

12+

Инженер



**КУРС «ПИТОН-
РАЗРАБОТЧИК»**

Инженер

Курс «Питон-разработчик»

«Издательские решения»

Инженер

Курс «Питон-разработчик» / Инженер — «Издательские решения»,

Курс представляет собой комплексную программу обучения, направленную на освоение языка с нуля до уровня Junior с переходом к Middle. В процессе обучения рассматриваются основы синтаксиса, работа с данными, функции, объектно-ориентированное программирование, взаимодействие с API, базы данных, тестирование и логирование. Особое внимание уделяется практическим заданиям и созданию реальных проектов, включая веб-приложения.

Содержание

1. Введение в Python	6
2. Установка и настройка окружения	8
3. Основы синтаксиса Python	10
4. Переменные и типы данных	12
5. Операторы и выражения	15
Конец ознакомительного фрагмента.	18

Курс «Питон-разработчик»

Инженер

© Инженер, 2026

ISBN 978-5-0070-3134-9

Создано в интеллектуальной издательской системе Ridero

1. Введение в Python

Python — это высокоуровневый язык программирования общего назначения, который широко используется в разработке программного обеспечения, анализе данных, автоматизации и искусственном интеллекте. Он известен простым и читаемым синтаксисом, благодаря чему подходит как для новичков, так и для опытных разработчиков.

Язык был создан Гвидо ван Россум в конце 1980-х годов и официально представлен в 1991 году. Основная идея Python — сделать код максимально понятным человеку.

Почему Python популярен

Python стабильно входит в число самых востребованных языков программирования благодаря следующим преимуществам:

— **Простота изучения** — лаконичный синтаксис, похожий на обычный английский язык

— **Универсальность** — подходит для разных задач

— **Большое сообщество** — множество библиотек и готовых решений

— **Кроссплатформенность** — работает на Windows, macOS и Linux

Однако есть и ограничения:

— Меньшая производительность по сравнению с компилируемыми языками

— Ограничения в мобильной разработке и низкоуровневом программировании

Где используется Python

Python применяется в самых разных сферах:

— **Веб-разработка** (с использованием фреймворков, например Django и Flask)

— **Анализ данных и машинное обучение**

— **Автоматизация задач и скрипты**

— **Разработка игр**

— **Научные вычисления**

Установка Python

Чтобы начать работу, необходимо установить интерпретатор Python.

— Перейдите на официальный сайт: ссылка в конце книги

— Скачайте последнюю версию Python

— Во время установки обязательно отметьте пункт **Add Python to PATH**

— Завершите установку

Проверить установку можно через командную строку:

```
python — version
```

Если система отображает версию Python — всё установлено корректно.

Среда разработки (IDE)

Для написания кода удобно использовать специальные редакторы:

— PyCharm — мощная IDE для Python

— Visual Studio Code — лёгкий и гибкий редактор

Выбор инструмента зависит от ваших предпочтений.

Первая программа

Традиционно изучение любого языка начинается с программы «Hello, World!».

Создайте файл main.py и добавьте следующий код:

```
print («Hello, World!»)
```

Запустите файл через терминал:

```
python main.py
```

Если всё сделано правильно, вы увидите:

Hello, World!

Как работает Python

Python — это интерпретируемый язык. Это означает, что код выполняется построчно с помощью интерпретатора, без предварительной компиляции.

Процесс выполнения выглядит так:

- Вы пишете код
- Интерпретатор Python читает его
- Код выполняется и вы получаете результат

Командная строка

Командная строка (терминал) — важный инструмент разработчика. С её помощью можно:

- запускать программы
- управлять файлами
- устанавливать библиотеки

Пример запуска интерактивного режима Python:

```
python
```

После этого вы можете вводить команды напрямую:

```
print(2 + 2)
```

Практические задания

- Установите Python на свой компьютер
- Установите один из редакторов (PyCharm или Visual Studio Code)
- Создайте файл hello.py
- Напишите программу, которая выводит ваше имя
- Запустите её через терминал

Итоги

В этой главе вы:

- узнали, что такое Python
- поняли, где он применяется
- установили интерпретатор
- настроили среду разработки
- написали первую программу

В следующей главе вы перейдёте к изучению основ синтаксиса Python и научитесь писать более сложные программы.

