Научное программирование на Python

Изучение основных задач программирования на профессиональном уровне с нуля с реальными научными примерами и решениями, взятыми из научной и инженерной практики. Студенты и исследователи всех уровней постепенно переходят на мощный язык программирования Python как альтернативу коммерческим программным продуктам. Этот интенсивный вводный курс позволяет пройти путь от основ до продвинутых концепций в одной книге, дающий возможность читателям быстро поднимать свой уровень профессиональной подготовки.

Книга начинается с общих концепций программирования, таких как циклы и функции в ядре Python 3, затем рассматриваются библиотеки NumPy, SciPy и Matplotlib для вычислительного программирования и визуализации данных. Обсуждается использование виртуального блокнота Jupyter Notebooks для создания мультимедийных совместно используемых документов для научного анализа. Отдельная глава посвящена анализу данных с использованием библиотеки pandas. В заключительной части представлены более сложные темы, такие как точность вычислений с применением чисел с плавающей точкой и обеспечение стабильности алгоритмов.

Кристиан Хилл (Christian Hill) – физик и специалист в области физической химии, в настоящее время работающий в Интернациональном агентстве по использованию атомной энергии (International Atomic Energy Agency). Он обладает более чем 25-летним опытом программирования в области физических наук и программирует на Python 15 лет. В своих исследованиях Кристиан использует Python для создания, анализа, обработки, управления и визуализации крупных наборов данных в области спектроскопии, физики плазмы и материаловедения.

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте **www.dmkpress.com**, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу **dmkpress@gmail.com**.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Благодарности

Эмме, Шарлотте и Лоренсу

Множество людей прямо или косвенно помогали написать эту книгу, в частности Джонатан Теннисон (Jonathan Tennyson) из UCL, Лоуренс Ротман (Laurence Rothman) и Айюли Гордон (Iouli Gordon) поддерживали меня во время годичного отпуска для научной работы в Центре астрофизики Гарварда и Смитсоновского института (Harvard-Smithsonian Center for Astrophysics).

Множество ошибок и промахов в первом издании этой книги было замечено и указано всего лишь несколькими людьми, которые всегда были готовы оказать мгновенную помощь: Стэффорд Бэйнс (Stafford Baines), Мэтью Гиллман (Matthew Gillman) и Стюарт Андерсон (Stuart Anderson). Те ошибки, которые все еще остаются в книге, – это, само собой, исключительно мои собственные ошибки.

Кроме того, особенно благодарен следующим людям: Хелен Рейнолдс (Helen Reynolds), Крис Пикар (Chris Pickard), Элисон Уайтли (Alison Whiteley), Джеймс Элиотт (James Elliott), Лианна Ишихара (Lianna Ishihara) и Мило Шаффер (Milo Shaffer). Как и любой человек, я очень нуждался в поддержке, поощрении и дружеском отношении Натали Хэйнс (Natalie Haynes).

Список листингов

Глава 1

Введение

1.1 Об этой книге

Эта книга предназначена для того, чтобы помочь ученым и инженерам освоить версию 3 языка программирования Python и связанных с ней библиотек: NumPy, SciPy, Matplotlib и pandas. Для чтения книги не требуется предварительный опыт программирования и научные знания в какой-либо конкретной области. Но знакомство с некоторыми математическими дисциплинами, такими как тригонометрия, комплексные числа и основы математического анализа, будет полезным при выполнении примеров и упражнений.

Python – мощный язык программирования со множеством расширенных функциональных возможностей и дополнительных пакетов поддержки. Основной синтаксис языка прост и понятен для изучения, но в полном объеме изучить его невозможно в книге такого размера. Таким образом, наша цель – сбалансированное, но достаточно подробное введение в самые главные функциональные возможности языка и самых важных библиотек. В текст включено множество примеров, связанных с научными исследованиями, в конце каждого раздела приведен список вопросов (коротких задач, предназначенных для проверки полученных знаний) и упражнений (более сложных задач, для решения которых обычно требуется написать небольшую компьютерную программу). Хотя нет необходимости выполнять абсолютно все упражнения, для читателей будет полезно попытаться выполнить хотя бы некоторые из них. Раздел, пример или упражнение, содержащие более сложный материал, который можно пропустить при первом чтении, обозначены символом ◊.

