

Роман Сузи



www.bhv.ru

www.bhv.kiev.ua

Python

- Библиотеки Python Image Library и Numeric Python
- Работа с базами данных
- Среда разработки Web-приложений Zope

Наиболее

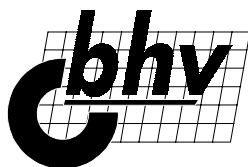
полное

руководство

В ПОДЛИННИКЕ

Роман Сузи

PYTHON



Санкт-Петербург

Дюссельдорф ♦ Киев ♦ Москва ♦ Санкт-Петербург

УДК 681.3.06

Описываются синтаксис и семантика типов данных, операций, конструкций и операторов языка Python; функции, специальные методы классов и исключения. Рассматриваются расширения языка: библиотека Numeric Python, позволяющая эффективно работать с массивами числовых данных, и библиотека Python Image Library, предназначенная для обработки растровых изображений. Обсуждаются работа с базами данных, возможности разработки графического интерфейса пользователя, интеграция программного продукта с модулями на языках SQL, Tcl/Tk, C, C++ и Java. Особое внимание уделено средствам Web-программирования.

Для программистов и Web-разработчиков

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Руководитель проекта	<i>Анатолий Адаменко</i>
Зав. редакцией	<i>Анна Кузьмина</i>
Редактор	<i>Семен Штейнер</i>
Компьютерная верстка	<i>Натальи Смирновой</i>
Корректор	<i>Наталья Першакова</i>
Дизайн обложки	<i>Игоря Цырульниковца</i>
Зав. производством	<i>Николай Тверских</i>

Сузи Р. А.

Python. — СПб.: БХВ-Петербург, 2002. — 768 с.: ил.

ISBN 5-94157-097-X

© Р. А. Сузи, 2002

© Оформление, издательство "БХВ-Петербург", 2002

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 17.12.01.

Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 61,92.

Тираж 3000 экз. Заказ №

"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.1.953.П.950.3.99 от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9 линия, 12.

Содержание

Предисловие	1
Введение	3
1. Как построена эта книга	3
2. Используемые обозначения	5
Часть I. Основы программирования	9
Глава 1. Интерактивный Python	11
1.1. Запускаем Python	11
1.2. Диалог с Python	12
1.3. Python как калькулятор	14
Глава 2. Первые программы	19
2.1. "Здравствуй, мир!"	19
2.2. Арифметика	20
2.3. Немного о функциях	21
2.4. Строки	23
2.5. Условия	26
2.6. Немного о логике	29
2.7. Циклы	31
2.8. Пишем комментарии	35
2.9. Учимся использовать списки	36
2.10. Работаем с текстом	40
2.11. Знакомимся с регулярными выражениями	41
2.12. Словари	47
2.13. Осваиваем форматированный вывод	50
2.14. Изучаем работу с файлами	54
2.15. Обработка ошибок	57
Глава 3. Функциональное программирование	60
3.1. Чистые функции и их композиция	60
3.2. Обработка последовательностей	64
3.3. Рекурсия	67
3.4. Функции над функциями	71
3.5. Функции могут все	74
Глава 4. Строим модули	77
4.1. Что такое модуль	77
4.2. Маленький проект	79

4.3. Интерфейсы модулей.....	83
4.4. Разрабатываем модули.....	86
4.5. Итоги нашего проекта.....	95
Глава 5. Объектный подход.....	96
5.1. Терминология.....	96
5.2. Наследование и агрегация.....	100
5.3. Используем готовые объекты.....	103
5.4. Строим свои классы.....	106
5.5. Модификация объектного проекта.....	112
5.6. Устойчивые объекты.....	114
5.7. Что дальше?.....	116
Глава 6. Графический интерфейс пользователя.....	117
6.1. Простой пример.....	118
6.2. Программа "Экзаменатор".....	120
6.3. Игра "Сапер".....	128
Часть II. ОПИСАНИЕ ЯЗЫКА PYTHON.....	135
Глава 7. Общие сведения о Python.....	137
7.1. История языка Python.....	137
7.2. О лицензиях.....	138
7.3. Области применения.....	139
7.4. Где взять Python и программное обеспечение.....	141
7.5. Установка Python.....	142
7.6. Где получить помощь.....	143
Глава 8. Описание языка.....	145
8.1. Структура программы.....	145
8.1.1. Физические и логические строки программы.....	146
8.1.2. Комментарии и пустые строки.....	147
8.1.3. Отступы и пробелы.....	149
8.2. Типы данных и объекты.....	151
8.2.1. Типы <i>None</i> и <i>Ellipsis</i>	154
8.2.2. Числа.....	154
8.2.3. Последовательности.....	157
Строки.....	158
Кортежи.....	162
Списки.....	163
8.2.4. Словари.....	167
8.2.5. Типы с поддержкой вызова.....	170
Функции.....	171
Методы.....	174
8.2.6. Классы и объекты.....	176
8.2.7. Модули.....	181

