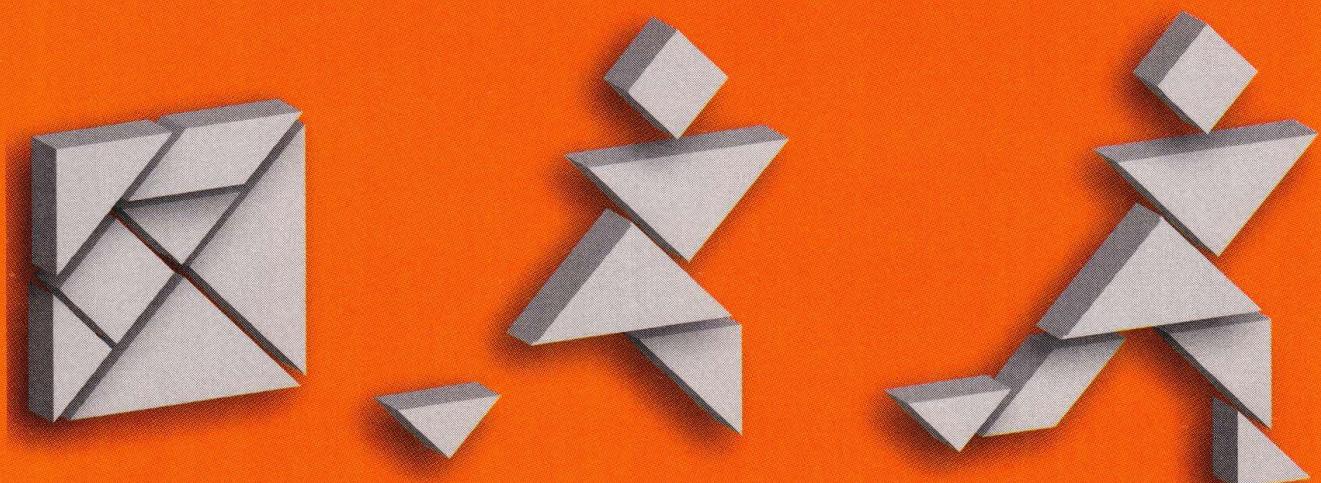


Анатолий Постолит

# Python, Django и PyCharm

*для начинающих*



Материалы  
на [www.bhv.ru](http://www.bhv.ru)

bhv®

**Анатолий Постолит**

# **Python, Django и PyCharm**

*для начинающих*

Санкт-Петербург  
«БХВ-Петербург»

2021

УДК 004.43  
ББК 32.973.26-018.1  
П63

**Постолит А. В.**

П63 Python, Django и PyChart для начинающих. — СПб.: БХВ-Петербург,  
2021. — 464 с.: ил. — (Для начинающих)

ISBN 978-5-9775-6779-4

Книга посвящена вопросам разработки веб-приложений с использованием языка Python, фреймворка Django и интерактивной среды разработки PyChart. Рассмотрены основные технологии и рабочие инструменты создания приложений, даны основы языка Python. Описаны фреймворк Django и структура создаваемых в нем веб-приложений. На простых примерах показаны обработка и маршрутизация запросов пользователей, формирование ответных веб-страниц. Рассмотрено создание шаблонов веб-страниц и форм для пользователей. Показано взаимодействие пользователей с различными типами баз данных через модели. Описана работа с базами данных через встроенные в Django классы без использования SQL-запросов. Приведен пошаговый пример создания сайта от формирования шаблона до его администрирования и развертывания в сети Интернет. Электронный архив на сайте издательства содержит коды всех примеров.

*Для программистов*

УДК 004.43  
ББК 32.973.26-018.1

**Группа подготовки издания:**

Руководитель проекта	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Людмила Гауль</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Дизайн серии	<i>Марины Дамбировой</i>
Оформление обложки	<i>Каринь Соловьевой</i>

Подписано в печать 30.04.21.  
Формат 70×100<sup>1/16</sup>. Печать офсетная. Усл. печ. л. 37,41.  
Тираж 1200 экз. Заказ №1045.  
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.  
Отпечатано с готового оригинал-макета  
ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-5-9775-6779-4

© ООО "БХВ", 2021  
© Оформление. ООО "БХВ-Петербург", 2021

# Оглавление

<b>Предисловие .....</b>	<b>9</b>
<b>Глава 1. Веб-технологии и инструментальные средства для разработки веб-приложений.....</b>	<b>14</b>
1.1. Базовые сведения о веб-технологиях .....	15
1.1.1. Технологии клиентского программирования .....	18
1.1.2. Технологии серверного программирования .....	18
1.2. Базовые сведения о HTML.....	19
1.2.1. Теги для представления текста на HTML-страницах.....	21
1.2.2. Списки.....	24
1.2.3. Таблицы .....	26
1.2.4. Гиперссылки.....	29
1.3. Каскадные таблицы стилей (CSS) .....	30
1.4. Возможности использования JavaScript .....	32
1.5. Интерпретатор Python .....	34
1.5.1. Установка Python в Windows.....	34
1.5.2. Установка Python в Linux .....	37
1.5.3. Проверка интерпретатора Python.....	38
1.6. Интерактивная среда разработки программного кода PyCharm.....	39
1.6.1. Установка PyCharm в Windows .....	40
1.6.2. Установка PyCharm в Linux .....	42
1.6.3. Проверка PyCharm .....	43
1.7. Установка пакетов в Python с использованием менеджера пакетов pip .....	45
1.7.1. Репозиторий пакетов программных средств PyPI.....	46
1.7.2. pip — менеджер пакетов в Python.....	46
1.7.3. Использование менеджера пакетов pip .....	47
1.8. Фреймворк Django для разработки веб-приложений.....	49
1.9. Фреймворк SQLiteStudio для работы с базами данных.....	53
1.10. Краткие итоги.....	55
<b>Глава 2. Основы языка программирования Python .....</b>	<b>56</b>
2.1. Первая программа в среде интерпретатора Python.....	57
2.2. Базовые операторы языка Python .....	61
2.2.1. Переменные .....	62

2.2.2. Функции .....	64
2.2.3. Массивы .....	69
2.2.4. Условия и циклы .....	70
Условия.....	70
Циклы.....	72
2.2.5. Классы и объекты.....	74
Классы.....	76
Объекты .....	78
2.2.6. Создание классов и объектов на примере автомобиля .....	80
2.2.7. Программные модули .....	82
Установка модуля .....	83
Подключение и использование модуля.....	84
2.3. Краткие итоги.....	84
<b>Глава 3. Знакомимся с веб-фреймворком Django.....</b>	<b>86</b>
3.1. Общие представления о Django.....	86
3.2. Структура приложений на Django .....	89
3.3. Первый проект на Django.....	91
3.4. Первое приложение на Django.....	99
3.5. Краткие итоги.....	104
<b>Глава 4. Представления и маршрутизация.....</b>	<b>105</b>
4.1. Обработка запросов пользователей.....	105
4.2. Маршрутизация запросов пользователей в функциях <i>path</i> и <i>re_path</i> .....	109
4.3. Очередность маршрутов .....	111
4.4. Основные элементы синтаксиса регулярных выражений .....	111
4.5. Параметры представлений .....	113
4.5.1. Определение параметров через функцию <i>re_path()</i> .....	113
4.5.2. Определение параметров через функцию <i>path()</i> .....	117
4.5.3. Определение параметров по умолчанию в функции <i>path()</i> .....	118
4.6. Параметры строки запроса пользователя .....	120
4.7. Переадресация и отправка пользователю статусных кодов .....	123
4.7.1. Переадресация .....	123
4.7.2. Отправка пользователю статусных кодов .....	125
4.8. Краткие итоги.....	126
<b>Глава 5. Шаблоны.....</b>	<b>127</b>
5.1. Создание и использование шаблонов .....	127
5.2. Класс <i>TemplateResponse</i> .....	136
5.3. Передача данных в шаблоны .....	137
5.4. Передача в шаблон сложных данных .....	140
5.5. Статичные файлы .....	142
5.5.1. Основы каскадных таблиц стилей .....	142
5.5.2. Использование статичных файлов в приложениях на Django.....	147
5.5.3. Использование класса <i>TemplateView</i> для вызова шаблонов HTML-страниц .....	153
5.5.4. Конфигурация шаблонов HTML-страниц.....	158
5.5.5. Расширение шаблонов HTML-страниц на основе базового шаблона .....	160

5.6. Использование специальных тегов в шаблонах HTML-страниц.....	163
5.6.1. Тег для вывода текущих даты и времени.....	163
5.6.2. Тег для вывода информации по условию.....	165
5.6.3. Тег для вывода информации в цикле.....	167
5.6.4. Тег для задания значений переменным.....	169
5.7. Краткие итоги.....	170
<b>Глава 6. Формы .....</b>	<b>171</b>
6.1. Определение форм.....	171
6.2. Использование в формах POST-запросов.....	176
6.3. Использование полей в формах Django .....	178
6.3.1. Настройка среды для изучения полей разных типов.....	178
6.3.2. Типы полей в формах Django и их общие параметры .....	179
6.3.3. Поле <i>BooleanField</i> для выбора решения: да\нет .....	184
6.3.4. Поле <i>NullBooleanField</i> для выбора решения: да\нет .....	185
6.3.5. Поле <i>CharField</i> для ввода текста .....	186
6.3.6. Поле <i>EmailField</i> для ввода электронного адреса.....	187
6.3.7. Поле <i>GenericIPAddressField</i> для ввода IP-адреса .....	189
6.3.8. Поле <i>RegexField</i> для ввода текста.....	189
6.3.9. Поле <i>SlugField</i> для ввода текста .....	190
6.3.10. Поле <i>URLField</i> для ввода универсального указателя ресурса (URL) .....	191
6.3.11. Поле <i>UUIDField</i> для ввода универсального уникального идентификатора UUID .....	192
6.3.12. Поле <i>ComboField</i> для ввода текста с проверкой соответствия заданным форматам .....	193
6.3.13. Поле <i>FilePathField</i> для создания списка файлов .....	194
6.3.14. Поле <i>FileField</i> для выбора файлов.....	196
6.3.15. Поле <i>ImageField</i> для выбора файлов изображений .....	198
6.3.16. Поле <i>DateField</i> для ввода даты .....	198
6.3.17. Поле <i>TimeField</i> для ввода времени .....	200
6.3.18. Поле <i>DateTimeField</i> для ввода даты и времени .....	201
6.3.19. Поле <i>DurationField</i> для ввода промежутка времени .....	201
6.3.20. Поле <i>SplitDateTimeField</i> для раздельного ввода даты и времени .....	202
6.3.21. Поле <i>IntegerField</i> для ввода целых чисел.....	203
6.3.22. Поле <i>DecimalField</i> для ввода десятичных чисел .....	204
6.3.23. Поле <i>FloatField</i> для ввода чисел с плавающей точкой .....	206
6.3.24. Поле <i>ChoiceField</i> для выбора данных из списка.....	206
6.3.25. Поле <i>TypedChoiceField</i> для выбора данных из списка.....	207
6.3.26. Поле <i>MultipleChoiceField</i> для выбора данных из списка .....	209
6.3.27. Поле <i>TypedMultipleChoiceField</i> для выбора данных из списка .....	210
6.4. Настройка формы и ее полей.....	211
6.4.1. Изменение внешнего вида поля с помощью параметра <i>widget</i> .....	211
6.4.2. Задание начальных значений полей с помощью свойства <i>initial</i> .....	213
6.4.3. Задание порядка следования полей на форме .....	213
6.4.4. Задание подсказок к полям формы .....	215
6.4.5. Настройки вида формы .....	216
6.4.6. Проверка (валидация) данных.....	217

6.4.7. Детальная настройка полей формы .....	222
6.4.8. Присвоение стилей полям формы.....	226
6.5. Краткие итоги.....	231
<b>Глава 7. Модели данных Django.....</b>	<b>232</b>
7.1. Создание моделей и миграции базы данных .....	233
7.2. Типы полей в модели данных Django .....	237
7.3. Манипуляция с данными в Django на основе CRUD .....	240
7.3.1. Добавление данных в БД.....	241
7.3.2. Чтение данных из БД .....	241
Метод <i>get()</i> .....	241
Метод <i>get_or_create()</i> .....	242
Метод <i>all()</i> .....	242
Метод <i>filter()</i> .....	242
Метод <i>exclude()</i> .....	242
Метод <i>in_bulk()</i> .....	243
7.3.3. Обновление данных в БД .....	243
7.3.4. Удаление данных из БД.....	245
7.3.5. Просмотр строки SQL-запроса к базе данных.....	245
7.4. Пример работы с объектами модели данных (чтение и запись информации в БД) .....	245
7.5. Пример работы с объектами модели данных: редактирование и удаление информации из БД .....	249
7.6. Организация связей между таблицами в модели данных.....	255
7.6.1. Организация связей между таблицами «один-ко-многим».....	255
7.6.2. Организация связей между таблицами «многие-ко-многим».....	261
7.6.3. Организация связей между таблицами «один-к-одному» .....	265
7.7. Краткие итоги.....	268
<b>Глава 8. Пример создания веб-сайта на Django .....</b>	<b>269</b>
8.1. Создание структуры сайта при помощи Django.....	269
8.2. Разработка структуры моделей данных сайта «Мир книг» .....	280
8.3. Основные элементы моделей данных в Django .....	282
8.3.1. Поля и их аргументы в моделях данных .....	283
8.3.2. Метаданные в моделях Django.....	285
8.3.3. Методы в моделях Django .....	286
8.3.4. Методы работы с данными в моделях Django .....	287
8.4. Формирование моделей данных для сайта «Мир книг».....	289
8.4.1. Модель для хранения жанров книг .....	290
8.4.2. Модель для хранения языков книг .....	290
8.4.3. Модель для хранения авторов книг .....	291
8.4.4. Модель для хранения книг .....	292
8.4.5. Модель для хранения отдельных экземпляров книг и их статуса.....	295
8.5. Административная панель Django Admin.....	303
8.5.1. Регистрация моделей данных в Django Admin .....	303
8.5.2. Работа с данными в Django Admin.....	304
8.6. Изменение конфигурации административной панели Django .....	314
8.6.1. Регистрация класса ModelAdmin .....	315
8.6.2. Настройка отображения списков .....	316

8.6.3. Добавление фильтров к спискам.....	317
8.6.4. Формирование макета с подробным представлением элемента списка.....	321
8.6.5. Разделение страницы на секции с отображением связанной информации.....	322
8.6.6. Встроенное редактирование связанных записей .....	324
8.7. Краткие итоги.....	328

## **Глава 9. Пример создания веб-интерфейса для пользователей сайта**

<b>«Мир книг».....</b>	<b>329</b>
9.1. Последовательность создания пользовательских страниц сайта «Мир книг».....	329
9.2. Определение перечня и URL-адресов страниц сайта «Мир книг» .....	330
9.3. Создание главной страницы сайта «Мир книг» .....	331
9.3.1. Создание URL-преобразования.....	331
9.3.2. Создание представления (view).....	333
9.3.3. Создание базового шаблона сайта и шаблона для главной страницы сайта «Мир книг» .....	334
9.4. Отображение списков и детальной информации об элементе списка .....	342
9.5. Краткие итоги.....	357

## **Глава 10. Расширение возможностей для администрирования сайта**

<b>«Мир книг» и создание пользовательских форм .....</b>	<b>358</b>
10.1. Сессии в Django .....	359
10.2. Аутентификация и авторизация пользователей в Django .....	363
10.2.1. Немного об аутентификации пользователей в Django.....	363
10.2.2. Создание отдельных пользователей и групп пользователей.....	364
10.2.3. Создание страницы регистрации пользователя при входе на сайт.....	369
10.2.4. Создание страницы для сброса пароля пользователя .....	375
10.3. Проверка подлинности входа пользователя в систему.....	381
10.4. Формирование страниц сайта для создания заказов на книги .....	383
10.5. Работа с формами .....	392
10.5.1. Краткий обзор форм в Django.....	393
10.5.2. Управление формами в Django.....	394
10.5.3. Форма для ввода и обновления информации об авторах книг на основе класса <i>Form()</i> .....	396
10.5.4. Форма для ввода и обновления информации о книгах на основе класса <i>ModelForm()</i> .....	406
10.6. Краткие итоги.....	414

## **Глава 11. Публикация сайта в сети Интернет.....** 415

11.1. Подготовка инфраструктуры сайта перед публикацией в сети Интернет .....	415
11.1.1. Окружение развертывания сайта в сети Интернет.....	416
11.1.2. Выбор хостинг-провайдера.....	417
11.2. Подготовка веб-сайта к публикации .....	418
11.3. Пример размещения веб-сайта на сервисе Heroku.....	420
11.3.1. Создание репозитория приложения на GitHub.....	421
11.3.2. Подключение веб-сервера Gunicorn .....	431
11.3.2. Пакеты для обеспечения доступа к базе данных на Heroku .....	433
11.3.3. Настройка доступа к базе данных Postgres на Heroku .....	435
11.3.4. Обслуживание статичных файлов на Heroku .....	436

11.3.5. Подключение библиотеки WitheNoise .....	437
11.3.6. Задание требований к Python .....	439
11.3.7. Настройка среды выполнения .....	440
11.3.8. Получение аккаунта на Heroku .....	442
11.3.9. Создание и запуск приложения на Heroku.....	446
11.4. Другие ресурсы для публикации веб-сайта .....	449
11.5. Краткие итоги.....	449
<b>Послесловие.....</b>	<b>450</b>
<b>Список источников и литературы.....</b>	<b>451</b>
<b>Приложение. Описание электронного архива.....</b>	<b>453</b>
<b>Предметный указатель .....</b>	<b>458</b>

# Предисловие

С развитием цифровых технологий и уходом в прошлое ряда популярных ранее специальностей молодые люди все чаще встают перед выбором перспективной, интересной и актуальной профессии. Международное интернет-сообщество считает, что одним из самых перспективных вариантов такого выбора является профессия веб-программиста. Именно эти специалисты формируют облик сети Интернет и создают технологические тренды будущего.