В главе 2 подробно рассматривается основной синтаксис, структуры данных и средства управления потоком выполнения в программе на языке Python. Глава 3 представляет собой небольшое отступление с описанием использования библиотеки pyplot для создания графических вариантов представления данных: это полезно для визуализации вывода программ в последующих главах. В главе 4 содержится более продвинутое описание ядра языка Python и краткое введение в объектно-ориентированное программирование. Далее следует еще одна короткая глава 5, представляющая широко известные виртуальные блокнотные среды IPython и Jupyter, далее – главы о научном программировании с использованием библиотек NumPy, SciPy, Matplotlib и pandas. В заключитель-

ной главе рассматриваются более общие темы научного программирования, включая арифметику с плавающей точкой, обеспечение стабильности алгоритмов и стиль программирования.

Читатели, уже знакомые с языком программирования Python, могут бегло просмотреть главы 2 и 4.

Примеры кода и решения упражнений можно загрузить с веб-сайта книги https://scipython.com/. Следует отметить, что хотя комментарии полностью включены в эти загружаемые программы, они не столь подробны в печатной версии данной книги: вместо этого исходный код описывается в тексте с помощью пронумерованных ссылок (например, ❶). Читатели, которые предпочитают вводить исходный код этих программ вручную, могут пожелать добавить собственные описательные комментарии в код.

1.2 Немного о Python

Python – это мощный язык программирования общего назначения, который разработал Гвидо ван Россум (Guido van Rossum) в 1989 году[1]. Python классифицируется как язык программирования высокого уровня, в котором автоматически обрабатывается большинство фундаментальных операций (таких как управление памятью), выполняемых на уровне процессора («машинный код»). Python считается языком более высокого уровня, чем, например, C, из-за его выразительного синтаксиса (который во многих случаях близок к естественному языку) и богатого разнообразия встроенных структур данных, таких как списки, кортежи, множества и словари. Например, рассмотрим следующую программу на Python, которая выводит список имен в отдельных строках.

Листинг 1.1. Вывод списка имен с использованием программы, написанной на Python

```
# eg1-names.py: вывод трех имен в консоли.

names = ['Isaac Newton', 'Marie Curie', 'Albert Einstein']
for name in names:
print(name)
```

Вывод:

```
Isaac Newton
Marie Curie
Albert Einstein
```

А теперь сравните исходный код из листинга 1.1 с кодом программы на C, которая делает то же самое.

[1] До настоящего момента Гвидо – «benevolent dictator for life (BDFL)» – «великодушный диктатор, управляющий жизнью» языка Python.

Листинг 1.2. Вывод списка имен с использованием программы, написанной на языке C

```
/* eg1-names.c: вывод списка имен в консоли. */
#include <stdio.h>
#include <stdlib.h>

const char *names[] = {"Isaac Newton", "Marie Curie", "Albert Einstein"};

int main(void)
{
  int i;

  for (i = 0; i < (sizeof(names) / sizeof(*names)); i++) {
    printf("%s\n", names[i]);
  }

  return EXIT_SUCCESS;
}
```

Даже если вы незнакомы с языком C, из листинга 1.2 можно понять, что написание кода на C даже для такой простой задачи связано с большими трудозатратами и издержками: две инструкции `include` для использования библиотек, не загружаемых по умолчанию, явные объявления переменных для хранения списка (в C это массив (array)) имен `names`, для счетчика `i`, а также явный проход по индексам этого массива в цикле `for`. Требуется даже вручную добавлять символы концов строк (`\n` – символ перехода на новую строку). Затем этот исходный код должен быть скомпилирован, т. е. преобразован в машинный код, который понимает процессор, прежде чем можно будет его выполнить. Более того, здесь огромное количество возможностей совершить ошибки (программные «баги» (bugs)): попытка вывода имени, хранящегося по адресу `name[10]`, вероятнее всего, приведет к выводу так называемого «мусора»: компилятор языка C не остановит вас при попытке доступа к несуществующему имени.

