Понятный Python

С наглядным кодом, для детей и начинающих

Артем Демиденко / ИИ

С подробным разбором кода

Артем Демиденко
Артем Демиденко
Искусственный Интеллект

Понятный Python. С наглядным кодом, для детей и начинающих

http://www.litres.ru/pages/biblio_book/?art=69576925
SelfPub; 2023

Аннотация

Это увлекательная и понятная книга о Python, созданная специально для начинающих взрослых и их детей. Эта книга предлагает простой и доступный путь в мир программирования, где даже самые молодые читатели могут освоить основы.Примеры кода для детей: Книга содержит множество примеров кода, которые легко понять и объяснить детям. Они иллюстрируют концепции программирования через интересные истории и задачи.Каждая глава вводит важные концепции программирования, такие как переменные, условия, циклы и функции, используя язык, понятный и детям, и начинающим взрослым.

Содержание

Артем Демиденко Понятный Python. С наглядным кодом, для детей и начинающих

Глава 1: Знакомство с Python

В первой главе нашей книги мы погрузимся в мир Python и научимся устанавливать его на ваш компьютер. Давайте начнем с самого начала!

В данной книге в примерах кода стоят .... они показывают количество пробелов, необходимое сделать перед тем или иным участком кода. Это сделано из-за особенностей публикации книги на платформе. В реальности вам надо заменить .... на 4 пробела.

1.1: Что такое Python?

Python – это компьютерный язык программирования. Давайте разберемся, что это значит и почему Python так важен.

Компьютерный Язык Программирования

Для начала, давайте представим, что компьютеры – это мощные машины, но они не могут понимать наш обычный язык, такой как английский или русский. Они понимают

только специальные инструкции, которые называются "кодом" или "программами". Вот где на помощь приходят компьютерные языки программирования, такие как Python.

Простота и Понятность

Одной из особенностей Python является его простота и понятность. В Python, код выглядит очень похоже на то, как мы общаемся друг с другом. Вот пример:

print("Привет, мир!")

Этот код говорит компьютеру вывести на экран фразу "Привет, мир!". Вы можете видеть, что он понятен даже без глубоких знаний программирования.

Популярность Python

Python очень популярен среди программистов по всему миру. Это потому, что он удобен для решения разных задач. Его используют для создания веб-сайтов, разработки приложений, анализа данных, и даже для искусственного интеллекта и машинного обучения.

Так что же такое Python? Это простой и понятный язык программирования, который помогает нам говорить с компьютером и делать интересные вещи. В этой книге мы будем изучать Python, чтобы вы тоже могли создавать свои собственные программы и приключения в мире компьютерного программирования!

1.2: Установка Python

Прежде чем начать программировать на Python, вам необходимо установить этот язык программирования на свой

компьютер. Ниже приведены подробные инструкции по установке Python на различные операционные системы:

Для Windows:

1. Перейдите на официальный сайт Python, используя ваш веб-браузер. Адрес сайта: https://www.python.org/downloads/windows/.
2. Выберите версию Python, которую хотите установить. Рекомендуется скачать последнюю стабильную версию (обычно она будет отображаться в верхней части страницы).
3. Скачайте установочный файл Python для Windows. Обратите внимание, что у вас может быть выбор между 32-битной и 64-битной версией. Если у вас 64-разрядная версия Windows, лучше выбрать 64-битную версию Python.
4. Запустите скачанный установочный файл Python. Убедитесь, что установщик имеет галочку "Добавить Python X.Y в PATH" (где X.Y – версия Python, например, 3.8). Это важно для того, чтобы вы могли запускать Python из командной строки.
5. Нажмите кнопку "Установить сейчас" и следуйте инструкциям на экране. Установка Python начнется и может занять несколько минут.
6. После завершения установки Python будет готов к использованию на вашем компьютере.

Для macOS:

1. Откройте терминал (Terminal) на вашем Mac. Вы можете найти его в папке "Утилиты" (Utilities), которая на-

ходится внутри папки "Приложения" (Applications).