Каждую секунду в Интернете появляется от 3 до 5 новых сайтов, а каждую минуту — 80 новых интернет-пользователей. Все это технологическое «цунами» управляемое разумом и умелыми руками веб-разработчиков. Зарплата этих специалистов вполне соответствует важности и актуальности их деятельности. Даже начинающие программисты на отечественном рынке могут рассчитывать на заработную плату от 50 тыс. рублей в месяц, а опытные программисты в России и за рубежом имеют доход, превышающий указанную цифру в десятки раз.

Специалисты ИТ-технологий уже много лет подряд занимают первые позиции в рейтингах кадровых агентств, как представители наиболее востребованных профессий на отечественном и мировом рынке труда. Постоянная нехватка квалифицированных программистов дает высокий шанс молодым и целеустремленным новичкам для входа в эту профессию.

Согласно данным Ассоциации предприятий компьютерных и информационных технологий (АПКИТ) спрос на ИТ-специалистов ежегодно остается на высоком уровне, однако кандидатов по-прежнему не хватает. В ближайшие несколько лет дефицит будет только нарастать, так как ИТ-специалисты нужны повсеместно. Если раньше ими интересовались сугубо профильные компании (разработчики софта и онлайн-платформ), то сейчас квалифицированные кадры ИТ-профилей требуются практически везде. И речь здесь идет не о дефиците, а о катастрофическом дефиците. Уже много лет подряд сервис по поиску работы SuperJob фиксирует огромный неудовлетворенный спрос на ИТ-специалистов.

Рассматриваемый в этой книге язык программирования Python — один из самых популярных языков программирования, и области его применения только расширяются. Последние несколько лет он входит в ТОП-3 самых востребованных язы-

ков. Задействуя Python, можно решать множество научных проблем и задач в области бизнеса. На Западе к нему обращаются ученые (математики, физики, биологи), так как изучить его не слишком сложно. Он используется при реализации нейронных сетей, для написания веб-сайтов и мобильных приложений. В целом это универсальный язык, входящий в тройку языков для анализа больших данных. В течение последних 5 лет Python-разработчики востребованы на рынке труда, специалистов в этой сфере до сих пор не хватает.

Еще одна причина, по которой следует обратить внимание на использование языка Python и фреймворка Django для веб-разработки, — это развитие систем искусственного интеллекта. На Python реализовано большое количество библиотек, облегчающих создание программного обеспечения для систем искусственного интеллекта, а Django обеспечивает развертывание приложений в сети Интернет. С использованием этого tandemа можно создавать приложения для беспилотных автомобилей, для интеллектуальных транспортных систем, для телемедицины, систем обучения, распознавания объектов в системах безопасности и т. п. Это очень востребованные и актуальные сферы, где наблюдается еще большая потребность в специалистах. Здесь молодые, талантливые и целеустремленные молодые люди могут найти как возможность самореализации, так и достойную оплату своего труда.

Каждая организация, принимающая на работу ИТ-специалиста, требует знаний, которые будут полезны именно в ее работе. Однако общее правило таково, что чем больше популярных и необходимых языков программирования, фреймворков и баз данных вы знаете (Js, HTML, C#, C++, Python, PHP, Django, SQL) и чем больше ваш опыт работы, тем выше шансы на удачное трудоустройство и достойную зарплату.

Перед теми, кто решил освоить специальность веб-программиста, встает непростой выбор — с чего же правильно начать. Конечно, всегда существует возможность получить полноценное ИТ-образование в одном из ведущих технических вузов ранга МГУ, МГТУ им. Н. Баумана, СПбГУ, МФТИ, ИТМО. Но обучение в них обойдется в круглую сумму от 60 до 350 тыс. рублей в год. Существует и более быстрый и дешевый вариант стать веб-разработчиком «с нуля», пройдя краткосрочные онлайн-курсы. Однако практикующие программисты уверяют, что на начальном этапе самый правильный способ обучиться веб-программированию — освоить его самостоятельно. Так можно не только избежать серьезных расходов, но и получить только те практические навыки, которые пригодятся в будущей работе. Полученные самостоятельно базовые знания впоследствии можно расширить, обучаясь уже на соответствующих курсах или в специализированном вузе.

Эта книга предназначена как для начинающих программистов (школьников и студентов), так и для специалистов с опытом, которые планируют заниматься или уже занимаются разработкой веб-приложений с использованием Python. Сразу следует отметить, что веб-программирование требует от разработчиков больших знаний, умений и усилий, чем программирование традиционных приложений. Здесь, кроме основного языка, который реализует логику приложения, требуется еще и знание структуры HTML-документов, основ работы с базами данных, принципов взаимодействия удаленных пользователей с веб-приложением. Если некоторые разделы

книги вам покажутся трудными для понимания и восприятия, то не стоит отчаиваться. Нужно просто последовательно, по шагам повторить приведенные в книге примеры. А когда вы увидите результаты работы программного кода, появится ясность — как работает тот или иной элемент изучаемого фреймворка.

В книге рассмотрены практически все элементарные действия, которые выполняют программисты, работая над реализацией веб-приложений, приведено множество примеров и проверенных программных модулей. Рассмотрены базовые классы фреймворка Django, методы и свойства каждого из классов и примеры их использования. Книга, как уже отмечалось, предназначена как для начинающих программистов, приступивших к освоению языка Python, так и специалистов, имеющих опыт программирования на других языках. В этом издании в меньшей степени освещены вопросы дизайна веб-приложений, а больше внимания уделено технологиям разработки веб-приложений на практических примерах.

*Первая глава* книги посвящена знакомству с веб-технологиями, рассмотрены принципиальные различия между приложениями, работающими на сервере и на стороне клиента. Приводятся базовые сведения о HTML-страницах, их структуре и основных тегах языка HTML, позволяющих выводить и форматировать информацию. Представлены перечень и краткие описания языков программирования, позволяющих создавать веб-приложения. Здесь же показана последовательность шагов по формированию инструментальной среды пользователя для разработки веб-приложений (установка и настройка программных средств). Это в первую очередь интерпретатор Python, интерактивная среда разработки программного кода PyCharm, фреймворк Django и менеджер работы с базами данных SQLite.

Во *второй главе* рассматриваются базовые элементы языка Python: переменные, функции, массивы, условия и циклы, классы и объекты, созданные на основе этих классов. Это самые простые команды, которых вполне хватит для понимания примеров, приведенных в последующих главах. Однако этих сведений не достаточно для полноценной работы с Python, да и целью книги является не столько описание самого Python, сколько специализированных библиотек для созданий веб-приложений. А для более глубокого изучения Python необходимо воспользоваться специальной литературой.

*Третья глава* посвящена основным понятиям и определениям, которые используются в фреймворке Django. В ней также описана структура приложений на Django. С использованием интерактивной среды PyCharm мы создадим здесь простейший первый проект на Django и сформируем в этом проекте приложение.

В *четвертой главе* приводятся основные понятия о представлениях (view) и маршрутизации запросов пользователей. Это весьма важные компоненты, которые определяют, что должно делать веб-приложение при поступлении различных запросов от удаленных пользователей. Здесь же описан синтаксис так называемых *регулярных выражений*, которые преобразуют запросы пользователей в адреса перехода к тем или иным страницам сайта.

В *пятой главе* рассмотрены вопросы создания шаблонов HTML-страниц. На конкретных примерах показано, как передать в шаблоны простые и сложные данные.

Приведены примеры использования статичных файлов в приложениях на Django. Рассмотрена возможность расширения шаблонов HTML-страниц на основе базового шаблона — это, по сути, основной прием, который применяется практически на всех сайтах. Приведены также примеры использования в шаблонах HTML-страниц специальных тегов (для вывода текущей даты и времени, вывода информации по условию, вывода информации в цикле и для задания значений переменным).

*В шестой главе* сделан обзор пользовательских форм. Формы — это основной интерфейс, благодаря которому удаленные пользователи имеют возможность вносить информацию в удаленную базу данных. Из этой главы вы узнаете, как использовать в формах POST-запросы, в ней также подробно описаны поля, которые можно задействовать для ввода данных, и приводятся короткие примеры применения каждого типа поля. Кроме того, рассмотрены примеры изменения внешнего вида полей с помощью виджетов (widget), настроек вида полей, валидации данных и стилизации полей на формах Django.

*Седьмая глава* посвящена изучению моделей Django. Модели, по своей сути, представляют собой описание таблиц и полей базы данных (БД), используемой веб-приложением. На основе моделей будут автоматически созданы классы, через свойства и методы которых приложение станет взаимодействовать с базой данных. Соответственно, через классы будут выполняться все процедуры работы с данными (добавление, модификация, удаление записей из таблиц БД), при этом отпадает необходимость в написании SQL-запросов — Django будет создавать их самостоятельно и задействовать при манипулировании данными. Кроме того, с использованием процедуры миграции Django в автоматическом режиме создаст необходимые таблицы в различных СУБД (SQLite, PostgreSQL, MySQL, Oracle). Из этой главы вы также узнаете, что в дополнение к официально поддерживаемым базам данных существуют сторонние серверы, которые позволяют использовать с Django другие базы данных (SAP SQL, Anywhere, IBM DB2, Microsoft SQL Server, Firebird, ODBC).

*В восьмой главе* мы создадим модели для достаточно простого «учебного» сайта «Мир книг» и соответствующие таблицы в системе управления базами данных (СУБД) SQLite. Возможность работы с данными этого сайта здесь будет показана через встроенную в Django административную панель.

*В девятой главе* приведен пример создания веб-интерфейса для пользователей сайта «Мир книг». Мы определим для этого сайта перечень страниц и их интернет-адресов (URL), создадим представления (views) для обработки запросов пользователей, базовый шаблон сайта и шаблоны прочих страниц сайта. Это простейшие страницы для получения информации из БД на основе запросов пользователей.

*В десятой главе* показано, как можно расширить возможности администрирования сайта «Мир книг» и обеспечить ввод и редактирование данных со стороны удаленных пользователей через формы Django. Здесь рассмотрено такое понятие, как *сессия*, представлены процедуры создания страниц авторизация пользователей, проверки подлинности входа пользователей в систему, изменение внешнего вида страниц для зарегистрированных пользователей. Подробно на примерах показано использование классов Form и ModelForm.

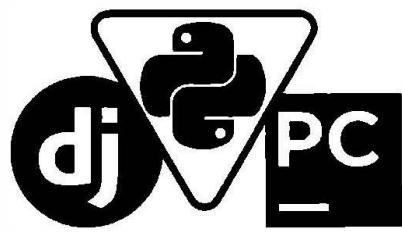
Заключительная, одиннадцатая глава посвящена теме публикации сайтов, разработанных на Django, в сети Интернет. Эта, казалось бы, несложная процедура требует от разработчика определенной квалификации и навыков. На удаленном сервере потребуется создать соответствующее окружение для Python, подгрузить необходимые библиотеки и модули, подключить нужную СУБД, настроить пути к статичным файлам рисунков и стилей. В качестве примера мы выполним процедуру публикации «учебного» сайта «Мир книг» на бесплатном сетевом ресурсе Heroku. В этой же главе вы найдете ссылки на инструкции по публикации сайтов, разработанных на Python и Django, на российских хостинг-ресурсах.

На протяжении всей книги раскрываемые вопросы сопровождаются достаточно упрощенными, но полностью законченными примерами. Ход решения той или иной задачи показан на большом количестве иллюстративного материала. Желательно изучение тех или иных разделов осуществлять, сидя непосредственно за компьютером, — тогда вы сможете последовательно повторять выполнение тех шагов, которые описаны в примерах, и тут же видеть результаты своих действий, что в значительной степени облегчит восприятие материала книги. Наилучший способ обучения — это практика. Все листинги программ приведены на языке Python, а шаблоны веб-страниц — в виде HTML-файлов. Это прекрасная возможность познакомиться с языком программирования Python и понять, насколько он прост и удобен в использовании.

### ЭЛЕКТРОННЫЙ АРХИВ

Сопровождающий книгу электронный архив содержит программный код ряда наиболее важных листингов книги (см. приложение). Архив доступен для закачки с FTP-сервера издательства «БХВ» по ссылке <ftp://ftp.bhv.ru/9785977567794.zip>, ссылка на него также ведет со страницы книги на сайте <https://bhv.ru>.

Итак, если вас заинтересовали вопросы создания веб-приложений с использованием Python, Django и PyCharm, то самое время перейти к изучению материалов этой книги.



## ГЛАВА 1

# Веб-технологии и инструментальные средства для разработки веб-приложений

В настоящее время веб-технологии стремительно развиваются, проникая в самые разнообразные сферы нашей жизни. Для компаний присутствие в сети Интернет — это возможность рассказать о своих товарах и услугах, найти потенциальных партнеров и клиентов, снизить издержки за счет интернет-торговли и использования «облачных» сервисов. Рядовые пользователи активно пользуются интернет-магазинами, интернет-банкингом, общаются в социальных сетях, могут получать через Интернет государственные услуги.

Для разработки веб-приложений существует множество языков программирования, каждый из которых имеет свои особенности. Но из них хочется выделить Python, как популярную универсальную среду разработки программного кода с тридцатилетней историей.

Python — интерпретируемый язык программирования — в конце 1989 года создал Гвидо Ван Россум, и он очень быстро стал популярным и востребованным у программистов. В подтверждение этого можно упомянуть компании-гиганты: Google, Microsoft, Facebook, Yandex и многие другие, которые используют Python для реализации глобальных проектов.

Область применения Python очень обширна — это и обработка научных данных, и системы управления жизнеобеспечением, а также игры, веб-ресурсы, системы искусственного интеллекта. За все время существования Python плодотворно использовался и динамично развивался. Для него создавались стандартные библиотеки, обеспечивающие поддержку современных технологий — например, работы с базами данных, протоколами Интернета, электронной почтой, машинным обучением и многим другим.

Для ускорения процесса написания программного кода удобно использовать специализированную инструментальную среду — так называемую *интегрированную среду разработки* (IDE, Integrated Development Environment). Эта среда включает полный комплект средств, необходимых для эффективного программирования на Python. Обычно в состав IDE входят текстовый редактор, компилятор или интер-

претатор, отладчик и другое программное обеспечение. Использование IDE позволяет увеличить скорость разработки программ (при условии предварительного обучения работе с такой инструментальной средой).

Возможность создавать веб-приложения не встроена в основные функции Python. Для реализации задач подобного класса на Python нужен дополнительный инструментарий. Таким инструментарием является веб-фреймворк Django. Django считается лучшим веб-фреймворком, написанным на Python. Этот инструмент удобно использовать для создания сайтов, работающих с базами данных, он делает веб-разработку на Python очень качественной и удобной.

Из материалов этой главы вы получите базовые сведения о языке HTML, а также узнаете:

- что такое веб-технологии;
- в чем различия технологий клиентского и серверного программирования;
- какие теги можно использовать для представления текста на HTML-страницах;
- что такое каскадные таблицы стилей;
- какие возможности таит в себе скриптовый язык JavaScript;
- что такое интерпретатор Python и как его установить;
- как начать использовать интерактивную среду разработки программного кода PyCharm;
- как можно установить различные дополнительные пакеты в Python с использованием менеджера пакетов pip;
- как можно установить и начать использовать веб-фреймворк Django;
- как установить фреймворк SQLite Studio для работы с базами данных.

