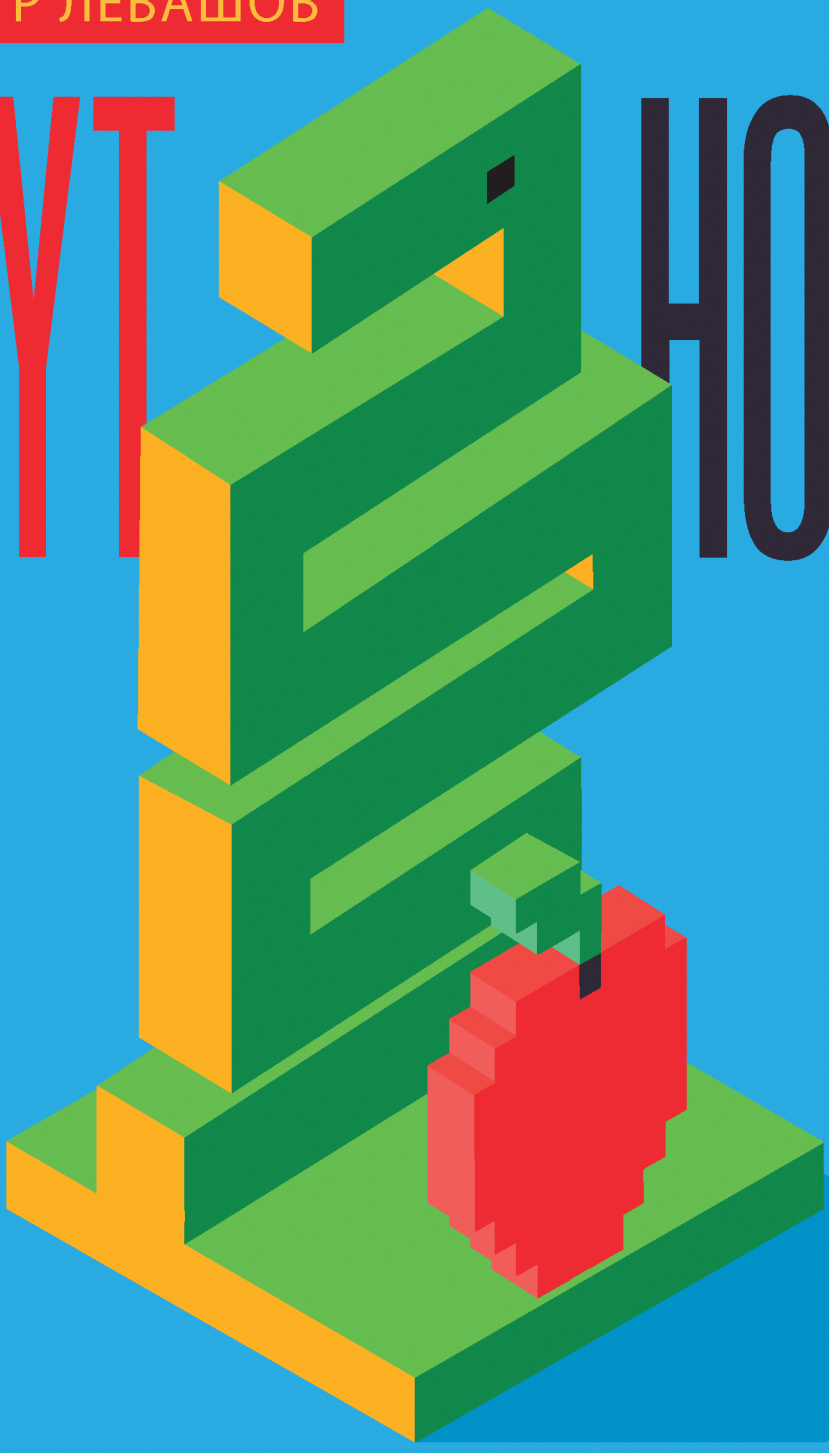
ПЕТР ЛЕВАШОВ

PYTHON



С НУЛЯ

Петр Левашов

Python с нуля

Серия «Библиотека программиста»

Руководитель дивизиона	*Ю. Сергиенко*
Руководитель проекта	*А. Питиримов*
Ведущий редактор	*Н. Гринчик*
Художественный редактор	*В. Мостипан*
Литературный редактор	*К. Тульцева*
Корректоры	*С. Беляева, Н. Викторова*
Верстка	*Л. Егорова*

