АНДРЕЙ МИЛЛИАРДОВ

Python

ПЕРВЫЙ ШАГ В ПРОГРАММИРОВАНИИ
ДЛЯ ДЕТЕЙ И НАЧИНАЮЩИХ

Андрей Миллиардов

Python. Первый шаг в программировании для детей и начинающих

«Автор»
2024

Миллиардов А.

Python. Первый шаг в программировании для детей и начинающих / А. Миллиардов — «Автор», 2024

Python. Первый шаг в программировании для детей и начинающих — это практическое руководство, которое откроет дверь в мир программирования для самых юных и начинающих разработчиков. Книга объясняет основы Python простым и понятным языком, сопровождая каждую тему живыми примерами, увлекательными задачами и веселыми проектами. Читатели узнают, как создавать программы, игры и графические рисунки, а также поймут, как решать задачи с помощью кода.Эта книга подойдёт всем, кто хочет сделать свои первые шаги в программировании — взрослым и детям, которые мечтают освоить один из самых популярных языков в мире. Начните путешествие в программирование с удовольствием и легкостью!

© Миллиардов А., 2024
© Автор, 2024

Содержание

Андрей Миллиардов
Python. Первый шаг в программировании для детей и начинающих

Глава 1: Введение в Python и первый код

Знакомство с Python и что такое программирование

Что такое программирование?

Когда ты смотришь на компьютер или планшет, тебе может показаться, что они умеют делать самые разные вещи, например, показывать видео, запускать игры, писать сообщения. Но как они это делают? Ответ прост – благодаря программированию!

Программирование – это способ общения с компьютером. Чтобы компьютер сделал то, что ты хочешь, ему нужно дать четкие инструкции. Представь, что ты работаешь с роботом, и хочешь, чтобы он нарисовал тебе картину. Для этого ты должен объяснить роботу, как именно рисовать: где начать, что рисовать и в каком порядке. То же самое происходит и в программировании. Тебе нужно написать код, который будет объяснять компьютеру, как выполнять задачу.

Зачем учить Python?

Ты можешь задаться вопросом: "Почему именно Python?" На самом деле существует много разных языков программирования, например, Java, C++, JavaScript и другие. Но Python – это один из самых популярных и удобных языков для начинающих.

Python легко понять и быстро научиться. Он похож на обычный язык, который мы используем в повседневной жизни, поэтому код на Python часто читается как обычный текст. Это делает его отличным выбором для первых шагов в программировании.

С помощью Python ты сможешь создавать игры, строить веб-сайты, анализировать данные, решать задачи и даже разрабатывать программы для роботов. И самое главное – Python будет с тобой на каждом шаге, помогая решать задачи с помощью кода.Программирование с Python – это как научиться читать и писать, только вместо слов ты учишься писать инструкции для компьютера.

Пайтон (персонаж-помощник):

На экране появляется маленький дружелюбный робот с круглыми глазами, который радостно машет рукой.

Привет, новичок! Я – Пайтон, твой помощник в мире программирования. Давай вместе шаг за шагом разберемся, как начать работать с Python. Это будет увлекательное путешествие, и ты удивишься, насколько просто создавать программы, если понять основы.Пайтон:

Как установить Python на компьютер?

Но не переживай, установка Python – это несложно! Просто следуй этим шагам:Перед тем как начать, нам нужно установить Python на твой компьютер.

Открой браузер и зайди на официальный сайт Python: python.org.Перейди на сайт Python.

На главной странице сайта ты увидишь большую кнопку "Download Python". Нажми на нее, и сайт предложит скачать последнюю версию Python. Это будет самая свежая версия, и с ней тебе будет проще работать.Выбери версию Python.

После скачивания установочного файла открой его. Важно! На первом экране установщика будет маленькая галочка с надписью "Add Python to PATH". Обязательно поставь эту галочку, чтобы Python мог работать на твоем компьютере без проблем.Установи Python.

После установки Python открой командную строку (для Windows это "CMD", для Mac и Linux – терминал). Напиши в командной строке:Запуск Python.

bash

python –version

Если ты увидишь ответ с номером версии Python, значит установка прошла успешно! Теперь ты готов начать работать с Python.

Что дальше?

Теперь, когда Python установлен, давай напишем нашу первую программу. Но сначала давай немного поговорим о том, что такое код.

Что такое код? Как его запускать?

Что такое код?

Код – это набор инструкций, которые мы даем компьютеру, чтобы он выполнял определенную задачу. Когда ты пишешь код, ты как бы пишешь "рецепт", который объясняет компьютеру, как действовать. Каждая строка кода – это одна маленькая инструкция.