8.2.8. Тип	182
8.2.9. Файлы	183
8.2.10. Другие встроенные типы	186
Объекты-срезы	187
Кодовые объекты	188
Фрейм-объекты	189
Трассировочные объекты	189
8.3. Выражения	190
8.3.1. Атомы	195
Идентификаторы	195
Константы (литералы)	197
Задание кортежей, списков, словарей	201
Преобразование к строке	203
8.3.2. Первичные выражения	204
Ссылка на атрибут	204
Индексные выражения	205
Вызовы функций и других объектов	205
8.3.3. Операции	207
Арифметические операции	207
Операция форматирования	209
Логические операции	212
Сравнения	214
Побитовые операции	216
8.3.4. Лямбда-выражение	219
8.4. Специальные имена методов	220
8.4.1. Базовая настройка	220
Инициализация объекта: метод <code>__init__()</code>	220
Деструктор объекта: метод <code>__del__()</code>	221
Представление объекта в виде строки: методы <code>__repr__()</code> и <code>__str__()</code>	222
Сравнение объектов	223
Хэш-метод <code>__hash__()</code>	224
Проверка на истинность: метод <code>__nonzero__()</code>	225
8.4.2. Настройка доступа к атрибутам	226
Чтение атрибута: метод <code>__getattr__()</code>	226
Запись атрибута: метод <code>__setattr__()</code>	227
Удаление атрибута: метод <code>__delattr__()</code>	228
8.4.3. Эмуляция числовых типов	228
Методы для арифметических операций	228
Методы для побитовых операций	230
Методы для организации преобразований типов	230
Методы <code>__oct__()</code> и <code>__hex__()</code>	231
8.4.4. Эмуляция последовательностей и отображений	231
Метод <code>__len__()</code>	232
Метод <code>__getitem__()</code>	232
Метод <code>__setitem__()</code>	234
Метод <code>__delitem__()</code>	234
Методы <code>__getslice__()</code> , <code>__setslice__()</code> и <code>__delslice__()</code>	234

8.4.5. Поддержка вызова	235
8.5. Операторы	235
8.5.1. Операторы обработки данных.....	236
Оператор-выражение.....	236
Оператор присваивания.....	237
Оператор <i>print</i>	238
Оператор <i>del</i>	240
8.5.2. Управляющие операторы.....	241
Оператор <i>if</i>	241
Оператор <i>while</i>	243
Оператор <i>for</i>	244
Оператор <i>break</i>	246
Оператор <i>continue</i>	246
Оператор <i>return</i>	247
Оператор <i>pass</i>	247
8.5.3. Операторы обработки исключений.....	247
Оператор <i>try</i>	248
Оператор <i>raise</i>	251
Утверждения.....	252
8.5.4. Декларативные операторы.....	253
Оператор <i>global</i>	253
8.5.5. Операторы определения объектов.....	253
Оператор <i>def</i>	253
Оператор <i>class</i>	256
8.5.6. Операторы исполнения.....	260
Оператор <i>import</i>	260
Оператор <i>exec</i>	263
8.6. Встроенные функции и исключения.....	264
8.6.1. Встроенные функции.....	264
Функции преобразования типов.....	265
Числовые и символьные функции.....	268
Функции обработки данных.....	272
Функции определения свойств.....	277
Функции для работы с системными объектами.....	280
Функции компиляции и исполнения.....	283
Функции ввода/вывода.....	288
Функции для работы с атрибутами.....	290
Другие функции.....	291
8.6.2. Встроенные исключения.....	292
Базовые классы <i>Exception</i> , <i>SystemExit</i> и <i>StandardError</i>	293
<i>ArithmeticError</i> (базовый класс).....	295
<i>AssertionError</i>	295
<i>AttributeError</i>	295
<i>EnvironmentError</i> (базовый класс).....	295
<i>FloatingPointError</i>	296
<i>ImportError</i>	296