Та же самая программа, написанная в три строки на языке Python, проста и выразительна: не нужно явно объявлять, что `names` – список строк, для цикла не требуется счетчик, подобный `i`, и не приходится включать отдельные библиотеки (инструкцией `import` в Python). Для запуска Python-программы нужно просто ввести команду `python eg1-names.py`, которая автоматически вызовет интерпретатор Python для компиляции и выполнения полученного байт-кода (bytecode) (это особый тип промежуточного представления программы между исходным кодом и конечным машинным кодом, который Python перенаправляет в процессор).

Синтаксис Python призван гарантировать, что «существует один – и преимущественно единственный – очевидный способ сделать это». Это отличает Python от некоторых других широко известных языков высокого уровня, таких как Ruby и Perl, в которых используется противоположный подход, выражающийся в кратком принципе «существует более одного способа сделать это». Например, существует (как минимум) четыре очевидных способа вывода того же списка имен на языке Perl[2].

[2] Уточняю: очевидных для программистов на языке Perl.

Листинг 1.3. Различные способы вывода списка имен с использованием программы, написанной на Perl

```
@names = ("Isaac Newton", "Marie Curie", "Albert Einstein");
# Метод 1
print "$_\n" for @names;

# Метод 2
print join "\n", @names;
print "\n";

# Метод 3
print map { "$_\n" } @names;

# Метод 4
$" = "\n";
print "@names\n";
```

(Также обратите внимание на знаменитый лаконичный, но иногда непонятный синтаксис языка Perl.)

1.2.1 Преимущества и недостатки языка Python

Ниже перечислены некоторые из основных преимуществ языка программирования Python, а также причины, по котором вы, возможно, захотите его использовать:

- ясный и простой синтаксис позволяет быстро писать программы на Python и в общем сводит к минимуму возможности совершения скрытых ошибок. При правильном подходе результатом является высококачественное программное обеспечение, которое легко сопровождать и расширять;
- сама рабочая программная среда Python и связанные с ней библиотеки бесплатны, а кроме того, представляют собой программное обеспечение с открытым исходным кодом, в отличие от коммерческих предложений, таких как Mathematica и MATLAB;
- поддержка многих платформ: Python доступен для каждой общедоступной компьютерной системы, в том числе Windows, Unix, Linux и macOS. Несмотря на то что существуют расширения, зависящие от конкретной платформы, всегда есть возможность написания кода, который будет работать на любой платформе без каких-либо изменений;
- для Python существует большая библиотека модулей и пакетов, которая расширяет его функциональность. Многие из этих модулей и пакетов доступны как часть «стандартной библиотеки» (Standard Library), предоставляемой вместе с интерпретатором языка Python. Другие, в том числе библиотеки NumPy, SciPy, Matplotlib и pandas, используемые в научных вычислениях, можно абсолютно бесплатно загрузить и установить;
- язык Python относительно прост для изучения. Синтаксис и ключевые слова для основных операций применяются постоянно и согласованно в большинстве случаев более продвинутого использования языка. Сообщения об ошибках в основном представляют собой разумные пред-

положения о том, что пошло не так, в отличие от обобщенных «крахов», характерных для компилируемых языков высокого уровня, подобных C;

- Python – гибкий язык: его часто описывают как язык «многих парадигм», в котором имеются наилучшие функциональные возможности для процедурного, объектно-ориентированного и функционального программирования. Он требует совсем небольшой подготовительной работы, обязательной в других языках, когда задачу можно решить лишь с применением одного из этих подходов.