ББК 32.973.2-018.1
УДК 004.43

Левашов Петр

Л34 Python с нуля. — СПб.: Питер, 2024. — 448 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-2145-8

Добро пожаловать в увлекательный мир программирования на языке Python! Независимо от того, начинающий вы или опытный программист, вы вооружитесь знаниями и навыками, необходимыми для успешного освоения языка. Python, известный своей простотой и универсальностью, завоевал огромную популярность среди разработчиков во всем мире. Благодаря удобному синтаксису и широкой библиотечной поддержке он идеально подходит для решения широкого спектра задач — от веб-разработки и анализа данных до программирования графических интерфейсов. Книга представляет собой комплексное руководство по изучению языка Python с нуля.

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

ISBN 978-5-4461-2145-8

Изготовлено в России. Изготовитель: ООО «Прогресс книга». Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург, Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 01.2024. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 — Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 30.11.23. Формат 70×100/16. Бумага офсетная. Усл. п. л. 36,120. Тираж 1000. Заказ 0000.

КРАТКОЕ СОДЕРЖАНИЕ

Глава 1

ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ НА PYTHON

Что такое Python

Python — это универсальный язык программирования высокого уровня, который легко изучать и читать и на котором просто писать. В 1991 году Гвидо ван Россум впервые опубликовал исходный код Python, и с тех пор он стал одним из самых популярных языков программирования в мире.

Python обладает рядом особенностей, которые делают его отличным выбором как для начинающих, так и для опытных программистов. Одна из ключевых особенностей Python — читаемость. Код на Python легко читается и понимается, что облегчает его изучение и сопровождение. Python также имеет простой и интуитивно понятный синтаксис, это означает, что разработчики могут писать код быстрее и с меньшим количеством ошибок.

Еще одна причина популярности языка — его универсальность. Python можно использовать для широкого спектра приложений, включая веб-разработку, научные вычисления, искусственный интеллект и анализ данных. Python также имеет большое и активное сообщество разработчиков, которые вносят свой вклад в создание различных библиотек и инструментов с открытым исходным кодом.

Python — интерпретируемый язык, а это означает, что его интерпретатор выполняет код напрямую, без необходимости компиляции. Это делает код Python легким для тестирования и отладки, а также позволяет разработчикам быстро повторять код и экспериментировать с ним.

Python — отличный выбор для всех, кто хочет научиться программированию. Его простота, читабельность и универсальность делают его отличным языком для начинающих, а мощные возможности и активное сообщество делают его ценным инструментом для опытных разработчиков.

История Python

История языка программирования Python началась в конце 1980-х. Гвидо ван Россум — голландский ученый-компьютерщик, который в то время работал в научно-исследовательском институте Centrum Wiskunde & Informatica в Амстердаме, начал работу над Python в качестве хобби. Он намеревался создать язык, который было бы легко читать и писать.

Первая версия Python, 0.9.0, была выпущена в феврале 1991 года. Это был простой язык с базовыми типами данных, управляющими структурами и функциями, но он быстро завоевал популярность среди небольшого сообщества разработчиков, которые его использовали.

В 1994 году была выпущена версия Python 1.0, где появились новые возможности: инструменты функционального программирования и система модулей. Эта версия Python также включала систему Python Enhancement Proposals (PEPs), которая позволяла разработчикам предлагать и обсуждать изменения в языке.

Python 2.0 был выпущен в 2000 году, в него были включены сборка мусора и поддержка Unicode. За Python 2.0 последовало несколько других релизов в серии 2.x, включая Python 2.7, который был выпущен в 2010 году и широко используется по сей день.

В 2008 году началась разработка Python 3.0, целью которой было устранение некоторых недостатков и несоответствий в Python 2.x. Python 3.0 был выпущен в 2008 году и внес в язык несколько серьезных изменений, включая новую функцию `print`, улучшенную поддержку Unicode и упрощенный синтаксис для определения классов.

Сегодня Python — один из самых популярных языков программирования в мире с большим и активным сообществом разработчиков, которые вносят свой вклад в создание различных библиотек и инструментов с открытым исходным кодом.

Установка Python и среды разработки

Для начала нужно установить Python на свой компьютер. Это бесплатный язык с открытым исходным кодом, его можно загрузить с официального сайта Python по адресу www.python.org.

Есть две основные версии Python: Python 2.x и Python 3.x. Хотя Python 2.x все еще используется в некоторых старых приложениях, для новых проектов рекомендую использовать Python 3.x, поскольку это последняя версия языка, имеющая несколько важных улучшений по сравнению с Python 2.x.