## 1.1. Базовые сведения о веб-технологиях

Под веб-технологиями в дальнейшем мы будем понимать всю совокупность средств для организации взаимодействия пользователей с удаленными приложениями. Поскольку в каждом сеансе взаимодействуют две стороны: сервер и клиент, то и веб-приложения можно разделить на две группы: приложения на стороне сервера (*server-side*) и приложения на стороне клиента (*client-side*). Благодаря веб-приложениям удаленному пользователю доступны не только статические документы, но и сведения из базы данных. Использование баз данных в сети Интернет приобрело огромную популярность и практически стало отдельной отраслью компьютерной науки.

Но для начала разберемся с основными понятиями веб-технологий: что такое веб-сайт и веб-страница. Часто неопытные пользователи их неправомерно смешивают. *Веб-страница* — это минимальная логическая единица в сети Интернет, которая представляет собой документ, однозначно идентифицируемый уникальным интернет-адресом (URL, Uniform Resource Locator, унифицированным указателем ресурс-

са). *Веб-сайт* — это набор тематически связанных веб-страниц, находящихся на одном сервере и принадлежащих одному владельцу. В частном случае веб-сайт может состоять из единственной веб-страницы. Сеть Интернет является совокупностью всех веб-сайтов.

Для формирования веб-страниц используется язык разметки гипертекста HTML (Hyper Text Markup Language). С его помощью осуществляется логическая (смысловая) разметка документа (веб-страницы). Для целей управления внешним видом веб-страниц используются каскадные таблицы стилей (от англ. Cascading Style Sheets, CSS). Сформированные на сервере HTML-документы можно просмотреть на компьютере клиента с помощью специальной программы — браузера.

Как было отмечено ранее, совокупность связанных между собой веб-страниц представляет собой веб-сайт. Сайты можно условно разделить на статические и динамические.

□ **Статический сайт** — это сайт с неизменным информационным наполнением. Основу статического сайта составляют неизменные веб-страницы, разработанные с использованием стандартной HTML-технологии. Страницы сайта хранятся в виде HTML-кода в файловой системе сервера. Естественно, на таком сайте могут присутствовать и различные видеоролики и анимация. Основная отличительная особенность статического сайта заключается в том, что веб-страницы такого сайта создаются заранее. Для редактирования содержимого страниц и обновления сайта страницы модифицируют вручную с применением HTML-редактора и затем заново загружают на сайт. Схема взаимодействия клиента со статическим сайтом приведена на рис. 1.1.

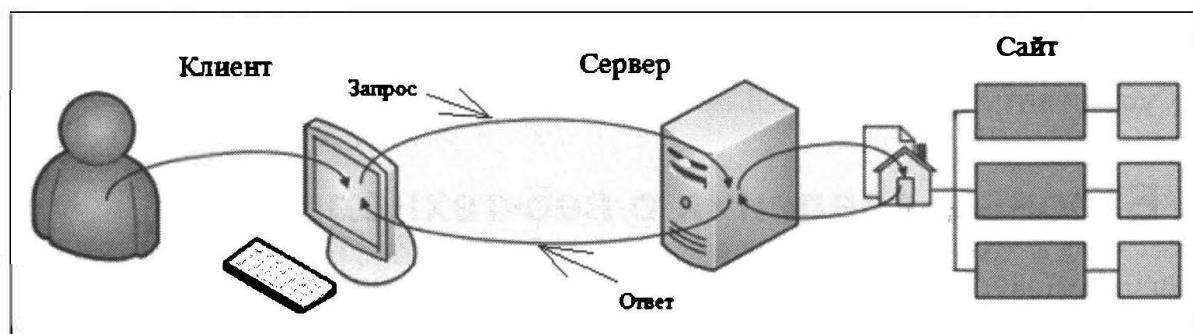


Рис. 1.1. Схема взаимодействия клиента со статическим сайтом

Такая схема приемлема в том случае, когда содержимое сайта довольно постоянно и изменяется сравнительно редко. Если же информация, размещенная на статическом сайте, требует постоянной актуализации и обновления, для его поддержания неизбежны внушительные трудозатраты. Таким образом, статический сайт дешевле в разработке и технической поддержке, но эти достоинства могут нивелироваться серьезными недостатками, связанными с необходимостью оперативного обновления актуальной информации и достаточно высокой трудоемкостью модификации.

□ **Динамический сайт** — это сайт с динамическим информационным наполнением. Динамические страницы также формируются с помощью HTML, но такие

страницы обновляются постоянно, нередко при каждом новом обращении к ним. Динамические сайты основываются на статических данных и HTML-разметке, но дополнительно содержат программную часть (скрипты), а также базу данных (БД), благодаря которым страница «собирается» из отдельных фрагментов в режиме реального времени. Это позволяет обеспечить гибкость в подборе и представлении информации, соответствующей конкретным запросам посетителей сайта.

Таким образом, динамический сайт состоит из набора различных блоков: шаблонов страниц, информационного наполнения (контента) и скриптов, хранящихся в виде отдельных файлов. Иными словами, динамическая веб-страница формируется из страницы-шаблона и добавляемых в шаблон динамических данных. Какие данные будут загружены в шаблон, будет зависеть от того, какой запрос сделал клиент. Схема взаимодействия клиента с динамическим сайтом приведена на рис. 1.2.

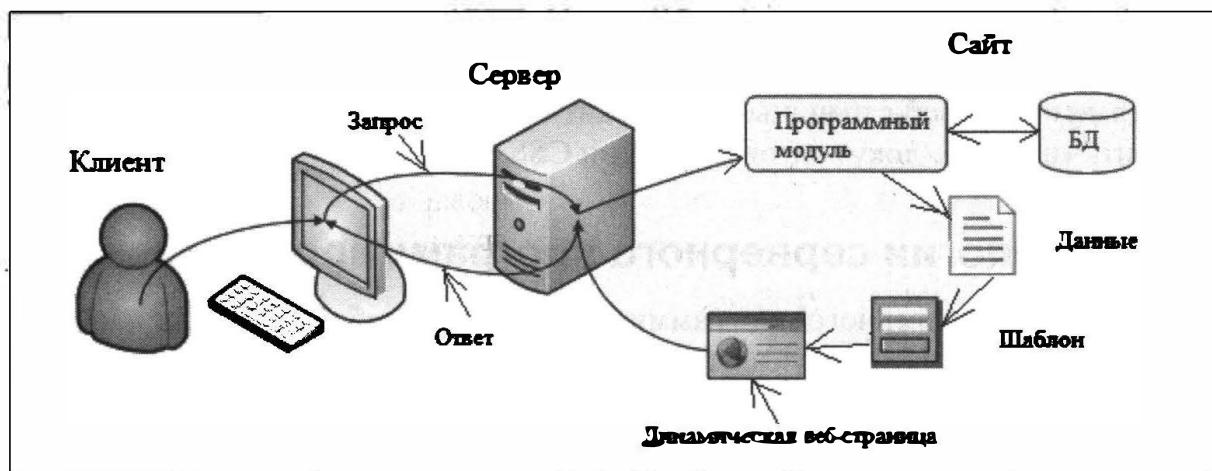


Рис. 1.2. Схема взаимодействия клиента с динамическим сайтом

Динамичность сайта заключается в том, что для изменения страницы достаточно изменить ее информационное наполнение, а сам механизм формирования и вывода страницы остается тем же. Кроме получения различных данных, удаленный клиент может изменить и содержание информационной части сайта. Для этого используются *веб-формы*. Клиент заполняет веб-форму своими данными, и эта информация вносится в БД сайта.

Динамические сайты различаются в зависимости от используемых технологий и процесса получения динамических страниц. Динамические страницы могут быть получены следующими способами:

- генерацией страницы на стороне сервера, осуществляющей серверными скриптами на языках PHP, Perl, ASP.NET, Java, Python и др. При этом информационное наполнение страниц хранится в базах данных;
- генерацией страницы на стороне клиента (JavaScript);
- комбинированной генерацией. Чаще всего на практике встречается именно комбинация первых двух способов.

### 1.1.1. Технологии клиентского программирования

Простейшим средством «оживления» веб-страниц, добавления динамических эффектов и задания реакции на пользовательские действия является скриптовый язык программирования JavaScript. Сценарии (скрипты) JavaScript внедряются непосредственно в веб-страницу или связываются с ней и после загрузки страницы с сервера выполняются браузером на стороне клиента. Все современные браузеры имеют поддержку JavaScript.

JavaScript — это компактный объектно-ориентированный язык для создания клиентских веб-приложений. Этот язык применяется для обработки событий, связанных с вводом и просмотром информации на веб-страницах. JavaScript обычно используется в клиентской части веб-приложений, в которых клиентом выступает браузер, а сервером — удаленный веб-сервер. Обмен информацией в веб-приложениях происходит по сети. Одним из преимуществ такого подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения технологии клиентского программирования представляют собой кросс-платформенные сервисы. Язык сценариев JavaScript позволяет создавать интерактивные веб-страницы и содержит средства управления окнами браузера, элементами HTML-документов и стилями CSS.

### 1.1.2. Технологии серверного программирования

Без использования серверного программирования нельзя обойтись, если необходимо изменять и сохранять какую-либо информацию, хранящуюся на сервере (например, организовать прием и сохранение отправляемых пользователями сообщений). Без серверных скриптов невозможно представить себе гостевые книги, форумы, чаты, опросы (голосования), счетчики посещений и другие программные компоненты, которые активно взаимодействуют с базами данных. Серверное программирование позволяет решать такие задачи, как регистрация пользователей, авторизация пользователей и управление аккаунтом (в почтовых веб-системах, социальных сетях и др.), поиск информации в базе данных, работа интернет-магазина и т. п. Серверные языки программирования открывают перед программистом большие функциональные возможности. Современные сайты зачастую представляют собой чрезвычайно сложные программно-информационные системы, решающие значительное количество разнообразных задач, и теоретически могут дублировать функции большинства бизнес-приложений.

Работа серверных скриптов зависит от платформы, т. е. от того, какие технологии поддерживаются сервером, на котором расположен сайт. Например, основной технологией, поддерживаемой компанией Microsoft, является ASP.NET (Active Server Pages, активные серверные страницы). Другие серверы могут поддерживать языки Perl, PHP, Python.

- **Perl** — высокоуровневый интерпретируемый динамический язык программирования общего назначения. Он является одним из наиболее старых языков, используемых для написания серверных скриптов. По популярности Perl сейчас

уступает более простому в освоении языку PHP. В настоящее время используются бесплатные технологии EmbPerl и mod\_perl, позволяющие обрабатывать HTML-страницы со встроенными скриптами на языке Perl.

- **PHP (Personal Home Page)** — язык сценариев общего назначения, в настоящее время интенсивно применяемый для разработки веб-приложений. PHP-код может внедряться непосредственно в HTML. Это язык с открытым кодом, крайне простой для освоения, но вместе с тем способный удовлетворить запросы профессиональных веб-программистов, т. к. имеет большой набор специализированных встроенных средств.
- **Python** — универсальный язык программирования, применимый в том числе и для разработки веб-приложений. Важно, что в Python приложение постоянно находится в памяти, обрабатывая множество запросов пользователей без «перезагрузки». Таким образом поддерживается правильное предсказуемое состояние приложения. В зависимости от того, какой фреймворк будет использоваться совместно с Python, взаимодействия могут существенно упрощаться. Например, Django, который сам написан на Python, имеет систему шаблонов для написания специальных HTML-файлов, которые могут включать код Python и взаимодействовать с данными из СУБД.

## 1.2. Базовые сведения о HTML

Как было отмечено ранее, язык разметки гипертекста HTML является основой для формирования веб-страниц. С помощью HTML осуществляется логическое форматирование документа, и он может использоваться только для этих целей.

HTML-документы строятся на основе тегов, которые структурируют документ. Обычно теги бывают парными, т. е. состоят из открывающего и закрывающего тега, хотя бывают и исключения. Имена открывающих тегов заключаются в угловые скобки `< ... >`, а закрывающие теги еще содержат знак слеш `</ ... >`.

Весь HTML-документ обрамляется парными тегами `<html>...</html>`. Кроме того, для обеспечения корректного отображения документа современный стандарт требует использования одиночного тега `<!DOCTYPE>`, который может иметь следующую структуру:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

В самом простом случае этот одиночный тег выглядит так:

```
<!DOCTYPE html>
```

Сами HTML-документы состоят из заголовка и тела. Заголовок документа описывается парой тегов `<head>...</head>`, а тело документа обрамляется парными тегами `<body>...</body>`. Таким образом, каркас HTML-документа будет иметь следующую структуру:

```
<!DOCTYPE HTML>
<html>
```

```

<head>
    СОДЕРЖАНИЕ ЗАГОЛОВКА ДОКУМЕНТА
</head>
<body>
    СОДЕРЖАНИЕ ТЕЛА ДОКУМЕНТА
</body>
</html>

```

Заголовок может включать в себя несколько специализированных тегов, основными из которых являются `<title>...</title>` и `<meta>...</meta>`.

Тег `<title>` содержит информацию, которая будет выводиться в заголовочной части окна браузера пользователя. Например, если в этом теге имеется следующий текст:

`<title>Мир книг</title>`

то он следующим образом отобразится в окне браузера (рис. 1.3).

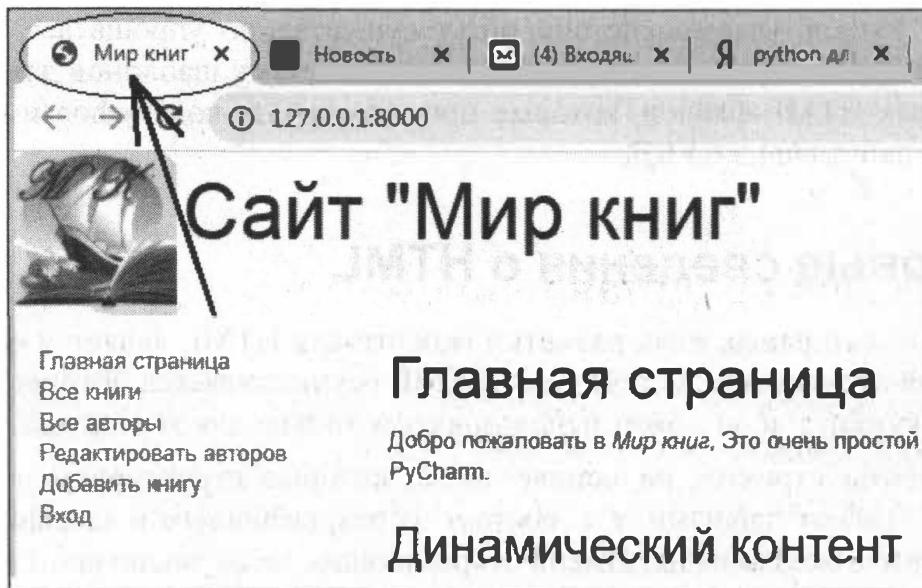


Рис. 1.3. Вывод содержимого тега `<title>` в браузере пользователя

Тег `<meta>` содержит специальную информацию:

- тип кодировки:

`<meta charset="utf-8" />`

- или список ключевых слов:

`<meta name="keywords" content="СУБД, БД, Базы данных">`

В первом случае этот тег обеспечивает поддержку необходимой кодировки, а во втором — позволяет поисковым машинам производить корректное индексирование страниц сайта по ключевым словам. Следует заметить, что современные поисковые системы могут игнорировать ключевые слова, но это не отменяет возможность их использования этого атрибута.

В рассмотренных тегах фрагменты `name="keywords"` и `content="список ключевых слов"` представляют собой *атрибуты* (параметры) тегов, которые конкретизируют их.

Например, атрибуты могут указывать, что текст, заключенный в том или ином теге, при отображении должен выравниваться по центру. Атрибуты записываются сразу после имени тега, причем значения атрибутов заключаются в кавычки. Атрибутов у тега может быть несколько, но могут они и вовсе отсутствовать.