<i>IndentationError</i>	296
<i>IndexError</i>	296
<i>IOError</i>	297
<i>KeyError</i>	297
<i>KeyboardInterrupt</i>	297
<i>LookupError</i> (базовый класс).....	298
<i>MemoryError</i>	298
<i>NameError</i>	298
<i>NotImplementedError</i>	298
<i>OSError</i>	299
<i>OverflowError</i>	299
<i>RuntimeError</i>	300
<i>SyntaxError</i>	300
<i>SystemError</i>	300
<i>TabError</i>	300
<i>TypeError</i>	301
<i>UnboundLocalError</i>	301
<i>UnicodeError</i>	301
<i>ValueError</i>	302
<i>WindowsError</i>	302
<i>ZeroDivisionError</i>	302
8.7. Пространства имен.....	302
8.8. Модули и пакеты.....	308
8.9. Оптимизация программ.....	312
8.10. О стиле программирования.....	314
8.11. Направления развития языка.....	318
Глава 9. Основные стандартные модули.....	322
9.1. Встроенная помощь: модуль <i>pydoc</i>	322
9.2. Сервисы периода исполнения.....	326
9.2.1. Модуль <i>sys</i>	326
9.2.2. Модуль <i>atexit</i>	329
9.2.3. Модуль <i>copy</i>	330
9.2.4. Модуль <i>gc</i>	333
9.2.5. Модуль <i>getpass</i>	335
9.2.6. Модуль <i>operator</i>	335
9.2.7. Модули <i>pickle</i> и <i>cPickle</i>	338
9.2.8. Модуль <i>pprint</i>	340
9.2.9. Модуль <i>repr</i>	341
9.2.10. Модуль <i>shelve</i>	343
9.2.11. Модуль <i>traceback</i>	344
9.2.12. Модуль <i>types</i>	346
9.2.13. Модуль <i>UserString</i>	348
9.2.14. Модули <i>UserList</i> и <i>UserDict</i>	349
9.2.15. Модули <code>__builtin__</code> и <code>__main__</code>	350

9.3. Обработка строк	350
9.3.1. Модуль <i>string</i>	350
9.3.2. Модуль <i>re</i>	355
9.3.3. Модули <i>StringIO</i> и <i>cStringIO</i>	364
9.3.4. Модуль <i>codecs</i> и работа с Unicode	366
9.3.5. Модуль <i>difflib</i>	371
9.3.6. Модуль <i>mmap</i>	374
9.4. Математические функции	376
9.4.1. Модуль <i>math</i>	376
9.4.2. Модуль <i>cmath</i>	378
9.4.3. Модули <i>random</i> и <i>whrandom</i>	379
9.5. Время и календарь	381
9.5.1. Модуль <i>time</i>	381
9.5.2. Модуль <i>calendar</i>	387
9.6. Массивы и структуры	388
9.6.1. Модуль <i>array</i>	389
9.6.2. Модуль <i>struct</i>	390
9.7. Взаимодействие с операционными системами (файлы, процессы)	394
9.7.1. Модуль <i>os</i>	394
Информация о системе.....	394
Параметры процесса	395
Создание и управление файловым объектом	397
Работа с файлами и каталогами.....	400
Создание и управление процессами.....	406
9.7.2. Модуль <i>os.path</i>	411
9.7.3. Модуль <i>dircache</i>	414
9.7.4. Модуль <i>errno</i>	414
9.7.5. Модуль <i>getopt</i>	416
9.7.6. Модуль <i>glob</i>	418
9.7.7. Модуль <i>popen2</i>	419
9.7.8. Модуль <i>shutil</i>	420
9.7.9. Модуль <i>select</i>	421
9.7.10. Модуль <i>signal</i>	423
9.7.11. Модуль <i>stat</i>	427
9.7.12. Модуль <i>tempfile</i>	430
9.7.13. Модуль <i>threading</i>	431
Класс <i>threading.Thread</i>	432
Класс <i>threading.Lock</i>	433
Класс <i>threading.RLock</i>	435
Семафоры: класс <i>threading.Semaphore</i>	435
Класс <i>threading.Event</i>	435
Класс <i>Condition</i>	437
9.7.14. Модуль <i>thread</i>	440
9.8. Простейшие базы данных. Архиваторы	443
9.8.1. Модуль <i>anydbm</i>	443
9.8.2. Модуль <i>whichdb</i>	444