Вы можете загрузить программу установки с официального сайта и следовать инструкциям по установке. В процессе установки можно выбрать глобальную

установку Python на своем компьютере или локальную в определенном каталоге.

Помимо самого Python вам может понадобиться установить среду разработки, которая поможет писать код Python и управлять им. Среда разработки — это приложение, которое предоставляет инструменты для написания, тестирования и отладки кода. Некоторые популярные среды разработки для Python включают PyCharm, Visual Studio Code и Sublime Text.

При выборе среды разработки следует учитывать ее простоту использования, поддержку библиотек и инструментов Python и совместимость с вашей операционной системой. Многие среды разработки предлагают автодополнение кода, подсветку синтаксиса и инструменты отладки, которые помогут писать код более эффективно и с меньшим количеством ошибок.

После установки Python и среды разработки можно приступать к написанию кода на Python! В следующем разделе рассмотрим основы написания и выполнения программ на Python.

Интерпретатор Python и REPL (Read-Eval-Print Loop)

Одна из уникальных особенностей Python — его интерактивный режим, который позволяет выполнять код и сразу же видеть результаты. Это возможно благодаря интерпретатору Python, который представляет собой программу, читающую и выполняющую код Python.

Когда вы устанавливаете Python на свой компьютер, он включает интерактивный интерпретатор цикл чтения-вывода-печати (REPL). REPL позволяет вводить код по одной строке за раз и сразу же видеть результаты. Чтобы запустить Python REPL, откройте терминал или командную строку и введите `python`.

Запустите интерпретатор Python, введите код и нажмите Enter, чтобы посмотреть результаты. Например, чтобы вывести на экране «Привет, мой новый друг», наберите такой код:

```
>>> print("Привет, мой новый друг!")
Привет, мой новый друг!
```

`>>>` — это подсказка, которую интерпретатор Python выводит на экран, чтобы показать, что он готов к вводу кода. Интерпретатор оценит код и выведет результат на следующей строке.

Помимо интерактивного режима, можно записать код в файл и выполнить его из командной строки. Для этого создайте новый текстовый файл и сохраните его с расширением `.py` (например, `myprogram.py`). Затем запустите программу, набрав `python myprogram.py` в командной строке.

Интерпретатор Python и REPL — это мощные инструменты для написания и тестирования кода Python. Используя интерактивный режим, можно быстро экспериментировать с различными фрагментами кода и сразу же видеть результаты. В следующем разделе рассмотрим основы написания кода в файле и его запуск из командной строки.

Ваша первая программа на Python

Теперь, когда у вас установлен Python и есть базовое понимание того, как использовать интерпретатор, напишем нашу первую программу на Python!

В Python программа — это набор операторов, которые выполняются по порядку. Оператор — это строка кода, которая выполняет определенное действие, например выводит сообщение на экране или вычисляет значение.

Откройте текстовый редактор (например, Notepad или Sublime Text) и создайте новый файл. Сохраните файл с расширением `.py`, которое указывает, что это программа на Python. Например, `program.py`.

Затем введите следующий код:

```
print("Я готов, хозяин!")
```

Код говорит Python вывести на экран сообщение «Я готов, хозяин!». Функция `print` — это встроенная функция в Python, позволяющая выводить текст на экран.

После того как вы ввели код, сохраните файл и выйдите из текстового редактора. Теперь откройте окно командной строки или терминала и перейдите в каталог, где сохранили файл. Введите `python program.py` и нажмите Enter.

Python выполнит код в файле и выведет на экран сообщение «Я готов, хозяин!». Поздравляю, вы только что написали и запустили свою первую программу на Python!

Конечно, это только начало того, что можно сделать с помощью Python. В следующем разделе рассмотрим некоторые основные концепции программирования в Python, включая переменные, типы данных и операторы.

Синтаксис и основные концепции программирования

Рассмотрим основные концепции программирования в Python, включая синтаксис, переменные, типы данных и операторы.

Синтаксис относится к правилам и рекомендациям по написанию кода Python. В этом языке утверждения обычно пишутся на отдельных строках, а для обозначения блоков кода используются отступы.

Например, следующий код создает переменную и присваивает ей значение:

```
x = 33
```

Знак равенства (=) используется для присвоения значения переменной. Переменные предназначены для хранения значений, которые могут быть использованы в программе позже.