Приведем пример простейшей веб-страницы:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Веб-страница</title>
    <meta charset="windows-1251">
    <meta name="keywords" content="веб-страница первая">
  </head>
  <body>
    Моя первая веб-страница
  </body>
</html>
```

Текст страницы может быть набран в любом редакторе и сохранен в файле с расширением `htm` или `html`. Такой файл можно открыть при помощи какого-либо браузера.

## 1.2.1. Теги для представления текста на HTML-страницах

Текст, содержащийся в теле документа, обычно обрамляется специальными тегами. Наиболее часто используются следующие виды тегов.

□ Тег `<hi>` — для вывода заголовков, где *i* имеет значения от 1 до 6. Например:

```
<h1> ... </h1>
<h2> ... </h2>
<h3> ... </h3>
<h4> ... </h4>
<h5> ... </h5>
<h6> ... </h6>
```

Заголовки отображаются жирными шрифтами. При этом у текста заголовка в теге `<h1>` наибольший размер шрифта, а у тега `<h6>` — наименьший. К этому тегу может быть добавлен параметр `align`, который определяет правило выравнивания заголовка относительно одной из сторон документа. Этот параметр может принимать следующие значения:

- `left` — выравнивание по левому краю;
- `center` — по центру;
- `right` — по правому краю;
- `justify` — выравнивание по обоим краям.

- Тег <P> — для вывода параграфов (абзацев). К этому тегу также может быть добавлен параметр align. Каждый новый тег <P> с этим параметром устанавливает выравнивание параграфа относительно одной из сторон документа.
- Тег <BR> — для перевода строки.
- Тег <HR> — для вывода горизонтальной линии. Параметр align этого тега определяет выравнивание линии относительно одной из сторон документа. Этот тег имеет еще несколько параметров:
  - size — высота линии;
  - width — ширина линии;
  - color — цвет линии;
  - noshade — отключает отбрасывание тени.
- Представленные далее теги меняют следующие параметры текста:
  - <B> — жирный текст;
  - <I> — наклонный текст;
  - <U> — подчеркнутый текст;
  - <TT> — моноширинный текст.
- Тег <PRE> — выводит заранее отформатированный текст. Используется он для того, чтобы текст с пробелами, символами перехода на новые строки и символами табуляции корректно отображался браузером. В секциях <PRE> могут присутствовать гипертекстовые ссылки, однако применять другие HTML-теги нельзя.
- Тег <FONT> — задает некоторые свойства шрифта. Эти свойства определяются следующими параметрами:
  - size — размер шрифта от 1 до 7;
  - face — начертание шрифта;
  - color — цвет шрифта.

Приведем пример использования некоторых тегов:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="windows-1251">
    <title>Моя первая веб-страница</title>
  </head>
  <body>
    <p>Моя первая веб-страница</p>
    <h1>Заголовок h1</h1>
    <h2>Заголовок h2</h2>
    <h3>Заголовок h3</h3>
    <h4>Заголовок h4</h4>
```

```
<h5>Заголовок h5</h5>
<h6>Заголовок h6</h6>
</body>
</html>
```

Открывающие теги могут содержать дополнительную информацию в виде параметров (атрибутов), которые существенно расширяют возможности представления информации. Параметры в открывающем теге записываются после названия тега в виде `параметр="значение"` и разделяются пробелами. Порядок следования параметров в теге не произвольный. Если параметр отсутствует, его значение принимается по умолчанию согласно спецификации.

Вот основные параметры тега `<BODY>`:

- `text` — устанавливает цвет текста документа, используя значение цвета в виде `RRGGBB` (например, `text="000000"` — черный цвет);
- `link` — устанавливает цвет гиперссылок, используя значение цвета в виде `RRGGBB` (например, `text="FF0000"` — красный цвет);
- `vlink` — устанавливает цвет гиперссылок, на которых пользователь уже побывал (например, `text="00FF00"` — зеленый цвет);
- `alink` — устанавливает цвет гиперссылок при нажатии (например, `text="0000FF"` — синий цвет);
- `bgcolor` — устанавливает цвет фона документа, используя значение цвета в виде `RRGGBB` (например, `text="FFFFFF"` — белый цвет);
- `background` — устанавливает изображение фона документа (например, `background="bg.gif"`);
- `topmargin` — устанавливает величину верхнего поля документа (например, `topmargin="0"`);
- `leftmargin` — устанавливает величину левого поля документа (например, `leftmargin="10"`).

#### **ПРИМЕЧАНИЕ**

В HTML-коде цвет определяется 6-значным кодом `RRGGBB` (красный, красный, зеленый, зеленый, синий, синий), а в JavaScript или в CSS — 6-значным кодом `RRGGBB` или английским названием цвета. Со значениями кодов, обозначающих цвета, можно ознакомиться по следующей ссылке: <https://www.rapidtables.com/web/color/html-color-codes.html>.

Вот пример использования в теге `<BODY>` указанных параметров:

```
<BODY text="black" bgcolor="white">
```

Несмотря на то что представленные здесь параметры форматирования еще весьма широко используются, нужно воздерживаться от их применения, т. к. для этих целей предназначены средства каскадных таблиц стилей, о чем речь пойдет в последующих разделах.

## 1.2.2. Списки

Современным стандартом HTML предусмотрены три основных вида списков:

- маркированные списки (unordered list);
- нумерованные списки (ordered list);
- списки определений (definition list).
- Маркированные списки* названы так потому, что перед каждым пунктом таких списков устанавливается тот или иной *маркер*. Иногда, прибегая к дословному переводу, их также называют *неупорядоченными списками* (unordered list). Маркированные списки задаются при помощи тегов <ul>...<ul>. Для задания элементов списка (item list) используются теги <il>...</il>. Например:

```
<ul>
<li>Пункт 1</li>
<li>Пункт 2</li>
<li>Пункт 3</li>
</ul>
```

Помимо элементов списка внутри тегов <ul>...<ul> можно размещать и другие теги — например, теги заголовков:

```
<ul>
<h3>Маркированный список</h3>
<li>Пункт 1</li>
<li>Пункт 2</li>
<li>Пункт 3</li>
</ul>
```

Тип маркера может задаваться с помощью атрибута `type`. Три его возможных значения: `circle` (незакрашенный кружок), `disk` (закрашенный кружок) и `square` (закрашенный квадрат). По умолчанию используется тип `disk`. Следует отметить, что различные модели браузеров могут по-разному отображать маркеры. Далее приводится пример использования маркера типа `circle`:

```
<ul type="circle">
<li>Пункт 1</li>
<li>Пункт 2</li>
<li>Пункт 3</li>
</ul>
```

В *нумерованных списках* (ordered list), которые иногда называют *упорядоченными*, каждому пункту присваивается номер. Создаются такие списки при помощи тегов <ol>...<ol>. Для элементов нумерованных списков, как и в случае маркированных списков, также используются теги <il>...</il>. В таких списках доступны пять типов маркеров, которые, так же как и в маркированных списках, определяются при помощи атрибута `type`, который может принимать следующие значения:

- i — строчные римские цифры;
- I — прописные римские цифры;
- 1 — арабские цифры;
- i — строчные арабские цифры;
- I — прописные арабские цифры;

- I — прописные римские цифры;
- a — строчные латинские буквы;
- A — прописные латинские буквы.

Далее приведены примеры нумерованных списков:

```
<ol>
<h3>Нумерованный список</h3>
<li>Пункт 1</li>
<li>Пункт 2</li>
<li>Пункт 3</li>
</ol>
<ol type="I">
<li>Пункт 1</li>
<li>Пункт 2</li>
<li>Пункт 3</li>
</ol>
```

*Списки определений* (*definition list*) применяются для того, чтобы организовать текст по примеру словарных статей. Они задаются с помощью тегов `<dl>...</dl>`, а определяемый термин или понятие (*definition term*) помещается в теги `<dt>...</dt>`. Определение понятия (*definition description*) заключается в теги `<dd>...</dd>`. В тексте, содержащемся внутри тегов `<dt>...</dt>`, не могут использоваться теги уровня блока, такие как `<p>` или `<div>`. Как и в предыдущих случаях, внутри списков определений могут использоваться теги заголовков и прочие теги:

```
<dl>
<h3>Список определений</h3>
<dt>Понятие 1</dt>
<dd>Определение понятия 1</dd>
<dt>Понятие 2</dt>
<dd>Определение понятия 2</dd>
</dl>
```

Как маркированные, так и нумерованные списки можно вкладывать друг в друга, причем допускается произвольное вложение различных типов списков. При вложении друг в друга маркированных и нумерованных списков следует быть внимательным. Далее приведен пример вложенных списков:

```
<ul>
<h3>Пример вложенных списков</h3>
<li>Пункт 1</li>
<ol>
<li>Пункт 1.1</li>
<li>Пункт 1.2</li>
</ol>
<li>Пункт 2</li>
<ol type="i">
<li>Пункт 2.1</li>
```

```
<li>Пункт 2.2</li>
<li>Пункт 2.3</li>
</ol>
<li>Пункт 3</li>
<ol type="I">
<li>Пункт 3.1</li>
</ol>
</ul>
```

### 1.2.3. Таблицы

Таблицы являются одной из основных структур, используемых для структурирования информации в HTML-документах. Кроме того, таблицы часто задействуются для организации структуры страницы, и хотя сейчас такое использование таблиц признано устаревшим и не рекомендуемым, оно до сих пор применяется многими веб-дизайнерами. Таблица создается при помощи тега `<TABLE>`. Этот тег может иметь следующие параметры:

- align — задает выравнивание таблицы (`align=left/center/right`);
- border — задает толщину линий таблицы (в пикселях);
- bgcolor — устанавливает цвет фона документа;
- background — устанавливает изображение фона документа;
- cellpadding — задает ширину промежутков между содержимым ячейки и ее границами (в пикселях), т. е. задает поля внутри ячейки;
- cellspacing — задает ширину промежутков между ячейками (в пикселях);
- width — задает ширину таблицы в пикселях, процентах или частях.

Для создания заголовка таблицы служит тег `<CAPTION>`. По умолчанию заголовки центрируются и размещаются либо над (`<CAPTION align="top">`), либо под таблицей (`<CAPTION align="bottom">`). Заголовок может состоять из любого текста и изображений. Текст будет разбит на строки, соответствующие ширине таблицы. Каждая новая строка таблицы создается тегом `<TR>` (Table Row), который может иметь дополнительные параметры:

- align — задает горизонтальное выравнивание информации в ячейках (`align=left/center/right/justify`);
- valign — задает вертикальное выравнивание информации в ячейках (`valign=top/middle/bottom`).

Внутри строки таблицы размещаются ячейки с данными, создаваемые тегами `<TD>`. Число тегов `<TD>` в строке определяет число ячеек (столбцов) таблицы. Тег `<TD>` может иметь следующие дополнительные параметры:

- align — задает горизонтальное выравнивание информации в ячейке (`align=left/center/right/justify`);

- valign — задает вертикальное выравнивание информации в ячейке (`valign=top/middle/bottom`);
- colspan — объединение столбцов в строке (например, `colspan=2`);
- rowspan — объединение строк в столбце (например, `rowspan=3`);
- width — задает ширину ячейки в пикселях или процентах;
- nowrap — запрещение перехода текста в ячейке на новую строку.

Для задания заголовков столбцов и строк таблицы служит тег заголовка `<th>` (Table Header). Этот тег аналогичен тегу `<td>` с той лишь разницей, что текст в теге `<th>` по умолчанию выделяется жирным шрифтом и располагается по центру ячейки.

Далее приведен пример простой таблицы:

```
<table border="3" cellpadding="7" cellspacing="3" height="80" width="50%">
<caption>Пример простой таблицы</caption>
<tr align="center">
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
<tr align="center">
<td align="center">2.1</td>
<td align="right">2.2</td>
<td>2.3</td>
</tr>
</table>
```

Благодаря наличию большого числа параметров, а также возможности создания границ нулевой толщины таблица может выступать в роли невидимой модульной сетки, относительно которой добавляется текст, изображения и другие элементы, организуя информацию на странице.

□ Возможности табличной верстки:

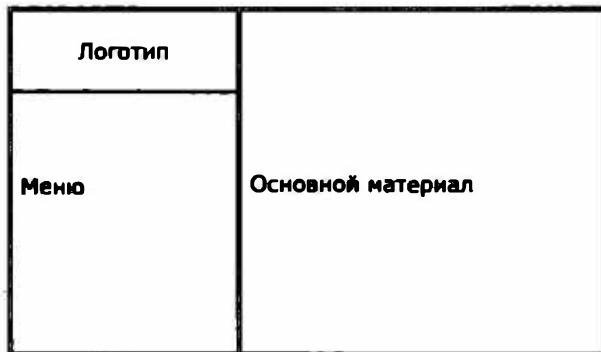
- создание колонок;
- создание «резинового» макета;
- «склейка» изображений;
- включение фоновых рисунков;
- выравнивание элементов.

□ Недостатки табличной верстки:

- долгая загрузка;
- громоздкий код;
- плохая индексация поисковиками;
- нет разделения содержимого и оформления;
- несоответствие стандартам.

Рассмотрим некоторые типовые модульные сетки.

- **Двухколонная модульная сетка** часто применяется на небольших информационных сайтах. Как правило, в первой колонке располагается логотип и меню сайта, а во второй — основной материал. Пример такой модульной сетки приведен на рис. 1.4.



**Рис. 1.4. Двухколонная модульная сетка**

Далее приведен HTML-код, реализующий эту структуру:

```

<TABLE width="760" cellpadding="10" cellspacing="10" border="1">
<TR>
<TD align="center">Логотип</TD>
<TD rowspan="2">Основной материал</TD>
</TR>
<TR>
<TD>Меню</TD>
</TR>
</TABLE>
  
```

- **Трехколонная модульная сетка** «резинового» макета применяется на крупных порталах с множеством информационных сервисов. Как правило, в первой колонке располагается меню сайта и дополнительная информация, во второй — основной материал, в третьей — дополнительные функции. Часто в модульную сетку сайта добавляют блоки «Заголовок сайта» и «Окончание сайта» (этот блок еще называют «подвалом»). Пример такой модульной сетки приведен на рис. 1.5.

Далее приведен HTML код, реализующий эту структуру:

```

<TABLE width="100%" cellpadding="10" cellspacing="10" border="0">
<TR align="center">
<TD colspan="3">Заголовок сайта</TD>
</TR>
<TR>
<TD width="200px">Меню</TD>
<TD width="*">Основной материал</TD>
<TD width="200px">Дополнительные функции</TD>
</TR>
<TR align="center">
  
```

```
<TD colspan="3">Окончание сайта</TD>
</TR>
</TABLE>
```



Рис. 1.5. Трехколонная модульная сетка

## 1.2.4. Гиперссылки

Для связи различных документов HTML предусматривает использование *ссылок*. Сам термин HTML (Hyper Text Markup Language) подразумевает их широкое использование. Для реализации ссылок в HTML служит тег `<a>...</a>`, который, как и большинство HTML-тегов, является контейнерным. Основной атрибут этого тега — `href`, собственно и содержащий адрес ресурса, на который указывает ссылка. Внутри тега `<a>...</a>` помещается текст ссылки.

В ссылках может использоваться как относительная, так и абсолютная адресация. В случае абсолютной адресации атрибуту `href` присваивается абсолютный URL-адрес ресурса:

```
<a href="http://server.com/doc3.htm">Ссылка на документ с абсолютным адресом  
http://server.com/doc3.htm</a> .
```

В случае относительной адресации указывается путь к документу относительно текущей страницы:

```
<a href="doc1.htm">Ссылка на документ с относительным адресом doc1.htm</a> .
```

Если в заголовочной части документа использован тег `<base>`, то отсчет будет вестись от адреса, заключенного в этом теге.

Помимо веб-страниц, допускается ссылаться и на другие интернет-ресурсы: e-mail, ftp, Gopher, WAIS, Telnet, newsgroup. Далее приведен пример ссылки на адрес электронной почты:

```
<a href="mailto:sss@mail.ru">Ссылка на адрес электронной почты sss@mail.ru</a>
```

В роли ссылок могут выступать и рисунки. Для этого сначала надо вставить рисунок с помощью тега `<img>`. У атрибута `src` этого тега устанавливается значение, соответствующее имени файла рисунка:

```

```

Далее, рисунок «обертывается» в тег ссылки:

```
<a href="link2.htm"></a>
```

## 1.3. Каскадные таблицы стилей (CSS)

Если HTML используется для логического форматирования документа, то для управления его отображением на экране монитора или выводом на принтер применяются каскадные таблицы стилей (CSS). Технология CSS реализует концепцию «Документ — представление» и позволяет отделять оформление HTML-документа от его структуры. Кроме того, CSS существенно расширяет возможности представления (оформления) документов, внося множество новых возможностей.