9.8.3. Модуль <i>gzip</i>	444
9.8.4. Модуль <i>zlib</i>	446
9.8.5. Модуль <i>zipfile</i>	448
9.9. Модули для UNIX.....	451
9.9.1. Модули <i>pwd</i> и <i>grp</i>	451
9.9.2. Модуль <i>fentl</i>	452
9.9.3. Модуль <i>resource</i>	455
9.9.4. Модуль <i>termios</i>	457
9.10. Редактирование в командной строке.....	459
9.10.1. Модуль <i>readline</i>	460
9.10.2. Модуль <i>rlcompleter</i>	462
9.11. Отладчик и профайлер.....	463
9.11.1. Модуль <i>pdb</i>	463
9.11.2. Модуль <i>profile</i>	465
9.12. Поддержка Internet. Протоколы.....	468
9.12.1. Модуль <i>cgi</i>	470
9.12.2. Модуль <i>urllib</i>	478
9.12.3. Модуль <i>urlparse</i>	481
9.12.4. Модуль <i>httplib</i>	483
9.12.5. Модуль <i>smtplib</i>	487
9.12.6. Модуль <i>poplib</i>	489
9.12.7. Модуль <i>telnetlib</i>	493
9.12.8. Модуль <i>socket</i>	495
9.13. Поддержка Internet. Форматы данных.....	502
9.13.1. Модуль <i>quopri</i>	502
9.13.2. Модуль <i>uu</i>	502
9.13.3. Модуль <i>base64</i>	503
9.13.4. Модуль <i>binhex</i>	504
9.13.5. Модуль <i>binascii</i>	504
9.13.6. Модуль <i>rfc822</i>	505
9.13.7. Модуль <i>mimetools</i>	508
9.13.8. Модуль <i>MimeWriter</i>	511
9.13.9. Модуль <i>multifile</i>	513
9.13.10. Модуль <i>mailbox</i>	515
9.14. Python и языки разметки (SGML, XML, HTML).....	516
9.14.1. Модуль <i>sgmllib</i>	517
9.14.2. Модуль <i>htmllib</i>	521
9.14.3. Пакет <i>xml</i>	525
Пакет <i>xml.sax</i>	527
Модуль <i>xml.parsers.expat</i>	539
Пакет <i>xml.dom</i>	542
9.15. Элементы мультимедиа.....	550
9.15.1. Модуль <i>wave</i>	551
9.15.2. Модуль <i>colorsys</i>	553
9.15.3. Модули <i>sndhdr</i> и <i>imghdr</i>	554

9.16. Элементы криптографии	555
9.16.1. Модуль <i>md5</i>	555
9.16.2. Модуль <i>rotor</i>	557
9.16.3. Модуль <i>crypt</i>	558
9.17. Защищенная среда исполнения: модуль <i>rexec</i>	559
9.18. Python о себе	562
9.18.1. Модуль <i>keyword</i>	564
9.18.2. Модули <i>py_compile</i> и <i>compileall</i>	564
9.18.3. Модуль <i>dis</i>	565
9.19. Графический интерфейс. Python и Tk	566
9.19.1. Классы модуля <i>Tkinter</i>	567
9.19.2. Виджеты в общем	569
Создание виджетов	570
Базовые методы виджетов	570
Конфигурирование виджетов	576
Стилевые опции	577
Содержимое виджета	578
Задание шрифта. Модуль <i>tkFont</i>	579
Цвета	581
Описания событий	582
9.19.3. Различные виджеты	585
Окна верхнего уровня	585
Этикетка (виджет <i>Label</i>)	587
Кнопка (виджет <i>Button</i>)	588
Переключатель (<i>Radiobutton</i>) и флажок (<i>Checkbutton</i>)	590
Поле ввода (виджет <i>Entry</i>)	592
Меню	595
Текст (виджет <i>Text</i>)	596
Рисунок (виджет <i>Canvas</i>)	605
Менеджеры расположения	612
Переменные	616
9.19.4. Черепашня графика: модуль <i>turtle</i>	618
9.20. Модули для MS Windows	620
9.20.1. Модуль <i>msvcrt</i>	620
9.20.2. Модуль <i>_winreg</i>	621
9.20.3. Модуль <i>winsound</i>	625
9.21. Интернационализация и локализация	626
9.21.1. Модуль <i>locale</i>	626
9.21.2. Модуль <i>gettext</i>	631
9.22. Используем Distutils	636
9.22.1. Установка модулей	637
В среде Windows	637
В среде UNIX	638
9.22.2. Распространение модулей	640
Сценарий настройки	640
Файл конфигурации	640
Создание дистрибутивов	642
9.22.3. Список команд и опций	643