Python поддерживает несколько типов данных, включая целые числа, числа с плавающей точкой, строки и булевы значения. Целочисленные данные — это целые числа (например, 1, 2, 3), числа с плавающей точкой — это десятичные числа (например, 9.99, 3.1415), а строки — это последовательности символов (например, «робот», «хозяин»). Булевы значения — `True` или `False`.

Операторы используются для выполнения операций над переменными и значениями. Python поддерживает несколько операторов, включая арифметические операторы (+, –, *, /), операторы сравнения (==, !=, <, >) и логические операторы (and, or, not). В следующем коде для выполнения вычислений используются арифметические операторы:

```
x = 30
y = 3
z = x + y
print(z) # Вывод: 33
```

В этом примере знак «плюс» (+) используется для сложения значений `x` и `y`, а результат сохраняется в переменной `z`. Затем функция `print` используется для вывода значения `z` на экран.

Изучив синтаксис, переменные, типы данных и операторы, вы сможете писать простые программы на Python, выполняющие полезные задачи. В следующем разделе рассмотрим управляющие структуры, которые позволяют управлять работой программы на основе условий и циклов.

Запуск программ на Python

После написания программы на Python ее нужно запустить, чтобы увидеть результаты. В этом разделе рассмотрим способы запуска программ.

Самый простой способ — использовать интерпретатор Python, о чем говорилось в предыдущем разделе. Откройте окно командной строки или терминала, перейдите в каталог, где сохранена программа на Python, и введите `python program.py`

(замените `program.py` именем файла вашей программы). Интерпретатор Python выполнит код, содержащийся в файле, и выведет результат.

Другой способ — использовать интегрированную среду разработки (IDE), например PyCharm или Visual Studio Code. IDE предоставляет более совершенную среду для написания, отладки и тестирования кода. Чтобы запустить программу в IDE, откройте файл программы и воспользуйтесь командой IDE `run` или `execute`.

Если вы пишете программу для веб-приложения или сервера, может понадобиться запустить программу с помощью веб-сервера, например Apache или Nginx. В этом случае для обработки запросов и ответов от веб-сервера обычно используется веб-фреймворк, например Django или Flask.

Независимо от того, как вы решите запустить свою программу, важно тщательно протестировать ее и убедиться, что она работает так, как ожидается. Это включает в себя тестирование пограничных состояний, обработку ошибок и исключений, а также проверку входных и выходных данных.

Основные методы отладки

Отладка — важная часть программирования, поскольку позволяет находить и исправлять ошибки в коде. В этом разделе рассмотрим некоторые основные методы отладки, которые можно использовать для выявления и устранения проблем в программах на Python.

Первый шаг в отладке — понять суть проблемы. Часто это включает в себя изучение выходных данных программы и поиск ошибок или неожиданного поведения. Можно использовать ведение журналов и операторы `print`, чтобы вывести информацию о состоянии программы в разных местах кода.

Следующий шаг после определения проблемы — изолировать ее причину. Для этого часто используется процесс исключения, чтобы определить части кода, вызывающие проблему. Можно использовать точки останова и пошаговую отладку, чтобы построчно изучить код и увидеть, как изменяются переменные и значения.

Помимо изучения кода, можно использовать сообщения об ошибках и трассировку стека, чтобы точно определить местоположение проблемы. Сообщения об ошибках предоставляют информацию о типе и месте ошибки, а трассировка стека показывает последовательность вызовов функций, которые привели к ошибке.

После выявления причины проблемы ее нужно устранить. Для этого вносят изменения в код и снова тестируют программу, чтобы убедиться, что проблема решена. Можно использовать инструменты автоматизированного тестирования

и код-ревью, чтобы убедиться, что изменения не приведут к появлению новых проблем.

Стиль кода Python и лучшие практики

Стиль кода и лучшие практики важны для написания читабельного, удобного и эффективного кода Python. В этом разделе рассмотрим ключевые принципы стиля и лучшие практики.

Во-первых, при написании кода на Python важно придерживаться последовательного стиля. PEP 8 — официальное руководство по стилю Python — содержит рекомендации по оформлению кода, отступам, соглашениям об именовании и т. д. Придерживаясь единого стиля, вы можете писать простой в чтении и понимании код.

Важно писать модульный и переиспользуемый код. Это предполагает разбиение кода на небольшие, независимые функции или модули, которые можно использовать в разных частях программы. Так вы облегчите тестирование и отладку кода, а также снизите риск ошибок.