2. Установка и настройка окружения

В этой главе вы более подробно разберётесь с настройкой рабочего окружения Python-разработчика. Правильно настроенная среда — основа комфортной и эффективной работы.

Что такое окружение разработки

Окружение разработки — это набор инструментов, которые вы используете для написания, запуска и отладки кода:

- интерпретатор Python
- редактор или IDE
- менеджер пакетов
- виртуальные окружения

Установка Python (повторение и углубление)

Если вы уже установили Python — отлично. Теперь важно убедиться, что всё работает корректно.

Проверка версии:

```
python — version
```

или (на некоторых системах):

```
python3 — version
```

Переменные окружения (PATH)

При установке Python важно, чтобы он был добавлен в переменную PATH. Это позволяет запускать Python из любой директории.

Проверка:

```
python
```

Если команда не найдена — Python не добавлен в PATH, и это нужно исправить.

Менеджер пакетов pip

Python использует встроенный менеджер пакетов **pip** для установки библиотек.

Проверка:

```
pip — version
```

Пример установки библиотеки:

```
pip install requests
```

Первая особенность — отступы

В Python отступы имеют значение и заменяют фигурные скобки { } из других языков.

Пример:

```
if True:
```

```
    print («Это выполнится»)
```

Важно:

- Используйте одинаковые отступы (обычно 4 пробела)

- Не смешивайте пробелы и табы

Виртуальные окружения

В реальной разработке проекты используют разные версии библиотек. Чтобы избежать конфликтов, применяются виртуальные окружения.

Создание окружения:

```
python -m venv venv
```

Активация:

- Windows:

```
venv\Scripts\activate
```

- macOS / Linux:

```
source venv/bin/activate
```

После активации все пакеты будут устанавливаться только внутри проекта.

Выбор и настройка IDE

Рассмотрим два популярных инструмента:

- PyCharm
- Visual Studio Code

Основные шаги настройки:

- Установите IDE
 - Откройте папку проекта
 - Укажите интерпретатор Python
 - Подключите виртуальное окружение
- Полезные плагины (для Visual Studio Code)

- Python (официальный плагин)
- Pylint (проверка кода)
- Black (форматирование)

Структура проекта

Простейшая структура Python-проекта:

```
project/
|
|—— venv/
|—— main.py
|—— requirements.txt
|—— README.md
```

- venv/ — виртуальное окружение
- main.py — точка входа
- requirements.txt — список зависимостей

Создание файла зависимостей:

```
pip freeze > requirements.txt
```

Работа с зависимостями

Установка зависимостей из файла:

```
pip install -r requirements.txt
```

Частые ошибки новичков

- Python не добавлен в PATH
- Путают python и python3
- Не активируют виртуальное окружение
- Устанавливают пакеты глобально вместо venv

Практические задания

- Создайте новый проект
- Настройте виртуальное окружение
- Установите библиотеку requests
- Создайте файл main.py, который делает HTTP-запрос
- Сохраните зависимости в requirements.txt

Итоги

В этой главе вы:

- разобрались с настройкой окружения
- научились работать с pip
- создали виртуальное окружение
- настроили IDE
- подготовили структуру проекта

Дальше вы перейдёте к самому важному — изучению синтаксиса Python.

3. Основы синтаксиса Python

В этой главе вы познакомитесь с базовыми правилами написания кода на Python. Понимание синтаксиса — это фундамент, без которого невозможно двигаться дальше.

Что такое синтаксис

Синтаксис — это набор правил, определяющих, как писать корректные программы. В Python он максимально простой и читаемый.

Главная особенность Python — **минимализм**: меньше служебных символов, больше смысла.

Комментарии

Комментарии используются для пояснения кода и игнорируются интерпретатором.

