

Мэтт
Харрисон



КАК УСТРОЕНЫ PYTHON

ГИД ДЛЯ РАЗРАБОТЧИКОВ, ПРОГРАММИСТОВ И ИНТЕРЕСУЮЩИХСЯ



ББК 32.973.2-018.1
УДК 004.43
Х21

Харрисон Мэтт

Х21 Как устроен Python. Гид для разработчиков, программистов и интересующихся. — СПб.: Питер, 2019. — 272 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-0906-7

Python в моде! Это самый популярный язык программирования. Вакансии для Python-разработчиков входят в список самых высокооплачиваемых, а благодаря бурному развитию обработки данных, знание Python становится одним из самых востребованных навыков в среде аналитиков.

Python — невероятный язык, популярный во многих областях. Он используется для автоматизации простых и сложных задач, цифровой обработки, веб-разработки, игр... Независимо от того, перешли ли вы на Python с другого языка, руководите группой программистов, работающих на Python, или хотите расширить свое понимание, имеет смысл подойти к изучению Python со всей серьезностью.

Готовы начать карьеру питониста? Не теряйте времени на поиск информации, перелопачивая блоги и сайты, списки рассылок и группы. Мэтт Харрисон использует Python с 2000 года. Он занимался научными исследованиями, сборкой и тестированием, бизнес-аналитикой, хранением данных, а теперь делится своими знаниями как с простыми пользователями, так и с крупными корпорациями. Приобщитесь к передовому опыту и узнайте секреты внутренней кухни Python, доступные только профи, работающим с этим языком на протяжении многих лет.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018.1
УДК 004.43

Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-1977921758 англ.
ISBN 978-5-4461-0906-7

© 2017 Matt Harrison
© Перевод на русский язык ООО Издательство «Питер», 2019
© Издание на русском языке, оформление ООО Издательство «Питер», 2019
© Серия «Библиотека программиста», 2019

Оглавление

Предисловие	13
От издательства	14
Глава 1. Почему Python?.....	15
Глава 2. Какая версия Python?.....	17
2.1. Установка Python	17
2.2. Какой редактор?	19
2.3. Итоги	19
2.4. Упражнения	20
Глава 3. Интерпретатор	21
3.1. REPL	22
3.2. Пример использования REPL.....	24
3.3. Итоги	27
3.4. Упражнения	27
Глава 4. Запуск программ.....	28
4.1. Запуск программ из IDLE	29
4.2. Усовершенствования для UNIX.....	31
4.3. Итоги	33
4.4. Упражнения	34
Глава 5. Запись и чтение данных	35
5.1. Простой вывод.....	35
5.2. Получение данных от пользователя.....	36
5.3. Итоги	37
5.4. Упражнения	37
Глава 6. Переменные	38
6.1. Изменение и состояние.....	38
6.2. Переменные Python как метки.....	39
6.3. Бирки.....	41
6.4. Повторное связывание переменных	43
6.5. Имена переменных	45
6.6. Дополнительные рекомендации по назначению имен	46
6.7. Итоги	49
6.8. Упражнения	50

Глава 7. Подробнее об объектах	51
7.1. Идентификатор	51
7.2. Тип	53
7.3. Изменяемость	55
7.4. Использование IDLE	57
7.5. Итоги	60
7.6. Упражнения	61
Глава 8. Числа	62
8.1. Сложение	63
8.2. Вычитание	65
8.3. Умножение	65
8.4. Деление	66
8.5. Остаток	66
8.6. Возведение в степень	68
8.7. Порядок операций	69
8.8. Другие операции	69
8.9. Итоги	69
8.10. Упражнения	70
Глава 9. Строки	71
9.1. Форматирование строк	74
9.2. Синтаксис форматных строк	74
9.3. Примеры форматирования	77
9.4. F-строки	78
9.5. Итоги	79
9.6. Упражнения	79
Глава 10. dir, help и pdb	80
10.1. Специальные методы	81
10.2. help	82
10.3. pdb	82
10.4. Итоги	84
10.5. Упражнения	84
Глава 11. Строки и методы	85
11.1. Основные строковые методы	89
11.2. endswith	89
11.3. find	91
11.4. format	91
11.5. join	92
11.6. lower	93
11.7. startswith	93