Используйте описательные имена переменных и комментарии для пояснения кода. Это облегчает другим разработчикам понимание вашего кода и снижает риск путаницы или ошибок.

Тщательно тестируйте код, чтобы убедиться, что все работает так, как ожидается. Это включает использование инструментов автоматизированного тестирования и написание юнит-тестов для проверки отдельных функций или модулей.

Наконец, следите за последними возможностями Python и лучшими практиками. Сообщество Python постоянно развивается, регулярно выпускаются новые функции и инструменты. Оставаясь в курсе последних событий, вы сможете писать более эффективный и действенный код.

Ресурсы для изучения Python

Python — популярный и широко используемый язык программирования, есть множество ресурсов для его изучения и освоения. Рассмотрим некоторые из лучших ресурсов для изучения Python: от онлайн-туториалов и курсов до книг и документации.

Онлайн-туториалы и курсы — отличный способ начать изучение Python. Они обеспечивают структурированную среду обучения и часто включают интерактивные примеры и упражнения. Популярные онлайн-ресурсы для изучения Python — Codecademy, Coursera, Udemy и edX.

Книги — это тоже отличный вариант, поскольку они дают всестороннее и глубокое представление о языке и его возможностях. Вот некоторые популярные книги для изучения Python: *Python Crash Course*[1] Эрика Мэтиза, *Python for Everybody* Чарльза Северанса и *Learning Python* Марка Лутца.

Документация — еще один важный ресурс для изучения, поскольку в ней содержится подробная информация о языке и его возможностях. Официальная документация Python доступна на сайте docs.python.org, и это всеобъемлющий ресурс для разработчиков Python всех уровней.

Наконец, сообщество Python — это ценный ресурс для освоения языка. Есть множество групп пользователей Python и онлайн-форумов, где разработчики общаются друг с другом, делятся знаниями и опытом, а также получают помощь по конкретным вопросам.

[1] *Мэтиз Э.* Изучаем Python. Программирование игр, визуализация данных, веб-приложения. — СПб.: Питер, 2016.

Глава 2

ПЕРЕМЕННЫЕ, ТИПЫ ДАННЫХ И ОПЕРАТОРЫ

Переменные — важная часть программирования на Python, они позволяют хранить значения и манипулировать ими в коде. *Переменная* — это, по сути, именованный контейнер, в котором хранится значение. Оно может быть числом, строкой или любым другим типом данных.

Чтобы создать переменную в Python, достаточно выбрать имя переменной и присвоить ей значение с помощью знака равенства (`=`). Например, следующий код создает переменную с именем `x` и присваивает ей значение `33`:

```
x = 33
```

В этом коде `x` — имя переменной, а `33` — значение, которое ей присваивается. После создания переменной ее можно использовать в коде для выполнения вычислений, хранения результатов или манипулирования данными.

Имена переменных в Python должны следовать определенным правилам и соглашениям. Например, имена переменных не могут начинаться с цифры и содержать пробелы. Хорошей практикой будет использование описательных имен переменных, которые отражают цель или смысл хранимого значения.

Переменные могут хранить различные типы данных в Python, включая числовые типы данных, — целые числа и числа с плавающей точкой, а также строковые типы и булевы значения. Далее рассмотрим некоторые распространенные типы данных в Python и работу с ними.

Соглашения об именовании переменных

Имена переменных в Python должны соответствовать определенным соглашениям, чтобы код был удобочитаемым и понятным другим разработчикам. Ниже приведены основные соглашения об именах, которым нужно следовать при создании переменных в Python.

1. Используйте описательные имена: имена переменных должны отражать цель или смысл хранимого значения. Это облегчает другим разработчикам понимание кода.
2. Используйте строчные буквы: в Python имена переменных следует писать строчными буквами, разделяя слова подчеркиванием. Например, `my_variable_name` — это правильное имя переменной в Python.
3. Избегайте использования зарезервированных слов: в Python есть набор зарезервированных слов, которые имеют особое значение в языке, например `if`, `else` и `while`. Не используйте эти слова в качестве имен переменных.
4. Используйте верблюжий регистр для имен классов: если вы создаете класс, имя должно быть написано в верблюжьем регистре, с заглавной первой буквой в каждом слове. Например, `MyClassName` — это правильное имя класса в Python.
5. Будьте последовательны: последовательность — важный момент в именовании переменных. Используйте одинаковые соглашения об именовании во всем коде, чтобы его было легко читать и понимать.