Так в чем же дело? А в том, что у Python имеются и некоторые недостатки, из-за которых он не подходит для абсолютно любого приложения:

- скорость выполнения программы на Python не так высока, как программ на других, полностью компилируемых языках, таких как C и Fortran. Для крупномасштабной вычислительной работы библиотеки NumPy и SciPy до некоторой степени способны облегчить ситуацию, используя «скрытый внутри» скомпилированный код C, но за счет некоторого снижения гибкости. Однако для множества приложений различия в скорости не так значимы, и снижение скорости выполнения почти полностью компенсируется гораздо более высокой скоростью разработки. Таким образом, процесс написания и отладки программы на Python занимает намного меньше времени, чем аналогичный процесс разработки на C, C++ или Java;
- трудно скрыть или замаскировать исходный код программы на Python, чтобы защитить ее от копирования и/или изменения. Но это не имеет особого значения, так как не существует успешных коммерческих программ на Python;
- на протяжении всей истории существования Python самыми частыми претензиями становились жалобы на излишне быстрое его развитие, приводящее к проблемам несовместимости между версиями. В действительности существуют два самых важных различия между версиями Python 2 и Python 3 (описанные в следующем разделе и в приложении Б), но претензии и жалобы основаны на том факте, что в группе версий Python 2 происходили основные усовершенствования и дополнения языка, из-за которых код, написанный в более поздней версии (например, в версии 2.7), не мог работать в более ранней версии (например, в версии 2.6), хотя код, написанный для более ранней версии Python, всегда будет работать в более поздней версии (в обеих ветвях – 2 и 3). Если вы используете самую последнюю версию Python (см. раздел 1.3), то, вероятнее всего, не столкнетесь с этой проблемой, но некоторые дистрибутивы операционных систем, в комплект которых входит Python, достаточно консервативны и устанавливают по умолчанию более старую версию.

1.2.2 Python 2 или Python 3

1 января 2020 года Python 2 завершил свой «жизненный путь»: он больше не будет обновляться и официально поддерживаться, поскольку более новая версия Python 3 уже активно развивается и сопровождается. Хотя различия между этими двумя версиями выглядят минимальными, код, написанный на Python 3, не будет работать в среде Python 2, и наоборот: Python 3 не обеспе-

чивает обратную совместимость со своим предшественником. В этой книге изучается Python 3.

Поскольку Python 3 появился в 2009 году, количество пользователей и поддержка библиотек для этой версии выросли до такого уровня, что новые пользователи не должны обнаружить особых преимуществ в изучении версии Python 2, за исключением необходимости поддержки старого кода.

Существует несколько обоснований внесения основных различий между версиями (с приведением в негодность пользовательского исходного кода не так-то легко примириться): Python 3 исправляет некоторые весьма неприятные особенности и нестыковки в языке, а также обеспечивает поддержку Юникода (Unicode) для всех строк (это полностью устраняет путаницу, возникавшую при обработке юникодных и неюникодных строк в Python 2). Unicode – это международный стандарт для представления текста в большинстве систем обработки и записи текстовых данных в мире.

Предвижу, что у большинства читателей этой книги не возникнет никаких проблем в процессе преобразования исходного кода между двумя версиями Python, если это будет необходимо. Список основных различий и дополнительную информацию см. в приложении Б.

1.3 Установка Python

Официальный сайт Python www.python.org содержит подробные и простые инструкции по загрузке (скачиванию) Python. Но существует несколько полноценных дистрибутивных комплектов, содержащих библиотеки NumPy, SciPy и Matplotlib (так называемый SciPy Stack). Эти дистрибутивы помогут избежать необходимости скачивания и установки библиотек по отдельности:

- пакет Anaconda доступен бесплатно (в том числе и для коммерческого использования) на сайте www.anaconda.com/distribution. Устанавливаются версии Python 2 и Python 3, но версию по умолчанию можно выбрать либо перед скачиванием, как указано на этой веб-странице, либо после скачивания с использованием команды `conda`;
- аналогичный дистрибутив Enthought Deployment Manager (EDM) существует в бесплатной версии и с различными компонентами для платных версий, включая техническую поддержку и программное обеспечение для разработки. Этот дистрибутив можно скачать здесь: https://assets.enthought.com/downloads/.