Это однострочный комментарий

```
«"»
```

Это многострочный комментарий
или строка документации (docstring)

```
«"»
```

Регистр имеет значение

Python чувствителен к регистру:

```
name = «Alex»
```

```
Name = «John»
```

```
print (name) Alex
```

```
print (Name) John
```

Это разные переменные.

Правила именования

Имена переменных должны:

— начинаться с буквы или _

— содержать только буквы, цифры и _

— не совпадать с ключевыми словами

Правильно:

```
user_name = «Ivan»
```

```
age = 25
```

Неправильно:

```
2name = «Ivan» ошибка
```

```
user-name = 25 ошибка
```

Ключевые слова

Это зарезервированные слова Python, которые нельзя использовать как имена переменных:

```
if, else, for, while, def, class, True, False, None
```

Инструкции и выражения

— **Инструкция** — это команда (например, присваивание)

— **Выражение** — это вычисление значения

```
x = 5 инструкция
```

```
y = x + 3 выражение
```

Разделение строк

Обычно одна инструкция — одна строка:

```
print («Hello»)
```

Но можно переносить строки:

```
total = (1 +2 +3 +  
4 +5)
```

Множественное присваивание

Python позволяет присваивать значения сразу нескольким переменным:

```
a, b, c = 1, 2, 3
```

Динамическая типизация

В Python не нужно заранее объявлять тип переменной:

```
x = 10 int
```

```
x = «text» теперь строка
```

Вывод данных

Функция `print ()` используется для вывода информации:

```
print («Привет, мир!»)
```

Можно выводить несколько значений:

```
name = «Anna»
```

```
age = 20
```

```
print («Имя:», name, «Возраст:», age)
```

Ввод данных

Для ввода используется функция `input ()`:

```
name = input («Введите имя:»)
```

```
print («Привет,», name)
```

Практические задания

- Напишите программу, которая выводит ваше имя и возраст
- Запросите у пользователя его имя через `input ()`
- Создайте переменные `a`, `b` и поменяйте их значения местами
- Попробуйте написать код с неправильными отступами и посмотрите на ошибку

Частые ошибки

- Неправильные отступы
- Использование зарезервированных слов
- Ошибки в именах переменных
- Путаница с регистрами

Итоги

В этой главе вы:

- изучили базовые правила синтаксиса
- поняли роль отступов
- научились писать комментарии
- познакомились с вводом и выводом данных

Следующий шаг — работа с переменными и типами данных, где вы начнёте управлять информацией в программах.

4. Переменные и типы данных

В этой главе вы научитесь хранить данные в программе и работать с основными типами данных Python.

Что такое переменная

Переменная — это имя, которое ссылается на некоторое значение в памяти.

```
x = 10
```

```
name = «Alice»
```

Здесь:

— x хранит число

— name хранит строку

Основные типы данных в Python

Python имеет несколько встроенных типов данных:

1. Числа (int, float)

```
a = 10 целое число (int)
```

```
b = 3.14 число с плавающей точкой (float)
```

2. Строки (str)

Строка — это текст:

```
text = «Привет»
```

Можно использовать одинарные или двойные кавычки:

```
s1 = «Hello»
```

```
s2 = «World»
```

3. Булевый тип (bool)

Используется для логики:

```
is_active = True
```

```
is_admin = False
```

4. None

Специальный тип, означающий «ничего»:

```
value = None
```

Определение типа

Функция `type ()` показывает тип данных:

```
x = 5
```

```
print (type (x)) <class 'int'>
```

Преобразование типов

Иногда нужно изменить тип данных:

```
x = «10»
```

```
y = int (x) строка → число
```

```
z = str (25) число → строка
```

Примеры:

```
print (int («5») +2) 7
```

```
print (str (5) + «2») «52»
```

Операции с числами

a = 10

b = 3

print (a + b) сложение
print (a — b) вычитание
print (a * b) умножение
print (a / b) деление
print (a // b) целочисленное деление
print (a % b) остаток
print (a ** b) степень

