Сэму, Оскару, Аве, Максу, Марго, Алексу и Коулу

Содержание

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, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и «Apress» очень серьезно относятся 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 электронной почты dmkpress@gmail.com.

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

Благодарности

Здесь будут фамилии тех, кто помогал изданию этой книги, прислав в издательство найденные ошибки или ссылку на подозрительные материалы.

Об авторе

Бернард Коритес имеет ученые степени университета Тафтса и Йельского университета. Всю свою карьеру он занимался инженерными и научными применениями компьютеров. Он был преподавателем, консультантом и автором более десяти книг по геометрическому моделированию, компьютерной графике, моделированию физических процессов и применению компьютеров в науке и технике.

Работал в компаниях Northrop Aviation, Океанографическом институте Вудс-Хоул, Arthur D. Little и Itek. Консультировал BMC США, Абердинский полигон и другие организации. Он был главным инженером экспедиции по отбору самого длинного образца керна отложений в Северной Атлантике на борту канадского ледокола «Джон Кэбот». Исследовал донную среду на борту подводного корабля «Элвин» и провел время на борту USBCF «Альбатрос». В начале своей карьеры он разработал программное обеспечение для обнаружения физических взаимодействий между системами твердых объектов. Это нашло широкое применение при проектировании энергетических установок, подводных лодок и других систем с плотно расположенными пространствами.

Он будет признателен за комментарии, советы и предложения по поводу этой книги или любого связанного с ней вопроса. На его веб-сайте www.korites.com есть вся контактная информация.

О техническом рецензенте

Андреа Гавана программирует на Python более 20 лет, а с конца 1990-х увлекается другими языками. Он окончил университет со степенью магистра химического машиностроения и в настоящее время является главным архитектором планирования развития, работая в компании TotalEnergies в Копенгагене, Дания.

Андреа любит программировать на работе и для развлечения, он участвовал во многих проектах с открытым исходным кодом, все на основе Python. Программирование на Python – одно из его любимых хобби, но он также любит кататься на велосипеде, плавать и ужинать с семьей и друзьями в уютной обстановке. Это его третья книга в качестве технического рецензента.

Благодарности

Я хотел бы поблагодарить Шонмирин П. А. из Apress за ее превосходную помощь и поддержку на протяжении всего процесса написания этой книги, а также Марка Пауэрса из Apress за его обнадеживающие комментарии, которые пришли как раз в нужное время.

Введение

В этой книге идет речь о том, как использовать встроенные графические примитивы Python (точки, линии и стрелки) для создания простой или сложной графики для визуализации двух- и трехмерных объектов, наборов данных и технических иллюстраций. Она предназначена для разработчиков Python, которые хотят создавать свои собственные графические изображения, а не ограничиваться функциями, доступными в существующих библиотеках Python.

В книге показано, как создать практически любой 2D- или 3D-объект либо иллюстрацию. Она показывает, как лучше представлять изображения; использовать цвет; перемещать, поворачивать, затенять объекты и добавлять тени, падающие от них на другие объекты; удалять скрытые линии; строить 2D- и 3D-данные; адаптировать линии, кривые и функции к наборам данных; отображать точки пересечения 2D- и 3D-объектов и таким образом создавать цифровые произведения искусства.

Книга также содержит приложения различных наук, включая астрономию, физику, демографию, изменение климата и управление природными ресурсами. Эти приложения предназначены для иллюстрации методов графического программирования на примерах, что является лучшим способом изучения языка.

Для всех иллюстраций в книге приводится исходный код Python с пояснениями.

Вооружившись примитивными графическими элементами Python и методами, описанными в этой книге, а также базовыми математическими навыками, в первую очередь геометрией, вы будете готовы создавать и настраивать детальные иллюстрации и визуализации данных.

