


Юлия Камалова

ПРОГРАММИРОВАНИЕ НА PYTHON С НУЛЯ



РОССИЙСКИЙ
КОМПЬЮТЕРНЫЙ
БЕСТСЕЛЛЕР

 **БОМБОРА**
ИЗДАТЕЛЬСТВО
Москва

УДК 004.43
ББК 32.973-018.2
К18

Камалова, Юлия Борисовна.

К18 Программирование на Python с нуля / Юлия Камалова. — Москва : Эксмо, 2026. — 448 с. — (Российский компьютерный бестселлер).

ISBN 978-5-04-214684-8

Книга «Программирование на Python с нуля» — это полное руководство для начинающих, студентов и всех, кто хочет освоить Python. Вы познакомитесь с основами языка, научитесь работать в IPython, Jupyter Notebook, Google Colab и Anaconda, разберетесь с ветвлениями, циклами, типами и структурами данных и объектно ориентированным программированием. Особое внимание уделено инженерии данных, работе с API и базами данных, визуализации (Matplotlib, Seaborn), а также применению Python в науке и машинном обучении (NumPy, Pandas, scikit-learn). Книга содержит множество примеров и советы по развитию карьеры.

УДК 004.43
ББК 32.973-018.2

ISBN 978-5-04-214684-8

© Камалова Ю.Б., текст, иллюстрации, 2026
© Оформление. ООО «Издательство «Эксмо», 2026

Оглавление

Об авторе	9
Введение	10
Глава 1. Введение в программирование на языке Python	14
Цели, задачи и основные понятия раздела	14
Более подробно о программном обеспечении	16
iPython	16
Jupyter Notebook	17
Google Colab	18
Anaconda	19
PyCharm	20
Интегрированная среда разработки (IDE)	20
Глава 2. Интерактивная среда iPython	22
Что такое Python?	22
Как описать язык программирования	22
История Python	24
Рождение языка	24
Развитие языка	25
Лицензия	26
Свободное распространение	27
Python Software Foundation License	27
Портируемость	27
Портируемость в контексте Python	28
Примеры портируемости	28
Режим работы Python	28
Стандартный режим	28
Интерактивный режим	29
Эффективность в Python	30
Выполнение	30
Разработка	31
Тестирование	32

4 Оглавление

Библиотеки Python	32
Стандартные библиотеки	32
Сторонние библиотеки	33
Примеры различий	34
Пакетный менеджер PyPI и pip	36
Сообщество и документация	36
Модульность и повторное использование кода	36
Примеры применения библиотек	37
Концепция Python Zen	37
Выводы	38
Глава 3. Управляющие конструкции в Python: ветвления, циклы, исключения	41
Алгоритм создания первой программы в Google Colab	41
Ветвления в Python	46
Циклы в Python	55
Цикл for	55
Цикл while	57
Операторы break и continue	59
Функция enumerate()	63
Функция zip()	64
Сочетание for с enumerate() и zip()	66
Обработка исключений	69
Выводы	71
Глава 4. Основные синтаксические конструкции Python	74
Арифметические операторы +, -, /, //, %, **	74
Основные правила приоритета операторов в Python (от высшего к низшему)	78
Операторы присваивания: =, +=, -=, *=, /=, //=, %=, **=	82
Правила и рекомендации по наименованию переменных	87
Типы данных: int, float и str	87
Правила определения типа переменной в Python	88
Определение типа переменной	88
Особенности и нюансы типов	89
Преобразование между типами	89
Важные нюансы	90
Строковые операции	90
Более сложные сценарии	92
Операторы сравнения	93