Для того чтобы таблица стилей влияла на вид HTML-документа, она должна быть подключена к нему. Подключение каскадных таблиц стилей может быть выполнено с использованием внешних, внутренних или локальных таблиц стилей:

- **внешние таблицы стилей, оформленные в виде отдельных файлов, подключаются к HTML-документу при помощи тега в заголовке документа.** Например:

```
<LINK rel="stylesheet" href="style.css" type="text/css"> .
```

- **внутренние таблицы стилей в составе HTML-документа помещаются в заголовок страницы при помощи тега <STYLE>.** Например:

```
<HEAD>
<STYLE>
  P {color: #FF0000}
</STYLE>
</HEAD>
```

- **локальные таблицы стилей объявляются непосредственно внутри тега, к которому они относятся, при помощи параметра style.** Например:

```
<P style="color: #FF0000">Каскадные таблицы стилей</p> .
```

Таблицы стилей записываются в виде последовательности тегов, для каждого из которых в фигурных скобках указывается его свойства по схеме параметр: свойство. Параметры внутри фигурных скобок разделяются точкой с запятой. Например:

```
P {color: #CCCCCC; font-size: 10px}
H1{color: #FF0000} .
```

С помощью каскадных таблиц стилей можно менять свойства следующих элементов: фона и цвета, шрифта, текста, полей и отступов, границ.

- Для задания свойств фона и цвета служат следующие параметры:

- background-color — цвет заднего плана;
- background-image — изображение заднего плана;
- background-repeat — дублирование изображения заднего плана (значения: repeat/repeat-x/repeat-y/no-repeat);

- `background-attachment` — фиксация изображения заднего плана (значения: `scroll/fixed`);
- `background-position` — начальное положение изображения заднего плана.

□ Для задания свойств шрифта служат следующие параметры:

- `font-style` — стиль шрифта (значения: `normal / italic`);
- `font-weight` — начертание шрифта (значения: `normal / bold`);
- `font-size` — размер шрифта;
- `font-family` — список имен шрифтов в порядке их приоритета.

□ Для задания свойств текста служат следующие параметры:

- `word-spacing` — установка промежутка между словами;
- `letter-spacing` — установка высоты строки;
- `text-indent` — установка абзацного отступа;
- `text-align` — выравнивание текста;
- `vertical-align` — установка вертикального выравнивания текста;
- `text-decoration` — преобразование текста (значение: `none/underline/overline/line-through/blink`).

□ Для задания свойств полей и отступов служат следующие параметры:

- `margin-top` — установка верхнего поля;
- `margin-right` — установка правого поля;
- `margin-bottom` — установка нижнего поля;
- `margin-left` — установка левого поля;
- `margin` — установка всех полей (например: `margin: 10px 10px 10px 0px`);
- `padding-top` — установка верхнего отступа;
- `padding-right` — установка правого отступа;
- `padding-bottom` — установка нижнего отступа;
- `padding-left` — установка левого отступа;
- `padding` — установка всех отступов (например: `padding: 10px 10px 10px 0px`).

□ Для задания свойств границ служат следующие параметры:

- `border-width` — установка ширины границы;
- `border-color` — установка цвета границы;
- `border-style` — установка стиля границы (значения: `none/dotted/dashed/solid/double`).

Для придания управлению элементами HTML большей гибкости используются **классы**, которые позволяют задавать различные стили для одного и того же тега.

В таблицах стилей имя класса записывается после тега и отделяется от него точкой. Например:

```
P.green {color: #00FF00}  
P.blue {color: #0000FF}
```

В HTML-коде соответствие тега определенному классу указывается при помощи параметра `class`. Например:

```
<p class="green">Зеленый текст</p>  
<p class="blue">Синий текст</p>
```

## 1.4. Возможности использования JavaScript

Чаще всего JavaScript используется как язык, встраиваемый в веб-страницы для получения программного доступа к их элементам. Сценарии JavaScript являются основой клиентской части веб-приложений. Они загружаются с сервера вместе с веб-страницами и выполняются браузером на компьютере пользователя. Обработка сценариев JavaScript осуществляется встроенным в браузер интерпретатором. Что может делать JavaScript?

- В первую очередь JavaScript умеет отслеживать действия пользователя (например, реагировать на щелчок мыши или нажатие клавиши на клавиатуре, на перемещение курсора, на скроллинг).
- Менять стили, добавлять (удалять) HTML-теги, скрывать (показывать) элементы.
- Посыпать запросы на сервер, а также загружать данные без перезагрузки страницы (эта технология называется AJAX).

JavaScript — это объектно-ориентированный язык. Он поддерживает несколько встроенных объектов, а также позволяет создавать или удалять свои собственные (пользовательские) объекты. Объекты JavaScript могут наследовать свойства непосредственно друг от друга, образуя цепочку объект — прототип.

Сценарии JavaScript бывают встроенные, т. е. их содержимое является частью документа, и внешние, хранящиеся в отдельном файле с расширением `js`. Сценарии можно внедрить в HTML-документ следующими способами:

- в виде гиперссылки;
- в виде обработчика события;
- внутрь элемента `<script>`.

Рассмотрим эти способы подробнее.

- Если подключать JavaScript в виде гиперссылки, то для этого код скрипта нужно разместить в отдельном файле, а ссылку на файл включить либо в заголовок:

```
<head>  
  <script src="script.js"></script>  
</head>
```

либо в тело страницы:

```
<body>
  <script src="script.js"></script>
</body>
```

Этот способ обычно применяется для сценариев большого размера или сценариев, многократно используемых на разных веб-страницах.

- Можно подключать JavaScript к обработчику события. Дело в том, что каждый HTML-элемент может иметь JavaScript-события, которые срабатывают в определенный момент. Нужно добавить необходимое событие в HTML-элемент как атрибут, а в качестве значения этого атрибута указать требуемую функцию. Функция, вызываемая в ответ на срабатывание события, и станет обработчиком события. В результате срабатывания события исполнится связанный с ним код. Этот способ применяется в основном для коротких сценариев — например, можно установить по нажатию на кнопку смену цвета фона :

```
<script>
var colorArray = ["#5A9C6E", "#A8BF5A", "#FAC46E", "#FAD5BB",
                  "#F2FEFF"]; // создаем массив с цветами фона
var i = 0;

function changeColor(){
  document.body.style.background = colorArray[i];
  i++;
  if( i > colorArray.length - 1){
    i = 0;
  }
}
</script>
<button onclick="changeColor()">Сменить цвет фона</button>
```

- Код внутри элемента `<script>` может вставляться в любое место документа. Код, который выполняется сразу после прочтения браузером или содержит описание функции, которая выполняется в момент ее вызова, располагается внутри тега. Описание функции можно располагать в любом месте — главное, чтобы к моменту ее вызова код функции уже был загружен.

Обычно код JavaScript размещается в заголовке документа (в элементе `<head>`) или после открывающего тега `<body>`. Если скрипт используется после загрузки страницы (например, код счетчика), то его лучше разместить в конце документа:

```
<footer>
  <script>
    document.write("Введите свое имя");
  </script>
</footer>
</body>
```

В примерах этой книги мы не будем задействовать возможности JavaScript, однако полезно знать о наличии этого языка и его возможностях при создании веб-страниц.

## 1.5. Интерпретатор Python

Язык программирования Python является весьма мощным инструментальным средством для разработки различных систем. Однако наибольшую ценность представляет даже не столько сам этот язык программирования, сколько набор подключающихся библиотек, на уровне которых уже реализованы все необходимые процедуры и функции. Разработчику достаточно написать несколько десятков строк программного кода, чтобы подключить требуемые библиотеки, создать набор необходимых объектов, передать им исходные данные и отобразить итоговые результаты.

Для установки интерпретатора Python на компьютер прежде всего надо загрузить его дистрибутив. Скачать дистрибутив Python можно с официального сайта, перейдя по ссылке: <https://www.python.org/downloads/> (рис. 1.6).

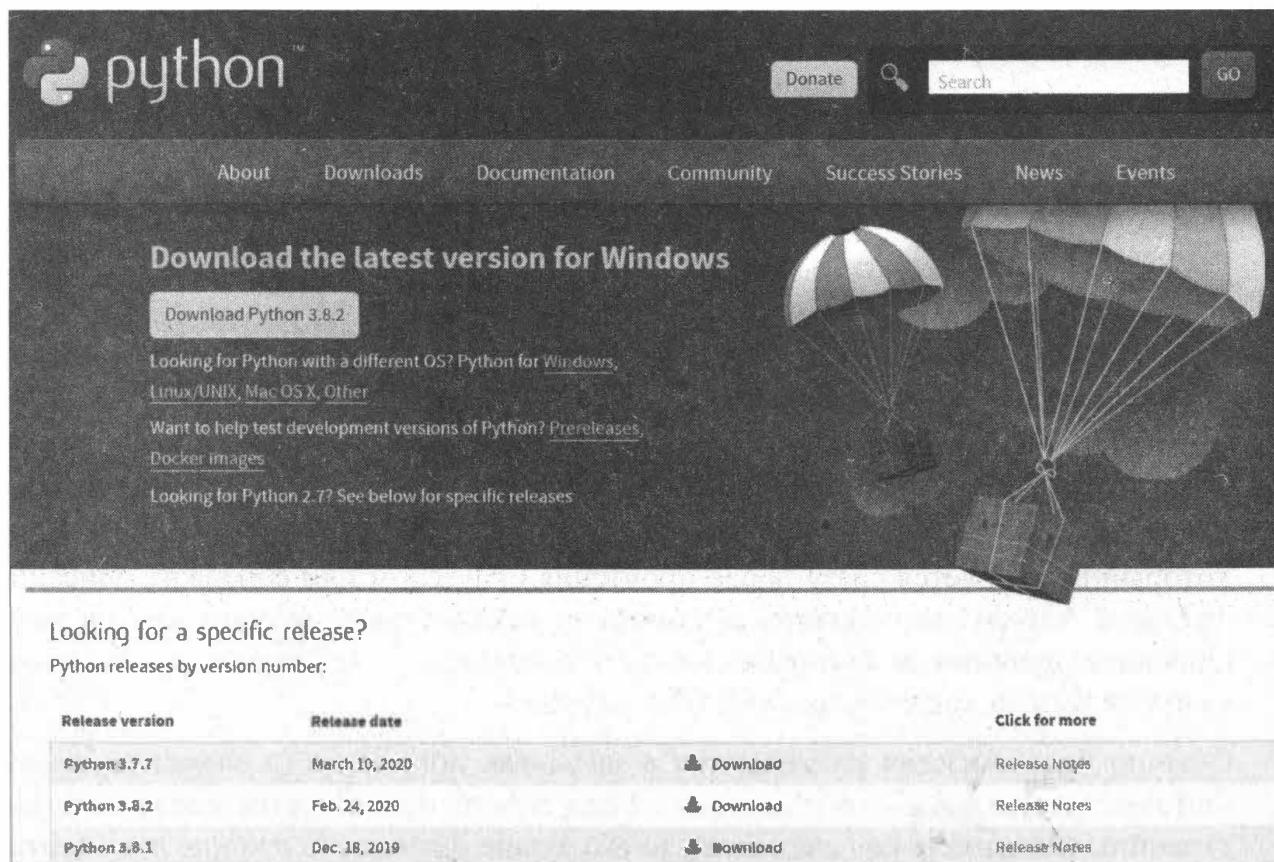


Рис. 1.6. Сайт для скачивания дистрибутива языка программирования Python

### 1.5.1. Установка Python в Windows

Для операционной системы Windows дистрибутив Python распространяется либо в виде исполняемого файла (с расширением `exe`), либо в виде архивного файла

(с расширением `zip`). На момент подготовки этой книги была доступна версия Python 3.8.3.

Порядок установки Python в Windows следующий:

1. Запустите скачанный установочный файл.
2. Выберите способ установки (рис. 1.7).

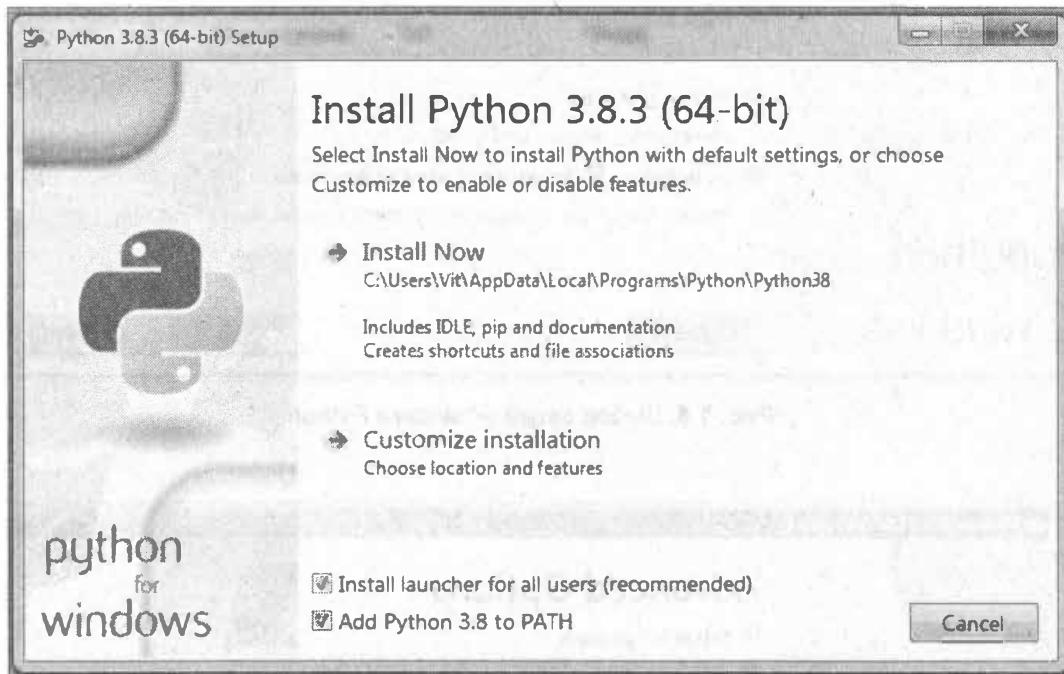


Рис. 1.7. Выбор способа установки Python

В открывшемся окне предлагаются два варианта: **Install Now** и **Customize installation**:

- при выборе **Install Now** Python установится в папку по указанному в окне пути. Помимо самого интерпретатора будут инсталлированы IDLE (интегрированная среда разработки), `pip` (пакетный менеджер) и документация, а также созданы соответствующие ярлыки и установлены связи (ассоциации) файлов, имеющих расширение `py`, с интерпретатором Python;
- **Customize installation** — это вариант настраиваемой установки. Опция **Add Python 3.8 to PATH** нужна для того, чтобы появилась возможность запускать интерпретатор без указания полного пути до исполняемого файла при работе в командной строке.

3. Отметьте необходимые опции установки, как показано на рис. 1.8 (доступно при выборе варианта **Customize installation**).

