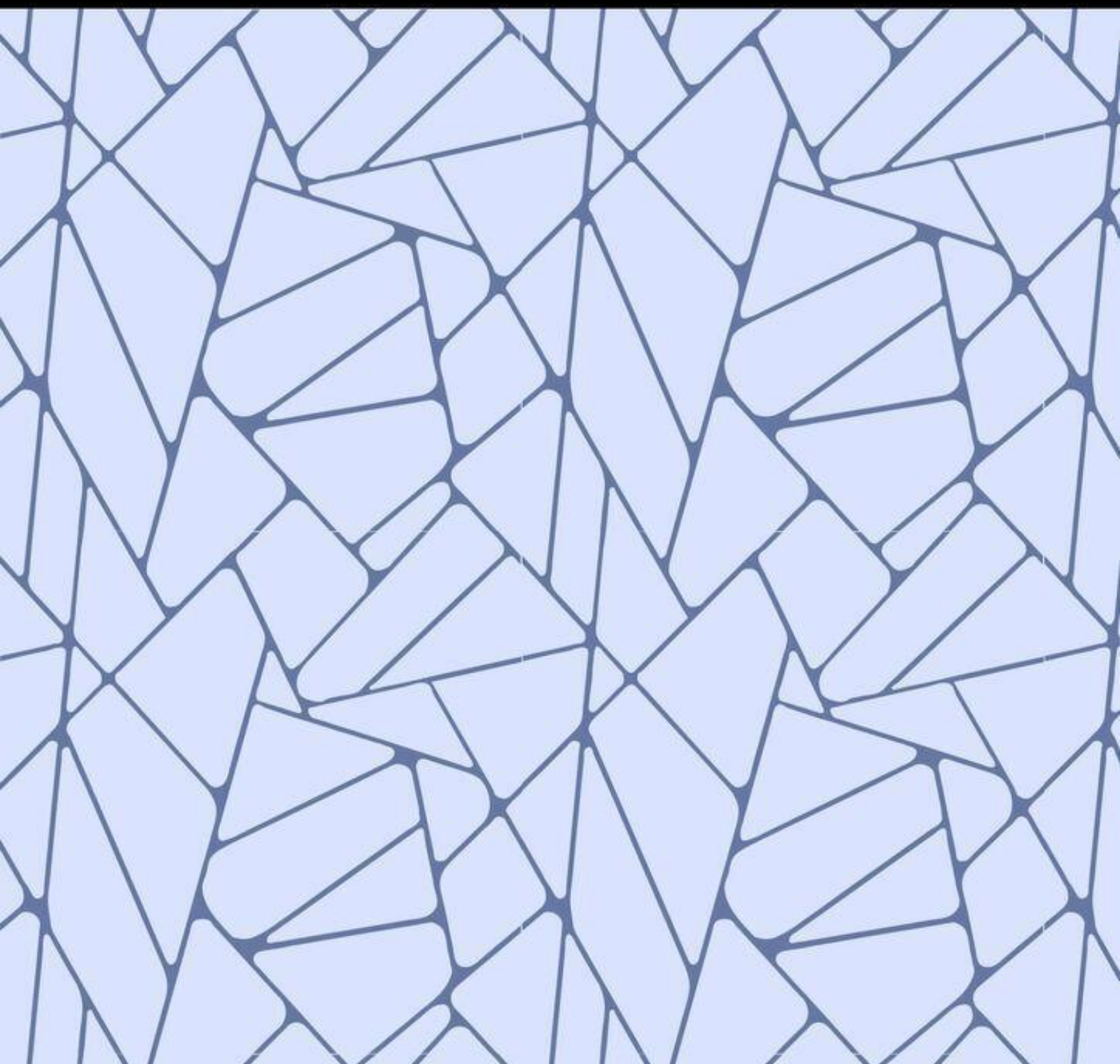


12+

Ленар Садыков

Программирование с нуля



Ленар Садыков

Программирование с нуля

«Издательские решения»

Садыков Л. И.