2. В терминале введите следующую команду и нажмите "Enter":

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Эта команда устанавливает Homebrew, менеджер пакетов, который поможет установить Python.

3. После установки Homebrew, введите следующую команду и нажмите "Enter", чтобы установить Python:

brew install python

Это установит Python на вашем Mac.

4. Для проверки успешной установки Python введите:

python3 –version

Вы должны увидеть версию Python.

Для Linux (Ubuntu):

1. Откройте терминал (командную строку).

2. В терминале выполните следующие команды:

sudo apt update sudo apt install python3

Эти команды обновят список пакетов и установят Python 3 на вашем Linux-компьютере.

3. Проверьте версию Python с помощью команды:

python3 –version

Это покажет установленную версию Python.

После завершения установки Python на вашем компьютере, вы будете готовы начать программировать на этом языке. Python успешно установлен, и теперь вы можете переходить

к созданию своих первых программ на Python.

1.3: Знакомство с интерфейсом Python

После успешной установки Python, мы можем начать использовать его интерактивный интерфейс. Этот интерфейс позволяет нам взаимодействовать с Python непосредственно и выполнять команды в режиме реального времени. Давайте более подробно рассмотрим, как работать с ним.

1. **Запуск Python интерпретатора**:

• **Windows**: Откройте меню "Пуск" и найдите программу "Python" или "IDLE (Python)". Запустите ее.

• **macOS**: Откройте "Terminal" (программа Терминал) и просто введите **python3** и нажмите "Enter". Иногда вам может потребоваться ввести **python** вместо **python3**, в зависимости от версии Python.

• **Linux**: Откройте терминал и введите **python3** или **python** в зависимости от вашей системы.

2. **Использование интерактивного режима** :

Когда вы запустите интерпретатор Python, вы увидите приглашение, похожее на **>>>**. Это место, где вы можете вводить команды Python.

Например, попробуйте ввести следующую команду:

print("Привет, мир!")

После нажатия "Enter" вы увидите вывод **"Привет, мир!"** непосредственно под вашей командой.

3. **Простой расчет**:

Python можно использовать как калькулятор. Попробуйте

выполнить арифметические операции:

2 + 3

Вы увидите результат **5**.

4. **Выход из интерпретатора**:

Чтобы выйти из интерпретатора Python и вернуться в обычный командный интерфейс, введите **exit()** или **quit()** и нажмите "Enter".

5. **Сохранение программы**:

Интерактивный режим полезен для быстрых проверок и экспериментов, но настоящие программы обычно пишутся в текстовых файлах с расширением **.py**. Вы создаете файл с кодом, сохраняете его, а затем запускаете через командную строку с помощью команды **python имя_файла.py**.

Этот интерфейс Python позволяет вам испытывать код по шагам и моментально видеть результаты выполнения каждой команды. Он будет вашим верным спутником в процессе изучения и использования Python.

1.4: Заключение

В первой главе нашей книги мы познакомились с Python, установили его на нашем компьютере и даже выполнили первую программу. Поздравляю вас с первым шагом в мир программирования!

Давайте подведем итоги того, что мы узнали:

1. **Python – это язык программирования**: Это средство, которое позволяет вам говорить с компьютером на его языке. Python легко читать и писать, что делает его отлич-

ным выбором для начинающих программистов.

2. **Установка Python**: Мы научились устанавливать Python на компьютер, что позволяет нам создавать и запускать программы.

3. **Интерфейс Python**: Мы познакомились с интерактивным интерфейсом Python, где мы можем вводить команды и видеть результаты немедленно. Это помогает нам экспериментировать и учиться.

Теперь, когда у нас есть Python на компьютере и небольшое представление о том, как им управлять, мы готовы перейти к более интересным аспектам программирования. В следующей главе мы начнем изучать базовые концепции, такие как переменные и типы данных. Эти знания помогут нам создавать более сложные программы.

Не бойтесь ошибок или неудач. Программирование – это искусство решать задачи, и ошибки – это часть этого процесса. Важно только продолжать учиться и экспериментировать.