На этом шаге нам предлагается отметить дополнения, устанавливаемые вместе с интерпретатором Python. Рекомендуется выбрать как минимум следующие опции:

- **Documentation** — установка документации;
- **pip** — установка пакетного менеджера `pip`;

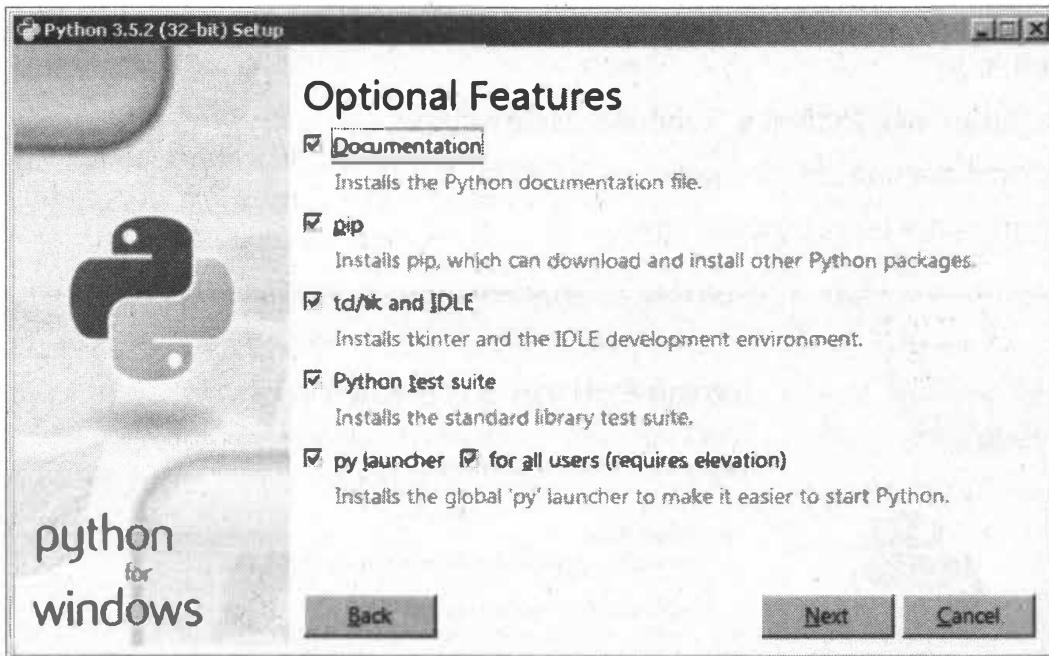


Рис. 1.8. Выбор опций установки Python

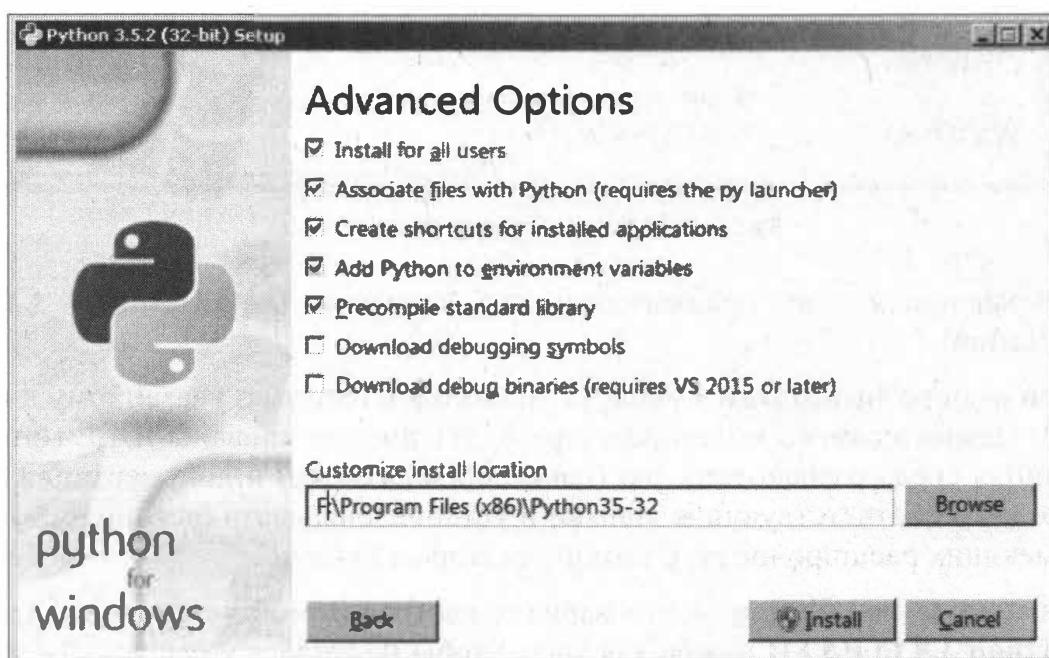


Рис. 1.9. Выбор места установки Python

- **tcl/tk and IDLE** — установка интегрированной среды разработки (IDLE) и библиотеки для построения графического интерфейса (tkinter).
4. На следующем шаге в разделе **Advanced Options** (Дополнительные опции) выберите место установки, как показано на рис. 1.9 (доступно при выборе варианта **Customize installation**).
- Помимо указания пути, этот раздел позволяет внести дополнительные изменения в процесс установки с помощью опций:

- **Install for all users** — установить для всех пользователей. Если не выбрать эту опцию, то будет предложен вариант инсталляции в папку пользователя, устанавливающего интерпретатор;
- **Associate files with Python** — связать файлы, имеющие расширение `py`, с Python. При выборе этой опции будут внесены изменения в Windows, позволяющие Python запускать скрипты по двойному щелчку мыши;
- **Create shortcuts for installed applications** — создать ярлыки для запуска приложений;
- **Add Python to environment variables** — добавить пути до интерпретатора Python в переменную `PATH`;
- **Precompile standard library** — провести перекомпиляцию стандартной библиотеки.

Последние два пункта связаны с загрузкой компонентов для отладки, их мы устанавливать не будем.

5. После успешной установки Python вас ждет следующее сообщение (рис. 1.10).



Рис. 1.10. Финальное сообщение после установки Python

## 1.5.2. Установка Python в Linux

Чаще всего интерпретатор Python уже входит в состав дистрибутива Linux. Это можно проверить, набрав в окне терминала команду:

> python

или

> python3

В первом случае вы запустите Python 2, во втором — Python 3. В будущем, скорее всего, во все дистрибутивы Linux, включающие Python, будет входить только третья его версия. Если у вас при попытке запустить Python выдается сообщение о том, что он не установлен или установлен, но не тот, что вы хотите, то у вас есть возможность взять его из репозитория.

Для установки Python из репозитория Ubuntu воспользуйтесь командой:

```
> sudo apt-get install python3
```

### 1.5.3. Проверка интерпретатора Python

Для начала протестируем интерпретатор в командном режиме. Если вы работаете в Windows, то нажмите комбинацию клавиш <Win>+<R> и в открывшемся окне введите: python. В Linux откройте окно терминала и в нем введите: python3 (или python).

В результате Python запустится в командном режиме. Выглядеть это будет примерно так, как показано на рис. 1.11 (иллюстрация приведена для Windows, в Linux результат будет аналогичным).

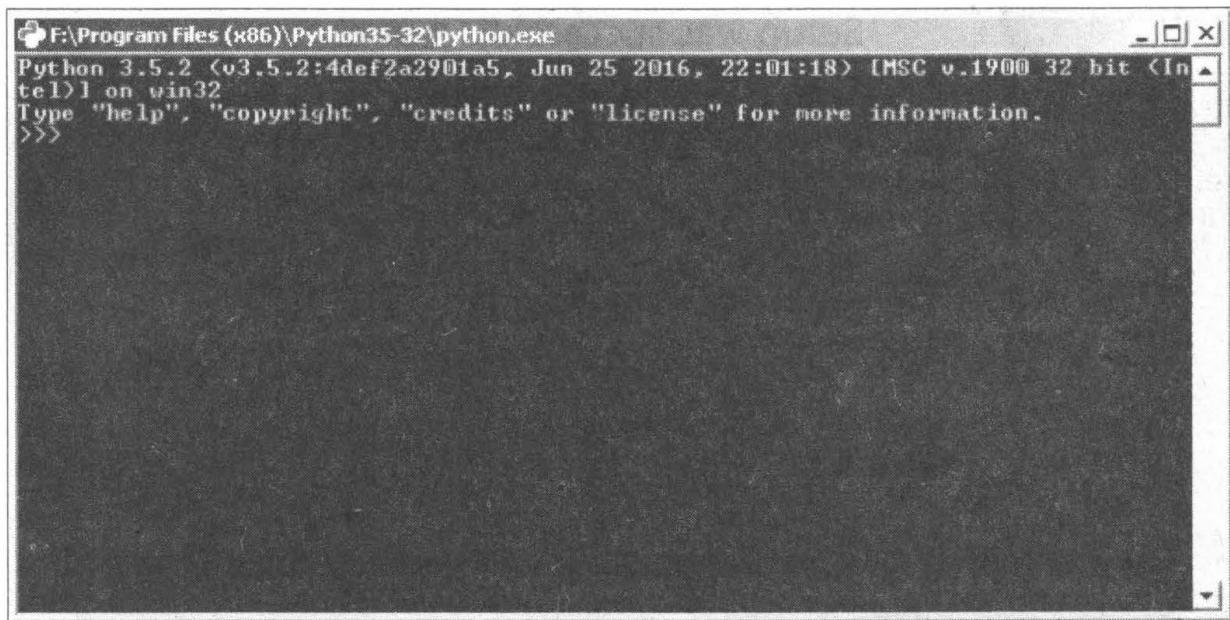


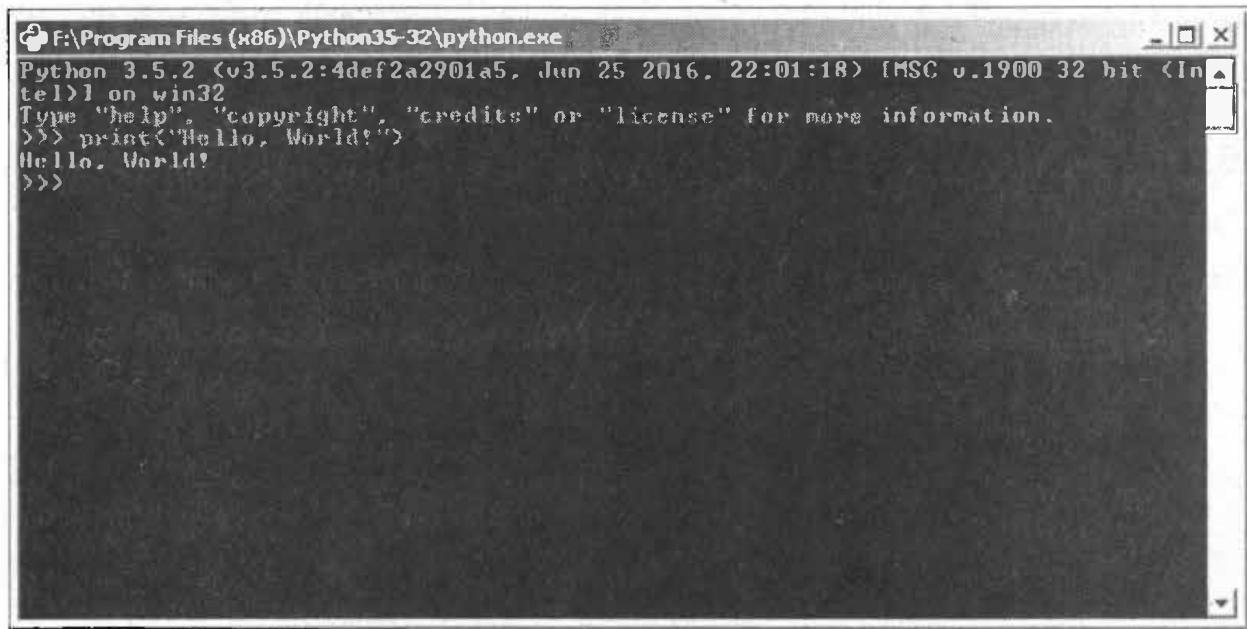
Рис. 1.11. Результат запуска интерпретатора Python в окне терминала

В этом окне введите программный код следующего содержания:

```
print("Hello, World!")
```

В результате вы увидите следующий ответ (рис. 1.12).

Получение такого результата означает, что установка интерпретатора Python прошла без ошибок.



```
F:\Program Files (x86)\Python35-32\python.exe
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>>
```

Рис. 1.12. Результат работы программы на Python в окне терминала

## 1.6. Интерактивная среда разработки программного кода PyCharm

В процессе разработки программных модулей удобнее работать в интерактивной среде разработки (IDE), а не в текстовом редакторе. Для Python одним из лучших вариантов считается IDE PyCharm от компании JetBrains. Для скачивания его дистрибутива перейдите по ссылке: <https://www.jetbrains.com/pycharm/download/> (рис. 1.13).



Рис. 1.13. Главное окно сайта для скачивания дистрибутива PyCharm

Эта среда разработки доступна для Windows, Linux и macOS. Существуют два вида лицензии PyCharm: **Professional** и **Community**. Мы будем использовать версию **Community**, поскольку она бесплатная и ее функционала более чем достаточно

для наших задач. На момент подготовки этой книги была доступна версия PyCharm 2020.1.2.

## 1.6.1. Установка PyCharm в Windows

1. Запустите на выполнение скачанный дистрибутив PyCharm (рис. 1.14).
2. Выберите путь установки программы (рис. 1.15).

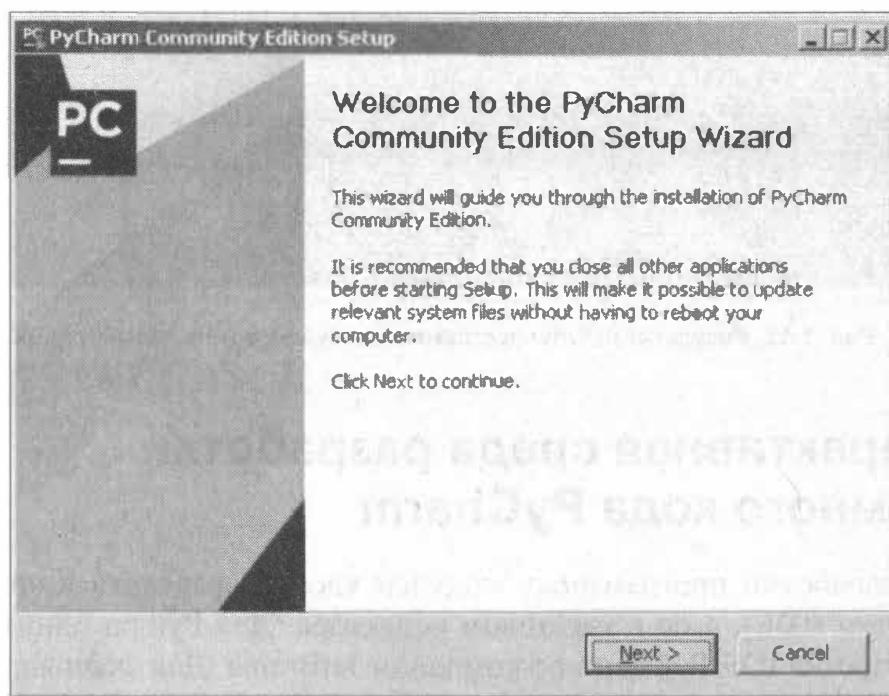


Рис. 1.14. Начальная заставка при инсталляции PyCharm

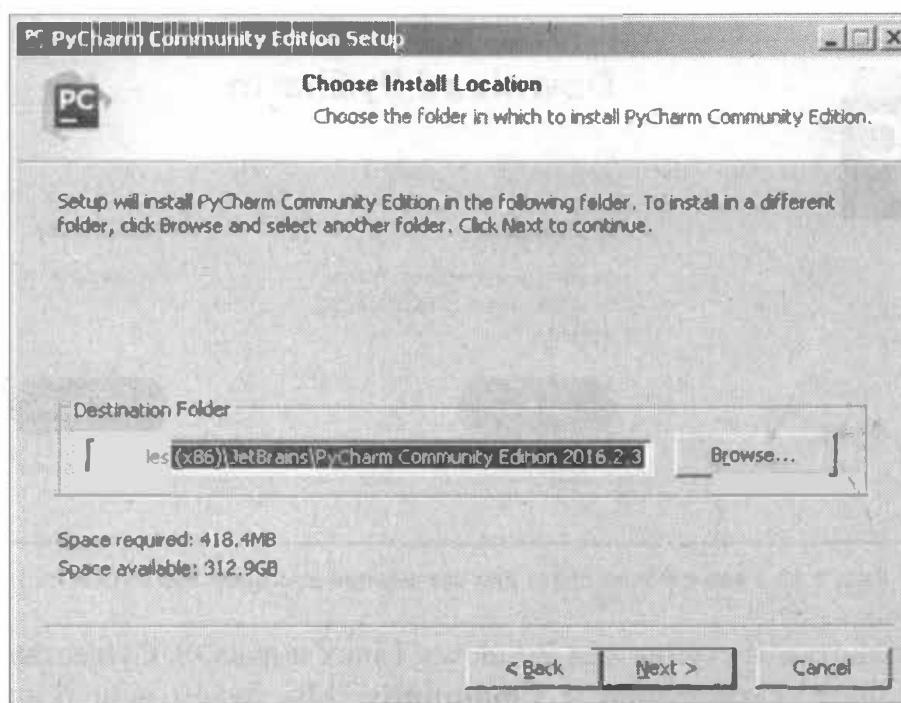


Рис. 1.15. Выбор пути установки PyCharm

3. Укажите ярлыки, которые нужно создать на рабочем столе (запуск 32- или 64-разрядной версии PyCharm), и отметьте флашком опцию **.py** в области **Create associations**, если требуется ассоциировать с PyCharm файлы с расширением **py** (рис. 1.16).
4. Выберите имя для папки в меню **Пуск** (рис. 1.17).
5. Далее PyCharm будет установлен на ваш компьютер (рис. 1.18).