Программирование с нуля / Л. И. Садыков — «Издательские решения»,

«Программирование с нуля» — практический учебник для тех, кто хочет последовательно освоить программирование и перейти от первых строк кода к созданию собственных прикладных проектов. Книга построена вокруг практического подхода: читатель не только знакомится с основными понятиями, но и закрепляет материал на примерах, упражнениях, небольших проектах и задачах.

© Садыков Л. И.

© Издательские решения

Содержание

15. ООП	10
1. Ключевой принцип	11
Практический пример	12
2. Ключевой принцип	13
Практический пример	14
3. Ключевой принцип	15
Практический пример	16
4. Ключевой принцип	17
Практический пример	18
5. Ключевой принцип	19
Практический пример	20
Упражнения	21
Самопроверка	22
2. Установка и рабочее окружение	23
1. Ключевой принцип	24
Практический пример	25
2. Ключевой принцип	26
Практический пример	27
3. Ключевой принцип	28
Практический пример	29
4. Ключевой принцип	30
Практический пример	31
5. Ключевой принцип	32
Практический пример	33
Упражнения	34
Самопроверка	35
3. Первая программа	36
1. Ключевой принцип	37
Практический пример	38
2. Ключевой принцип	39
Практический пример	40
3. Ключевой принцип	41
Практический пример	42
4. Ключевой принцип	43
Практический пример	44
5. Ключевой принцип	45
Практический пример	46
Упражнения	47
Самопроверка	48
4. Переменные и типы данных	49
1. Ключевой принцип	50
Практический пример	51
2. Ключевой принцип	52
Практический пример	53
3. Ключевой принцип	54
Практический пример	55

4. Ключевой принцип	56
Практический пример	57
5. Ключевой принцип	58
Практический пример	59
Упражнения	60
Самопроверка	61
5. Операторы и выражения	62
1. Ключевой принцип	63
Практический пример	64
2. Ключевой принцип	65
Практический пример	66
3. Ключевой принцип	67
Практический пример	68
4. Ключевой принцип	69
Практический пример	70
5. Ключевой принцип	71
Практический пример	72
Упражнения	73
Самопроверка	74
6. Условия	75
1. Ключевой принцип	76
Практический пример	77
2. Ключевой принцип	78
Практический пример	79
3. Ключевой принцип	80
Практический пример	81
4. Ключевой принцип	82
Практический пример	83
5. Ключевой принцип	84
Практический пример	85
Упражнения	86
Самопроверка	87
7. Циклы	88
1. Ключевой принцип	89
Практический пример	90
2. Ключевой принцип	91
Практический пример	92
3. Ключевой принцип	93
Практический пример	94
4. Ключевой принцип	95
Практический пример	96
5. Ключевой принцип	97
Практический пример	98
Упражнения	99
Самопроверка	100
8. Строки	101
1. Ключевой принцип	102
Практический пример	103
2. Ключевой принцип	104

Практический пример	105
3. Ключевой принцип	106
Практический пример	107
4. Ключевой принцип	108
Практический пример	109
5. Ключевой принцип	110
Практический пример	111
Упражнения	112
Самопроверка	113
9. Списки, кортежи, множества и словари	114
1. Ключевой принцип	115
Практический пример	116
2. Ключевой принцип	117
Практический пример	118
3. Ключевой принцип	119
Практический пример	120
4. Ключевой принцип	121
Практический пример	122
5. Ключевой принцип	123
Практический пример	124
Упражнения	125
Самопроверка	126
Конец ознакомительного фрагмента.	127