Например, если ты хочешь, чтобы компьютер вывел на экран слово, ты пишешь команду, которая говорит "напечатай это слово". В языке Python такая команда будет выглядеть так:

python

print("Привет, мир!")

Здесь:

print – это команда Python, которая говорит компьютеру "напечатай".

"Привет, мир!" – это то, что мы хотим напечатать на экране. Это строка текста, и она заключена в кавычки, чтобы Python знал, что это не число, а именно текст.

Как запустить код?

Чтобы запустить код, нужно выполнить несколько простых шагов:

Ты можешь использовать специальную программу, такую как IDLE, которая идет вместе с Python, или Visual Studio Code, который тоже легко установить. Эти редакторы позволяют писать код и сразу же запускать его.Открой редактор кода.

В редакторе напиши:Напиши свой код.

python

print("Привет, мир!")

После того как ты написал код, нужно его запустить. В IDLE для этого просто нажми кнопку "Run", а в Visual Studio Code используй команду "Run Python File".Запусти код.

На экране появится сообщение "Привет, мир!" – это твоя первая программа!Что увидишь?

Пайтон (персонаж-помощник):

Ну вот, ты написал свою первую программу! Это всего лишь начало. Теперь ты можешь использовать Python для создания гораздо более интересных вещей.Пайтон подмигивает и говорит:

Изменяем код и добавляем новые инструкции

Теперь, когда ты знаешь, как запускать код, давай поэкспериментируем и добавим что-то интересное!

Изменяем текст программы.

Вместо того чтобы выводить просто "Привет, мир!", давай напишем что-то свое. Открой программу и измени строку кода:

python

print("Привет, я учусь программировать!")

Теперь, когда ты запустишь программу, на экране появится твое сообщение! Ты только что сделал программу, которая выводит текст, который ты сам написал.

Давай добавим еще одну строку!

Ты можешь добавить еще один вывод, чтобы программа вывела сразу два сообщения. Пример:

python

print("Привет, я учусь программировать!")

print("Это моя первая программа на Python!")

Теперь программа напечатает два сообщения подряд! Ты только что научился работать с несколькими командами в одном коде.

Пайтон (персонаж-помощник):

Молодец! Ты уже можешь писать программы, которые выводят на экран текст. А теперь давай попробуем сделать что-то еще более интересное!Пайтон танцует радостно.

Практическое задание и что дальше?

Твоя первая задача!

Попробуй изменить свой код еще раз. Напиши программу, которая выводит твоё имя и твой возраст. Например:

python

print("Меня зовут Анна, мне 10 лет!")

Заменяй текст в кавычках на свой!

Что мы узнали в этой главе?

Мы узнали, что такое программирование и что такое Python.

Мы установили Python на компьютер и настроили все для работы.

Написали первую программу "Привет, мир!" и узнали, как её запускать.

Поняли, как использовать команду print для вывода текста на экран.

Изменили код, чтобы программа выводила наше сообщение.

Что будет дальше?

Теперь, когда ты знаешь, как писать код и запускать его, можно двигаться дальше! В следующей главе мы будем изучать переменные и типы данных. Мы узнаем, как сохранять информацию и использовать её в программе.

Пайтон (персонаж-помощник):

До встречи в следующей главе! Ты уже на пути к тому, чтобы стать настоящим программистом.Пайтон машет рукой на прощание.

Глава 2: Переменные и типы данных

Программирование – это не только про команды, которые мы пишем для компьютера, но и про то, как работать с информацией. В этой главе мы познакомимся с двумя важными концепциями, которые помогут нам создавать более сложные программы: переменныеи типы данных.

Что такое переменные?

Переменная – это место в памяти компьютера, где мы можем хранить информацию. Представь себе коробку, в которую можно положить что угодно: игрушку, книгу или даже деньги. Так вот, переменная – это такая коробка, в которой мы можем хранить данные. И точно так же, как мы можем легко менять содержимое коробки, в переменной можно хранить разные значения.

Чтобы создать переменную, нужно дать ей имя и присвоить значение. Например, давай создадим переменную с именем age, которая будет хранить твой возраст:

python
age = 10

Здесь:

age – это имя переменной.

10 – это значение, которое мы присваиваем переменной.

Теперь переменная age хранит число 10, и ты можешь использовать его в дальнейшем в программе. Например, если ты хочешь вывести на экран свой возраст, можно сделать так:

python
print(age)

Программа выведет на экран число 10, потому что именно это содержимое хранится в переменной age.