Часть III. РАСШИРЕНИЯ PYTHON	649
Глава 10. Python и C	651
Глава 11. Numeric Python	658
11.1. Модуль <i>Numeric</i>	658
11.2. Модуль <i>LinearAlgebra</i>	672
Глава 12. Обработка изображений: Python Image Library	675
12.1. Модуль <i>Image</i>	675
12.1.1. Примеры	676
12.1.2. Функции	677
12.1.3. Методы объектов класса <i>Image</i>	678
12.2. О других модулях	682
Глава 13. Связь с базами данных	686
13.1. Описание DB API 2.0	686
13.2. Модуль <i>MySQLdb</i>	691
13.3. Модуль для работы с PostgreSQL	695
Глава 14. Среда Zope	700
14.1. С точки зрения администратора	700
14.1.1. Установка Zope	701
14.1.2. Политика безопасности	702
14.1.3. Добавляем ресурс и пользователей	703
14.2. С точки зрения программиста	704
14.2.1. Zope-объекты	704
14.2.2. Знакомимся с языком DTML	705
14.2.3. Пишем сценарий	709
14.3. С точки зрения менеджера по содержанию	712
Глава 15. Jython — Python на Java	715
Часть IV. ПРИЛОЖЕНИЯ	723
Приложение 1. Переводы сообщений об ошибках	725
Приложение 2. Оболочка IDLE	728
Список литературы	735
Предметный указатель	737

Глава 1

Интерактивный Python



В этой главе мы познакомимся с Python — новым интересным языком программирования. Он задумывался для тех, кто хочет с легкостью и изяществом решать даже трудные задачи, и подходит для тех, кто хочет научиться программировать.

1.1. Запускаем Python

Для диалога с Python нам необходимо прежде всего запустить интерпретатор Python. Упрощенно, *интерпретатор* — это такая программа, которая последовательно, одну за другой, читает команды языка программирования (они называются операторами), и переводит их в машинные команды, которые исполняет процессор компьютера.

A screenshot of the Python Shell window. The title bar reads '*Python Shell*'. The menu bar contains 'File', 'Edit', 'Debug', 'Windows', and 'Help'. The main text area displays the following text:

```
Python 2.1 (#15, Apr 16 2001, 18:25:49) [MSC 32 bit (Intel)] on win32
Type "copyright", "credits" or "license" for more information.
IDLE 0.8 -- press F1 for help
>>>
```

The status bar at the bottom right shows 'Ln: 4 | Col: 4'.

Рис. 1.1. Первый запуск IDLE

В системе Windows интерпретатор обычно запускается вместе с графической оболочкой IDLE (рис. 1.1) из **Программы | Python | IDLE (Python GUI)**, но можно запустить только интерпретатор, выбрав команду **Программы | Python | Python (Command line)**. И то, и другое даст нам возможность вводить команды, которые Python будет исполнять.

В системе UNIX и Linux также можно запустить Python просто командой `python`, а можно запустить IDLE (если вы любите работать в X Window) — командой `idle`. (В UNIX режим командной строки гораздо богаче возможностями редактирования, чем в Windows).

Сразу после запуска Python нас приветствует, например, так:

```
Python 2.1 (#1, Apr 21 2001, 13:46:58)
[GCC egcs-2.91.66 19990314/Linux (egcs-1.1.2 release)] on linux2
Type "copyright", "credits" or "license" for more information.
>>>
```

Три знака ">" — *приглашение ко вводу*. Мы можем вводить наши команды.