Глава 2: Первая программа

В предыдущей главе мы познакомились с Python и установили его на наш компьютер. Теперь пришло время создать нашу первую программу! На этом этапе мы будем использовать Python как инструмент для написания команд, которые компьютер будет выполнять.

Наша Первая Программа

Давайте начнем с чего-то простого. Мы создадим программу, которая выводит на экран слова "Привет, мир!". Это традиционный способ начать изучение программирования.

print("Привет, мир!")

Разбор:

• **print** – это функция Python, которая выводит текст на экран.

• **"Привет, мир!"** – это строка, которую мы хотим вывести. Строки обычно заключаются в двойные кавычки.

Сохранение и Запуск

Теперь, когда у нас есть программа, давайте научимся ее сохранять и запускать.

Сохранение Программы

1. Откройте текстовый редактор на вашем компьютере. Это может быть Notepad, Visual Studio Code, PyCharm или другой редактор по вашему выбору.

2. Вставьте код программы в текстовый редактор:

print("Привет, мир!")

3. Сохраните файл с расширением **.py**. Назовите его, например, **hello.py**.

Запуск Программы

Теперь, когда наш файл сохранен, давайте научимся его запускать.

1. Откройте командную строку (на Windows она может называться "Командная строка" или "PowerShell", на MacOS и Linux – "Терминал").

2. Перейдите в каталог, где находится ваш файл **hello.py**. Это можно сделать с помощью команды **cd** (change directory). Например:

cd Путь\к\каталогу

3. Когда вы находитесь в каталоге с файлом **hello.py**, выполните команду:

python hello.py

Вы увидите на экране:

Привет, мир!

Python выполнил нашу программу последовательно, начиная с верхней строки и двигаясь вниз. Поздравляю, вы только что написали, сохранили и запустили свою первую программу на Python!

Это было ваше первое знакомство с написанием и выполнением программ на Python. Не волнуйтесь, если что-то пока кажется непонятным. С практикой и дальнейшим изучением вы будете понимать Python лучше.

Глава 3: Переменные и Типы Данных

Что такое переменные?

В программировании переменные – это контейнеры, которые позволяют вам хранить и работать с данными. Мы можем представить переменные как ярлыки, на которых написано имя, и эти ярлыки можно прикрепить к разным вещам. Например, мы можем создать переменную с именем "возраст" и присвоить ей значение, такое как 10. Теперь мы знаем, что возраст равен 10, и мы можем использовать это значение в наших программах.

Создание переменных в Python

В Python, создание переменных очень просто. Вы можете выбрать имя для переменной (помните, что имена переменных могут содержать буквы, цифры и подчеркивания, но не могут начинаться с цифры) и затем использовать знак равенства (=) для присвоения значения этой переменной. Например:

возраст = 10

Теперь у нас есть переменная "возраст", которая содержит значение 10.

Типы Данных

В Python существует несколько различных типов данных,

и они определяют, какие виды значений может хранить переменная. Вот некоторые из наиболее распространенных типов данных:

1. **Целые числа (integers)**: Этот тип данных используется для хранения целых чисел, например, 10, -5, 1000 и так далее.

2. **Десятичные числа (floats)**: Этот тип данных используется для хранения чисел с плавающей точкой, таких как 3.14, -0.5, 2.0 и другие числа с десятичной точкой.

3. **Строки (strings)**: Строки используются для хранения текста. Например, "Привет, мир!" – это строка.

4. **Булевы значения (booleans)**: Булевы значения могут быть только двух видов: True (истина) или False (ложь). Они используются для представления логических условий.

Примеры использования переменных и типов данных

Давайте посмотрим на примеры использования переменных и типов данных:

имя = "Алиса"
возраст = 12
рост = 150.5
прошел_тест = True

В этом примере мы создали четыре переменные:

- "имя" содержит строку "Алиса".
- "возраст" содержит целое число 12.
- "рост" содержит число с плавающей точкой 150.5.

- "прошел_тест" содержит булевое значение True.

Изменение Значения Переменных

Вы можете изменять значения переменных, просто присваивая им новые значения. Например:

возраст = 13

Теперь значение переменной "возраст" равно 13, а не 12.