11.8. strip.....	93
11.9. upper	94
11.10. Другие методы	94
11.11. Итоги	94
11.12. Упражнения	94
Глава 12. Комментарии, логические значения и None	96
12.1. Комментарии	96
12.2. Логические значения	97
12.3. None	101
12.4. Итоги	103
12.5. Упражнения	103
Глава 13. Условия и отступы	105
13.1. Объединение условных выражений.....	107
13.2. Команды if	109
13.3. Команды else.....	109
13.4. Множественный выбор.....	110
13.5. Пробелы	110
13.6. Итоги	112
13.7. Упражнения	112
Глава 14. Контейнеры: списки, кортежи и множества	113
14.1. Списки	113
14.2. Индексы.....	114
14.3. Вставка в список	115
14.4. Удаление из списка	116
14.5. Сортировка списков	116
14.6. Полезные советы по работе со списками.....	117
14.7. Кортежи	122
14.8. Множества	125
14.9. Итоги	128
14.10. Упражнения	128
Глава 15. Итерации	130
15.1. Перебор с индексом	131
15.2. Выход из цикла	134
15.3. Пропуск элементов в цикле.....	134
15.4. Оператор in может использоваться для проверки принадлежности.....	135
15.5. Удаление элементов из списков при переборе	135
15.6. Блок else.....	137
15.7. Циклы while.....	137
15.8. Итоги	139
15.9. Упражнения	139

Глава 16. Словари	140
16.1. Присваивание в словарях.....	140
16.2. Выборка значений из словаря.....	142
16.3. Оператор in.....	143
16.4. Сокращенный синтаксис словарей.....	144
16.5..setdefault.....	144
16.6. Удаление ключей.....	147
16.7. Перебор словаря.....	147
16.8. Итоги.....	150
16.9. Упражнения.....	150
Глава 17. Функции	152
17.1. Вызов функций.....	155
17.2. Область видимости.....	156
17.3. Множественные параметры.....	158
17.4. Параметры по умолчанию.....	159
17.5. Правила выбора имен для функций.....	161
17.6. Итоги.....	162
17.7. Упражнения.....	162
Глава 18. Индексирование и срезы	163
18.1. Индексирование.....	163
18.2. Срезы.....	164
18.3. Приращения в срезах.....	166
18.4. Итоги.....	168
18.5. Упражнения.....	168
Глава 19. Операции ввода/вывода с файлами	169
19.1. Открытие файлов.....	169
19.2. Чтение текстовых файлов.....	171
19.3. Чтение двоичных файлов.....	172
19.4. Перебор при работе с файлами.....	173
19.5. Запись файлов.....	174
19.6. Закрытие файлов.....	175
19.7. Проектирование на основе файлов.....	177
19.8. Итоги.....	178
19.9. Упражнения.....	178
Глава 20. Юникод	180
20.1. Историческая справка.....	180
20.2. Основные этапы в Python.....	183
20.3. Кодирование.....	185

20.4. Декодирование	187
20.5. Юникод и файлы.....	188
20.6. Итоги	190
20.7. Упражнения	190
Глава 21. Классы	192
21.1. Планирование класса.....	195
21.2. Определение класса	196
21.3. Создание экземпляра класса	201
21.4. Вызов метода	204
21.5. Анализ экземпляра.....	205
21.6. Приватный и защищенный доступ.....	206
21.7. Простая программа, моделирующая поток посетителей	207
21.8. Итоги	209
21.9. Упражнения	209
Глава 22. Создание подклассов.....	211
22.1. Подсчет остановок	214
22.2. super	215
22.3. Итоги	217
22.4. Упражнения	218
Глава 23. Исключения.....	219
23.1. «Посмотри, прежде чем прыгнуть»	220
23.2. «Проще просить прощения, чем разрешения»	221
23.3. Несколько возможных исключений	223
23.4. finally	225
23.5. Секция else	227
23.6. Выдача исключений	228
23.7. Упаковка исключений.....	229
23.8. Определение собственных исключений.....	232
23.9. Итоги	233
23.10. Упражнения	234
Глава 24. Импортирование библиотек.....	235
24.1. Способы импортирования	236
24.2. Конфликты имен при импортировании.....	239
24.3. Массовое импортирование	240
24.4. Вложенные библиотеки.....	241
24.5. Организация импортирования.....	241
24.6. Итоги	243
24.7. Упражнения	243