Глава 1

Основные команды и функции Python

В этой главе вы изучите основные команды и функции Python, которые понадобятся вам для создания иллюстраций, приведенных в данной книге. Вы научитесь использовать базовые функции построения графиков Python, настраивать область построения, создавать набор двумерных осей координат и применять базовые примитивы построения графиков (точку, линию и стрелку), являющиеся строительными блоками, с которыми вы будете работать для создания изображений на протяжении всей книги. В главе 2 вы узнаете, как обращаться с этими примитивами для создания двумерных изображений, а затем перемещать и поворачивать их. В главе 3 вы расширите эти методы на три измерения. Вы также узнаете о цветах, о том, как применять текст к графикам, включая использование команд LaTeX, а также использование листингов и массивов. К последней главе вы сможете создавать изображения, подобные изображениям Сатурна на рис. 1.1a и 1.1b. Они были созданы программой SATURN в главе 10 и показывают Сатурн в различной ориентации относительно Солнца.

Рис. 1.1a ❖ Сатурн

Рис. 1.1b ❖ Сатурн

Стиль программирования

Прежде всего несколько слов о стиле программирования, используемом в этой книге. У всех нас есть предпочтения, когда дело касается этого стиля. Я предпочитаю ясный, открытый и нисходящий стиль[1]. Многие программисты стараются сократить свой код до как можно меньшего количества строк. На практике это может быть хорошо, но в учебном тексте, таком как этот, я считаю, что лучше действовать медленно, небольшими, простыми шагами. Цель состоит в том, чтобы все было открыто, ясно и понятно. Это также облегчает поиск ошибок в коде Python во время разработки новой программы. Поскольку я не знаю уровня навыков читателей и хочу сделать эту книгу доступной как можно более широкой аудитории, то обычно начинаю каждую тему с элементарного уровня, хотя и предполагаю некоторое знакомство с языком Python. Если вы только изучаете Python, вам будет полезен материал первой главы. Прочитав эту книгу, вы узнаете больше о возможностях Python. Они будут разъясняться по мере использования. Некоторые разработчики Python рекомендуют использовать длинные описательные имена для переменных, таких как `temperature`, а не `T`. Я считаю, что слишком длинные имена переменных затрудняют чтение кода. Это вопрос предпочтений. При использовании относительно коротких программ, таких как представленные в этой книге, нет необходимости в сложном программировании. Постарайтесь выбрать простой, но прочный стиль, а не элегантный, но хрупкий.

---

[1] Подход «сверху вниз» представляет собой разбиение системы на части для получения понимания ее композиционных подсистем методом обратного проектирования. При нисходящем подходе формулируется обзор системы с указанием, но не детализацией, любых подсистем первого уровня. Затем каждая подсистема уточняется еще более подробно, пока вся спецификация не сводится к базовым элементам. Подход «сверху вниз» начинается с общей картины, а потом разбивается на более мелкие сегменты. – *Прим. ред.*

Как вы увидите на протяжении всей книги, мои программы обычно имеют схожую структуру. Ниже показана типичная структура на примере программы DOTART:

```
1 " " "
2 DOTART
3 " " "
4
5 import matplotlib.pyplot as plt
6 import numpy as np
7 import random
8
'
' program statements
'
52 plt.show()
```

Оператор `import matplotlib.pyplot as plt` добавляет в программу библиотеку функций, полезных при построении графиков. Образцы можно увидеть в программе DOTART в строках 9–17 и в других местах программы. Обратите внимание, что каждой функции в теле программы предшествует `plt`, который мы определили в строке 5 выше как сокращение для matplotlib, как в строке `11 plt.axis()` и строке `22 plt.scatter()`. Функции `axis()` и `scatter()` включены в библиотеку `matplotlib`.

Содержимое библиотеки matplotlib можно найти по адресу https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html. В библиотеке более 130 функций. Стоит просмотреть их, чтобы увидеть, какие вы можете использовать в своих программах.

Аналогично `import numpy as np` добавляет библиотеку математических функций. Каждой функции из `numpy` должен предшествовать `np.`, как в строке 25 для `x in np.arange()`. Библиотеку numpy можно найти по адресу https://numpy.org/doc/stable/.