1.2. Диалог с Python

Наш Python готов к диалогу, о чем говорит >>>. Для начала спросим его о погоде на сегодня:

```
>>> Какая сегодня погода?
File "<stdin>", line 1
    Какая сегодня погода?
        ^
SyntaxError: invalid syntax
>>>
```

К сожалению, уважаемый Python не понимает русского языка, и нам придется учить его язык. Но что же он нам сказал, указав на нашу ошибку? `SyntaxError` — синтаксическая ошибка, а более конкретно `invalid syntax` — неверный синтаксис. Высказавшись, Python снова предложил продолжить общение.

```
>>> 2+2
4
```

Вот это ему понятно. Пробуем дальше (`print` означает "печатай"):

```
>>> print "Сегодня хороший день!"
Сегодня хороший день!
>>> print "Сегодня хороший день!", 1, "мая!"
Сегодня хороший день! 1 мая!
```

```
>>> print "Завтра будет", 1+1, "мая!"
Завтра будет 2 мая!
>>> print "Послезавтра будет", 1+ "мая!"
Послезавтра будет
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsupported operand types for +
>>>
```

Кажется, мы увлеклись и забыли, что компьютер — вещь точная, и требует четкости при работе с ним. На этот раз произошла ошибка типа `TypeError`, потому что нельзя складывать числа и строки, как это случайно произошло в диалоге выше. Ничего страшного, мы исправим ошибку и продолжим общение:

```
>>> print "Послезавтра будет", 1+2, "мая!"
Послезавтра будет 3 мая!
>>>
```

Кстати, строки с текстом тоже можно "складывать":

```
>>> print "Послезавтра" + "будет" + "май!"
Послезавтрабудетмай!
>>> print "Послезавтра " + "будет " + "май!"
Послезавтра будет май!
```

А вот арифметические операции в кавычках исполняться, естественно, не будут:

```
>>> print "2+2"
2+2
```

Python все равно, использованы ли кавычки или апострофы. Кроме того, для записи строк текста можно применять утроенные кавычки или апострофы:

```
>>> print """"Текст можно писать и в таких кавычках, """" 'и в таких.'
Текст можно писать и в таких кавычках, и в таких.
>>> print "Но не в таких:", `2+2`
Но не в таких: 4
```

Надеемся, что читатель неплохо провел время в легкой беседе с интерпретатором Python и готов к более серьезному общению в следующем разделе.

Упражнение

Сложите яблоки с апельсинами.

1.3. Python как калькулятор

В этом разделе мы научимся использовать Python в качестве калькулятора. Запустите Python (см. *разд. 1.1*, если вы забыли, как это делается) и за дело.

Python умеет производить простейшие арифметические операции: сложение, вычитание, умножение, деление и даже получение остатка от деления. Причем делать он это может как с целыми числами, так и с десятичными дробями, а также с комплексными числами. В записываемых числовых примерах (их грамотно называть *выражениями*) можно расставлять скобки так, как нужно для правильных вычислений. В отличие от многих калькуляторов, Python знает, что умножение, деление и получение остатка от деления имеет бóльший приоритет, чем сложение или вычитание. Давайте посмотрим, как Python справится со следующими выражениями:

```
>>> 2+3*4
14
>>> 2*3+4
10
>>> (2+3)*4
20
>>> 10/2
5
>>> 10/3
3
>>> 12.48/2
6.24
>>> 13%5
3
```

В первых примерах все ясно. Операция умножения обычно обозначается в языках программирования как "*", деление — как "/", а операция получения остатка от деления — как "%". Вы, наверное, заметили, что десятичные дроби пишутся через точку (12.48), несмотря на то, что в России принято писать через запятую. Запись дробей с точкой традиционна для многих языков и систем программирования. Кроме того, результат 10/3 равен 3. Дело в том, что и 10, и 3 — целые числа, и потому Python осуществляет деление нацело. Для того чтобы разделить числа "обычным" образом, нужно записать хотя бы одно из них как десятичную дробь — с точкой:

```
>>> 10.0/3
3.333333333333333
>>> 11./3
```



```
3.666666666667
>>> 10/3.
3.333333333333
```

Теперь можно решать примеры любой сложности:

```
>>> (12.3 + 12)/(12 - (78.3+5*6))
-0.2523336448598
>>> _ + 5
4.7476635514
```

Одинокое подчеркивание "_" является специальным именем для обозначения результата предыдущего вычисления.

Упражнение

Составьте выражение на Python, которое вычисляет стоимость покупки яблок, мандаринов и бананов, если известны их вес и цена за килограмм.