Глава 25. Библиотеки: пакеты и модули	244
25.1. Модули	244
25.2. Пакеты.....	244
25.3. Импортирование пакетов	245
25.4. PYTHONPATH.....	246
25.5. sys.path.....	247
25.6. Итоги	248
25.7. Упражнения	249
Глава 26. Полноценный пример	250
26.1. cat.py	250
26.2. Что делает этот код?.....	253
26.3. Типичная структура	254
26.4. #!	255
26.5. Строка документации.....	256
26.6. Импортирование	257
26.7. Метаданные и глобальные переменные.....	257
26.8. Операции с журналом	259
26.9. Другие глобальные переменные	259
26.10. Реализация	259
26.11. Тестирование	259
26.12. if __name__ == '__main__':	260
26.13. __name__	261
26.14. Итоги	262
26.15. Упражнения	263
Глава 27. В начале пути.....	264
Приложение А. Перемещение по файлам.....	265
А.1. Mac и UNIX	265
А.2. Windows	266
Приложение Б. Полезные ссылки	267
Об авторе.....	268
Научные редакторы.....	269

1

Почему Python?

Python в моде! Это самый популярный язык, которому учат в университетах. Вакансии для разработчиков Python входят в число самых высокооплачиваемых. Из-за бурного развития теории обработки данных знание Python быстро становится одним из самых желанных навыков для аналитиков. Операционные отделы также осваивают Python для управления подсистемами баз данных. Они осознают то, что давно известно веб-разработчикам, уже использующим Python, а именно то, что Python делает их работу более продуктивной.

В жизни Python наступил переломный момент. Его область применения уже не ограничивается небольшими, динамичными стартапами. Стремясь извлечь пользу из его мощи и эффективности, крупные предприятия также переходят на Python. За последний год я преподавал Python сотням матерых разработчиков с многолетним опытом работы в крупных компаниях, переходивших на Python.

Python повышает производительность программирования. Я перешел на Python из мира Perl. На работе меня включили в команду с коллегой, хорошо владевшим Tcl. Ни один из нас не хотел переходить в другой лагерь, хотя мы оба были заинтересованы в изучении Python. За три дня наш прототип был готов — намного быстрее, чем ожидалось, и мы оба моментально забыли свои предыдущие «goto-языки». Меня привлекло в Python прежде всего то, что этот язык казался мне абсолютно логичным. Я твердо уверен, что каждый, у кого есть хоть какой-то опыт программирования, сможет изучить основы Python всего за несколько дней.

Python легок в освоении. Для начинающих программистов Python станет отличным трамплином. Научиться писать простые программы очень легко, однако Python также хорошо масштабируется для сложных «корпоративных» систем. Наконец, Python подойдет для любого возраста — я сам видел, как люди в возрасте от 7 до 80+ лет изучали основы программирования на примере Python.

2

Какая версия Python?

Эта книга написана на основе Python 3. Версия Python 2 верно служила нам много лет. Фонд Python Software Foundation, управляющий выпуском новых версий, заявил, что эпоха Python 2 подошла к концу. Соответственно, после 2020 года язык поддерживаться не будет.

Версия Python 3 существует уже в течение некоторого времени, и, как выяснилось, она не обладает полной обратной совместимостью с линейкой 2. Если разработка начинается с нуля, беритесь за Python 3. Если вам приходится иметь дело с унаследованными системами, написанными на Python 2, не огорчайтесь. Большая часть материала книги идеально подходит для Python 2. Если же вы захотите сосредоточиться на Python 2, найдите предыдущее издание этой книги.

2.1. Установка Python

Python 3 не устанавливается по умолчанию на большинстве платформ. Некоторые дистрибутивы Linux включают Python 3, но пользователям Windows и Mac придется установить его отдельно.