Библиотека `random` – это библиотека случайных функций. Когда одна из этих функций используется в программе, ей должен предшествовать `np.`, как в программе DOTART, строки 42, 43 и 44. Описание модуля ramdom можно найти по адресу www.geeksforgeeks.org/python-random-module/.

Чтобы избежать использования префикса (т. е. `plt.`, `np.`, `random.`) перед функциями, вы можете импортировать функции напрямую. Например, если посмотреть на программу 4BOXES, строка 7, оператор `from math import sin(), cos, radians` импортирует `sin, cos` и `radians`. Когда функции таким образом импортируются напрямую, нет необходимости использовать префикс при их применении в программе. Я часто импортирую из математической библиотеки явно с помощью оператора, например `from math import sin, cos, radians, sqrt`. Тогда я смогу использовать эти функции в программе без префикса.

После импорта необходимых функций я чаще всего определяю область построения посредством `plt.axis([0,150,100,0])`. Как объяснено в разделе 1.2, эти значения `[0,150,100,0]`, где ось x (150) на 50 % шире, чем ось y (100), создают круг и квадрат без уменьшения размера области построения, по крайней

мере на моем компьютере и мониторе, делают математически правильное описание круга и квадрата. Компьютерные системы часто различаются по интервалу горизонтальных и вертикальных растров, поэтому вам, возможно, придется поэкспериментировать с этими значениями в вашей настройке. Позже вы увидите множество примеров кругов и квадратов. На этом этапе при желании можно разметить оси координат и заголовок графика. Далее я обычно определяю параметры (такие как диаметры, постоянные времени и т. д.) и составляю списки. Списки будут обсуждаться в разделе 1.19.5. Затем я определяю пользовательские функции (функции, которые вы определяете самостоятельно), если таковые имеются. На данном этапе в этом может не быть необходимости, поскольку пользовательские функции могут быть определены в любой момент в самой программе с помощью функции `def`, как показано в программе LTP, строки 28, 31, 34, 38, 56, 156 и 174. Другой пример: в программе PERSPECTIVE строки 32–47 создают функцию с именем `plothouse()`, которая строит дом. Необходимо построить 15 ребер, для каждого из которых требуется одна строка кода. Поскольку дом требует построения изображения до и после поворота и преобразования перспективы, использование функции `plothouse()` позволяет сэкономить на вводе 15 строк кода, тогда как для вызова функции `plothouse()` после ее определения требуется всего одна строка.

Наконец, в длинных программах, использующих необязательные входные данные, я могу разместить внизу раздел управления, вызывающий параметры. См. программу KEYBOARDDATAENTRY, строки 115–127.

Что касается осей координат, они необходимы в качестве ориентира для построения графиков. `plt.axis('on')` строит оси; `plt.grid(True)` строит сетку. Это очень удобные варианты при разработке графики. Однако если я не хочу, чтобы оси или сетка отображались в окончательном выводе, то заменяю эти команды на `plt.axis('off')` и `plt.grid(False)`. Синтаксис должен быть таким, как показано здесь, включая кавычки или их отсутствие, а также использование заглавных букв в `True` и `False`. См. раздел 1.10, чтобы узнать, как создавать собственные линии сетки, если вас не устраивают настройки Python по умолчанию.

Я часто начинаю разработку графики с использования функции `scatter()`, которая создает то, что я называю *точками рассеяния*[1]. Функцию `scatter()` необходимо импортировать напрямую, как объяснялось выше, или использовать с префиксом, как объяснялось ранее. Точки рассеяния быстры и просты в использовании и очень полезны на этапе разработки. Если точки достаточно малы и расположены близко друг к другу, из них можно получить удобочитаемые линии и кривые, хотя иногда они могут выглядеть немного размытыми. Итак, после того как у меня все начинает работать правильно, я часто возвращаюсь и заменяю точки короткими отрезками линий, используя либо стрелки через `plt.arrow()`, либо линии через `plt.plot()`. Как объяснено в разделе 1.19.3, стрелку можно преобразовать в линию, исключив параметры, определяющие размер ее наконечника.

