



# Урок 24. Python: Ввод, заполнение и перебор

## Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 24 из 33 · 45 минут · Модуль «Массивы и списки»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Два ученика проверяют одну программу на разных случайных данных и получают разные ответы. Как сохранить точный вход и повторить неудачный запуск?

Результат урока: Прочитать список заданного размера, проверить вход и получить повторяемую последовательность учебных данных.

### 01. Проблемная ситуация

0–5 минут. Строка "4 7 0 -2 5" означает: сначала размер, затем элементы. Сравните с "4 7 0 -2" и "3 7 0 -2 5". Назовите, какие записи надо отклонить и почему.

### 02. Понятия и разобранный пример

5–14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

#### Размер и фактическое число элементов должны совпасть

Вход — одна строка: размер  $n$  от 0 до 8, затем ровно  $n$  целых от -9 до 9, разделённых пробельными символами. `split()` выделяет слова, `int` преобразует их; `ValueError` перехватывается. После чтения проверяем количество, затем диапазоны. Исходная строка сохраняется в журнале вместе с результатом.

`append` добавляет элемент в конец. Перебор `for x in a` читает значения; `enumerate(a)` даёт пару индекс–значение. Нельзя подменять индекс значением: в `[7,0,-2,5]` число 7 не обозначает допустимую позицию.

`random.Random(seed)` создаёт отдельный псевдослучайный генератор. Если заново создать его с тем же `seed` и повторить те же вызовы в той же версии Python, последовательность повторится. Продолжение работы старого объекта даёт следующие числа. Для точного обмена между версиями и языками сохраняйте сам список, `seed` и сведения о среде.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

"4 7 0 -2 5" превращается в список `[7,0,-2,5]`. Размер 4 совпал с количеством элементов. Проверка диапазона проходит; журнал индексов: `0→7,1→0,2→-2,3→5`. "0" даёт `[]`, а пустая строка не содержит даже размера.

```
def read_values(line):
    try:
        data = [int(t) for t in line.split()]
    except ValueError:
        return None
    if not data:
        return None
    n = data[0]
    a = data[1:]
    if not 0 <= n <= 8 or len(a) != n:
        return None
    if any(x < -9 or x > 9 for x in a):
        return None
    return a
```

```
line = input()
a = read_values(line)
print(a)
if a is not None:
    for i, x in enumerate(a):
        print(i, x)
```



|         |                             |
|---------|-----------------------------|
| Строка  | 4 7 0 -2 5                  |
| Размер  | n = 4                       |
| Список  | [7, 0, -2, 5]               |
| Индексы | 0 → 7; 1 → 0; 2 → -2; 3 → 5 |

Разбор учебной строки: первый токен задаёт размер, остальные заполняют список по порядку.

### 03. Практика 1. Проверяем материал

#### Полный журнал ввода и вывода

14–25 минут. Восемь раз запустите программу с таблицей. Для успешного непустого входа перепишите все пары индекс–значение; для отказа укажите причину.

| Вход                     | Список или отказ |
|--------------------------|------------------|
| "4 7 0 -2 5"; "0"        | [7,0,-2,5]; []   |
| "4 7 0 -2"; "3 7 0 -2 5" | None; None       |
| "1 10"; "2 1 x"          | None; None       |
| ""; "-1"                 | None; None       |

2 минуты — прогноз, 4 — ввод и запуск, 3 — временно замените проверку `a is not None` на `if a` и объясните потерю различия между пустым результатом и ошибкой, 2 — восстановите код. Критерии: все входы, причины отказов, порядок и индексы.

### 04. Практика 2. Перенос способа

#### Генератор с собственным состоянием

25–33 минуты. Создайте функцию `generated(n,seed)`: `n` от 0 до 8; список из `n` чисел 0–9, новый `Random(seed)` на каждый вызов. За 2 минуты объясните ожидаемое равенство двух вызовов `generated(6,17)`, за 4 реализуйте, за 2 проверьте `n=0,8,9` и `seed=1`. Запишите `seed` и весь вывод. Критерии: размер, диапазон, повторение, обработка неверного `n`.

### 05. Обсуждение и границы вывода

33–40 минут. За 2 минуты сравните отчёты «ошибка на случайном списке» и «`seed=17`, вход=[8,6,4,5,4,2]». За 3 перечислите, что нужно коллеге для повторения. За 2 объясните, почему случайные тесты не заменяют пустой вход и границы.

### 06. Индивидуальная проверка

40–45 минут. Для "3 -1 0 1" укажите список, пару с индексом 2, результат `read_values("0")` и способ гарантированно передать тот же тест другому ученику. Четыре пункта — четыре балла.



## Часть 2. Ответы и материалы учителя

### Практика 1: разбор и критерии

Недостаток и избыток элементов — разные некорректные записи. 10 выходит за диапазон, x не преобразуется, пустой ввод не содержит n, -1 недопустим. if а ложно и для [], и для None: при ветке обработки успеха это смешивает допустимый нулевой размер с отказом.

### Практика 2: разбор и критерии

В проверенной среде Python для n=6, seed=17 получено [8,6,4,5,4,2], для seed=1 — [2,9,1,4,1,7]. Два свежих вызова с seed=17 совпадают; n=0 даёт [], n=9 — None. Не требуйте совпадения этих чисел с другим языком или иной реализацией генерации.

```
import random

def generated(n, seed):
    if n < 0 or n > 8:
        return None
    rng = random.Random(seed)
    a = []
    for _ in range(n):
        a.append(rng.randint(0, 9))
    return a

for n, seed in [(6,17),(6,17),(6,1),(0,17),(8,17),(9,17)]:
    print(n, seed, generated(n, seed))
```

### Обсуждение: возможный вывод

Нужны код, точный список, версия среды и ожидаемый ответ; seed полезен дополнительно. Псевдослучайный набор может не включить нужную границу. Проверки нулевой длины, недопустимого числа и неверного количества выполняются отдельно.

### Индивидуальная проверка: ответ и критерии

[-1,0,1]; (2,1); []; передать полную строку/список и ожидаемый результат. Один seed без описания среды и алгоритма не гарантирует тот же тест во всех системах.

### Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Python 3.11+ и IDLE или редактор; запуск python lesson.py (либо python3 lesson.py). Внешние библиотеки не нужны. Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 8 класс. Предпосылки: списки, append, цикл for, функции, int. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.

| Минуты | Работа                  | Результат                  |
|--------|-------------------------|----------------------------|
| 0–5    | Проблема                | Вопрос и первое решение    |
| 5–14   | Понятия и пример        | Способ проверки            |
| 14–25  | Практика 1              | Готовый продукт и проверка |
| 25–33  | Практика 2              | Перенос способа            |
| 33–40  | Обсуждение              | Аргумент и ограничение     |
| 40–45  | Индивидуальная проверка | Самостоятельный ответ      |



## 07. Соответствие исходной программе

| Пункт программы                                        | Действие ученика                                                                                             | Проверяемый результат                                    |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| Массивы и списки                                       | Прочитать список заданного размера, проверить вход и получить повторяемую последовательность учебных данных. | Исходник, трассировка и таблица проверок                 |
| Создание, ввод и обработка массивов; перебор элементов | Генератор с собственным состоянием                                                                           | Самостоятельный алгоритм, ключ и индивидуальная проверка |

## Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Python: псевдослучайная генерация и воспроизводимость

<https://docs.python.org/3/library/random.html>

Python: разделение входной строки

<https://docs.python.org/3/library/stdtypes.html#str.split>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 18.10.2026.