Если вы используете Windows, откройте раздел загрузок на сайте Python¹ и найдите ссылку `Python 3.6 Windows Installer`. По ссылке загружается файл `.msi`, который устанавливает Python на машину с системой Windows. За-

¹ <https://www.python.org/download>

грузите файл, откройте его двойным щелчком и выполните инструкции, чтобы завершить установку.

ПРИМЕЧАНИЕ

В установочной программе для Windows имеется флажок «Add Python to PATH» (Добавить Python в переменную PATH). Проследите за тем, чтобы этот флажок был установлен. В этом случае при запуске из режима командной строки система будет знать, где найти исполняемый файл Python. Если флажок все же не будет установлен, откройте свойства системы (нажмите клавиши **WIN+Pause** или выполните команду **environ** из меню Пуск), откройте вкладку **Дополнительно** и щелкните на кнопке **Переменные среды**. Обновите переменную PATH и добавьте следующие пути:

`C:\Program Files\Python 3.6;C:\Program Files\Python 3.6\Scripts`

Если в вашей системе Windows включен механизм UAC (User Account Control), то путь будет выглядеть так:

`C:\Users\<имя_пользователя>\AppData\Local\Programs\Python\Python36`

Пользователи Mac загружают с сайта Python установочную программу для Mac.

ПРИМЕЧАНИЕ

Другой способ установки Python основан на использовании дистрибутива Anaconda¹. Он работает в Windows, Mac и Linux, а также предоставляет много заранее построенных двоичных файлов для выполнения научных вычислений. Традиционно устанавливать эти библиотеки было утомительно, потому что в них были упакованы библиотеки, написанные на C и Fortran, и это требовало дополнительной настройки для компиляции.

Пользователи Mac также могут присмотреться к Homebrew-версии². Если вы уже знакомы с Homebrew, проблема решается простой командой **brew install python3**.

¹ <https://www.anaconda.com/download/>

² <https://brew.sh>

2.2. Какой редактор?

Кроме установки Python вам понадобится текстовый редактор. В нем вы будете писать код. Настоящий мастер не жалеет времени на то, чтобы как следует изучить свои инструменты, и это время не пропадет даром. Умение пользоваться всеми возможностями текстового редактора упростит вашу работу. Во многих современных редакторах предусмотрена некоторая степень поддержки Python.

Если же вы только делаете первые шаги в изучении Python и у вас еще нет особого опыта в работе с текстовыми редакторами, в большинство установок Python включается среда разработки IDLE, которая работает в Windows, Mac и Linux.

При выборе редактора следует обратить внимание на интеграцию со средой Python REPL¹. Вскоре мы рассмотрим пример для IDLE. Желательно, чтобы выбранный вами редактор обладал сходной функциональностью.

Среди популярных редакторов с достойной поддержкой Python можно выделить Emacs, Vim, Atom, Visual Studio Code и Sublime Text. Если вас интересуют более мощные редакторы со встроенной поддержкой рефакторинга и автозавершения, обратите внимание на популярные PyCharm и Wing IDE.

2.3. Итоги

Python 3 — актуальная версия языка Python. Если только вы не работаете над унаследованным кодом, вам стоит отдать предпочтение именно этой версии. Новейшую версию можно загрузить на сайте Python.

Во многих современных редакторах реализована некоторая степень поддержки Python. Разные редакторы и среды разработки предоставляют разную функциональность. Если вы только начинаете осваивать программирование, опробуйте редактор IDLE. На первых порах это именно то, что нужно.

¹ REPL — сокращение от Read, Evaluate, Print Loop (цикл «чтение-вычисление-вывод»). Вскоре мы рассмотрим пример использования REPL.

2.4. Упражнения

1. Установите Python 3 на своем компьютере. Убедитесь в том, что Python успешно запускается.
2. Если вы привыкли работать в конкретном редакторе, узнайте, реализована ли в нем поддержка Python. В частности, умеет ли он:
 - автоматически выделять элементы синтаксиса Python;
 - выполнять код Python в REPL;
 - осуществлять пошаговое выполнение кода Python в отладчике.