---

[1] Элементы построения линий или сплошных фигур, а также фона диаграммы. – *Прим. ред.*

Есть еще один аспект выбора точек или линий – наложение объектов: что накладывается на что? Вряд ли вы захотите создавать что-то с точками, а затем обнаружить линии, закрывающие вашу работу. Это обсуждается в разделе 1.14.

Некоторые варианты Python требуют оператора `plt.show()` в конце программы для построения графики. Моя установка, Anaconda со Spyder и Python 3.5 (инструкции по установке см. в приложении А), не требует этого, но я все равно включаю его, поскольку он служит удобным маркером конца программы. Наконец, нажмите клавишу **F5** или кнопку **Run** вверху, чтобы увидеть, что вы создали. После того как вы будете удовлетворены, вы можете сохранить график, наведя на него курсор и кликнув правой кнопкой мыши. Затем укажите, где вы хотите его сохранить.

Что касается использования списков, кортежей и массивов, они могут оказаться большим подспорьем, особенно при графическом программировании, включающем большое количество точек данных. Они объяснены в разделе 1.19.5. Их понимание, а также несколько основных графических команд и приемов, описанных в этой главе, – это все, что вам нужно для создания иллюстраций и изображений, которые вы видите в этой книге.

Область построения диаграмм

Дисплей компьютера с двумерной системой координат показан на рис. 1.2. В этом примере начало осей координат x,y (x=0, y=0) расположено в центре экрана. Положительная ось x проходит от начала координат вправо; ось y проходит от начала координат вертикально вниз. Как вы вскоре увидите, в вашей программе можно изменить местоположение начала координат, а также направления осей x и y. Также показана точка p в координатах (x,y), которые относятся к осям x и y.

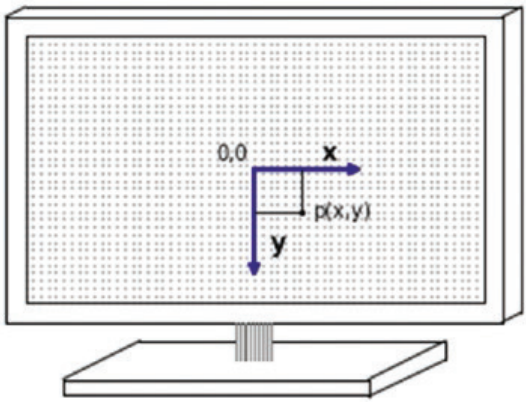

Рис. 1.2 ❖ Двумерная система координат x,y с началом координат (0,0) в центре экрана. Точка p показана в координатах (x,y) относительно x,y

Направление оси Y вниз на рис. 1.2 может показаться немного необычным. При построении графика данных или функций, таких как `y=cos(x)` или `y=exp(x)`, мы обычно думаем, что ось y направлена вверх. Но при создании технической графики, особенно в трех измерениях, как вы увидите позже, более интуитивно понятно, чтобы ось y была направлена вниз. Это также соответствует старым вариантам BASIC, где ось x проходила вдоль верхней части экрана слева направо, а ось y – вниз по левой стороне. Как вы увидите, вы все равно можете определить направление y вверх или вниз, в зависимости от того, что лучше всего соответствует представляемому вами материалу.