Основные операторы сравнения	93
Типы данных логических выражений	96
Проверка четности числа в Python	96
Оператор членства in	99
Методы lower() и upper()	101
Логические операторы and и or	102
Оператор and	103
Оператор or	104
Использование логических операторов в условиях	104
Дополнения к логическим операторам and и or	105
Использование в списковых выражениях (list comprehensions)	106
Практический пример с пользовательским вводом	108
Циклы	110
Циклы со счетчиком	110
Цикл for	110
Цикл for	111
Конструкция range()	112
Операторы break и continue в цикле for	115
Цикл while	118
Выводы	121
Глава 5. Структуры данных в Python	124
Строки	124
Индексы и срезы строк	129
Изменяемость строк	133
Длина строки	134
Словари	135
Словари как инструмент для хранения данных	149
Проход по словарю в цикле	150
Изменение словаря во время итерации	153
Вложенные словари в Python	154
Кортежи	160
Характеристики кортежей	160
Создание кортежа из одного элемента	162
Доступ к элементам кортежа	163
Списки	164
Создание списков	165
Доступ к элементам списка	165
Основные операции со списками	168

6 Оглавление

Функция <code>len()</code> в списках	172
Функция <code>type()</code> в списках	179
Индексы и срезы в списках	183
Методы работы со списками	184
Выводы	190
Глава 6. Инженерия данных: манипуляции с файлами и каталогами	195
Текстовые файлы	195
Фундаментальные аспекты работы с файлами	195
Чтение текстового файла	196
Обработка исключительных ситуаций при работе с файлами	201
Манипуляции со структурированными данными	202
Чтение и запись данных в формате JSON	210
Работа с XML-документами	216
Обработка медиафайлов	223
Работа с изображениями	223
Работа со звуковыми файлами	234
Установка необходимых библиотек	235
Взаимодействие с веб-API	241
Библиотека <code>requests</code> для HTTP-запросов	241
Основная терминология	241
Библиотеки для работы с веб-API	242
Практические примеры автоматизации	249
Выводы	252
Глава 7. Основы объектно ориентированного программирования	
в Python	257
Основные принципы ООП	257
Инкапсуляция	257
Защита данных и свойства статуса	258
Свойства и методы	260
Синглтоны и управляемый доступ	262
Защита состояний через свойства	264
Уровни доступа	267
Наследование	269
Логика наследования	269
Множественное наследование	270
Суперклассы и метод <code>super()</code>	272
Полиморфное наследование и изменение поведения	274
Класс-обертка (<code>wrapper class</code>)	277

Полиморфизм	279
Паттерн «Стратегия»	284
Интерфейсы через duck typing	286
Обобщенные функции и классы	288
Абстракция	290
Классы и объекты: строительные блоки ООП	302
Инкапсуляция, наследование и полиморфизм (дополнительные примеры)	309
Инкапсуляция	309
Наследование	313
Полиморфизм	316
Обработка исключений и итераторы	322
Итераторы	322
Декораторы и метаклассы	323
Декораторы классов	323
Метаклассы	324
Паттерны проектирования	325
Singleton (Одиночка)	325
Фабричный метод	325
Наблюдатель	326
Выводы	327
Глава 8. Работа с базами данных в Python	331
Подключение к базам данных и выполнение запросов	331
ORM и SQLAlchemy: абстракция вместо SQL	332
Обработка результатов запросов и отображение данных	334
Преобразование результатов в словари с использованием SQLite	334
Использование библиотеки Pandas для анализа данных	336
Исторический экскурс: PostgreSQL vs NoSQL	337
Реляционные базы данных (PostgreSQL)	337
NoSQL-базы данных	337
Введение в NoSQL: MongoDB на примере	338
Выводы	339
Глава 9. Графическое программирование	341
Разработка графических приложений	341
Альтернативные подходы в Google Colab	341
Визуализация данных	343
Matplotlib	343

8 Оглавление

Seaborn — расширенные возможности визуализации	363
Выводы	381
Глава 10. Применение Python в научных вычислениях, анализе данных и машинном обучении	384
Научные вычисления и анализ данных	384
Библиотека NumPy: основы работы с многомерными массивами	384
Библиотека Pandas: инструменты для анализа табличных данных	401
Машинное обучение и искусственный интеллект	419
Библиотека scikit-learn: от предобработки данных до оценки моделей	419
Выводы	432
Рекомендации по дальнейшему изучению	433
Заключение	433
Глава 11. Путь Python-разработчика	434
Пути дальнейшего обучения	434
Углубление в специализации	434
Книги для вдохновения	435
Участие в open-source проектах	436
Почему это важно?	436
Как начать?	436
Заключение	437
Список источников	438