В большинстве случаев один из этих дистрибутивов – это все, что вам нужно. Ниже приведены некоторые замечания по отдельным платформам.

Исходный код (и бинарные файлы для некоторых платформ) для пакетов NumPy, SciPy, Matplotlib и IPython по отдельности доступен на следующих сайтах:

- NumPy: https://github.com/numpy/numpy;
- SciPy: https://github.com/scipy/scipy;
- Matplotlib: https://matplotlib.org/users/installing.html;
- IPython: https://github.com/ipython/ipython;
- Jupyter Notebook и JupyterLab: https://jupyter.org/.

Windows

Для пользователей Windows существует пара дополнительных возможностей установки полного стека SciPy: Python(x,y) (https://python-xy.github.io) и WinPython (https://winpython.github.io/). Оба дистрибутива бесплатные.

macOS

Операционная система macOS (бывшая Mac OS X), основанная на Unix, включает в комплект Python, но обычно более старой версии Python 2. Вы не должны удалять или обновлять этот вариант установки (он необходим операционной системе), но можно выполнить приведенные выше инструкции по установке Python 3 и стека SciPy. В macOS нет встроенного менеджера пакетов (приложения для установки и сопровождения программного обеспечения), но существуют два широко известных независимых менеджера пакетов: Homebrew (https://brew.sh/) и MacPorts (www.macports.org), которые поддерживают Python 3 и соответствующие пакеты, если вы предпочитаете данный вариант.

Linux

Почти все дистрибутивы Linux обычно содержат версию Python 2, а не Python 3, поэтому, возможно, потребуется установка новой версии по приведенным выше ссылкам: дистрибутивы Anaconda и Enthought предлагают специальные версии Linux. В большинстве дистрибутивов Linux имеется собственный менеджер программных пакетов (например, `apt` в Debian и `rpm` в RedHat). Менеджер пакетов можно использовать для установки Python 3 и всех необходимых библиотек, хотя при поиске репозиториев пакетов может потребоваться некоторое исследование ресурсов интернета. Будьте внимательны: не заменяйте и не обновляйте системный вариант установки, так как от него могут зависеть другие приложения.

1.4 Командная строка

Большинство примеров исходного кода в этой книге написаны как независимые программы, которые можно запускать из командной строки (command line) (или из интегрированной среды разработки (integrated development environment – IDE), если вы используете одну из таких сред: см. раздел 10.3.2). Для доступа к интерфейсу командной строки (также известного как консоль или терминал) на различных платформах выполните инструкции, приведенные ниже:

- Windows 7 и более ранние версии: **Пуск** > **Все программы** > **Командная строка**. Другой вариант: в окне ввода **Пуск** > **Выполнить** ввести команду `cmd`;
- Windows 8: **Preview** (нижний левый угол экрана) > **Windows System: All apps**. Другой вариант: ввод команды `cmd` в окне поиска, спускающегося из верхнего правого угла экрана;
- Windows 10: из меню **Пуск** (значок Windows в нижнем левом углу экрана) > **Служебные Windows** > **Командная строка**. Другой вариант: ввод команды `cmd` в окне поиска, вызываемого значком лупы в нижнем левом углу экрана рядом со значком Windows;

- Mac OS X и macOS: **Finder** > **Applications** > **Utilities** > **Terminal**;
- Linux: если вы не используете графический пользовательский интерфейс, то вы уже в командной строке. При использовании графического интерфейса найдите приложение Terminal (в разных дистрибутивах по-разному, но обычно терминал находится в разделе *System Utilities* или в разделе *System Tools*).

Команды, вводимые в командной строке, интерпретируются приложением, которое называется командной оболочкой (shell), позволяющей пользователю перемещаться по файловой системе и запускать разнообразные приложения. Например, команда

```
python myprog.py
```

сообщает командной оболочке о необходимости вызова интерпретатора языка Python с передачей в него файла *myprog.py* как скрипта для выполнения. Затем результат работы этой программы возвращается в командную оболочку и выводится в консоли (в терминале).