Типы данных в Python

Когда мы создаём переменные, важно понимать, с каким типом данных мы работаем. В Python есть несколько основных типов данных, которые позволяют хранить разные виды информации. Давайте рассмотрим три самых важных типа данных, с которыми мы будем работать.

Числа (integers и floats)

Числа – это один из самых распространенных типов данных. В Python мы можем работать как с целыми числами (например, 10, 100, 2024), так и с дробными числами (например, 3.14, 0.5, 2.7).

Целые числа (integers) – это числа, которые не имеют десятичных знаков. Пример: 1, -5, 2023.

Дробные числа (floats) – это числа, у которых есть десятичная точка. Пример: 3.14, 9.99, -0.5.

Пример использования чисел:

python
age = 10 # переменная age хранит целое число
height = 1.75 # переменная height хранит дробное число
print(age)
print(height)

Строки (strings)

Строки – это тип данных, который хранит текст. Строки заключаются в кавычки, и могут быть как одиночными (одинарные кавычки), так и двойными (двойные кавычки).

Пример строки:

python
name = "Анна" # строка с двойными кавычками
city = 'Москва' # строка с одинарными кавычками
print(name)
print(city)

Обратите внимание, что Python не делает различий между одинарными и двойными кавычками. Ты можешь использовать любые, главное – чтобы они были одинаковыми с обеих сторон строки.

Булевы значения (booleans)

Булевы значения – это тип данных, который может быть либо True (истина), либо False(ложь). Этот тип данных часто используется для проверки условий, например, в играх или логических задачах.

Пример:
python
is_sunny = True # переменная, которая хранит логическое значение
is_raining = False # переменная, которая хранит противоположное значение
print(is_sunny)
print(is_raining)

Булевы значения полезны, когда нужно принимать решения, например, если на улице солнечно, то мы идём гулять.

Работа с переменными и типами данных: практические примеры

Теперь, когда мы понимаем, что такое переменные и типы данных, давай рассмотрим несколько примеров, которые помогут нам лучше понять, как с ними работать.

Пример с числами

Предположим, что ты хочешь узнать свой возраст через 5 лет. Для этого можно создать переменную с текущим возрастом и прибавить к ней 5:

python
age = 10
future_age = age + 5
print(future_age)

В этом примере:

Мы создаем переменную age, в которой храним твой текущий возраст.

Создаем новую переменную future_age, в которой вычисляем твой возраст через 5 лет.

Используем оператор +, чтобы прибавить 5 к текущему возрасту и записать результат в новую переменную.

После того как программа выполнится, на экране появится результат: 15 – твой возраст через 5 лет.

Пример с текстами

Теперь давай создадим программу, которая будет составлять приветственное сообщение с использованием переменных. Мы создадим две переменные: одну с именем, другую – с городом, и объединим их в строку:

python
name = "Анна"
city = "Москва"
message = "Привет, " + name + "! Ты живешь в городе " + city + "."
print(message)

Здесь:

Мы используем оператор +, чтобы соединить несколько строк вместе.

Переменные name и city вставляются в строку, и в результате получается полное приветственное сообщение, которое выводится на экран.

Привет, Анна! Ты живешь в городе Москва.На экране появится:

Пример с булевыми значениями

Предположим, что ты решаешь, можно ли идти на улицу, основываясь на погоде. Если на улице солнечно, то можно идти гулять, а если идет дождь – нет. Мы будем использовать булевы значения:

python
is_sunny = True
is_raining = False
if is_sunny:
print("Можно идти гулять!")
else:
print("Лучше остаться дома.")

В этом примере:

Мы проверяем значение переменной is_sunny. Если оно True, программа выведет "Можно идти гулять!", если False – "Лучше остаться дома."

Оператор if проверяет, правда ли, что на улице солнечно.

Можно идти гулять!На экране появится:

Преобразования типов данных

В Python можно легко преобразовывать данные из одного типа в другой. Например, можно превратить строку в число или наоборот.

Преобразование строки в число:

Если у тебя есть строка, содержащая число, ты можешь превратить её в настоящий числовой тип с помощью функции int() или float():

python
number_str = "10"
number = int(number_str)
print(number)

Преобразование числа в строку:

Ты можешь превратить число в строку с помощью функции str():

python
age = 10
age_str = str(age)
print(age_str)

Мы рассмотрели основные типы данных: числа, строки и булевы значения. Теперь ты можешь использовать переменные для хранения информации в своих программах, а также легко работать с различными типами данных. Важно помнить, что каждый тип данных имеет свои особенности, и они используются в разных ситуациях.