3

Интерпретатор

Python традиционно относится к семейству *интерпретируемых* языков (другой термин для описания интерпретируемого языка — *язык сценариев*). Чтобы программа могла выполняться на компьютерном процессоре, она должна существовать в формате, понятном для этого процессора — а именно в *машинном коде*. Интерпретируемые языки не *компилируются* в машинный код напрямую; вместо этого в системе существует промежуточная прослойка — *интерпретатор*, — которая выполняет эту функцию.

У такого подхода есть как достоинства, так и недостатки. Как нетрудно понять, трансляция «на ходу» может занимать много времени. Интерпретируемый код — такой, как программы Python, — может работать в 10–100 раз медленнее программ на языке C. С другой стороны, написание кода на Python оптимизирует время разработки. Программы на языке Python нередко получаются в 2–10 раз короче своих аналогов на языке C. Кроме того, этап компиляции может занимать довольно много времени и отвлекать программиста между разработкой и отладкой.

Многие разработчики и компании охотно идут на этот компромисс. Небольшие программы (то есть содержащие меньше строк кода) быстрее пишутся и создают меньше проблем с отладкой. Труд программистов обходится дорого — если удастся переложить часть работы на оборудование, это может обойтись дешевле, чем привлечение дополнительных специалистов. Отладить 10 строк кода проще, чем отладить 100 строк кода. Исследования показали, что количество ошибок в коде пропорционально количеству строк. Следовательно, если язык позволяет написать меньше строк кода для решения некоторой задачи, то, скорее всего, программа

будет содержать меньше ошибок. Иногда скорость выполнения программы не столь важна, и во многих практических случаях Python работает достаточно быстро. Кроме того, были предприняты проекты, направленные на ускорение работы интерпретатора Python, например PyPy¹.

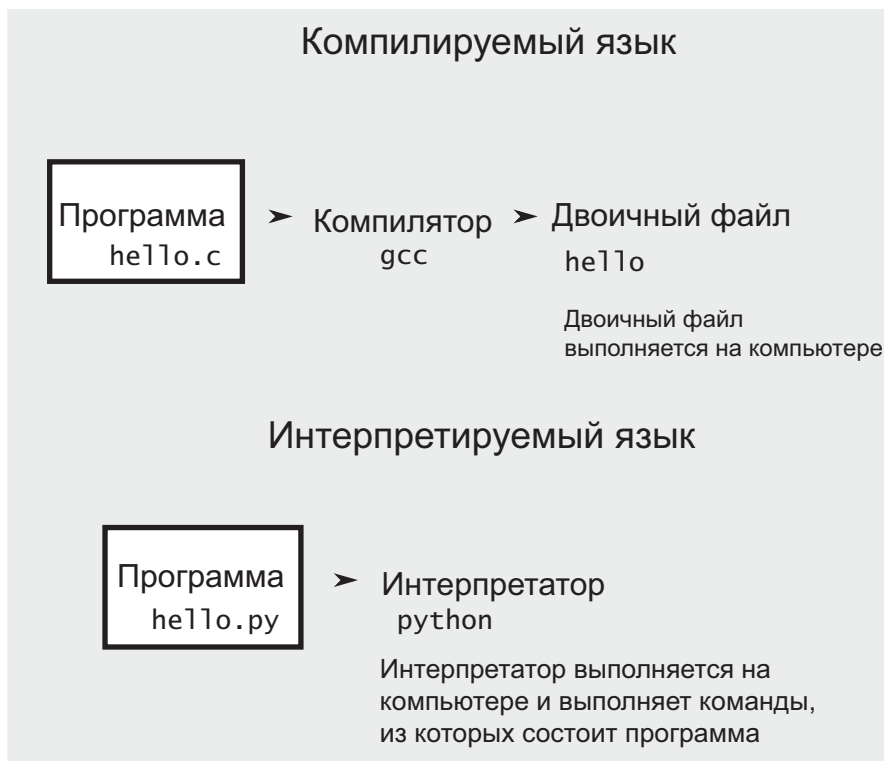


Рис. 3.1. Различия между компилируемым и интерпретируемым языком. Компилятор обрабатывает программный код и создает исполняемый файл. Интерпретатор создает исполняемый файл, который загружает программный код и управляет его выполнением

3.1. REPL

Для Python также существует *интерактивный интерпретатор*, который называется *REPL* (Read Evaluate Print Loop — цикл «чтение-вычисление-вывод»). REPL в цикле ожидает, пока появятся входные данные, читает