Об авторе

Камалова Юлия Борисовна — опытный преподаватель и инженер, специализирующийся на информационных технологиях и анализе данных. Сейчас она занимает должность старшего преподавателя на Кафедре информационных технологий в Финансовом университете при Правительстве Российской Федерации, где обучает студентов математике, управлению данными с использованием PostgreSQL, программированию на Python и машинному обучению. Юлия Борисовна активно участвует в научной работе. Она преподавала в Центре дополнительного образования МГТУ имени Н. Э. Баумана, где вела занятия, посвященные Python и информационной безопасности.

Карьера Юлии Борисовны началась в 2008 году, когда она работала лаборантом в компании, занимающейся экспертизой промышленной безопасности. С тех пор она накопила опыт работы на различных кафедрах ИжГТУ, включая «Приборы и методы контроля качества» и «Прикладная математика». В дополнение к этому Юлия Борисовна преподавала английский язык, что позволило ей расширить свои педагогические навыки.

В 2020 году работала в коммерческой дирекции завода «Нефтемаш» в качестве специалиста по тендерной документации и переводчика. В том же году она стала инженером 1 категории в Ижевском нефтяном научном центре («Роснефть»), где занималась разработкой информационной системы и созданием базы данных. Этот опыт позволил ей углубить свои знания в области автоматизации и внедрения новых технологий.

Юлия Камалова продолжает следовать своим интересам в области образования и инженерии, стремясь делиться знаниями и опытом с будущими специалистами.

Введение

В современном мире информационных технологий знание языка программирования Python становится все более важным и востребованным навыком. Python — это универсальный язык, который широко используется для обработки данных, автоматизации процессов, разработки веб-приложений и во многих других областях. Простота и читаемость делают Python идеальным выбором как для начинающих программистов, так и для опытных разработчиков.

Python поддерживает множество популярных библиотек и фреймворков, таких как NumPy, Pandas, Matplotlib и Django, что позволяет разработчикам и аналитикам эффективно решать самые разнообразные задачи. Эти возможности делают Python одним из самых популярных языков программирования на рынке труда.

В последние годы Python стал одним из самых востребованных языков программирования благодаря своей простоте, богатой экосистеме и применимости в различных областях. Рассмотрим ключевые направления, где Python находит свое применение, — от веб-разработки и до машинного обучения.

Веб-разработка

Python активно используется для создания веб-приложений благодаря своим мощным фреймворкам, таким как Django и Flask. Эти инструменты позволяют разработчикам:

- *Быстро создавать и разрабатывать веб-сайты и веб-приложения.* Django, например, предлагает множество встроенных функций, таких как управление пользователями, работа с базами данных и безопасность, что позволяет сократить время разработки.
- *Создавать RESTful API.* Python позволяет легко разрабатывать API для взаимодействия с фронтенд-приложениями, что делает его хорошим выбором для разработки многоуровневых систем.
- *Пользоваться простотой и ясностью синтаксиса.* Это облегчает разработку и поддержку кода, что делает его идеальным для командной работы.

Анализ данных и визуализация

Python стал стандартом де-факто в области анализа данных. Благодаря таким библиотекам, как Pandas, NumPy и Matplotlib, разработчики могут эффективно обрабатывать, анализировать и визуализировать данные. Возможности включают:

- *Обработку и анализ больших объемов данных.* Pandas предоставляет мощные инструменты для манипуляций с таблицами данных, что позволяет легко выполнять такие операции, как фильтрация, агрегация и преобразование.
- *Визуализацию данных.* Библиотеки, такие как Matplotlib и Seaborn, позволяют создавать информативные графики и диаграммы для представления результатов анализа.
- *Работу с данными из разных источников.* Python может взаимодействовать с базами данных, API, веб-сайтами и файлами в различных форматах (CSV, Excel, JSON и т. д.).
- *Машинное обучение и искусственный интеллект.*