Дальше мы будем учиться работать с более сложными концепциями, такими как условияи циклы, которые позволят нам создавать ещё более интересные программы.

Глава 3: Операции с числами

Числа – это один из самых важных типов данных в программировании. С их помощью мы можем выполнять математические вычисления, создавать калькуляторы, решать задачи и анализировать данные. В этой главе мы познакомимся с арифметическими операциями, создадим свой первый калькулятор и узнаем, как использовать встроенные математические функции Python.

Арифметические операции

Python поддерживает все основные арифметические операции, такие как сложение, вычитание, умножение и деление. Давайте рассмотрим каждую из них подробнее.

Сложение используется для того, чтобы объединить два числа:Сложение (+)

python

result = 5 + 3

print(result) # Выведет 8

С помощью вычитания мы можем узнать разницу между числами:Вычитание (-)

python

result = 10 – 4

print(result) # Выведет 6

Операция умножения используется для умножения двух чисел:Умножение (*)

python

result = 6 * 7

print(result) # Выведет 42

Деление всегда возвращает результат в виде дробного числа (даже если деление нацело):Деление (/)

python

result = 15 / 3

print(result) # Выведет 5.0

Если тебе нужно получить только целую часть от деления, можно использовать оператор //:Целочисленное деление (//)

python

result = 17 // 3

print(result) # Выведет 5

Оператор % возвращает остаток от деления двух чисел:Остаток от деления (%)

python

result = 17 % 3

print(result) # Выведет 2

С помощью оператора ** можно возводить число в степень:Возведение в степень ()**

python

result = 2 ** 3

print(result) # Выведет 8 (2 в кубе)

Создание калькулятора

Теперь, когда мы знаем основные арифметические операции, давай создадим простую программу-калькулятор. Она будет запрашивать два числа у пользователя и выполнять над ними математические операции.

Пример программы:

python

Простой калькулятор

print("Добро пожаловать в калькулятор!")

```
number1 = float(input("Введите первое число: "))
number2 = float(input("Введите второе число: "))
print("Выберите операцию: +, -, *, /, //, %, **")
operation = input("Введите операцию: ")
if operation == "+":
result = number1 + number2
elif operation == "-":
result = number1 – number2
elif operation == "*":
result = number1 * number2
elif operation == "/":
if number2 != 0:
result = number1 / number2
else:
result = "Ошибка: деление на ноль!"
elif operation == "//":
if number2 != 0:
result = number1 // number2
else:
result = "Ошибка: деление на ноль!"
elif operation == "%":
result = number1 % number2
elif operation == "**":
result = number1 ** number2
else:
result = "Ошибка: неизвестная операция!"
print("Результат:", result)
```

Как работает эта программа:

Пользователь вводит два числа.

Выбирает одну из доступных операций.

Программа вычисляет результат и выводит его на экран.

Попробуй выполнить эту программу с разными числами и операциями!

Математические функции

Python обладает мощным модулем для работы с математикой – math. Этот модуль включает в себя множество полезных функций, которые делают вычисления проще и удобнее. Чтобы использовать эти функции, нужно сначала импортировать модуль:

```
python
import math
```

Вот несколько популярных функций из модуля math:

math.sqrt(x) – вычисление квадратного корня:

```
python
import math
result = math.sqrt(16)
print(result) # Выведет 4.0
```

math.pow(x, y) – возведение числа x в степень y (аналог **):

```
python
import math
result = math.pow(2, 3)
print(result) # Выведет 8.0
```

math.ceil(x) – округление числа вверх:

python
import math
result = math.ceil(4.2)
print(result) # Выведет 5

math.floor(x) – округление числа вниз:

python
import math
result = math.floor(4.8)
print(result) # Выведет 4

math.pi – значение числа π (пи):

python
import math
print(math.pi) # Выведет 3.141592653589793

Пример использования математических функций

Давай создадим программу, которая будет рассчитывать длину окружности и площадь круга по введенному радиусу. Для этого нам понадобится число π и несколько функций из модуля math.

python
import math
# Ввод радиуса
radius = float(input("Введите радиус круга: "))
# Вычисление длины окружности
circumference = 2 * math.pi * radius
# Вычисление площади круга
area = math.pi * math.pow(radius, 2)
# Вывод результатов
print("Длина окружности:", circumference)
print("Площадь круга:", area)

Программа спрашивает у пользователя радиус, а затем вычисляет длину окружности и площадь круга, используя введенное значение. Это пример реального использования математических функций в программировании.