Определение размера области построения диаграмм

Область построения диаграмм содержит графическое изображение. Она всегда имеет одинаковый *физический* размер при отображении на панели вывода Spyder. Spyder – это среда программирования (см. приложение А). Однако *числовой* размер области построения и значений, определяющих определения точки (точки), линии и стрелки в области построения, может быть любым. Прежде чем приступить к построению графика, вы должны сначала установить числовой размер области. Это отличается от физического размера на вашем мониторе. Например, ваш монитор может иметь физическую ширину 20 дюймов, но числовые размеры вашей диаграммы могут составлять тысячи километров. Также необходимо указать местоположение начала системы координат и направления осей координат. В качестве иллюстрации в листинге 1.1 показана программа PLOTTING_AREA, которая использует функцию `plt.axis([x1,x2,y1,y2])` в строке 8 для настройки области от x=-10 до +10; y=-10 до +10. Остальная часть скрипта будет объяснена в ближайшее время.

Кстати, все изображения в этой книге, такие как монитор выше, были созданы с использованием Python.

Листинг 1.1 ❖ Программа PLOTTING_AREA

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x1=-10
5 x2=10
6 y1=-10
7 y2=10
8 plt.axis([x1,x2,y1,y2])
9
10 plt.axis('on')
11 plt.grid(True)
12
13 plt.show()
```

Листинг 1.1 создает область построения, показанную на рис. 1.3. Она имеет ширину по горизонтали 20 и высоту по вертикали 20. Я мог бы сделать эти числа 200 и 200, и область будет отображаться на панели вывода с тем же же физическим размером, но с другими числовыми значениями по осям. Строка 13 содержит команду `plt.show()`. Цель этой команды – отобразить результаты работы программы на панели вывода. В современных версиях Python это не требуется, поскольку графики автоматически отображаются при запуске программы. В более старых версиях они могут отображаться, а могут и не отображаться. Команду `plt.show()` также можно поместить в тело программы, чтобы показывать графики, создаваемые по мере ее выполнения. Даже если в этом нет необходимости, рекомендуется включить команду `plt.show()` в конец вашего скрипта, поскольку она служит удобным маркером окончания вашей программы. Строки 1, 2, 10 и 11 в листинге 1.1 обсуждались ранее, но будут объяснены далее в следующих разделах. Эти команды или их варианты будут присутствовать во всех наших программах, создающих графику.

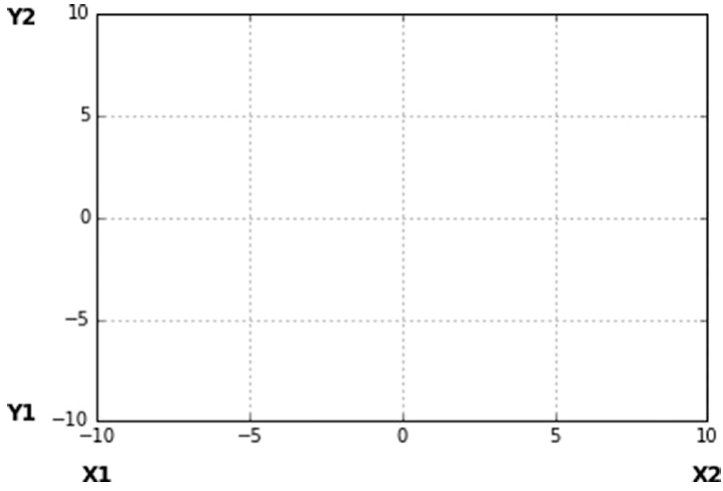

Рис. 1.3 ❖ Область построения, полученная в листинге 1.1 с началом координат (0,0), расположенным в центре

Импорт команд построения диаграмм

Хотя в Python имеется множество встроенных команд и функций, чтобы просмотреть то, что вы узнали выше, необходимо импортировать некоторые математические и графические команды. Это делают строки 1 и 2 в листинге 1.1. Оператор `import numpy as np` в строке 1 импортирует математические функции, такие как `sin(φ)`, `exp()` и т. д. Сокращение `np` в этом выражении – аббревиатура, которую можно использовать при ссылке на функцию `numpy`.