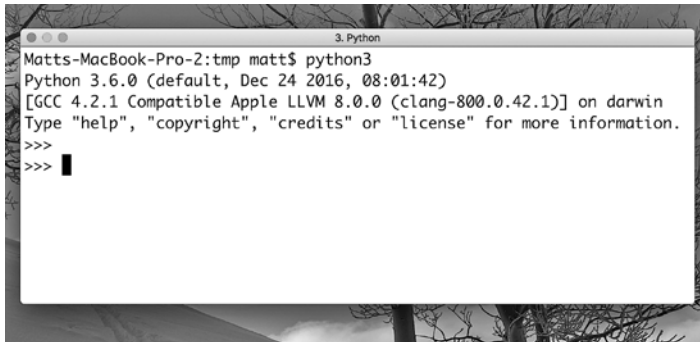
¹ <https://www.pypy.org>

и обрабатывает (интерпретирует) их, после чего выводит результат. Запуская исполняемый файл `python3`, вы запускаете интерактивный интерпретатор Python. Другие среды, например IDLE, также содержат встроенный интерактивный интерпретатор.

ПРИМЕЧАНИЕ

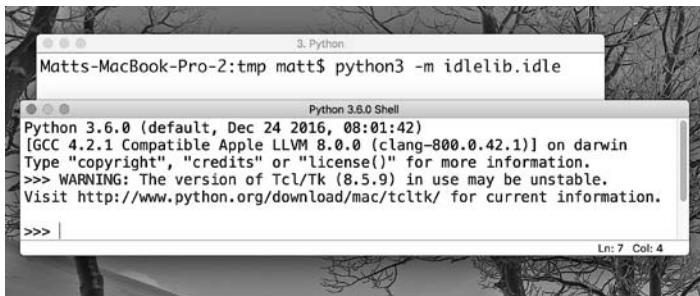
В этой книге Python 3 обычно запускается командой `python3`. В системе Windows исполняемому файлу присвоено имя `python`. Если вы работаете в Windows, замените имя `python3` именем `python`. В системе UNIX менять ничего не нужно.

При запуске интерпретатор выводит версию Python, информацию о сборке и несколько подсказок по использованию. Наконец, интерпретатор выдает приглашение `>>>`.



```
3. Python
Matts-MacBook-Pro-2:tmp matt$ python3
Python 3.6.0 (default, Dec 24 2016, 08:01:42)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> █
```

Рис. 3.2. Чтобы запустить REPL, введите в приглашении командной строки команду `python3`. Команда открывает сеанс Python



```
3. Python
Matts-MacBook-Pro-2:tmp matt$ python3 -m idlelib.idle

Python 3.6.0 Shell
Python 3.6.0 (default, Dec 24 2016, 08:01:42)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
>>> █
```

Рис. 3.3. Чтобы запустить REPL из IDE, щелкните на значке IDLE или введите команду `python3 -m idlelib.idle`

IDLE (редактор, включенный в поставку Python) также можно запустить командой `python3 -m idlelib.idle`.

ПРИМЕЧАНИЕ

Некоторые дистрибутивы Linux включают не все библиотеки из стандартной библиотеки Python. Это неприятно, но на то есть своя причина: на сервере не нужны библиотеки для создания клиентских приложений. По этой причине Ubuntu и Arch (среди прочих) в установке по умолчанию не включают библиотеки графического интерфейса, необходимые для IDLE.

Если вы увидите ошибку, которая выглядит примерно так:

```
$ python3 -m idlelib.idle
** IDLE can't import Tkinter.
Your Python may not be configured for Tk. **
```

это означает, что в системе отсутствует библиотека `tkinter`.

В Ubuntu следует выполнить команду:

```
$ sudo apt-get install tk-dev
```

В Arch эта команда выглядит так:

```
$ sudo pacman -S tk
```

3.2. Пример использования REPL

Следующий пример показывает, почему интерактивный интерпретатор REPL получил свое название. Введите команду `python3` в командной строке¹ или запустите IDLE; вы увидите приглашение `>>>`.