Основные типы данных

Python поддерживает несколько основных типов данных, включая числовые, строковые и булевы значения. Вот краткий обзор некоторых типов данных.

1. Целочисленные значения: целые числа, 1, 2, 3 и т. д. Они представлены с помощью типа данных `int`.
2. Числа с плавающей точкой: десятичные числа, 9,99 или 3,1415. Они представлены с помощью типа данных `float`.
3. Строка: последовательность символов, например `"робот"` или `"хозяин"`. Они представлены с помощью типа данных `str`.
4. Булевы значения: `True` или `False`. Они представлены с помощью типа данных `bool`.
5. `None`: специальный тип данных, который представляет отсутствие значения. Он часто используется для представления переменных, которые еще не инициализированы или не имеют значения.

В Python переменные могут содержать различные типы данных в зависимости от значения, которое им присваивается. Например, следующий код создает две переменные, `x` и `y`, и присваивает им значения `33` и `3.1415` соответственно:

```
x = 33
y = 3.1415
```

В этом коде `x` — целочисленная переменная, а `y` — переменная с плавающей точкой.

Далее рассмотрим, как работать с этими типами данных, выполнять вычисления и манипулировать строками.

Числовые типы данных

Числовые типы данных используются для представления чисел. Python поддерживает несколько числовых типов данных, включая целые числа, числа с плавающей точкой и комплексные числа.

Целочисленные значения — это целые числа, такие как 1, 2, 3. Они представлены с помощью типа данных `int`. Над целыми числами можно выполнять арифметические операции: сложение (+), вычитание (–), умножение (*) и деление (/). Например, следующий код выполняет несколько основных арифметических операций над целыми числами:

```
x = 30
y = 3
print(x + y) # Вывод: 33
print(x – y) # Вывод: 27
print(x * y) # Вывод: 90
print(x / y) # Вывод: 10.0
```

Знак «плюс» (`+`) используется для сложения значений `x` и `y`, знак «минус» (`-`) — для вычитания значения `y` из `x`, звездочка (`*`) — для умножения, а слеш (`/`) — для деления `x` на `y`.

Числа с плавающей точкой — это десятичные числа. Они представлены с помощью типа данных `float`. Вы можете выполнять арифметические операции над числами с плавающей точкой так же, как и над целыми числами. Однако из-за того, как числа с плавающей точкой представлены в памяти, вы можете столкнуться с ошибками округления или неточностями в вычислениях.

Комплексные числа — это числа, которые имеют как действительную, так и мнимую часть. Они представлены с помощью типа данных `complex`. Вы можете создать комплексное число, указав действительную и мнимую части с помощью суффикса `j`. Например, следующий код создает комплексное число:

```
z = 30 + 3j
```

В этом коде `z` — комплексное число с действительной частью `30` и мнимой частью `3`.

Строковый тип данных

Строка — это последовательность символов, например `"робот"` или `"хозяин"`. В Python строки представлены с помощью типа данных `str`. Вы можете создать

строку, заключив текст в одинарные или двойные кавычки. Например, следующий код создает строку:

```
string1 = "Привет, хозяин!"
```

В этом коде `string1` — это строковая переменная, которая содержит значение `Привет, хозяин!`.

Строки в Python рассматриваются как объекты, это означает, что над ними можно выполнять операции и методы. Вот некоторые общие операции и методы, которые можно использовать со строками в Python:

- **Конкатенация:** можно объединить две строки или более вместе с помощью оператора «плюс» (`+`). Следующий код объединяет две строки:

```
first_name = "Петр"
last_name = "Левашов"
full_name = first_name + " " + last_name
print(full_name) # Вывод: Петр Левашов
```

- **Длина:** можно найти длину строки с помощью функции `len()`. Следующий код определяет длину строки:

```
string2 = "Привет, хозяин!"
length = len(string2)
print(length) # Вывод: 14
```

- **Индексирование:** можно получить доступ к отдельным символам в строке с помощью индексации. Индексация в Python начинается с `0`, это означает, что первый символ в строке имеет индекс `0`. Следующий код получает доступ ко второму символу в строке:

```
string3 = "Привет, робот!"
второй_знак = string3[1]
print(second_char) # Вывод: е
```

- **Нарезка:** можно извлечь подстроку из строки с помощью нарезки (слайсинга). Нарезка позволяет указать диапазон индексов для извлечения из строки. Следующий код извлекает подстроку из строки:

```
string4 = "Робот готов!"
substring = string4[0:5]
print(substring) # Вывод: Робот
```

Булев тип данных

Булево значение — это значение, являющееся либо истинным, либо ложным. Они представлены с помощью типа данных `bool`. Булевы значения часто используются в программировании для представления условий или состояний.