Операции со строками

name = «Python»

print (name + " Developer») конкатенация
print (name * 3) повторение

Длина строки

text = «Hello»

print (len (text)) 5

Индексация строк

text = «Python»

print (text [0]) P

print (text [-1]) n

Срезы (slicing)

text = «Python»

print (text [0:3]) Pyt

print (text [2:]) thon

print (text [:4]) Pyth

Изменяемость

Строки — **неизменяемый тип**:

text = «Hello»

text [0] = «h» ошибка

Форматирование строк

Современный способ — f-строки

name = «Ivan»

age = 25

print (f"Имя: {name}, возраст: {age}»)

Практические задания

- Создайте переменные разных типов (int, float, str, bool)
- Напишите программу, которая складывает два числа
- Запросите у пользователя число и выведите его квадрат
- Объедините имя и фамилию в одну строку

— Выведите первую и последнюю букву строки

Частые ошибки

- Путаница типов («5» +5)
- Деление / вместо //
- Ошибки в индексации
- Попытка изменить строку

Итоги

В этой главе вы:

- узнали, что такое переменные
- изучили основные типы данных
- научились преобразовывать типы
- освоили базовые операции

Далее вы перейдёте к операторам и выражениям — это позволит строить более сложную логику программ.

5. Операторы и выражения

В этой главе вы изучите операторы — основные инструменты для выполнения вычислений и построения логики в программе.

Что такое операторы

Оператор — это символ или конструкция, выполняющая операцию над данными.

Выражение — это комбинация значений и операторов, которая возвращает результат.

`x = 5 + 3` выражение

Арифметические операторы

Используются для работы с числами:

`a = 10`

`b = 3`

`print (a + b)` сложение

`print (a — b)` вычитание

`print (a * b)` умножение

`print (a / b)` деление

`print (a // b)` целая часть

`print (a % b)` остаток

`print (a ** b)` степень

Операторы сравнения

Возвращают True или False

`a = 5`

`b = 10`

`print (a == b)` равно

`print (a != b)` не равно

`print (a > b)` больше

`print (a < b)` меньше

`print (a >= b)` больше или равно

`print (a <= b)` меньше или равно

Логические операторы

Используются для объединения условий:

`x = True`

`y = False`

`print (x and y)` И

`print (x or y)` ИЛИ

`print (not x)` НЕ

Операторы присваивания

`x = 5`

`x += 3` `x = x + 3`

`x -= 2`

`x *= 4`

`x /= 2`

Приоритет операторов

Операторы выполняются в определённом порядке:

- **
- *, /, //, %
- +, —
- сравнения
- логические операторы

Пример:

result = 2 + 3 * 4 14, а не 20

Скобки меняют порядок:

result = (2 + 3) * 4 20

Операторы принадлежности

Проверяют наличие элемента

text = «Python»

```
print («P» in text) True
```

```
print («z» not in text) True
```

Операторы тождественности

Проверяют, ссылаются ли переменные на один объект:

```
a = [1, 2]
```

```
b = a
```

```
c = [1, 2]
```

```
print (a is b) True
```

```
print (a is c) False
```

Короткое замыкание (short-circuit)

Python не вычисляет выражение полностью, если результат уже известен:

```
x = 0
```

```
print (x != 0 and 10 / x > 1) не будет ошибки
```

Практические задания

- Напишите программу, которая проверяет, является ли число чётным
- Сравните два числа, введённые пользователем
- Используйте логические операторы для проверки диапазона числа
- Попробуйте разные комбинации операторов и предскажите результат
- Напишите выражение с несколькими операторами и скобками

Частые ошибки

- Путаница = и ==
- Неправильный порядок операций
- Деление на ноль
- Неверное использование and / or

Итоги

В этой главе вы:

- изучили основные операторы
- научились сравнивать значения
- разобрались с логикой условий

— поняли приоритет операций

Далее вы перейдёте к условным конструкциям, где начнёте управлять поведением программы в зависимости от условий.

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.