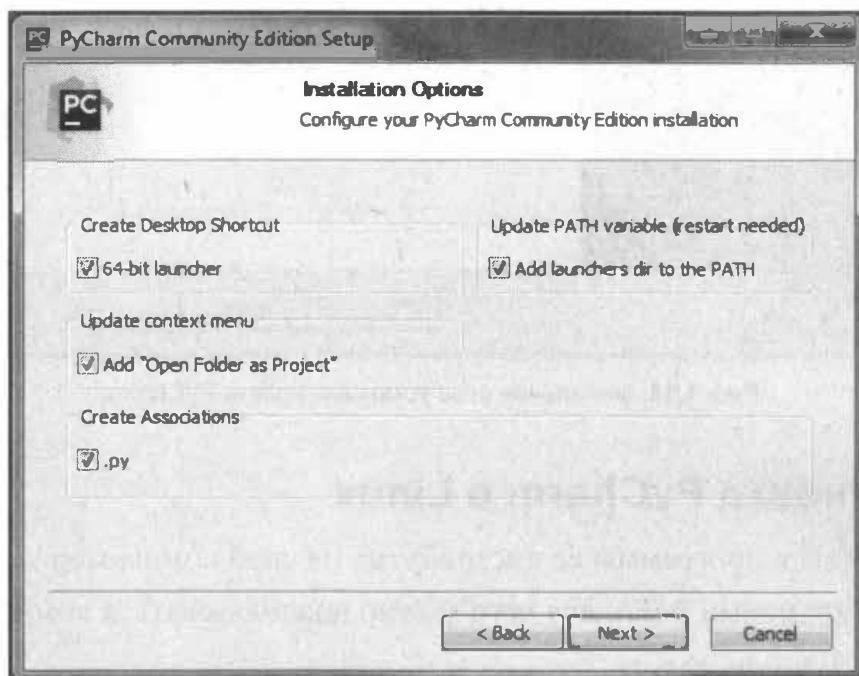


Рис. 1.16. Выбор разрядности устанавливаемой среды разработки PyCharm

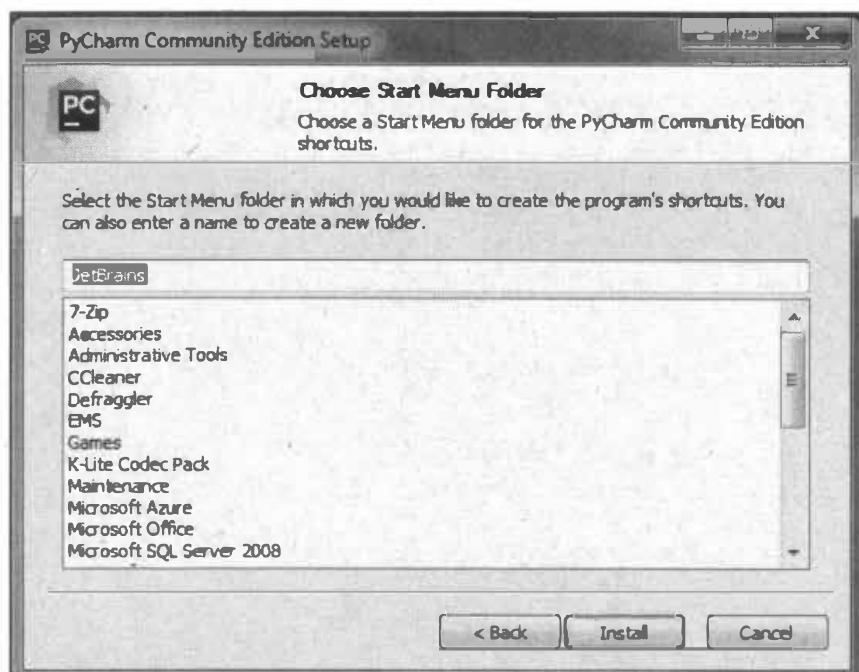


Рис. 1.17. Выбор имени папки для PyCharm в меню Пуск

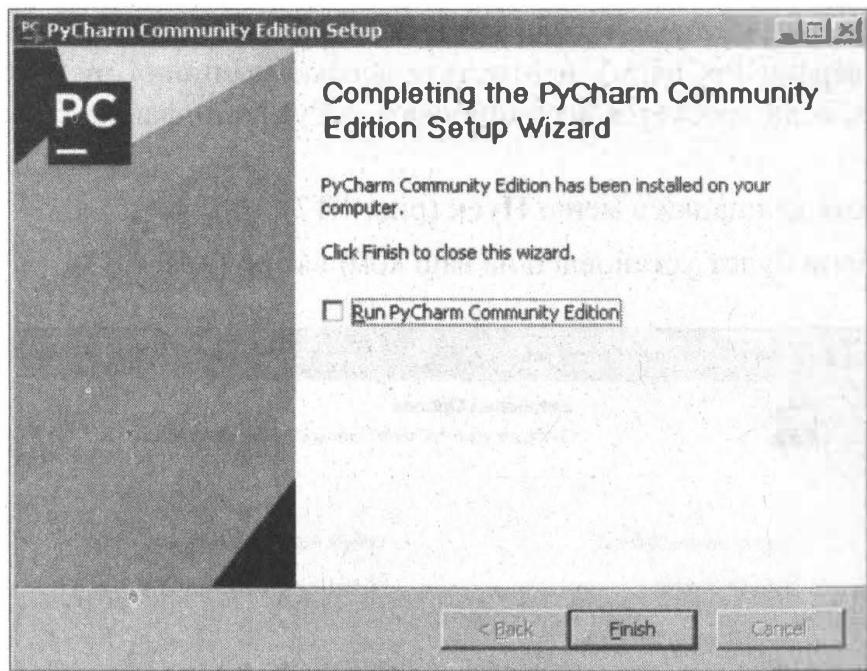


Рис. 1.18. Финальное окно установки пакета PyCharm

## 1.6.2. Установка PyCharm в Linux

1. Скачайте с сайта программы ее дистрибутив на свой компьютер.
2. Распакуйте архивный файл, для чего можно воспользоваться командой:

```
> tar xvf имя_архива.tar.gz
```

Результат работы этой команды представлен на рис. 1.19.

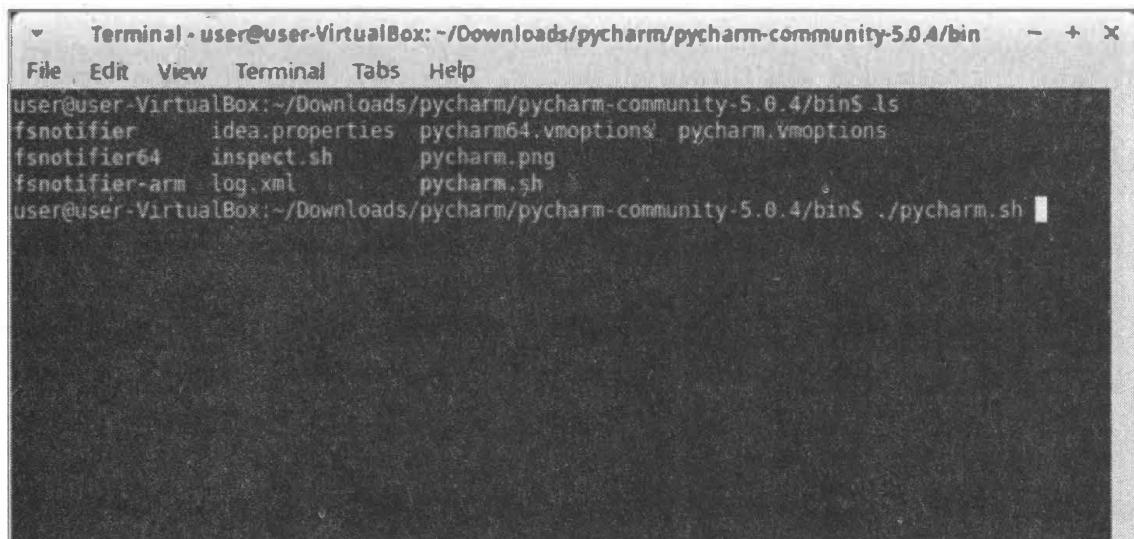
A screenshot of a terminal window titled "Terminal - user@user-VirtualBox: ~/Downloads/pycharm". The window shows the command "tar xvf pycharm-community-5.0.4.tar.gz" being run in the directory "~/Downloads/pycharm". The output of the command is visible in the terminal window.

Рис. 1.19. Результат работы команды распаковки архива PyCharm

3. Перейдите в каталог, который был создан после распаковки дистрибутива, найдите в нем подкаталог `bin` и зайдите в него. Запустите установку PyCharm командой:

```
> ./pycharm.sh
```

Результат работы этой команды представлен на рис. 1.20.



The screenshot shows a terminal window titled "Terminal - user@user-VirtualBox: ~/Downloads/pycharm/pycharm-community-5.0.4/bin". The window contains the following text output from the command:

```
File Edit View Terminal Tabs Help
user@user-VirtualBox:~/Downloads/pycharm/pycharm-community-5.0.4/bin$ ls
fsnotifier    idea.properties  pycharm64.vmoptions  pycharm.vmoptions
fsnotifier64   inspect.sh      pycharm.png
fsnotifier-arm log.xml        pycharm.sh
user@user-VirtualBox:~/Downloads/pycharm/pycharm-community-5.0.4/bin$ ./pycharm.sh
```

Рис. 1.20. Результаты работы команды инсталляции PyCharm

### 1.6.3. Проверка PyCharm

1. Запустите PyCharm и выберите вариант **Create New Project** в открывшемся окне (рис. 1.21).

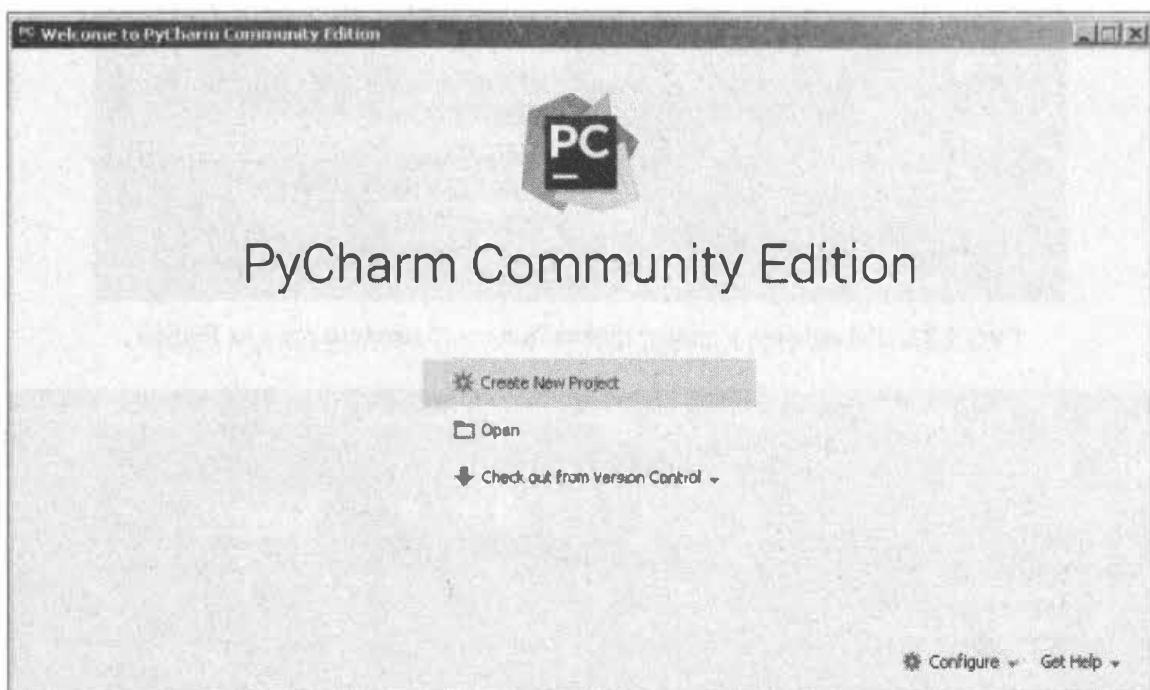
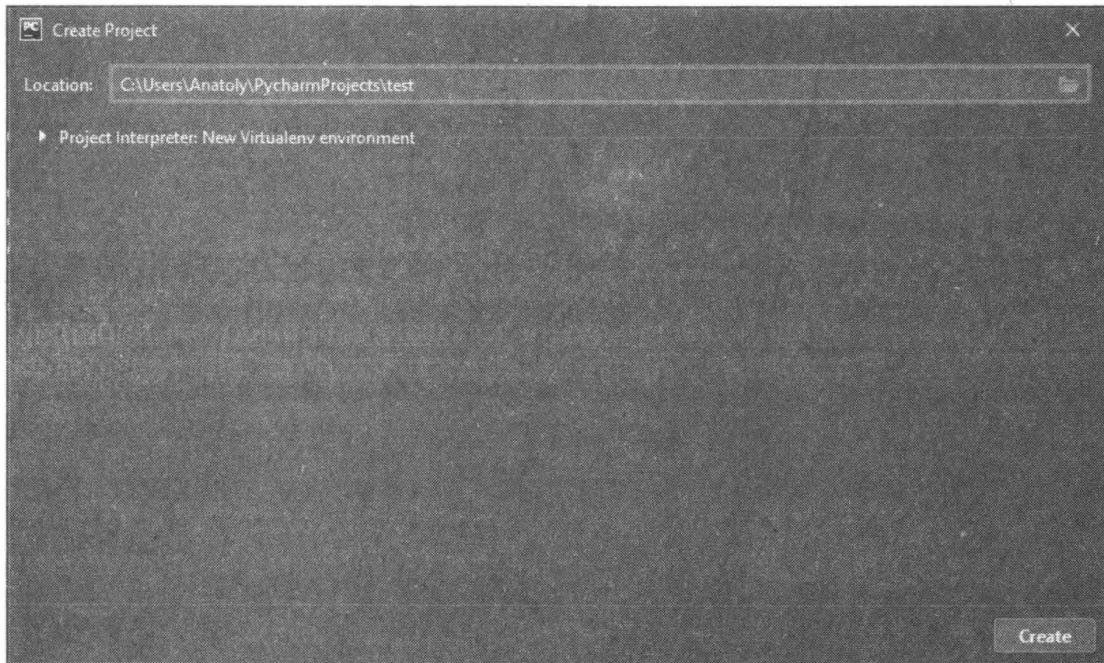
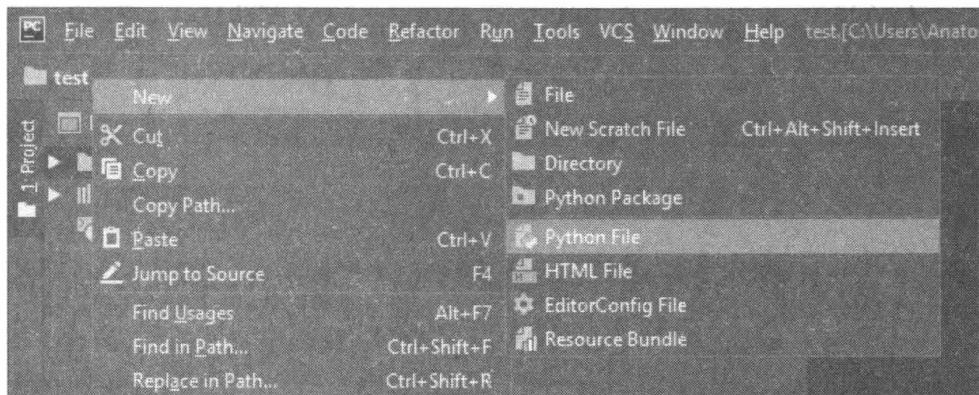


Рис. 1.21. Создание нового проекта в среде разработки PyCharm

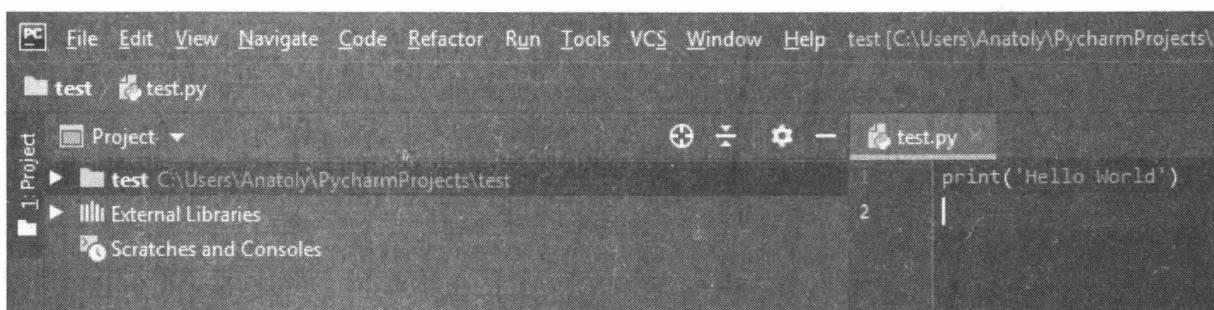


**Рис. 1.22.** Указание пути до проекта в среде разработки PyCharm

2. Укажите путь до создаваемого проекта Python и интерпретатор, который будет использоваться для его запуска и отладки (рис. 1.22).
3. Добавьте в проект файл, в котором будет храниться программный код Python (рис. 1.23).
4. Введите одну строчку кода программы (рис. 1.24).