Например, можно использовать булево значение для представления того, является условие истинным или ложным. Следующий код создает булеву переменную `is_robot` и присваивает ей значение `True`:

```
is_robot = True
```

В этом коде `is_robot` — булева переменная, которая имеет значение `True`.

В Python можно выполнять булевы операции с помощью следующих операторов:

- **Логическое И:** логический оператор `AND` (И) возвращает значение `True`, если оба операнда равны `True`. Например:

  ```
  a = True
  b = False
  print(a and b) # Вывод: False
  ```

- **Логическое ИЛИ:** логический оператор `OR` (ИЛИ) возвращает `True`, если хотя бы один операнд равен `True`. Например:

  ```
  a = True
  b = False
  print(a or b) # Вывод: True
  ```

- **Логическое НЕ:** логический оператор `NOT` (НЕ) возвращает противоположный операнд. Например:

  ```
  a = True
  print(not a) # Вывод: False
  ```

Булевы значения также можно сравнивать с помощью операторов сравнения, таких как оператор равенства (`==`) или неравенства (`!=`). Например:

```
x = 30
y = 3
print(x == y) # Вывод: False
print(x != y) # Вывод: True
```

В этом коде оператор `==` возвращает `False`, потому что `x` не равен `y`, а оператор `!=` возвращает `True`, потому что `x` не равен `y`.

Приведение типов

Приведение, или преобразование, типов — это процесс преобразования одного типа данных в другой. В Python преобразование типов можно выполнять с помощью встроенных функций, предназначенных для каждого типа данных.

Например, можно преобразовать строку в целое число с помощью функции `int()`. Следующий код преобразует строковую переменную `string5` в целое число:

```
string5 = "33"
int5 = int(string5)
```

В этом коде `string5` — это строковая переменная, которая содержит значение `33`, а `int5` — целочисленная переменная, которая содержит преобразованное значение.

Аналогично можно преобразовать целое число в строку с помощью функции `str()`. Следующий код преобразует целочисленную переменную `int6` в строку:

```
int6 = 33
string6 = str(int6)
```

В этом коде `int6` — это целочисленная переменная, которая содержит значение `33`, а `string6` — строковая переменная, которая содержит преобразованное значение.

Можно выполнять преобразование типов между числовыми типами данных, например преобразовать целое число в число с плавающей точкой с помощью функции `float()`.

Важно отметить, что не все преобразования типов являются действительными или осмысленными. Например, вы не сможете преобразовать строку, содержащую буквы, в целое число, поскольку она не имеет числового значения.

Преобразование типов в Python поможет манипулировать типами данных и преобразовывать их по мере необходимости. При этом важно знать ограничения и правила преобразования типов, чтобы избежать ошибок и неожиданного поведения в коде.

Арифметические операторы

Арифметические операторы используются для выполнения основных математических операций. Python поддерживает следующие арифметические операторы.

1. Сложение (`+`): складывает два значения вместе.
2. Вычитание (`-`): вычитает одно значение из другого.
3. Умножение (`*`): перемножает два значения вместе.
4. Деление (`/`): делит одно значение на другое.
5. Получение остатка от деления (`%`): возвращает остаток от операции деления.
6. Возведение в степень (`**`): возводит одно значение в степень другого значения.

Следующий код использует арифметические операторы для выполнения некоторых базовых вычислений:

```
x = 30
y = 3
# Сложение
result1 = x + y
print(result1) # Вывод: 33
```

```
# Вычитание
result2 = x – y
print(result2) # Вывод: 27
# Умножение
result3 = x * y
print(result3) # Вывод: 90
# Деление
result4 = x / y
print(result4) # Вывод: 10.0
# Остаток от деления
result5 = x % y
print(result5) # Вывод: 0
# Возведение в степень
result6 = x ** y
print(result6) # Вывод: 27000
```

В этом коде переменные `x` и `y` содержат значения `30` и `3` соответственно. Арифметические операторы используются для выполнения операций сложения, вычитания, умножения, деления, получения остатка от деления и возведения в степень. Результаты сохраняются в разных переменных.