При использовании в программе эти функции должны быть идентифицированы как исходящие из numpy. Например, если вы хотите закодировать v=e^α, оператор программы будет v=np.exp(α), где α было определено ранее. Вам не нужно записывать полную длину numpy.exp(α), поскольку вы определили сокращение np для numpy в строке 1. Для этого подойдет np.numpy. Графические команды обрабатываются аналогично. Оператор import matplotlib.pyplot as plt импортирует библиотеку pyplot, содержащую графические команды. plt – это сокращение от pyplot. Например, если вы хотите построить точку в координатах x,y, вы пишете plt.scatter(x,y). Я расскажу подробнее о plt.scatter() в ближайшее время.

Функции также можно импортировать непосредственно из numpy. Оператор from numpy import sin, cos, radians импортирует функции sin(), cos() и radians(). При явном импорте таким способом их можно использовать без префикса np, как в x=sin(alpha). Существует еще библиотека math, которая работает аналогичным образом. Например, from math import sin, cos, radians эквивалентно импорту из numpy. Вы будете использовать все эти варианты в следующих программах.

Существует также графическая библиотека glib, содержащая графические команды. Библиотека glib использует другой синтаксис, чем pyplot. Поскольку pyplot применяется более широко, вы будете использовать здесь в своей работе именно ее.

Строка 8 в листинге 1.1, plt.axis([x1,x2,y1,y2]), представляет собой стандартную форму команды, которая устанавливает область построения. Она из библиотеки pyplot, поэтому вначале стоит префикс plt.. У этой команды есть параметры, а также другие способы определения области печати, в частности команда linspace(), но форма в строке 8 достаточна для большинства целей, и именно ее вы и будете использовать. Параметры x1 и x2 определяют значения левой и правой сторон области построения соответственно; y1 и y2 определяют низ и верх соответственно. Используя числовые значения в строках 8–11, вы получаете область построения, показанную на рис. 1.3. Точки x1, x2, y1 и y2 всегда имеют местоположения, показанные на рис. 1.3, если они определены с помощью функции plt.axis(x1,x2,y1,y2), как в строке 8. То есть x1 и y1 всегда находятся в нижнему левом углу, y2 на другом конце оси y, точка x2 на другом конце оси x. Их значения могут меняться, но они всегда находятся в этих местах. Они могут быть и отрицательными, как показано на рис. 1.4.

Поскольку значения x и y, указанные в строках 4–7, симметричны в обоих направлениях x и y (т. е. –10, +10), эта область построения имеет точку (x=0, y=0) посередине между ними. В этом случае центр области будет началом координат, используемым в качестве опоры для построения системы координат. Так как x1 < x2, положительное направление оси x – горизонтально, слева направо. Аналогично, поскольку y1 < y2, положительное направление оси y – вертикально вверх. Но ранее я сказал, что нам нужно, чтобы положительное направление y было вертикально вниз. Это можно сделать, изменив значения y на y1=10, y2=-10. В этом случае вы получите область, показанную на рис. 1.4, где положительная ось X по-прежнему идет слева направо, но положительная ось Y теперь направлена вниз. Центр, как и прежде, находится в середине области построения.

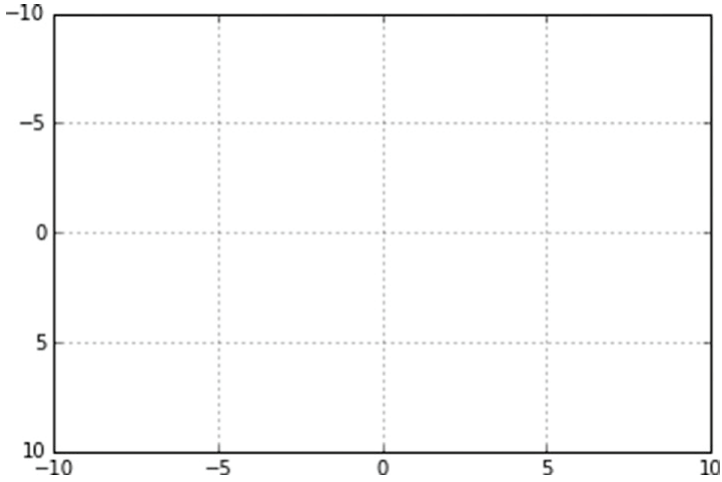

Рис. 1.4 ❖ Область построения с началом координат (0,0), расположенным в центре, положительное направление y – вниз