**Рис. 1.23.** Добавление в проект файла для программного кода на Python



**Рис. 1.24.** Одна строка программного кода на Python в среде разработки PyCharm

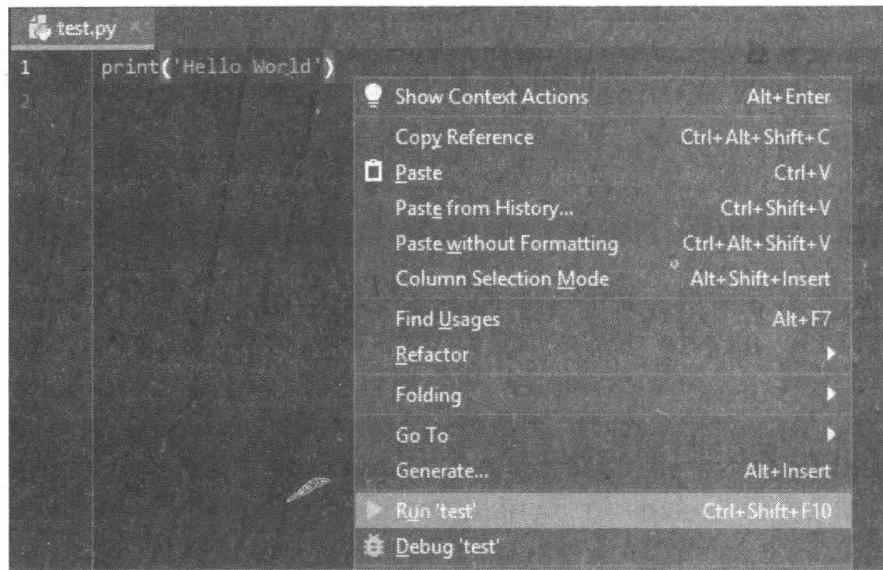


Рис. 1.25. Запуск строки программного кода на Python в среде разработки PyCharm

5. Запустите программу командой **Run** (рис. 1.25).

В результате в нижней части экрана должно открыться окно с выводом результатов работы программы (рис. 1.26).

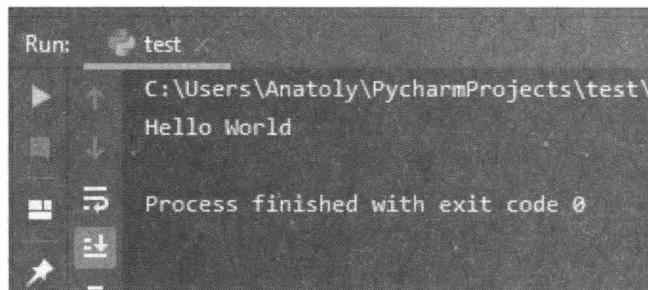


Рис. 1.26. Вывод результатов работы программы на Python в среде разработки PyCharm

## 1.7. Установка пакетов в Python с использованием менеджера пакетов pip

В процессе разработки программного обеспечения на Python часто возникает необходимость воспользоваться пакетом (библиотекой), который в текущий момент отсутствует на вашем компьютере.

В этом разделе вы узнаете о том, откуда можно взять нужный вам дополнительный инструментарий для разработки ваших программ. В частности:

- где взять отсутствующий пакет;
- как установить pip — менеджер пакетов в Python;
- как использовать pip;
- как установить пакет;
- как удалить пакет;

- как обновить пакет;
- как получить список установленных пакетов;
- как выполнить поиск пакета в репозитории.

### 1.7.1. Репозиторий пакетов программных средств PyPI

Необходимость в установке дополнительных пакетов возникнет довольно часто, поскольку решение практических задач обычно выходит за рамками базового функционала, который предоставляет Python. Это, например, создание веб-приложений, обработка изображений, распознавание объектов, нейронные сети и другие элементы искусственного интеллекта, геолокация и т. п. В таком случае необходимо узнать, какой пакет содержит функционал, который вам необходим, найти его, скачать, разместить в нужном каталоге и начать использовать. Все указанные действия можно выполнить и вручную, однако этот процесс поддается автоматизации. К тому же скачивать пакеты с неизвестных сайтов может быть весьма опасно.

В рамках Python все эти задачи автоматизированы и решены. Существует так называемый Python Package Index (PyPI) — репозиторий, открытый для всех разработчиков на Python, в котором вы можете найти пакеты для решения практически любых задач. При этом у вас отпадает необходимость в разработке и отладке сложного программного кода — вы можете воспользоваться уже готовыми и проверенными решениями огромного сообщества программистов на Python. Вам нужно просто подключить нужный пакет или библиотеку к своему проекту и активировать уже реализованный в них функционал. В этом и заключается преимущество Python перед другими языками программирования, когда небольшим количеством программного кода можно реализовать решение достаточно сложных практических задач. Там также есть возможность выкладывать свои пакеты. Для скачивания и установки нужных модулей в ваш проект используется специальная утилита, которая называется `pip`. Сама аббревиатура, которая на русском языке звучит как «пип», фактически раскрывается как «установщик пакетов» или «предпочитаемый установщик программ». Это утилита командной строки, которая позволяет устанавливать, переустанавливать и деинсталлировать PyPI пакеты простой командой `pip`.

### 1.7.2. pip — менеджер пакетов в Python

`pip` — утилита консольная (без графического интерфейса). После того, как вы ее скачаете и установите, она пропишется в PATH и будет доступна для использования. Эту утилиту можно запускать как самостоятельно — например, через терминал Windows или в терминальном окне PyCharm командой:

```
> pip <аргументы>
```

`pip` можно запустить и через интерпретатор Python:

```
> python -m pip <аргументы>
```

Ключ `-m` означает, что мы хотим запустить модуль (в нашем случае `pip`).

При развертывании современной версии Python (начиная с Python 2.7.9 и более поздних версий) pip устанавливается автоматически. В PyCharm проверить наличие модуля pip достаточно просто — для этого нужно войти в настройки проекта через меню **File | Settings | Project Interpreter**. Модуль pip должен присутствовать в списке загруженных пакетов и библиотек (рис. 1.27).

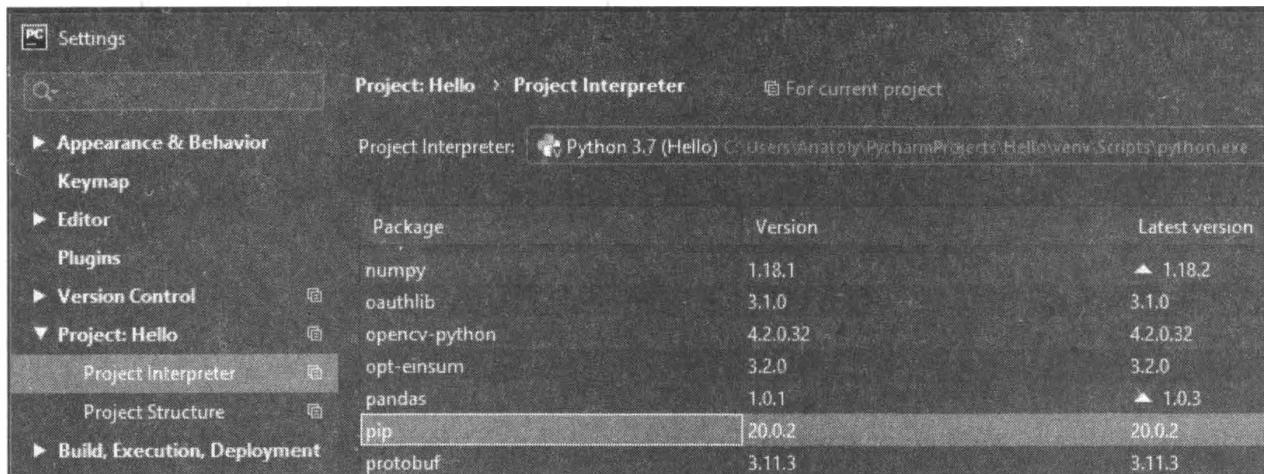


Рис. 1.27. Проверка наличия в проекте модуля pip

В случае отсутствия в списке этого модуля последнюю его версию можно загрузить, нажав на значок + в правой части окна и выбрав модуль pip из списка (рис. 1.28).

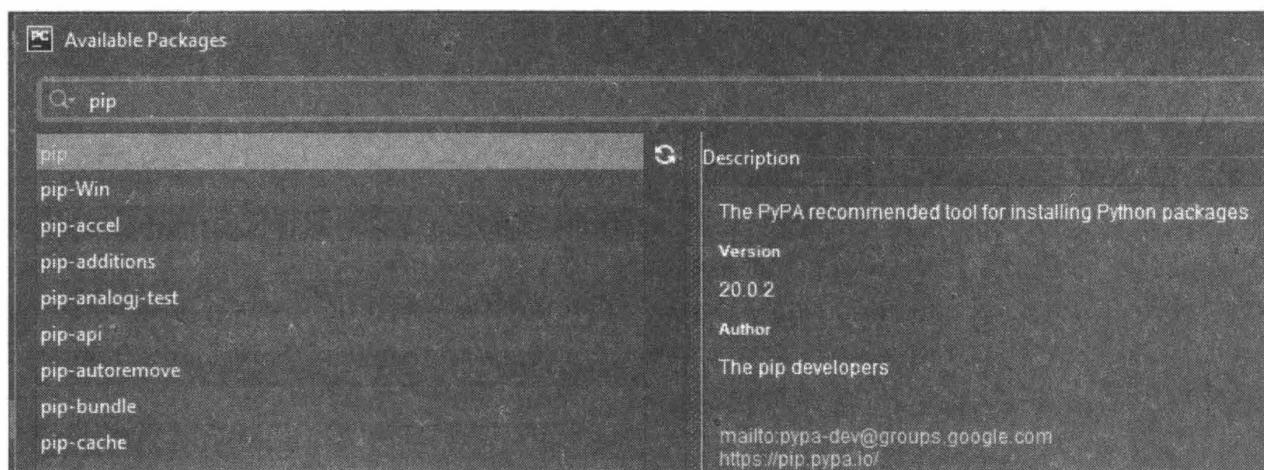


Рис. 1.28. Загрузка модуля pip

### 1.7.3. Использование менеджера пакетов pip

Здесь мы рассмотрим основные варианты использования pip: установку пакетов, удаление и обновление пакетов.

Pip позволяет установить самую последнюю версию пакета, конкретную версию или воспользоваться логическим выражением, через которое можно определить, что вам, например, нужна версия не ниже указанной. Также есть поддержка уста-

новки пакетов из репозитория. Рассмотрим, как использовать эти варианты (здесь Name — это имя пакета).

□ Установка последней версии пакета:

```
> pip install Name
```

□ Установка определенной версии:

```
> pip install Name==3.2
```

□ Установка пакета с версией не ниже 3.1:

```
> pip install Name>=3.1
```

□ Для того чтобы удалить пакет, воспользуйтесь командой:

```
> pip uninstall Name
```

□ Для обновления пакета используйте ключ --upgrade:

```
> pip install --upgrade Name
```

□ Для вывода списка всех установленных пакетов служит команда:

```
> pip list
```

□ Если вы хотите получить более подробную информацию о конкретном пакете, то используйте аргумент show:

```
> pip show Name
```

□ Если вы не знаете точного названия пакета или хотите посмотреть на пакеты, содержащие конкретное слово, то вы можете это сделать, используя аргумент search:

```
> pip search "test"
```

Если вы запускаете pip в терминале Windows, то терминальное окно автоматически закроется после того, как эта утилита завершит свою работу. При этом вы просто не успеете увидеть результаты ее работы. Чтобы терминальное окно не закрывалось автоматически, команды pip нужно запускать в нем с ключом /k. Например, запуск процедуры установки пакета tensorflow должен выглядеть так, как показано на рис. 1.29.

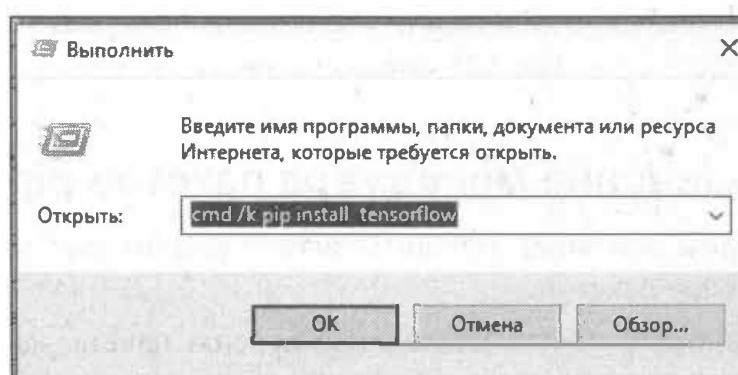
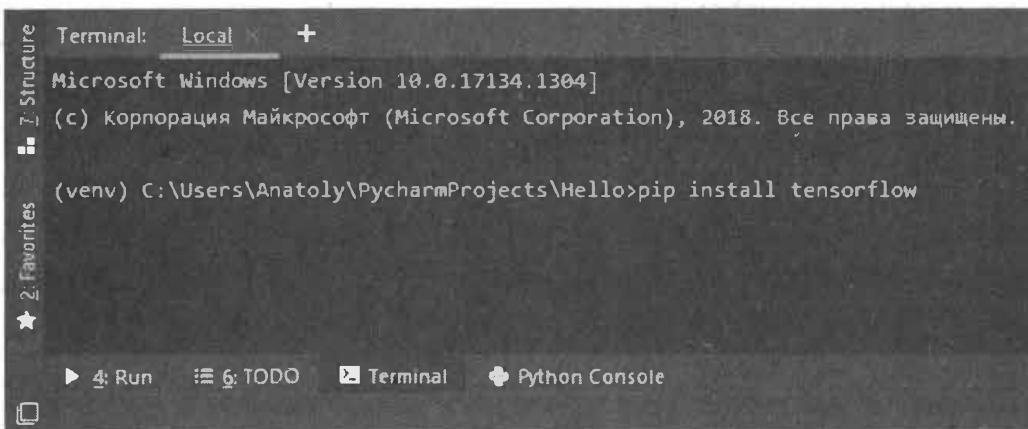


Рис. 1.29. Выполнение команды модуля pip в терминальном окне Windows

Если же пакет pip запускается из терминального окна PyCharm, то в использовании дополнительных ключей нет необходимости, т. к. терминальное окно после завершения работы программы не закрывается. Пример выполнения той же команды в терминальном окне PyCharm показан на рис. 1.30.



Terminal: Local +  
Microsoft Windows [Version 10.0.17134.1304]  
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.  
(venv) C:\Users\Anatoly\PycharmProjects>Hello>pip install tensorflow  
▶ 4: Run ⌂ 6: TODO Terminal Python Console

Рис. 1.30. Выполнение команды модуля pip в терминальном окне PyCharm

## 1.8. Фреймворк Django для разработки веб-приложений

Итак, основной инструментарий для разработки программ на языке Python установлен, и мы можем перейти к установке дополнительных библиотек. В этом разделе мы установим веб-фреймворк Django, который позволяет разрабатывать веб-приложения.

Запустим среду разработки PyCharm и создадим в ней новый проект Web\_1. Для этого в главном меню среды выполните команду **File | New Project** (рис. 1.31).

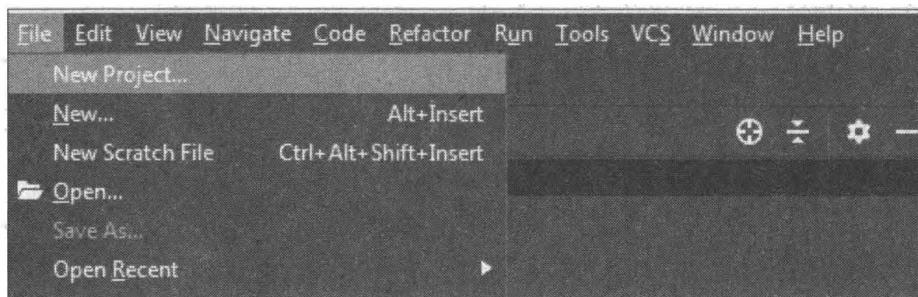


Рис. 1.31. Создание нового проекта в среде разработки PyCharm

Откроется окно, где вы можете задать имя создаваемому проекту, определить виртуальное окружение для этого проекта и указать каталог, в котором находится интерпретатор Python. Задайте новому проекту имя Web\_1 и нажмите кнопку **Create** (рис. 1.32).

Будет создан новый проект. Это, по сути дела, шаблон проекта, в котором пока еще ничего нет (рис. 1.33).