Вывод Значений переменных

Чтобы увидеть значения переменных, вы можете использовать функцию **print()**. Например:

print(имя)

print(возраст)

Этот код выведет значения переменных "имя" и "возраст" на экран.

В этой главе мы узнали, что такое переменные, как их создавать, какие существуют типы данных, и как изменять значения переменных. Понимание переменных и типов данных – это фундаментальная часть программирования, и это позволяет нам хранить и манипулировать информацией в наших программах.

Глава 4: Операторы и Выражения

1: Арифметические операторы

Арифметические операторы – это специальные символы в Python, которые используются для выполнения математических операций. Вот некоторые из них:

Сложение (+): Оператор сложения используется для сложения двух чисел. Например:

```
результат = 5 + 3
print(результат) # Выведет: 8
```

Вычитание (-): Оператор вычитания используется для вычитания одного числа из другого. Например:

```
результат = 7 – 2
print(результат) # Выведет: 5
```

Умножение (*): Оператор умножения используется для умножения двух чисел. Например:

```
результат = 4 * 6
print(результат) # Выведет: 24
```

Деление (/): Оператор деления используется для деления одного числа на другое. Важно помнить, что результат деления всегда будет числом с плавающей точкой (десятичной дробью). Например:

```
результат = 8 / 2
print(результат) # Выведет: 4.0
```

Деление по модулю (%): Оператор деления по моду-

лю возвращает остаток от деления одного числа на другое. Представь, у тебя есть 10 конфет, и ты хочешь поделить их между тремя своими друзьями. Ты делишь конфеты и видишь, что у тебя осталась 1 конфета. Это деление по модулю! Например:

остаток = 10 % 3

print(остаток) # Выведет: 1

Операция деления по модулю – это как посчитать остаток, который остается после того, как мы поделили одно число на другое.

2: Операторы сравнения

Операторы сравнения используются для сравнения двух значений. Они всегда возвращают либо **True** (истина), либо **False** (ложь). Вот некоторые операторы сравнения:

Равно (==) – это когда две вещи идентичны:

Представь, у тебя есть две машины, и они выглядят точно одинаково: у них одинаковые цвета, одинаковые колеса и одинаковые фары. Ты можешь сказать, что эти две машины **равны**, потому что они одинаковы по своему внешнему виду.

Теперь давай посмотрим, как это работает на Python:

машина1 = "красная, 4 колеса"

машина2 = "красная, 4 колеса"

если_равны = (машина1 == машина2)

если_равны будет True, потому что машина1 и машина2 одинаковы по своему описанию – обе красные и имеют 4 колеса.

Так что оператор "Равно (==)" используется, чтобы сравнивать две вещи и убедиться, что они идентичны, как две одинаковые машины. Если они идентичны, то результатом сравнения будет "True" (истина); если нет, то "False" (ложь).

"Не равно" (!=) – это как сравнение игрушек:

Допустим, у тебя и у твоего друга есть коробки с игрушками. Ты хочешь узнать, разные ли у вас игрушки или одинаковые. Если игрушки в твоей коробке не такие же, как у друга, то ты говоришь "не равно"!

Давай на Python создадим пример:

```
твои_игрушки = ["машинка", "кукла", "мяч"]
другие_игрушки = ["машинка", "кубики", "мяч"]

если твои_игрушки != другие_игрушки:
....print("Твои игрушки не такие же, как у друга!")
else:
....print("Твои игрушки такие же, как у друга!")
```

В этом примере мы используем оператор "не равно" (**!=**) для сравнения твоих игрушек и игрушек друга. Если они не одинаковые (то есть, хотя бы одна игрушка разная), мы выводим сообщение, что у вас разные игрушки.

Итак, оператор "не равно" (**!=**) используется для сравнения двух вещей и говорит нам, разные ли они или нет.

Больше (>) – это как игра в сравнение чисел. Давай представим, что у нас есть два числа, скажем, 7 и 3. Оператор "больше (>)" помогает нам определить, кто из них больше, как в конкурсе.