Введите `2 + 2`, как показано ниже, и нажмите клавишу `Enter`:

```
$ python3
>>> 2 + 2
4
>>>
```

¹ Чтобы вызвать приглашение командной строки, в меню Пуск системы Windows введите `cmd` (или нажмите клавиши `Win+R`). Чтобы быстро вызвать окно терминала на компьютере Mac, нажмите `Command+Space`, введите `Terminal` и нажмите `Return`. Если вы установили Python 3, то теперь сможете запустить его на любой платформе командой `python3`.

В этом примере мы ввели команду `python3`, которая запустила интерпретатор. Первое приглашение `>>>` можно рассматривать как первую часть названия (R — чтение): Python ожидает входных данных. Мы ввели `2 + 2`, интерпретатор прочитал и *обработал* (E — обработка) их. Далее *выводится* (P — вывод) результат этого выражения — 4. Второе приглашение `>>>` относится к циклу (L — цикл): интерпретатор ожидает новых входных данных.

REPL по умолчанию направляет результат выражения в стандартный вывод (если результат отличен от `None`, но об этом позднее). Такое поведение отличается от обычных программ Python, в которых для вывода данных необходимо вызвать функцию `print`. В REPL это экономит несколько нажатий клавиш.

ПРИМЕЧАНИЕ

Приглашение `>>>` используется только в первой строке входных данных. Если команда, введенная в REPL, занимает более одной строки, следует приглашение `...`:

```
>>> sum([1, 2, 3, 4, 5,
... 6, 7])
```

Эти приглашения определяются в модуле `sys`:

```
>>> import sys
>>> sys.ps1
'>>> '
>>> sys.ps2
'... '
```

О том, что такое модули, будет рассказано в одной из следующих глав. А пока знайте, что внешний вид приглашений определяется специальными переменными.

REPL — весьма полезный инструмент. При помощи интерактивного интерпретатора можно писать небольшие функции, тестировать фрагменты кода и даже выполнять вычисления, как на калькуляторе. А еще интереснее пойти в другом направлении: запустите свой код Python в REPL. Код будет выполнен, а вы сможете проверить его состояние в REPL (скоро мы покажем, как сделать это в IDLE).

Символы `>>>` образуют *приглашение*. Здесь вы вводите свою программу. Введите после `>>>` команду `print("hello world")` и нажмите клавишу Enter. Проследите за тем, чтобы перед словом `print` не было ни пробелов, ни табуляций. Результат должен выглядеть так:

```
>>> print("hello world")
hello world
```

Если все получилось именно так — поздравляем, вы написали свою первую программу на Python. Считайте, что отныне официально приобрелись к миру программирования. Вы только что запустили программу «hello world» — классическую программу, которую многие по традиции пишут в начале знакомства с новым языком. Чтобы выйти из REPL в терминале, введите `quit()`. Пользователи UNIX также могут нажать клавиши `Ctrl+D`.

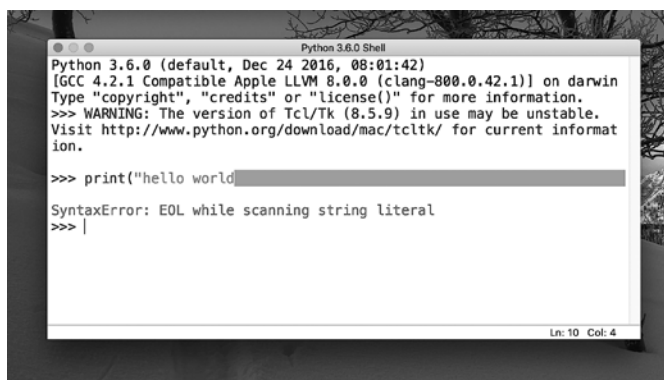


Рис. 3.4. IDLE пытается указать, где произошла ошибка. Цветовое выделение после `world` обозначает место, в котором ошибка была обнаружена

ПРИМЕЧАНИЕ

Программирование требует точности. Если при вводе `print("hello world")` пропустить всего один символ, результат может оказаться совсем другим, например, таким:

```
>>> print("hello world
      File "<stdin>", line 1
        print("hello world
              ^
SyntaxError: EOL while scanning string literal
```