Python стал ведущим языком для разработки приложений в области машинного обучения и искусственного интеллекта благодаря своему удобству и множеству библиотек, таких как TensorFlow, Keras и Scikit-learn. Здесь Python предоставляет:

- *Разработку и реализацию алгоритмов машинного обучения.* Упрощает создание моделей, обучение на данных и тестирование их производительности.
- *Глубокое обучение.* Библиотеки, такие как TensorFlow и PyTorch, позволяют создавать сложные нейронные сети для обработки изображений, текста и других типов данных.
- *Простоту интеграции с другими инструментами.* Python легко сочетается с другими языками и инструментами, что упрощает создание сложных систем и прототипов.

Автоматизация и скриптование

Python также очень популярен в сфере автоматизации задач благодаря своей простоте и мощным библиотекам для работы с системами и веб-сервисами. Возможности включают:

- *Создание скриптов для автоматизации рутинных задач.* Python позволяет быстро писать скрипты, которые могут выполнять задачи, такие как обработка файлов, взаимодействие с веб-сервисами и автоматизация рабочих процессов.
- *Создание ботов и парсеров.* С помощью библиотек, например BeautifulSoup и Scrapy, Python позволяет извлекать и обрабатывать данные из веб-страниц, что очень полезно для сбора информации и анализа.

Научные вычисления и инженерия

Python использует в своих приложениях, связанных с научными исследованиями и инженерией:

- *Численные вычисления:* NumPy и SciPy, которые предоставляют мощные инструменты для работы с многомерными массивами и численными задачами, что важно в таких областях, как физика и инженерия.
- *Симуляции и моделирование.* Python позволяет легко создавать модели для тестирования гипотез и проведения экспериментов.

Таким образом, Python предлагает большие возможности для разработчиков, независимо от их уровня подготовки. Простота, обширная экосистема библиотек и активное сообщество делают его подходящим для решения множества задач, будь то создание веб-приложений, анализ данных, машинное обучение или автоматизация процессов. Освоив Python, вы откроете для себя широкие горизонты в мире программирования и технологий.

Цель книги

Цель этой книги — научить читателей основам работы с Python, позволяя им уверенно использовать язык для решения повседневных задач, связанных с программированием и анализом данных. Мы рассмотрим базовые концепции языка, синтаксис, структуры данных и основные алгоритмы.

Темы, такие как оптимизация производительности кода, проектирование сложных приложений и работа с базами данных, не менее важны, но для начинающих программистов лучше сначала освоить основы языка. Поэтому акцент сделан на практических примерах и задачах, чтобы читатели могли постепенно углублять свои знания и уверенно переходить к более сложным аспектам Python.

Программное обеспечение для работы с Python

Для работы с Python и выполнения примеров использовалось бесплатное программное обеспечение (ПО), такое как Anaconda [1], или стандартный интерпретатор Python, доступный для всех основных операционных систем, включая Windows, Linux и MacOS. Эти инструменты предоставляют удобную среду для разработки и позволяют легко устанавливать необходимые библиотеки и модули.

Для изучения языка программирования Python вам потребуется установить на свой компьютер необходимое ПО. Вы также можете воспользоваться готовыми решениями в виде онлайн-сервисов, которые позволяют запускать код Python и работать с библиотеками без необходимости установки локальной среды.

Основная среда разработки для Python

Python — это язык программирования, для которого существует множество сред разработки. Одна из самых популярных — PyCharm. Это мощная интегрированная среда разработки (IDE), которая предлагает множество инструментов для работы с Python, включая автодополнение кода, отладку и взаимодействие с системами контроля версий.

Официальный сайт: <https://www.jetbrains.com/pycharm> [2].

Альтернативные среды разработки

Если вы предпочитаете легкие решения, можно использовать Jupyter Notebook [3]. Этот инструмент предназначен для работы с документами, содержащими живой код, визуализации и текстовую информацию. Jupyter Notebook особенно популярен среди исследователей и аналитиков данных.

Jupyter можно установить через пакетный менеджер Anaconda или с помощью команды `pip install notebook`.