Комбинирование арифметики и математических функций

Ты можешь комбинировать арифметические операции с функциями модуля math, чтобы решать более сложные задачи. Например, вот программа, которая вычисляет гипотенузу треугольника по двум его сторонам, используя теорему Пифагора:

python
import math
# Ввод длин сторон
a = float(input("Введите длину первой стороны: "))
b = float(input("Введите длину второй стороны: "))
# Вычисление гипотенузы
hypotenuse = math.sqrt(math.pow(a, 2) + math.pow(b, 2))
# Вывод результата
print("Гипотенуза треугольника:", hypotenuse)

Здесь мы используем функцию math.sqrt для вычисления квадратного корня и math.pow для возведения в квадрат. Таким образом, программа реализует формулу гипотенузы: c=a2+b2c = \sqrt{a^2 + b^2}c=a2+b2.

Практическое задание

Напиши программу, которая вычисляет площадь прямоугольника. Пользователь вводит длину и ширину, а программа возвращает площадь.

Создай программу, которая определяет, является ли число чётным или нечётным. Используй оператор %.

Допиши калькулятор, добавив в него возможность вычислять квадратный корень и округление чисел.

Эта глава научила нас, как работать с числами, использовать арифметические операции, создавать калькуляторы и применять математические функции. Теперь ты можешь легко решать задачи, связанные с числами, и использовать эти знания в своих проектах.

Глава 4: Условия (if/else)

Программирование – это не только про выполнение команд, но и про принятие решений. Представь, что ты создаёшь игру, где игрок должен выбрать, идти ли ему налево или направо. Или разрабатываешь программу, которая проверяет, достиг ли пользователь нужного возраста, чтобы получить права. Такие задачи решаются с помощью условий.

Как работают условия в Python?

Условия позволяют программе проверять определённые утверждения и действовать в зависимости от их правдивости. Например, если на улице идёт дождь, мы возьмём зонтик; если нет, выйдем без него. В программировании такая проверка выглядит как конструкция if/else.

Основной синтаксис условия в Python:

python
if условие:
# код, который выполнится, если условие истинно
else:
# код, который выполнится, если условие ложно

Давай разберём это подробнее:

if – проверяет условие.

else – выполняет альтернативный код, если условие оказалось ложным.

После if и else всегда идёт двоеточие :, а код внутри них записывается с отступом (обычно 4 пробела).

Пример:

python
age = 18
if age >= 18:
print("Ты совершеннолетний!")
else:
print("Ты ещё ребёнок.")

Если переменная age больше или равна 18, программа напечатает: "Ты совершеннолетний!", иначе: "Ты ещё ребёнок."

Операторы сравнения

Чтобы проверять условия, используются операторы сравнения. Вот основные:

> – больше.

< – меньше.

>= – больше или равно.

<= – меньше или равно.

== – равно.

!= – не равно.

Примеры:

python
x = 10
y = 5
print(x > y) # True, потому что 10 больше 5
print(x == y) # False, потому что 10 не равно 5
print(x != y) # True, потому что 10 не равно 5

Эти операторы используются в условиях для проверки различных утверждений.

Пример: Проверка возраста

Давай создадим программу, которая спрашивает у пользователя возраст и сообщает, может ли он водить машину. В большинстве стран водить машину можно с 18 лет.

Пример программы:

python

```
age = int(input("Сколько тебе лет? "))
if age >= 18:
print("Ты можешь водить машину!")
else:
print("Ты ещё не можешь водить машину.")
```

Как это работает:

Программа запрашивает у пользователя возраст с помощью функции input(). Так как ввод возвращает строку, мы используем int() для преобразования в число.

Сравниваем возраст с числом 18.

Если возраст больше или равен 18, программа выводит сообщение: "Ты можешь водить машину!".

В противном случае программа выводит: "Ты ещё не можешь водить машину."

Попробуй запустить программу с разными значениями возраста и убедись, что всё работает правильно!

Расширение условий: elif

Иногда нужно проверить несколько условий. Например, ты создаёшь программу, которая оценивает, насколько пользователь близок к возрасту, необходимому для получения прав. В Python для этого используется конструкция elif (сокращение от "else if").

Пример:

python

```
age = int(input("Сколько тебе лет? "))
if age >= 18:
print("Ты можешь водить машину!")
elif 16 <= age < 18:
print("Ты почти можешь водить машину. Потерпи ещё немного!")
```

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.