Следующий пример показывает, что некоторые операции над большими числами приводят к переполнениям, о которых Python тут же заявляет:

```
>>> 1000000000*1000000000
Traceback (innermost last):
  File "<stdin>", line 1, in ?
OverflowError: integer multiplication
```

Это было сказано для того, чтобы, пользуясь имеющейся в Python операцией возведения в степень (обозначается **), вы не удивлялись возникающим переполнениям:

```
print "1 килобайт =", 2**10, "байт"
1 килобайт = 1024 байт
print "1 мегабайт =", 1024 ** 2, "байт"
1 мегабайт = 1048576 байт
print "1 гигабайт =", 1024 ** 3, "байт"
1 гигабайт = 1073741824 байт
>>> 2**100
Traceback (innermost last):
  File "<stdin>", line 1, in ?
OverflowError: integer pow()
```

Упражнение

В какую наибольшую степень можно возвести число 2 без переполнения?

Python может стать "инженерным" калькулятором, если перед вычислениями подать специальную команду, загружающую в память модуль с различными математическими функциями (см. разд. 9.4.1). Ниже мы будем использовать `sqrt` (квадратный корень), `sin` (синус), `cos` (косинус), `atan` (арктангенс).

```
>>> from math import *
>>> sqrt(16)
4.0
>>> sqrt(18.49)
4.3
>>> sin(3)
0.14112000806
>>> sin(1.22)**2 + cos(1.22)**2
1.0
```

Упражнение

Поработав с функцией `fabs()`, попытайтесь установить, что она делает.

Работая с обычными калькуляторами, бывает очень обидно, что с таким трудом вычисленные значения нужно стирать перед выполнением других вычислений. В Python результатам вычислений можно давать *имена*:

```
>>> from math import *
>>> x = sin(0.5) + atan(2.3)
>>> y = 7
>>> x*y
11.480661674
```

В этом примере Python, вычислив `sin(0.5) + atan(2.3)`, дает полученному результату имя `x`. Аналогично происходит и с `y`, хотя там уже никаких сложных вычислений нет. Говорят, что (переменной) `y` присваивается значение 7. А дальше мы просто используем имена переменных в вычисляемых выражениях наравне с числами.

Кстати, все тот же модуль `math` содержит имена, соответствующие приближенным значениям известных констант π и e :

```
>>> from math import *
>>> print "pi =", pi
pi = 3.14159265359
>>> print "e =", e
e = 2.71828182846
```

Так что площадь круга радиуса r можно вычислить так:

```
>>> from math import *
>>> r = 3
>>> print "S =", pi * r**2
S = 28.2743338823
```

Упражнение

Воспользуйтесь интерпретатором Python для вычисления по какой-либо известной вам формуле.

За исключением примера с переполнением, в наших вычислениях участвовали сравнительно небольшие числа. Оказывается, для представления очень больших или очень близких нулю чисел существует свой вид записи — *запись в экспоненциальном формате*. В такой записи записываются лишь действительные числа (числа "с точкой"):

```
>>> 0.0001
0.0001
>>> 0.00001
1e-05
>>> 10.0**-6
1e-06
>>> 100000000000.
100000000000.
>>> 1000000000000.
1e+12
>>> 10.0**13
1e+13
>>> 3.14e5
314000.0
>>> 3.14e-5
3.14e-05
>>> 3.14e-3
0.00314
```

Секрет записи раскрывается просто: число получается в результате умножения *мантиссы* (стоит перед "e") на 10 в степени, указанной после "e". Указанное после "e" число называется *порядком*.

Упражнение

Как в Python запишется миллион миллиардов?

Если же нас интересуют большие целые числа, мы можем использовать целые числа с произвольной точностью. В Python они (скромно) названы *длинными целыми*. Запись длинного целого отличается от записи обычного целого только добавлением "L" на конце:

```
>>> print "1 терабайт =", 1024L ** 4, "байт"  
1 терабайт = 1099511627776L байт  
>>> print "1 петабайт =", 1024L ** 5, "байт"  
1 петабайт = 1125899906842624L байт
```

Числа с произвольной точностью ограничены лишь размером памяти, доступной интерпретатору Python, и степенью вашего терпения при ожидании результатов вычислений.

Упражнение

Вычислите с помощью целых чисел произвольной точности число 2 в десяти-тысячной степени. А в стотысячной?

Кстати, выполняющиеся вычисления можно прервать с помощью комбинации клавиш <Ctrl>+<C> (Linux) или <Ctrl>+<Break> (Windows). Прервемся и мы, чтобы в следующей главе заняться настоящими программами.