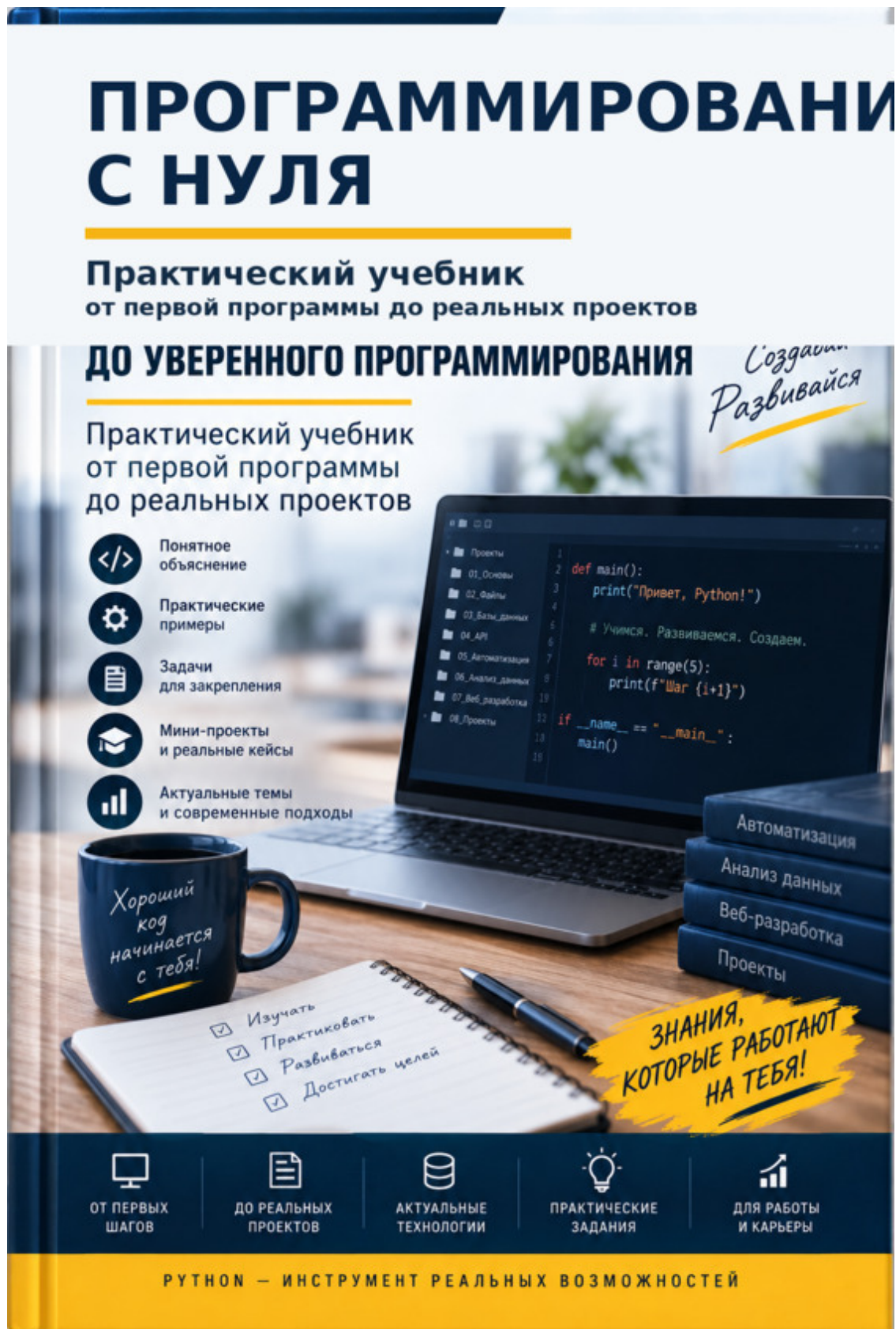
Программирование с нуля

Ленар Ильдарович Садыков

© Ленар Ильдарович Садыков, 2026

ISBN 978-5-0071-3584-9

Создано в интеллектуальной издательской системе Ridero



Содержание

1. Как устроено программирование и Python
2. Установка и рабочее окружение

3. Первая программа
4. Переменные и типы данных
5. Операторы и выражения
6. Условия
7. Циклы
8. Строки
9. Списки, кортежи, множества и словари
10. Функции
11. Ошибки и исключения
12. Файлы и pathlib
13. Модули и пакеты
14. venv, pip и зависимости