Глава 2

Ядро языка Python I

2.1 Командная оболочка Python

В этой главе рассматриваются основы синтаксиса, структуры и типы данных языка программирования Python. В нескольких первых разделах предполагается написание не более чем нескольких строк кода Python, которые могут быть выполнены с использованием командной оболочки (shell) языка Python. Это интерактивная рабочая среда: пользователь вводит инструкции на языке Python, которые выполняются немедленно сразу после нажатия клавиши **Enter**.

Действия, необходимые для получения доступа к встроенной командной оболочке языка Python, отличаются в различных операционных системах. Для запуска командной оболочки из командной строки сначала необходимо открыть окно терминала (консоли), воспользовавшись инструкциями из раздела 1.4, затем ввести команду `python`.

Для выхода из командной оболочки Python выполните команду `exit()`.

После запуска командной оболочки Python вы увидите приветственное сообщение (которое может меняться в зависимости от используемой операционной системы и версии Python). В моей системе это сообщение выглядит так:

```
Python 3.7.5 (default , Oct 25 2019, 10:52:18)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda , Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Три правые угловые скобки (`>>>`) – это приглашение, или промпт (prompt), после этих символов приглашения вы будете вводить команды языка Python. В данной книге используется версия Python 3, поэтому вы должны проверить номер версии в первой строке – `Python 3.X.Y`: здесь не особенно важны значения `X` и `Y`, представляющие минорный номер версии.

Многие дистрибутивные пакеты Python содержат немного более продвинутую командную оболочку, которая называется IDLE, с дополнительными функциями завершения по нажатии клавиши **Tab** и подсветкой синтаксиса (syntax highlighting) (ключевые слова выделяются различными цветами, когда вы вводите их). Но мы не будем рассматривать это приложение, а предпочтем ему более новую и более усовершенствованную рабочую программную среду IPython, обсуждаемую в главе 5.

Кроме того, для многих вариантов установки существует возможность (особенно в Windows) запуска командной оболочки Python непосредственно из приложения, установленного в процессе установки самого интерпретатора Python. Некоторые варианты установки даже добавляют ярлык со значком на рабочий стол, так что можно открыть командную оболочку Python щелчком (или двойным щелчком) по этому ярлыку.

2.2 Числа, переменные, операции сравнения и логические операции

2.2.1 Типы числовых значений

Одним из самых простых объектов языка Python являются числовые значения, для которых определены три типа: целые числа (тип: int), числа с плавающей точкой (тип: float) и комплексные числа (тип: complex).

Целые числа

Целые числа (integers) – это математические целые числа, такие как 1, 8, –72 и 3 847 298 893 721 407. В версии Python 3 нет ограничений по их величине (кроме ограничения по доступности оперативной памяти компьютера). Арифметика целых чисел является точной.

Для удобства разрешается разделять группы цифр (разрядов) целого числа символом подчеркивания «_». Например, запись `299_792_458` интерпретируется как целое число `299 792 458`.

Числа с плавающей точкой

Числа с плавающей точкой (floating-point numbers) являются представлением действительных чисел, таких как 1.2, –0.36 и $1\ 67263 \times 10^{-7}$. Вообще говоря, не существует точных значений действительных чисел в представлении Python, но эти числа хранятся в бинарном (двоичном) виде с определенной точностью (в большинстве систем точность составляет 15–16 разрядов)[3], как описано в разделе 10.1. Например, дробь 4/3 хранится как двоичное равнозначное представление 1.33333333333333325931846502..., которое приблизительно (но не точно) равно бесконечно повторяющемуся десятичному представлению дроби 4/3 = 1.3333.... Более того, даже числа, для которых существует точное десятичное представление, могут не иметь точного бинарного представления: например, дробь 1/10 представлена бинарным числом, равным значению 0.10000000000000000555111512.... Из-за такой ограниченной точности арифметика чисел с плавающей точкой не является абсолютно точной, но при аккуратном отношении она «достаточно точна» для большинства научных приложений.

Любое отдельное число, содержащее символ точки (.), рассматривается в Python как представление числа с плавающей точкой. Также поддерживается научный формат записи чисел с использованием буквы `e` или `E` для отделения

[3] Это соответствует реализации по международному стандарту представления чисел с плавающей точкой двойной точности IEEE-754.

значимой части числа (мантиссы) от показателя степени: например, `1.67263e-7` представляет число $1\ 67263 \times 10^{-7}$.

Как и для целых чисел, группы разрядов можно разделять символами подчеркивания. Например, `1.602_176_634e-34`.

Комплексные числа

Комплексные числа, такие как 4+3j, состоят из действительной и мнимой частей (в Python мнимая часть обозначается буквой `j`), каждая из которых сама по себе является числом с плавающей точкой (даже если записана без символа точки). Таким образом, арифметика комплексных чисел не является точной, но обеспечивает такую же конечную ограниченную точность, что и числа с плавающей точкой (`float`).

Комплексное число может быть определено «сложением» действительного числа с мнимым (обозначенным суффиксом `j`): `2.3 + 1.2j` – или посредством отдельной передачи действительной и мнимой частей при вызове `complex`, например `complex(2.3, 1.2)`.

Пример П2.1. Ввод числа после приглашения командной оболочки Python просто возвращает это же число

