



Урок 19. Python: Строка и число

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 19 из 33 · 45 минут · Модуль «Символьные строки»

Уровень: начинающий программист; Python 3.

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

"007" — текст, а 7 — число. При превращении в число ведущие нули исчезают. Как проверить ввод до вычислений и не принять цену 3.5 за целое количество?

Результат урока: Проверить строгий формат целого неотрицательного числа, преобразовать строку и объяснить потерю форматирования.

01. Проблемная ситуация

0–5 минут. Для "007", "7", "7", "+7", "3.5", "" решите, что примет учебный формат: от 1 до 3 символов ASCII 0...9, значение 0...120. Не ориентируйтесь на то, что умеет int: контракт здесь строже.

02. Понятия и разобранный пример

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

Сначала формат, потом значение

int умеет преобразовывать многие записи, включая пробелы и знак. Но наша задача требует только 1...3 ASCII-цифры 0...9 без пробелов и знака. Поэтому сначала проверяем длину и каждый символ. isdigit здесь недостаточно точен: он допускает некоторые символы других числовых систем, которые контракт исключает.

После проверки формата int(s) безопасен для этой короткой строки. Затем отдельно проверяем значение ≤120. В ответе используем None для ошибки и целое 0...120 для успеха. Ноль — корректный результат: проверять успех через if value нельзя, нужна проверка value is None.

Преобразование не сохраняет внешний вид. int("007")=7, str(7)="7", исходные нули не восстанавливаются сами. Если значение — идентификатор, а не количество, превращать его в число может быть неправильно. Для точных сотых удобнее целые единицы минимального разряда, без float.

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

"007": длина 3, все символы цифры, int даёт 7, диапазон соблюден. "121": формат верный, но значение вне диапазона. "3.5": неверный формат ещё до int. "000" даёт допустимый 0.

```
def parse_count(s):
    if not 1 <= len(s) <= 3:
        return None
    for ch in s:
        if ch < "0" or ch > "9":
            return None
    value = int(s)
    if value > 120:
        return None
    return value

for s in ["007", "0", "120", "121", "7", "+7", "3.5", ""]:
    value = parse_count(s)
    if value is None:
        print(repr(s), "error")
    else:
        print(repr(s), value, value + 1)
```



"007"	формат верен → число 7
"121"	формат верен → диапазон нарушен
"3.5"	не все символы — ASCII-цифры
"000"	число 0 — успех, не ошибка

Проверка формата отделена от проверки диапазона; None обозначает ошибку, 0 остаётся допустимым числом.

03. Практика 1. Проверяем материал

Текстовое представление и арифметика

14–25 минут. Проверьте 12 строк таблицы, для каждого отказа назовите этап: длина, символы или значение.

Строки	Результат parse_count
"0";"000";"007"	0;0;7
"120";"121"	120;None
"";"1234"	None;None
" 7";"+ 7";"3.5"	None;None;None
"2";"з" (не ASCII)	None;None

2 минуты — прогноз; 4 — запуски; 3 — замените value is None на not value в выводе и найдите ошибку на "0"; 2 — восстановите, сравните "007" и str(7). Критерии: формат, диапазон, правильный ноль, объяснение ведущих нулей.

04. Практика 2. Перенос способа

Строгая сумма с двумя знаками после точки

25–33 минуты. Учебная сумма задаётся ровно пятью символами DD.DD: две ASCII-цифры, точка, две ASCII-цифры. Верните целое число сотых: "03.50"→350; неправильный формат→None. За 2 минуты рассчитайте "00.00", "99.99", "3.50", "03,50", "0a.50"; за 4 напишите parse_cents без float; за 2 проверьте. Критерии: длина, точка, цифры, точные сотые.

05. Обсуждение и границы вывода

33–40 минут. За 2 минуты сравните код ученика "007" как идентификатор и как количество деталей, за 3 обсудите последствия int, за 2 объясните, почему строгая проверка может отвергать запись, которую Python понимает.

06. Индивидуальная проверка

40–45 минут. Найдите parse_count("012"), str этого результата, parse_cents("12.03") и объясните, почему parse_count("000") нельзя считать ошибкой. Четыре пункта — четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

В таблице три допустимых значения 0,0,7 и ещё 120. Длина отклоняет"" и"1234"; значение отклоняет"121"; остальные отказы относятся к символам. not value ошибочно считает допустимый 0 ошибкой. str(7) возвращает"7", а не"007".

Практика 2: разбор и критерии

0;9999;None;None;None. Для"03.50" расчёт $3 \cdot 100 + 50 = 350$. "3.50" не подходит, потому что требуются две цифры до точки. Формат учебный, не универсальный денежный ввод.

```
def parse_cents(s):
    if len(s) != 5 or s[2] != ".":
        return None
    for i in [0,1,3,4]:
        if s[i] < "0" or s[i] > "9":
            return None
    return int(s[:2]) * 100 + int(s[3:])

for s in ["03.50", "00.00", "99.99", "3.50", "03,50", "0a.50"]:
    print(repr(s), parse_cents(s))
```

Обсуждение: возможный вывод

Количество 7 и 007 одинаково по величине, но идентификаторы могут различаться и должны храниться строками. Возможности int шире нашего контракта; проверка нужна для установленного формата. Удаление пробелов или знаков без требования меняет правила приёма данных.

Индивидуальная проверка: ответ и критерии

12;"12";1203 сотых;"000" соответствует формату и значению 0, ошибка обозначается None. Ведущий ноль при преобразовании исчезает.

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: строки, for, if, функции, позиционная запись десятичных чисел. 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
Символьные строки	Проверить строгий формат целого неотрицательного числа, преобразовать строку и объяснить потерю форматирования.	Исходник, таблица прогнозов и фактических результатов
Символьные строки, обращение к символам, операции и преобразования	Строгая сумма с двумя знаками после точки	Самостоятельное решение и индивидуальная проверка по четырём критериям

Источники и права на изображения

Исходная программа курса

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

Python: преобразование int

<https://docs.python.org/3/library/functions.html#int>

Python: строковые проверки и ASCII-границы

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

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.