Вы можете переместить начало системы координат из центра, манипулируя x1, x2, y1 и y2. Например, чтобы переместить точку x=0 до предела влево, вы можете указать x1=0, x2=20. Чтобы переместить точку (x=0, y=0) в левый нижний угол, вы можете указать x1=0, x2=20, y1=0, y2=20. Но это сделало бы направление y указывающим вверх; а нам нужно, чтобы ось y была направлена вниз, что можно сделать, задав y2=0, y1=20. Это приведет к тому, что начало координат появится в *верхнем левом* углу. Вы можете разместить точку (0,0) где угодно, изменить направление положительных значений x и y, а также масштабировать числовые значения осей координат в соответствии с изображением, которое пытаетесь создать. Числовые значения, которые вы здесь используете, могут быть любыми. Физический размер графика, созданного Python, не изменится; изменятся только значения координат на нем.

Отображение области построения диаграммы

В строке 10 листинга 1.1 оператор `plt.axis('on')` отображает область построения с ее рамкой и числовыми значениями. Если вы пропустите эту команду, рамка по-прежнему будет отображаться с числовыми значениями. Так зачем включать эту команду? Потому что при создании диаграммы иногда желательно отключить рамку. Для этого замените `plt.axis('on')` на `plt.axis('off')`. Наличие команды заранее позволяет легко набирать `'off'` вместо `'on'` и наоборот, чтобы включать и отключать показ рамки. Кроме того, после того как вы закончили работу с диаграммой, вам может понадобиться использовать ее в документе, и в этом случае рамка вам может не понадо-

биться. Обратите внимание, что `'on'` и `'off'` должны заключаться в кавычки, **одинарные или двойные**.

Сетка построения диаграмм

Строка 11 листинга 1.1, `plt.grid(True)` включает пунктирные линии сетки, которые могут помочь при построении графика, особенно когда приходит время позиционировать текстовую информацию. Если вы не включите эту команду, линии сетки не будут отображаться. Чтобы отключить линии сетки, измените значение `True` на `False`. Обратите внимание, что первая буква в словах `True` и `False` написана с заглавной буквы. `True` и `False` *не заключаются* в кавычки. Как и в случае с `plt.axis()`, наличие команд `plt.grid(True)` и `plt.grid(False)` позволяет легко переключать режим показа.

Сохранение диаграммы

Самый простой способ сохранить диаграмму, которая появляется на панели вывода, – это навести на нее курсор и кликнуть правой кнопкой мыши. Появится окно, в котором вы сможете дать ей имя и указать, где ее следует сохранить. Она будет сохранена в формате .png, который представляет собой файл растрового изображения. Если вы планируете использовать его в такой программе, как Photoshop, формат .png вполне подойдет. Но некоторым программам обработки текстов и документов может потребоваться файл .eps. Если это так, сохраните ее в формате .png, далее откройте в программе, способной конвертировать форматы файлов, например Photoshop[1], и повторно сохраните в формате .eps. Вы также можете конвертировать ее в .jpg, тоже файл растрового изображения. Этот тип файла может называться и как .jpg, и как .jpeg.

Единственная причина, по которой расширение файла .jpg имеет длину три символа, а не четыре, заключается в том, что ранние версии Windows требовали трехбуквенного расширения для имен файлов. На всякий случай используйте .jpeg, поскольку эта форма сейчас применяется более широко.

Цвет сетки

У команды `plt.grid()` есть несколько опций. Вы можете изменить цвет линий сетки с помощью атрибута `color='color'`. Например, `plt.grid(True, color='b')`

[1] Также Adobe Illustrator или GIMP. – *Прим. ред.*

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru