

Содержание

Часть I. Введение в программирование	10
Глава 1. Вступление.....	10
Как построена эта книга	11
Сначала эндшпиль	11
Вы не одиноки	12
Преимущество самообучения	12
Почему вы должны программировать.....	13
Продолжайте этим заниматься.....	13
Оформление этой книги.....	13
Технологии, используемые в этой книге.....	14
Скачивание файлов примеров.....	14
Словарь терминов.....	15
Практикум	15
Глава 2. Начало работы.....	15
Что такое программирование	15
Что такое Python	16
Установка Python	16
Исправление проблем.....	17
Интерактивная оболочка	17
Сохранение программ.....	18
Запуск программ-примеров.....	18
Словарь терминов.....	19
Практикум	19
Глава 3. Введение в программирование	19
Примеры	20
Комментарии.....	20
Вывод	21
Строки кода.....	22
Ключевые слова	22
Отступы	22
Типы данных.....	23
Константы и переменные.....	25
Синтаксис.....	28
Ошибки и исключения.....	28
Арифметические операторы	29
Операторы сравнения	32
Логические операторы	33
Условные инструкции.....	35
Инструкции.....	39
Словарь терминов.....	41
Практикум	42
Глава 4. Функции	43
Синтаксис.....	43
Функции.....	44
Определение функций.....	44
Встроенные функции	46
Многократное использование функций	48

Обязательные и необязательные параметры.....	50
Область видимости.....	51
Обработка исключений.....	53
Строки документации.....	56
Используйте переменные, только когда это необходимо.....	57
Словарь терминов.....	57
Практикум.....	58
Глава 5. Контейнеры	58
Методы.....	59
Списки.....	59
Кортежи.....	63
Словари.....	65
Контейнеры внутри контейнеров.....	69
Словарь терминов.....	72
Практикум.....	72
Глава 6. Операции со строками	72
Тройные строки.....	73
Индексы.....	73
Строки неизменяемы.....	74
Конкатенация.....	74
Умножение строк.....	75
Изменение регистра.....	75
Метод format.....	76
Метод split.....	77
Метод join.....	77
Метод strip.....	78
Метод replace.....	78
Поиск индекса.....	78
Ключевое слово in.....	79
Управляющие символы.....	79
Новая строка.....	80
Извлечение среза.....	81
Словарь терминов.....	82
Практикум.....	82
Глава 7. Циклы	83
Циклы for.....	83
Функция range.....	87
Циклы while.....	87
Инструкция break.....	88
Инструкция continue.....	90
Вложенные циклы.....	90
Словарь терминов.....	92
Практикум.....	93
Глава 8. Модули	93
Импорт встроенных модулей.....	93
Импорт других модулей.....	95
Словарь терминов.....	96
Практикум.....	96
Глава 9. Файлы	97
Запись в файлы.....	97
Автоматическое закрытие файлов.....	98

Чтение из файлов	99
CSV-файлы.....	99
Словарь терминов.....	101
Практикум.....	101
Глава 10. Практикум. Часть I.....	102
Игра «Виселица»	103
Практикум.....	106
Глава 11. Дополнительная информация	106
Для прочтения.....	107
Другие ресурсы.....	107
Получение помощи.....	107
Часть II. Введение в объектно-ориентированное программирование	108
Глава 12. Парадигмы программирования	108
Состояние	108
Процедурное программирование	108
Функциональное программирование.....	110
Объектно-ориентированное программирование	111
Словарь терминов.....	116
Практикум.....	117
Глава 13. Четыре столпа объектно-ориентированного программирования	117
Инкапсуляция.....	117
Абстракция.....	120
Полиморфизм.....	120
Наследование	122
Композиция	125
Словарь терминов.....	126
Практикум.....	126
Глава 14. Еще об объектно-ориентированном программировании	127
Переменные класса и переменные экземпляра.....	127
Магические методы	129
Ключевое слово <code>is</code>	130
Словарь терминов.....	131
Практикум.....	131
Глава 15. Практикум. Часть II.....	132
Карты	132
Колода.....	134
Игрок	135
Игра.....	135
«Пьяница»	137
Часть III. Введение в инструменты программирования.....	141
Глава 16. bash	141
Выполнение примеров.....	142
Запуск bash	142
Команды	142
Последние команды.....	143
Относительные и абсолютные пути	143
Навигация	144
Флаги.....	146
Скрытые файлы	146

Вертикальная черта.....	146
Переменные окружения.....	147
Пользователи.....	148
Узнайте больше.....	148
Словарь терминов.....	148
Практикум.....	149
Глава 17. Регулярные выражения.....	149
Настройка.....	150
Простое совпадение.....	151
Совпадение в начале и в конце.....	152
Поиск совпадений с несколькими символами.....	153
Совпадения цифр.....	154
Повторение.....	155
Управляющие символы.....	157
Инструмент для создания регулярных выражений.....	158
Словарь терминов.....	158
Практикум.....	158
Глава 18. Системы управления пакетами.....	158
Пакеты.....	158
Pip.....	159
Виртуальные окружения.....	161
Словарь терминов.....	161
Практикум.....	161
Глава 19. Управление версиями.....	162
Репозитории.....	162
Начинаем.....	163
Помещение и извлечение данных.....	164
Пример помещения данных.....	165
Пример извлечения данных.....	167
Откат версий.....	168
Команда <code>git diff</code>	169
Дальнейшие шаги.....	170
Словарь терминов.....	170
Практикум.....	171
Глава 20. Практикум. Часть III.....	171
HTML.....	171
Парсинг контента с сайта Google Новости.....	173
Словарь терминов.....	176
Практикум.....	176
Часть IV. Введение в информатику.....	177
Глава 21. Структуры данных.....	177
Структуры данных.....	177
Стеки.....	177
Изменение порядка символов строки при помощи стека.....	179
Очереди.....	180
Очередь за билетами.....	182
Словарь терминов.....	184
Практикум.....	184
Глава 22. Алгоритмы.....	185
FizzBuzz.....	185
Последовательный поиск.....	186

Палиндром	187
Анаграмма	187
Подсчет вхождений символов	188
Рекурсия	189
Словарь терминов.....	190
Практикум	191
Часть V. Получение работы	192
Глава 23. Лучшие практические советы по программированию.....	192
Написание кода – крайнее средство	192
НПС.....	192
Ортогональность	193
У каждого фрагмента данных должно быть одно представление.....	193
У функции должна быть одна задача	193
Если на это уходит много времени, вероятно, вы совершаете ошибку	194
Делайте все самым лучшим способом	194
Соблюдайте соглашения.....	194
Используйте мощную IDE.....	194
Логирование	195
Тестирование.....	195
Анализ кода	196
Безопасность	196
Словарь терминов.....	197
Глава 24. Ваша первая работа программистом.....	198
Выберите путь	198
Получите начальный опыт	199
Запишитесь на собеседование	199
Собеседование.....	199
Подготовьтесь к собеседованию	200
Глава 25. Работа в команде	200
Освойте базис.....	201
Не задавайте вопросы, ответы на которые можете найти в Google	201
Изменение кода.....	201
Синдром самозванца	201
Глава 26. Дальнейшее обучение	202
Классика	202
Онлайн-курсы	202
Платформа Hacker News	203
Глава 27. Следующие шаги	203
Найдите себе наставника.....	203
Копайте глубже.....	203
Другие советы.....	204
Предметный указатель.....	205

*Я посвящаю эту книгу своим родителям,
Эбби и Джеймсу Альтхоф, за то, что всегда поддерживали меня.*

ЧАСТЬ I

Введение в программирование

Глава 1. Вступление

Большинство хороших программистов делают свою работу не потому, что ожидают вознаграждения или признания, а потому, что получают удовольствие от программирования.

Линус Торвалдс

Я изучал политологию в Университете Клемсона. Подумывая заняться информатикой, в первый учебный год я даже записался на курс «Введение в программирование», но быстро бросил его. Это было слишком сложно. Живя в Силиконовой долине после окончания университета, я решил, что нужно научиться программировать. Год спустя я работал в eBay на должности инженера-программиста второй категории (выше начального уровня, но ниже старшего инженера-программиста). Я не хочу, чтобы у вас создалось впечатление, будто это было легко. Это было невероятно трудно. А в промежутках между швырянием вещей в стену это было еще и очень весело.

Я начал свой путь постижения программирования с популярного языка Python. Впрочем, моя книга не направлена на обучение программированию на определенном языке. Основное внимание в ней уделяется тому, чему прочие ресурсы вас не научат. Речь идет о тех вещах, которыми мне пришлось овладеть самостоятельно. Эта книга задумывалась не для тех, кому нужно обычное введение в программирование, чтобы писать код на досуге, в качестве хобби. Она написана для тех, кто планирует заниматься программированием профессионально. Независимо от того, хотите ли вы стать инженером-программистом, предпринимателем или использовать полученные навыки в другой профессии, эта книга написана для вас.

Изучение языка программирования — лишь часть битвы; чтобы говорить на языке компьютерных ученых, вам понадобятся и другие навыки. Я научу вас всему, что постиг на пути от новичка до профессионального программиста. Я написал эту книгу, чтобы обрисовать начинающим кодерам общий контур того, что

им необходимо знать. Будучи программистом-самоучкой, я не знал, что именно мне нужно прочитать и узнать. Книги по введению в программирование все одинаковые. Они предлагают читателям лишь основы программирования на Python, либо на Ruby, а затем отправляют вас восвояси. Отклики, которые я слышал от людей, закончивших читать подобные книги, звучали так: «И что мне делать теперь? Я еще не программист, но не знаю, чему учиться дальше». Эта книга — мой ответ на подобный вопрос.

Как построена эта книга

Многим темам, охваченным всего в одной главе этой книги, посвящены целые отдельные книги. У меня не было цели детально раскрывать все темы, которые вам нужно знать. Моей целью было дать вам карту — набросок всех навыков, которыми нужно овладеть, чтобы начать программировать профессионально. Эта книга состоит из пяти частей.

Часть I. Введение в программирование. Вы напишете свою первую программу как можно скорее, надеюсь, прямо сегодня.

Часть II. Введение в объектно-ориентированное программирование. Я освещаю различные парадигмы программирования, фокусируясь на ООП. Вы создадите игру, которая продемонстрирует вам возможности программирования. После этого раздела вы «подсядете» на программирование.

Часть III. Введение в инструменты программирования. Вы научитесь использовать различные инструменты, чтобы поднять свою производительность на новый уровень. К этому моменту вы уже влюблены в программирование и хотите стать лучше. Вы узнаете больше о своей операционной системе, о том, как использовать регулярные выражения для повышения своей производительности, как устанавливать программы и пользоваться ими, а также как взаимодействовать с другими инженерами при помощи управления версиями.

Часть IV. Введение в информатику. Эта часть представляет собой краткое введение в информатику. Я касаюсь здесь двух важных тем: алгоритмов и структур данных.

Часть V. Получение работы. В заключительном разделе речь пойдет о лучших методах программирования, получении места инженера-программиста, командном взаимодействии и профессиональном совершенствовании. Я даю советы о том, как пройти собеседование, работать в команде, а также насчет того, как усовершенствовать свои навыки в будущем.

Если у вас нет опыта программирования, перед каждым разделом вы должны самостоятельно практиковаться в этом, и как можно больше. Не пытайтесь читать мою книгу слишком быстро, вместо этого используйте ее как руководство, практикуясь столько, сколько нужно в перерывах между разделами.

Сначала эндшпиль

Я учился программировать способом, полностью противоположным тому, который обычно практикуется при изучении информатики, и я структурировал книгу, следуя этому подходу. Традиционно ученики проводят много времени, изучая теорию; так много, что многие выпускники школ компьютерных наук

не знают, как программировать. В своем блоге «Почему программисты не умеют... программировать?» Джефф Этвуд пишет: «Как и меня, автора тревожит тот факт, что 199 из 200 соискателей программистских вакансий вообще не умеют писать код. Повторяю: они вообще не способны написать какой-либо код». Это подтолкнуло Этвуда к созданию задачи по кодингу **FizzBuzz** — теста на знание программирования, используемого на собеседованиях для отсеивания кандидатов. Большинство людей проваливают этот тест, вот почему в моей книге читатель тратит столько времени на получение навыков, которые пригодятся ему на практике. Не волнуйтесь, вы научитесь проходить тест FizzBuzz.

В книге «Искусство учиться» Джош Вайцкин, прославленный в фильме «В поисках Бобби Фишера», описывает, как он учился играть в шахматы в обратном порядке. Вместо того чтобы постигать первые ходы, он начал с изучения эндшпиля (когда на доске остается лишь несколько фигур). Эта стратегия позволила ему лучше понять игру, и позже он выиграл множество чемпионатов. Точно так же и я полагаю, что было бы более эффективно в первую очередь научиться программировать, а затем, когда вы уже будете до смерти хотеть знать, как там все устроено, приниматься за теорию. Вот почему я не затрагиваю теорию информатики вплоть до четвертой части книги, да и там свожу ее к минимуму. Хотя теория важна, она станет еще ценнее, когда у вас появится опыт программирования.

Вы не одиноки

Изучение программирования за пределами учебного заведения становится все более распространенным явлением. Опрос, проведенный в 2015 году Stack Overflow (онлайн-сообществом программистов), показал, что 48 % респондентов не имеют ученой степени в области компьютерных наук¹.

Преимущество самообучения

Когда меня приняли на работу в компанию eBay, я попал в одну команду с программистами из Стэнфорда, Университета Калифорнии и Дьюка, у которых были степени по информатике, а также с двумя докторами наук. В 25 лет было страшно осознавать, что мои 21-летние товарищи по команде знают о программировании и об информатике в 10 раз больше меня.

Как бы ни было страшно работать с людьми, у которых есть степень бакалавра, магистра или доктора по информатике, никогда не забывайте, что у вас есть, как я люблю это называть, «преимущество самообучения». Вы читаете эту книгу не потому, что вам так сказал преподаватель, вы читаете ее потому, что у вас есть стремление учиться, и это стремление — наибольшее преимущество из тех, что у вас есть. Кроме того, не забывайте, что некоторые наиболее успешные люди обучались программированию самостоятельно. Стив Возняк, основатель Apple, — программист-самоучка. Как и Маргарет Гамильтон, получившая Президентскую медаль Свободы за работу над программой «Аполлон» НАСА, Дэвид Карп, основатель Tumblr, Джек Дорси, основатель Твиттера, и Кевин Систром, основатель Instagram.

¹ www.infoworld.com/article/2908474/application-development/stack-overflow-survey-finds-nearly-half-have-no-degree-in-computer-science.html

Почему вы должны программировать

Программирование поможет вашей карьере независимо от выбранной профессии. Обучение программированию расширяет ваши возможности. Мне нравится придумывать новые идеи, и у меня всегда есть проекты, над которыми я хочу работать. Как только я научился программировать, я смог сесть и реализовать свои идеи, не утруждаясь поиском кого-то, кто сделал бы это за меня.

Программирование поможет вам стать лучше во всем, что вы делаете. Шутки в сторону. Существует не так много областей, в которых не пригодились бы идеально отработанные навыки решения задач. К примеру, недавно мне пришлось заниматься утомительным вопросом поиска жилья на Крейгслисте², и я смог написать программу, которая сделала эту работу за меня и отослала мне результаты по электронной почте. Знание программирования навсегда избавит вас от решения рутинных задач.

Если вы действительно хотите стать инженером-программистом, знайте, что спрос сейчас растет, а квалифицированных специалистов недостаточно для заполнения доступных вакансий. К 2020 году будет примерно миллион рабочих мест, связанных с программированием³. Даже если ваша цель не состоит в том, чтобы становиться программистом, работодатели в таких областях, как наука и финансы, тоже начинают отдавать предпочтение кандидатам с опытом в программировании.

Продолжайте этим заниматься

Если у вас нет опыта в программировании и вы чувствуете себя неуверенно, я хочу, чтобы вы знали: вам это по силам. Относительно программистов существует распространенное заблуждение, будто все они по-настоящему хороши в математике. Это не так. Чтобы научиться программировать, не нужно быть математическим гением, но все же это тяжелая работа. К счастью, большую часть материала, описанного в этой книге, легче освоить, чем вы думаете.

Чтобы улучшить свои навыки программирования, вы должны практиковаться каждый день. Единственное, что не даст вам двигаться дальше, это нежелание придерживаться учебного процесса, так что давайте рассмотрим пару способов, которые помогут вам продолжить работу.

Когда я только начинал, я составлял расписание и следил за тем, чтобы практиковаться каждый день — это помогало мне оставаться сосредоточенным.

Если вам нужна дополнительная помощь, Тим Феррисс, эксперт по производительности труда, рекомендует следующую технику поддержки мотивации: отдавайте деньги другу и члену семьи, указывая, что они должны вернуть вам их после достижения определенной цели в установленные вами сроки, а в противном случае пожертвовать организации, которая вам не нравится.

Оформление этой книги

Главы этой книги опираются друг на друга. Имейте в виду, что я стараюсь избегать повторного объяснения понятий. Когда я в первый раз ввожу новые терми-

² Крейгслист (Craigslist) — популярный в США сайт электронных объявлений. — *Прим. ред.*

³ www.wsj.com/articles/computer-programming-is-a-trade-lets-act-like-it-1407109947?mod=e2fb

ны, они выделяются жирным шрифтом. Каждое такое слово определено в конце каждой главы в словаре терминов. Также в конце глав есть задания для развития навыков программирования вместе со ссылками на их решения.

Технологии, используемые в этой книге

В этой книге представлены технологии, позволяющие как можно больше погрузиться в практику программирования. Я стараюсь быть технологическим «агностиком», сосредоточиваясь на концепциях вместо технологий.

В некоторых случаях мне пришлось выбирать между множеством различных технологий. В главе 19 «Управление версиями» (для тех читателей, которые не знакомы с управлением версиями, я все объясню позже) я описываю основы использования системы Git. Я выбрал ее, поскольку считаю отраслевым стандартом для управления версиями. Я использую Python для большинства примеров в программировании, потому что это широко изучаемый язык и его просто читать, даже если вы им никогда не пользовались. Кроме того, практически в каждой отрасли на разработчиков Python наблюдается огромный спрос.

Чтобы следовать примерам в этой книге, вам понадобится компьютер. У компьютера есть **операционная система**, программа-посредник между физическими компонентами компьютера и вами. То, что вы видите, когда смотрите на экран, называется **графическим интерфейсом пользователя** или ГИП, и это часть вашей операционной системы.

Для настольных компьютеров и ноутбуков есть три распространенных операционных системы: **Windows**, **Unix** и **Linux**. Windows — операционная система Microsoft. Unix была создана в 1970-х годах. Современная операционная система корпорации Apple основана на Unix. С этого момента, когда я использую слово «Unix», я имею в виду операционную систему корпорации Apple. Linux является операционной системой с **открытым исходным кодом**, она используется большинством мировых **серверов**. Сервер — это компьютер или компьютерная программа, выполняющая задачи вроде хостинга (расположения) веб-сайта. Открытый исходный код подразумевает, что программное обеспечение не принадлежит компании или отдельному лицу и его можно изменить или заново распространить. Linux и Unix принадлежат к **Unix-подобным операционным системам**, из чего следует, что они очень похожи. Эта книга предполагает, что вы используете компьютер под управлением операционной системы Windows, Unix или Ubuntu (популярной версии Linux).

Скачивание файлов примеров

Скачать все файлы примеров из этой книги вы можете по ссылке eksmo.ru/files/self_taught_programmer_althoff.zip. Распакуйте скачанный файл с помощью программы-архиватора, например WinZip или WinRAR, и вы увидите структуру папок, названных согласно номерам глав в этой книге. В папке Глава 02 вы найдете файлы примеров из главы 2 и т.д.

В тексте книги рядом с листингами кода указаны имена файлов. Так, пример программы «Привет, мир!» из главы 2 вы найдете в файле *Python_ex1.py* в папке *Глава 02*.

Словарь терминов

FizzBuzz: тест на программирование, используемый на собеседованиях для отсеивания кандидатов.

Linux: операционная система с открытым исходным кодом, используемая на большинстве серверов по всему миру.

Unix: операционная система, созданная в 1970-х годах. Операционная система macOS корпорации Apple основана на Unix.

Unix-подобные операционные системы: Unix и Linux.

Windows: операционная система корпорации Microsoft.

Графический интерфейс пользователя (ГИП): часть операционной системы, которую вы видите, когда смотрите на экран компьютера.

Операционная система: программа, которая является посредником между физическими компонентами компьютера и вами.

Открытый исходный код: характеристика программного обеспечения, которое не принадлежит компании или отдельному лицу, но вместо этого поддерживается группой добровольцев.

Сервер: компьютер или компьютерная программа, которая выполняет такие задачи, как хостинг веб-сайта.

Практикум

Составьте расписание на день, которое предусматривает практику программирования.

Глава 2. Начало работы

Хороший программист — это тот, кто смотрит в обе стороны, переходя дорогу с односторонним движением.

Даг Линдер

Что такое программирование

Программирование — это создание инструкций, которые выполняет компьютер. Инструкции могут указывать компьютеру вывести строку `Привет, мир!`, найти данные в Интернете или считать содержимое файла и сохранить его в базе данных. Эти инструкции называются кодом. Программисты пишут код на разных языках программирования. Раньше программирование было намного сложнее, поскольку программисты были вынуждены использовать крайне сложные **низкоуровневые языки программирования**, такие как **язык ассемблера**. Когда язык программирования является низкоуровневым, это значит, что он ближе к двоичной записи (в нулях и единицах), чем **высокоуровневый язык программирования** (язык, который больше напоминает английский), и поэтому его сложнее понять. Ниже приведен пример простой программы, написанной на ассемблере:

```
global _start
section .text
                                        bash_ex00.sh
```

```

_start:
    mov     rax, 1
    mov     rdi, 1
        mov     rsi, message
    mov     rdx, 13
    syscall
; exit(0)
    mov     eax, 60
    xor     rdi, rdi
    syscall

message:
    db     "Привет, мир!", 10

```

Ниже показана та же программа, написанная на современном языке программирования:

Python_ex001.py

```
1 | print("Привет, мир!")
```

Как видите, сегодня программистам приходится гораздо проще. Вам не нужно тратить время на изучение сложных языков низкоуровневого программирования. Вместо этого вы научитесь использовать легко читаемый язык программирования Python.

Что такое Python

Python — это язык программирования с открытым исходным кодом, созданный голландским программистом Гвидо ван Россумом и названный в честь британской труппы комиков «Монти Пайтон» (Monty Python). Одним из ключевых соображений ван Россума было то, что программисты тратят больше времени на чтение кода, чем на его написание, поэтому он решил создать легко читаемый язык. Python является одним из самых популярных и простых в освоении языков программирования в мире. Он работает на всех основных операционных системах и компьютерах и применяется везде, где только можно — от создания веб-серверов до настольных приложений. Благодаря популярности этого языка, на программистов Python сегодня существует большой спрос.

Установка Python

Чтобы следовать примерам в этой книге, вам необходимо установить Python 3. Вы можете загрузить Python для Windows и Unix по адресу python.org/downloads. Если у вас Ubuntu, Python 3 уже установлен по умолчанию. Убедитесь, что вы загружаете версию Python 3, а не Python 2. Некоторые примеры из этой книги не будут работать в версии Python 2.

Python доступен как для 32-разрядных, так и для 64-разрядных компьютеров. Если компьютер был приобретен после 2007 года, то он, скорее всего, имеет 64-битную разрядность. Если вы не уверены, поиск в Интернете поможет вам разобраться.

Если у вас операционная система Windows или macOS, загрузите 64- или 32-разрядную версию Python, откройте файл и следуйте инструкциям. Вы также можете посетить сайт theselftaughtprogrammer.io/installpython и посмотреть видео, объясняющие, как установить Python в вашей операционной системе.

Исправление проблем

Начиная с этого момента, у вас должен быть установлен Python. При возникновении проблем с установкой, перейдите к главе 11 в раздел «Получение помощи».

Интерактивная оболочка

Python поставляется с программой IDLE (сокращение от interactive development environment – интерактивная среда разработки). Кроме того, это фамилия Эрика Айдла (Eric Idle), одного из членов «Летающего цирка Монти Пайтона». IDLE – это то, где вы будете вводить свой код на Python. После загрузки Python, выполните поиск IDLE в Проводнике (Windows), Finder (macOS) или Nautilus (Ubuntu). Советую создать ярлык на Рабочем столе, чтобы упростить поиск.

Щелкните мышью по ярлычку IDLE и откроется программа со следующими строками (впрочем, к моменту чтения книги все могло поменяться, так что не беспокойтесь, если сообщение отсутствует или отличается):

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

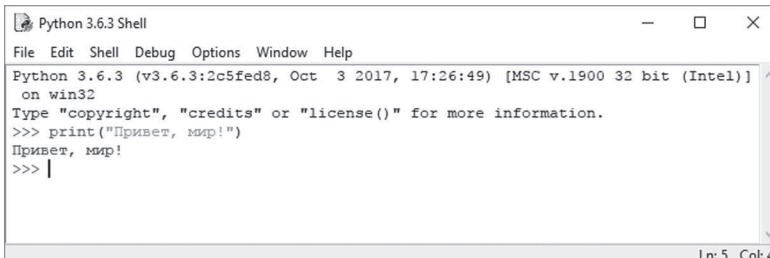
Эта программа называется интерактивной оболочкой. Вы можете вводить код Python непосредственно в интерактивную оболочку, и она выведет результаты. После приглашения >>> введите:

```
1 | print("Привет, мир!")
```

Затем нажмите клавишу **Enter**.

IDLE может отвергать код, копируемый из Kindle, других электронных книг или текстовых редакторов, таких как Microsoft Word. Если вы скопировали и вставили код, но получили сообщение об ошибке, попробуйте набрать код вручную непосредственно в оболочке. Необходимо вводить код точно так, как он написан в примере, включая кавычки, круглые скобки и любые другие знаки препинания.

Интерактивная оболочка ответит выводом строки **Привет, мир!**



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print("Привет, мир!")
Привет, мир!
>>> |
```

Ln: 5 Col: 4