Операторы сравнения

Операторы сравнения используются для сравнения двух значений и возвращают булево значение (`True` или `False`) на основе результата сравнения. Python поддерживает следующие операторы сравнения.

1. Равно (`==`): возвращает `True`, если два значения равны.
2. Не равно (`!=`): возвращает `True`, если два значения не равны.
3. Больше чем (`>`): возвращает `True`, если первое значение больше второго.
4. Меньше чем (`<`): возвращает `True`, если первое значение меньше второго.
5. Больше или равно (`>=`): возвращает `True`, если первое значение больше или равно второму.
6. Меньше или равно (`<=`): возвращает `True`, если первое значение меньше или равно второму.

В следующем коде используются операторы для сравнения двух значений и возврата булева значения:

```
x = 30
y = 3
# Равно
result1 = x == y
print(result1) # Вывод: False
# Не равно
result 2 = x != y
print(result2) # Вывод: True
# Больше, чем
result 3 = x > y
print(result3) # Вывод: True
```

```
# Меньше, чем
result 4 = x < y
print(result4) # Вывод: False
# Больше или равно
result 5 = x >= y
print(result5) # Вывод: True
# Меньше или равно
result6 = x <= y
print(result6) # Вывод: False
```

В этом коде переменные `x` и `y` имеют значения `30` и `3` соответственно. Операторы сравнения используются для сравнения значений и возвращают булево значение, основанное на результате сравнения.

Логические операторы

Логические операторы используются для выполнения логических операций над булевыми значениями. Python поддерживает следующие логические операторы.

1. Логическое И (AND): возвращает значение `True`, если оба операнда равны `True`.
2. Логическое ИЛИ (OR): возвращает `True`, если хотя бы один операнд равен `True`.
3. Логическое НЕ (NOT): возвращает противоположный операнд.

Например, в следующем коде логические операторы используются для выполнения логических операций над булевыми значениями:

```
a = True
b = False
# Логическое И
result = a and b
print(result) # Вывод: False
# Логическое ИЛИ
result = a or b
print(result) # Вывод: True
# Логическое НЕ
result = not a
print(result) # Вывод: False
```

В этом коде переменные `a` и `b` содержат булевы значения. Логические операторы используются для выполнения логических операций над этими значениями и возвращают булево значение, основанное на результате операции.

Можно использовать круглые скобки для группировки логических операций и управления порядком вычисления:

```
a = True
b = False
c = True
# Сгруппированные логические операции
result = (a and b) or c
print(result) # Вывод: True
```

Здесь круглые скобки используются для группировки логической операции AND между `a` и `b` и логической операции OR между результатом операции AND и переменной `c`.

Приоритет и ассоциативность операторов

Приоритет и ассоциативность операторов определяют порядок, в котором в Python выполняются операторы. Когда выражение содержит несколько операторов, Python использует правила приоритета операторов и ассоциативности, чтобы определить, какой оператор выполняется первым и как вычисляется выражение.

Приоритет операторов определяет порядок оценки операторов с разными приоритетами. Python следует тем же правилам, что и большинство языков программирования, где умножение, деление и получение остатка от деления имеют более высокий приоритет, чем сложение и вычитание. Например, в выражении `1 + 3 * 33` Python сначала делает умножение, а затем сложение, в результате чего получается значение `100`.

Ассоциативность операторов определяет порядок выполнения операторов с одинаковым приоритетом. Для большинства операторов Python следует ассоциативности слева направо, это означает, что операторы с одинаковым приоритетом выполняются слева направо. Например, в выражении `33 – 3 – 3` сначала выполняется первая операция вычитания (в результате получается `30`), а затем вторая (в результате получается `27`).

Некоторые операторы имеют ассоциативность справа налево, например оператор возведения в степень (`**`). Это означает, что выражение `2 ** 3 ** 2` выполняется как `2 ** (3 ** 2)`, в результате чего получается `512`.

Важно понимать правила приоритета и ассоциативности операторов в Python, чтобы избежать ошибок и неожиданного поведения в коде. Используйте круглые скобки, чтобы контролировать порядок вычислений и отменять правила по умолчанию. Например, выражение `(33 – 3) * 4` даст значение `120`, поскольку сложение внутри круглых скобок выполняется первым.

Задания для самопроверки

1. Назовите основные типы данных в Python. Приведите пример каждого из них.
2. Выберите подходящие имена переменных для следующих описаний:
   а) имя пользователя;
   б) общее количество товаров в корзине;
   в) цена одного товара.