Онлайн-сервисы для работы с Python

Если вы хотите избежать установки ПО, то можете воспользоваться онлайн-сервисами, такими как Google Colab. Этот бесплатный инструмент позволяет писать и запускать код Python в браузере и предоставляет доступ к облачным ресурсам, включая графические процессоры для обучения моделей машинного обучения.

Официальный сайт: <https://colab.research.google.com> [4].

Пользуясь этими инструментами, вы сможете легко начать изучать Python и использовать его возможности для решения различных задач.

Весь код, графики, рисунки, диаграммы, приведенные в этой книге, можно взять здесь: <https://colab.research.google.com/drive/1etWWCRVv9qC9xq9S5fgGmirPWhQgKnx?usp=sharing> или здесь: https://addons.eksmo.ru/it/Python_snulya_code.7z.

Глава 1

Введение в программирование на языке Python

Цели, задачи и основные понятия раздела

Цель этой главы в том, чтобы познакомить читателя с основами языка Python, его возможностями и функциональностью.

Python — это высокоуровневый язык программирования, который зарекомендовал себя как один из самых популярных и универсальных языков, подходящих для решения широкого спектра задач, от веб-разработки до анализа данных и машинного обучения [5].

Что значит «высокоуровневый язык программирования»? Это язык, который предоставляет разработчикам удобный и абстрактный способ взаимодействия с компьютером [6].

Высокоуровневые языки программирования характеризуются следующими признаками:

1. *Читаемость и простота.* Они имеют синтаксис, который ближе к естественному языку или математическим записям. Это делает код более понятным и легким для чтения, что упрощает процесс написания и сопровождения программ.

2. *Абстракция от аппаратуры.* Эти языки абстрагируют детали работы с аппаратным обеспечением, позволяя программистам сосредоточиться на логике решения задач, не углубляясь в работу процессора, памяти и других низкоуровневых компонентов системы.

3. *Богатая стандартная библиотека.* Высокоуровневые языки обычно поставляются с обширными библиотеками встроенных функций и модулей, что позволяет программистам быстро разрабатывать приложения, не занимаясь реализацией базовых функций с нуля.

4. *Портативность*. Программы, написанные на высокоуровневых языках, обычно могут быть запущены на различных платформах с минимальными изменениями. Это значительно упрощает разработку и развертывание приложений.

5. *Управляемая память*. Многие высокоуровневые языки, такие как Python, имеют встроенные механизмы управления памятью, например сборщик мусора, который автоматически освобождает память, когда она больше не нужна. Это снижает риск ошибок, связанных с управлением памятью, вроде утечек памяти.

Среди высокоуровневых языков программирования можно назвать Java, C, Ruby и т. д.

В противовес этому низкоуровневые языки, такие как ассемблер и машинные коды, требуют от программистов более глубокого понимания архитектуры компьютера и работы с его ресурсами, что делает их менее доступными для начинающих разработчиков.

Перед тем как погружаться в более сложные концепции программирования, важно освоить инструмент, который позволит вам эффективно взаимодействовать с Python.

Вот несколько задач, которые нам предстоит решить:

1. *Изучить основы работы в интерактивной среде iPython* [7]. Это мощная интерактивная оболочка для Python, которая предлагает удобный способ выполнения кода, позволяет разработчикам получать мгновенный отклик на введенные команды и облегчает процесс разработки и отладки. Важно отметить, что iPython прекрасно подходит для экспериментов и быстрого прототипирования, что особенно полезно для начинающих программистов.

Почему iPython?

Он позволит вам:

- *Выполнять код по строкам*. Можно вводить и выполнять команды по одной строке, что позволяет тестировать различные фрагменты кода без необходимости создания полного скрипта.
- *Работать с переменными*. Можно создавать переменные прямо в интерактивной среде и обращаться к ним в любое время, что значительно ускоряет рабочий процесс.
- *Использовать «магические» команды*. iPython предлагает много «магических» команд (начинаются с %) и функций, которые облегчают выполнение определенных задач. Например, можно быстро получить информацию о переменной, вызвать графические интерфейсы, управлять файлами и выполнять другие операции без написания лишнего кода.