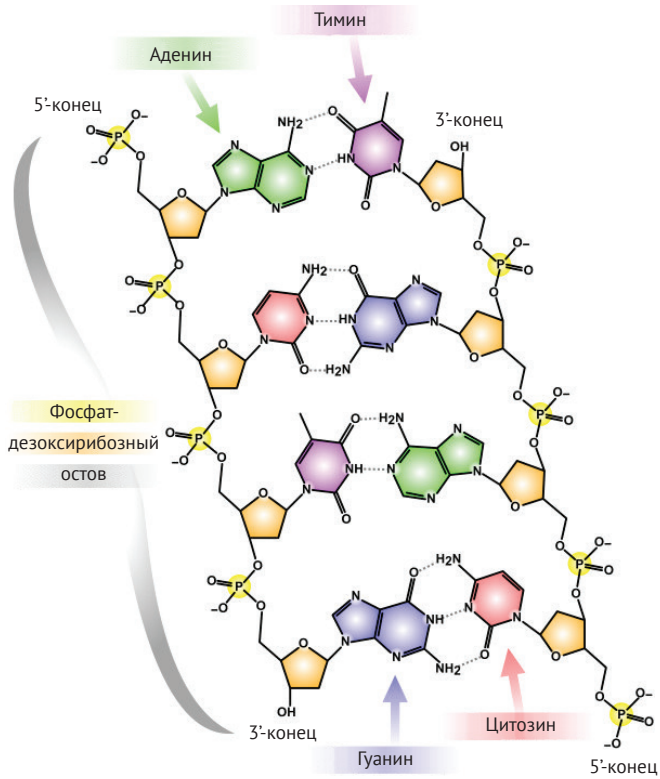


Рис. 1.1 ❖ Химическая структура ДНК (автор Маделин Прайс Болл, изображение используется в соответствии с лицензией общего пользования CC0)

[1] См. https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology.

Чтобы получить доступ к этой информации, ДНК *транскрибируется* в родственную молекулу, которая называется матричной рибонуклеиновой кислотой, или мРНК. Наконец, мРНК *транслируется* в белки, «рабочие лошадки» клетки (рис. 1.2). Участок ДНК, в котором закодирована информация, дающая белок (через мРНК), называется геном.

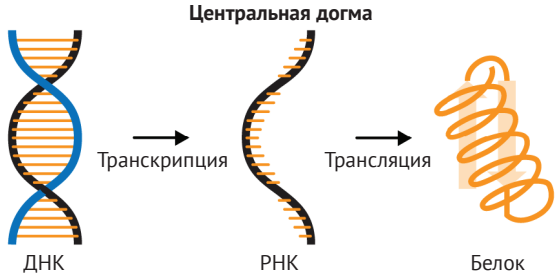


Рис. 1.2 ❖ Центральная догма молекулярной биологии

Количество мРНК, получаемых из конкретного гена, называется *экспрессией* этого гена. В идеальном случае мы хотели бы измерить уровни белка. Однако это намного более трудная задача, чем измерение мРНК. К счастью, уровни экспрессии мРНК и уровни соответствующего ей белка обычно коррелируются[1]. Поэтому мы обычно измеряем уровни мРНК и на этом основании проводим наши исследования. Как вы увидите ниже, зачастую это не имеет значимой разницы, потому что уровни мРНК используются из-за их способности предсказывать биологические исходы, избавляя от необходимости делать определенно сформулированные утверждения о белках.

Важно отметить, что ДНК в каждой клетке вашего тела идентична. Поэтому различия между клетками являются результатом дифференциальной экспрессии этой ДНК на РНК: в разных клетках разные участки ДНК обрабатываются в нисходящие молекулы (рис. 1.3). Аналогичным образом, как мы увидим в этой и следующей главах, дифференциальная экспрессия может идентифицировать различные виды рака.

Современная технология измерения количества мРНК основывается на методе секвенирования (расшифровки) РНК, который носит сокращенное название РНК-сек (RNA-seq). РНК извлекается из образца ткани (например, в результате взятой у пациента биопсии), обратно транскрибируется в ДНК (чья стабильность выше) и затем прочитывается с использованием химически модифицированных оснований, которые светятся[2] при включении их в последовательность ДНК.

---

[1] См.: *Майер Т., Гвелл М., Серрано Л.* Корреляция мРНК и белка в составных биологических образцах (*Tobias Maier, Marc Güell, and Luis Serrano*. Correlation of mRNA and protein in complex biological samples. FEBS Letters 583, no. 24 (2009). https://www.sciencedirect.com/science/article/pii/S0014579309008126).

[2] См.: И все-таки ДНК светится // https://22century.ru/biology-and-biotechnology/31442. – *Прим. перев.*

В настоящее время высокопроизводительные секвенирующие машины способны прочитать лишь короткие фрагменты (как правило, приблизительно 100 оснований). Эти короткие последовательности называются «прочтениями»[1]. Мы измеряем миллионы прочтений и затем на основе их последовательностей подсчитываем, сколько прочтений пришло из каждого гена (рис. 1.4). Мы начнем наш анализ непосредственно с этих количественных данных.

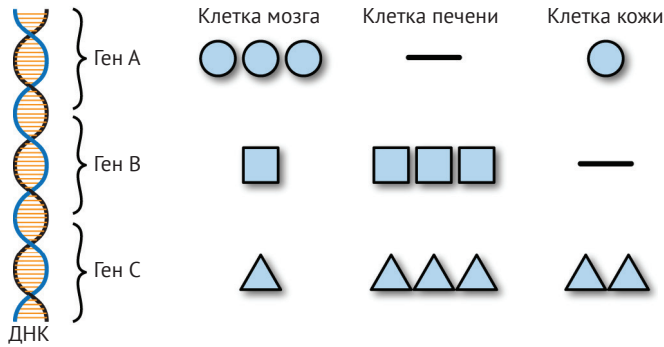


Рис. 1.3 ❖ Экспрессия гена

Таблица 1.1 показывает минимальный пример количественных данных об экспрессии генов.

Таблица 1.1. Количественные данные об экспрессии генов

	Тип клетки A	**Тип клетки B**
Ген 0	100	200
Ген 1	50	0
Ген 2	350	100

Эти данные представляют собой таблицу количеств прочтений в виде целых чисел, показывающих, сколько прочтений наблюдалось относительно каждого гена в каждом типе клетки. Вы можете заметить, насколько эти количества по каждому гену разнятся в зависимости от типов клетки. Эта информация может использоваться для того, чтобы узнать различия между этими двумя типами клеток.

Одним из способов представить эти данные в Python является список списков:

```
gene0 = [100, 200]
gene1 = [50, 0]
gene2 = [350, 100]
expression_data = [gene0, gene1, gene2]
```

[1] Прочтение, или рид (read) – это отсеквенированная последовательность коротких отрезков, полученных при разбиении молекулы, в данном случае молекулы мРНК. – *Прим. перев.*