Если мы спрашиваем Python, "7 больше, чем 3?" – Python ответит "Да!" и покажет нам True (правда). Это как сказать, что "7 – это больше, чем 3!"

Таким образом, оператор "больше (>)" помогает нам сравнивать числа и определять, какое из них больше. Если число слева от оператора больше числа справа, Python скажет "Да" и выведет True. Если не так, то Python скажет "Нет" и выведет False (ложь).

```
результат = 7 > 4
print(результат) # Выведет: True
```

Меньше (<): Так же как и больше, только наоборот. Проверяет, меньше ли одно значение, чем другое.

```
результат = 3 < 6
print(результат) # Выведет: True
```

Больше или равно (>=) – это способ сравнения двух чисел или величин, чтобы узнать, одно из них больше или равно другому.

Давай представим, что у нас есть два стакана с мармеладками. В первом стакане у нас 5 мармеладок, а во втором стакане 3 мармеладки. Мы можем использовать оператор "больше или равно (>=)" для сравнения количества мармеладок в этих стаканах.

Если мы скажем: "Мармеладок в первом стакане больше или равно, чем во втором стакане?", то это значит, что мы проверяем, есть ли в первом стакане хотя бы столько же мармеладок, сколько во втором, или даже больше.

Если это правда (если в первом стакане 5 мармеладок, а во втором 3), то оператор "больше или равно (>=)" даст нам ответ "да" или **True**. Но если бы в первом стакане было бы, например, только 2 мармеладки, то оператор дал бы нам ответ "нет" или **False**.

Так что "больше или равно (>=)" говорит нам о том, есть ли у нас достаточно или даже больше чего-то, чем в чем-то ещё.

количество_мармеладок_в_первом_стакане = 5
количество_мармеладок_во_втором_стакане = 3

если_больше_или_равно = количество_мармеладок_в_первом_стакане >= количество_мармеладок_во_втором_стакане

если_больше_или_равно:
....print("Мармеладок в первом стакане больше или равно, чем во втором стакане.")
else:
....print("Мармеладок в первом стакане меньше, чем во втором стакане.")

В этом примере сначала мы устанавливаем количество

мармеладок в первом и втором стаканах (5 и 3 соответственно). Затем мы используем оператор "больше или равно (>=)" для сравнения этих значений и сохраняем результат в переменной **если_больше_или_равно**.

Затем мы проверяем значение **если_больше_или_равно** с помощью условной конструкции **if**. Если значение равно **True**, мы выводим сообщение "Мармеладок в первом стакане больше или равно, чем во втором стакане." В противном случае **(else)**, если значение **False**, мы выводим "Мармеладок в первом стакане меньше, чем во втором стакане."

Этот код демонстрирует, как оператор "больше или равно (>=)" используется для сравнения значений в Python.

Меньше или равно (<=): Проверяет, меньше или равно ли одно значение другому.

Давайте представим ситуацию, где у вас есть яблоки и вы хотите узнать, сколько из них можно поделить между детьми так, чтобы каждый получил хотя бы одно яблоко. Вот как можно использовать оператор **<=**:

```
# Количество яблок и количество детей
количество_яблок = 10
количество_детей = 3

# Проверяем, можно ли поделить яблоки равномерно между детьми
если количество_яблок <= количество_детей:
....print("Яблок хватит на всех!")
```

else:

....print("Яблок не хватит на всех.")

В этом примере мы сравниваем **количество_яблок** с **количество_детей**. Если **количество_яблок** меньше или равно **количество_детей**, то программа выводит "Яблок хватит на всех!".

Представьте, что у вас есть 10 яблок и 3 ребенка. Поскольку 10 яблок меньше или равно 3 детям, то программа сообщит, что яблок хватит на всех.

Это простой способ использования оператора **<=** для сравнения чисел и принятия решений в коде.

3: Логические операторы

Логические операторы используются для комбинирования условий и выполняют логические операции. Вот некоторые из них:

Конец ознакомительного фрагмента.

Текст предоставлен ООО «ЛитРес».

Прочитайте эту книгу целиком, купив полную легальную версию на ЛитРес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.