```
>>> 5
5
>>> 5.
5.0
>>> 0.10
0.1                                                    ❶
>>> 0.0001
0.0001
>>> 0.0000999
9.99e-05                                               ❷
```

Следует отметить, что интерпретатор Python выводит числа стандартным способом. Например:

❶ Внутреннее представление числа 0.1, описанное выше, округляется до `0.1`, т. е. до самого короткого числового значения в этом представлении.

❷ Числа, меньшие по величине, чем 0.0001, выводятся в научном формате.

Число любого типа может быть создано из числа другого типа с помощью соответствующего конструктора (constructior):

```
>>> float(5)
5.0
>>> int(5.2)
5
>>> int(5.9)
5                                                      ❶
>>> complex(3.)
(3+0j)                                                 ❷
>>> complex(0., 3.)                                    ❸
3j
```

❶ Обратите внимание: положительное число с плавающей точкой округляется с недостатком (в меньшую сторону) во время преобразования его в целое число. Более общее правило: метод `int` округляет в сторону нуля: `int(-1.4)` дает результат `-1`.
❷ Создание объекта типа `complex` из объекта типа `float` генерирует комплексное число с мнимой частью, равной нулю.
❸ Для генерации абсолютно мнимого числа необходимо явно передать два числа в метод `complex`, при этом первая, действительная часть должна быть равна нулю.

2.2.2 Использование командной оболочки Python в качестве калькулятора

Простые арифметические действия

С описанными в предыдущем разделе тремя основными типами чисел можно использовать командную оболочку Python как простой калькулятор, применяя операторы, описанные в табл. 2.1. Это бинарные операторы, поскольку они работают с двумя числами (операндами) для создания третьего числа (например, при вычислении `2**3` получается результат `8`).

Таблица 2.1. Основные арифметические операторы языка Python

+	Сложение
–	Вычитание
*	Умножение
/	Деление с плавающей точкой
//	Целочисленное деление
%	Деление по модулю (взятие остатка)
**	Возведение в степень

В версии Python 3 существует два типа операции деления: деление чисел с плавающей точкой (/) всегда возвращает как результат число с плавающей точкой (или комплексное число), даже если применяется к целым числам. Целочисленное деление (//) всегда округляет результат с недостатком (в меньшую сторону), т. е. к ближайшему меньшему целому числу (floor division). Типом результата является `int`, только если оба операнда имеют тип `int`, иначе возвращается значение типа `float`. Ниже приведены примеры, помогающие понять эти правила.

Обычное («истинное») деление с плавающей точкой с использованием оператора (/):

```
>>> 2.7 / 2
1.35
>>> 9 / 2
4.5
>>> 8 / 4
2.0
```