15. ООП

- 16. Генераторы и включения
- 17. Декораторы
- 18. Дата и время
- 19. Регулярные выражения
- 20. JSON и CSV
- 21. HTTP и API
- 22. SQLite
- 23. Конфигурация и переменные окружения
- 24. Логирование
- 25. Тестирование
- 26. Отладка
- 27. Git
- 28. Командная строка
- 29. Автоматизация
- 30. Проект: инженерный калькулятор
- 31. Проект: менеджер задач
- 32. Проект: журнал измерений
- 33. Проект: API-клиент
- 34. Проект: база оборудования
- 35. Финальный проект
- 36. Python для веб-разработки
- 37. Анализ данных
- 38. Python и ИИ
- 39. Python в инженерной автоматизации
- 40. Подготовка программы к публикации
- 41. Как продолжать обучение

Примечание. Python является зарегистрированным товарным знаком Python Software Foundation в ряде юрисдикций. Названия библиотек, платформ и других продуктов в книге используются для идентификации технологий и не означают официальную связь автора с их правообладателями. Настоящее издание является независимым учебно-практическим материалом.

Приложение А. Шпаргалка

Приложение В. 150 практических задач

Приложение С. Словарь

Приложение D. Чек-лист публикации

1. Как устроено программирование и Python

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Как устроено программирование и Python», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Как устроено программирование и Python», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Как устроено программирование и Python», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Как устроено программирование и Python», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Как устроено программирование и Python», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

2. Установка и рабочее окружение

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Установка и рабочее окружение», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Установка и рабочее окружение», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Установка и рабочее окружение», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Установка и рабочее окружение», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Установка и рабочее окружение», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

3. Первая программа

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Первая программа», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Первая программа», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Первая программа», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Первая программа», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Первая программа», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

4. Переменные и типы данных

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Переменные и типы данных», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Переменные и типы данных», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Переменные и типы данных», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Переменные и типы данных», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Переменные и типы данных», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

5. Операторы и выражения

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Операторы и выражения», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Операторы и выражения», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Операторы и выражения», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Операторы и выражения», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Операторы и выражения», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

6. Условия

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Условия», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Условия», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Условия», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Условия», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Условия», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

7. Циклы

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Циклы», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Циклы», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Циклы», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Циклы», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Циклы», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

8. Строки

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Строки», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Строки», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Строки», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Строки», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Строки», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.
- Могу проверить результат.

9. Списки, кортежи, множества и словари

В этом разделе мы изучаем тему пошагово. Сначала формулируем задачу обычными словами, затем разбираем синтаксис Python, после чего пишем небольшую программу и проверяем её на нормальных и ошибочных данных. Главная цель — научиться не копировать код, а понимать, почему каждая строка нужна.

1. Ключевой принцип

Перед кодированием выпишите входные данные, ожидаемый результат и ограничения. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

2. Ключевой принцип

Разбивайте задачу на маленькие действия и проверяйте каждое отдельно. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

3. Ключевой принцип

Используйте понятные имена переменных и функций. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

4. Ключевой принцип

Ошибку сначала локализуйте по traceback, а уже потом исправляйте. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

5. Ключевой принцип

После рабочего решения попробуйте изменить входные данные и проверить граничные случаи. На практике это означает, что решение нужно проверять не только на одном удобном примере, но и на граничных значениях.

Практический пример

```
def check (value, low, high):  
    if not (low <= value <= high):  
        return «Вне диапазона»  
    return «Норма»  
  
print (check (25, 0, 100))
```

Упражнения

1. Составьте собственный небольшой пример по теме «Списки, кортежи, множества и словари», запустите его и проверьте минимум два разных входных набора.
2. Составьте собственный небольшой пример по теме «Списки, кортежи, множества и словари», запустите его и проверьте минимум два разных входных набора.
3. Составьте собственный небольшой пример по теме «Списки, кортежи, множества и словари», запустите его и проверьте минимум два разных входных набора.
4. Составьте собственный небольшой пример по теме «Списки, кортежи, множества и словари», запустите его и проверьте минимум два разных входных набора.
5. Составьте собственный небольшой пример по теме «Списки, кортежи, множества и словари», запустите его и проверьте минимум два разных входных набора.

Самопроверка

- Могу объяснить тему своими словами.
- Могу написать пример без копирования.
- Могу обработать типичную ошибку.

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.