



Урок 16. Python: числовая задача с циклом

Часть 1. Материалы ученика

Программирование на Python и C++ · 7 класс

Урок 16 из 34 · 45 минут · Модуль «Марафон по программированию»

Уровень: начинающий программист; Python 3.

Программа: <https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

В копилке уже достаточно жетонов, но программа делает ещё один шаг. Как найти именно первый подходящий день?

Результат урока: Самостоятельно построить цикл накопления, доказать остановку и проверить минимальное число шагов.

01. Проблемная ситуация

Учебная копилка получает 5 жетонов в первый день, затем каждый день на 3 больше. Цель 30. Один алгоритм отвечает 3 дня, другой 4. За 5 минут вручную выпишите первые четыре взноса и суммы, выберите ответ и укажите, какая сумма была непосредственно перед достижением цели.

02. Понятия и разобранный пример

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

Состояние цикла и условие остановки

Состояние хранит три величины: `total` — уже накопленное, `day` — число выполненных взносов, `amount` — следующий взнос. До первой итерации это 0,0,5. В теле сначала добавляем текущий взнос, затем увеличиваем день и готовим следующий `amount`. Порядок важен: преждевременное увеличение `amount` пропустит 5.

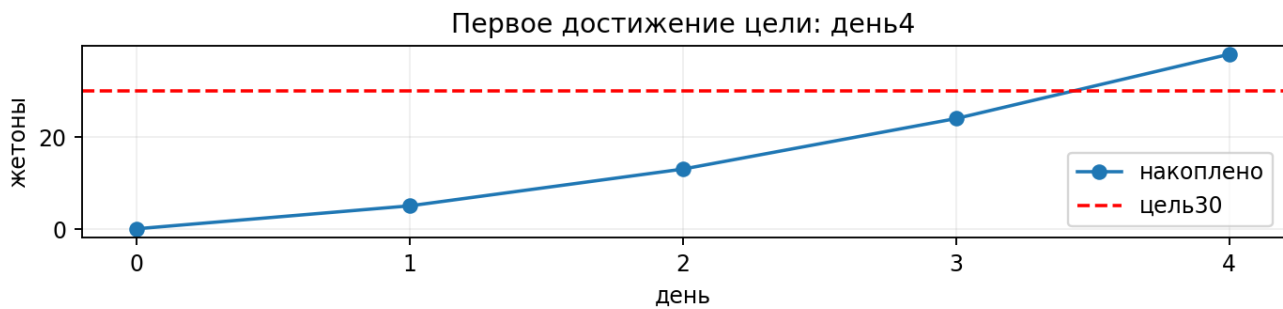
Условие `total < target` означает «денег ещё недостаточно». При точном достижении цели цикл не должен продолжаться. Нулевая цель требует 0 шагов. Для допустимого `target` от 0 до 1000 каждый взнос положителен и не меньше 5, значит цикл завершится. За пределами диапазона выводим `error`; вход — одна целая строка.

Минимальность проверяют двумя фактами: итоговая сумма не меньше цели, предыдущая — меньше. Просто получить большую сумму недостаточно: можно было добавить лишние дни. Все суммы — жетоны учебной модели, это не прогноз реального дохода и не предложение накопительного продукта.

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

Для 30: после дней 1,2,3,4 суммы 5,13,24,38. Поэтому ответ 4 38. После третьего дня $24 < 30$, после четвёртого $38 \geq 30$. Для 24 ответ 3 24: замена `<` на `<=` ошибочно добавила бы четвёртый взнос.

```
target = int(input())
if not 0 <= target <= 1000:
    print("error")
else:
    total = 0
    day = 0
    amount = 5
    while total < target:
        total += amount
        day += 1
        amount += 3
    print(day, total)
```



Взносы 5,8,11,14 дают суммы 5,13,24,38; цель 30 впервые достигнута на четвёртом шаге.

03. Практика 1. Проверяем материал

Первый день достижения цели

14–25 минут. Сначала закройте код примера и по описанию составьте свои строки: ввод, проверка, начальное состояние, условие, три обновления, вывод.

Цель	Ожидаемые день и сумма
0	0 0
5	1 5
6	2 13
24	3 24
30	4 38
-1	error

2 минуты — трассировка, 4 — самостоятельный код, 3 — тесты таблицы, 2 — объяснение минимальности для 30. Сверьте реализацию с примером только после первой попытки. Продукт — программа, шесть фактических выводов и две неравенства для цели 30. Критерии: правильное состояние, отсутствие лишнего шага, обработка 0 и неверного диапазона, обоснование остановки.

04. Практика 2. Перенос способа

Новый старт и другой прирост

25–33 минуты. Новая копилка начинает с 7 жетонов; первый новый взнос 4, каждый следующий на 2 больше. Диапазон целей тот же 0...1000. Для цели не выше 7 дополнительных дней 0. Самостоятельно измените начальное состояние и прирост, сохранив правило минимальности. Проверьте цели 7,8,17,18. Критерии: учтён начальный запас, правильная последовательность взносов и точное достижение.

05. Обсуждение и границы вывода

33–40 минут. Ученик проверяет лишь цель 1000. Достаточно ли этого для доказательства правильности? Предложите альтернативу: малые граничные случаи, трассировка состояния и объяснение остановки.

06. Индивидуальная проверка

40–45 минут. В исходной модели цель 25. Запишите полный вывод, предыдущую сумму и объясните, почему `while total <= target` неверен для цели 24. Три критерия: вывод, предыдущая сумма, объяснение строгого условия.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Ответы приведены в таблице. Допустимо другое расположение day и amount, если прибавляются именно 5,8,11... и сохраняется минимальность. Если программа печатает только day, она не выполняет данный формат результата; нужны оба числа. Рабочий полный вариант приведён ниже.

```
target = int(input())
if not 0 <= target <= 1000:
    print("error")
else:
    total = 0
    day = 0
    amount = 5
    while total < target:
        total += amount
        day += 1
        amount += 3
    print(day, total)
```

Практика 2: разбор и критерии

Для 7 ответ 0 7; для 8 —1 11; для 17 —2 17; для 18 —3 25. Здесь total=7, amount=4, прирост 2. Взносы 4,6,8, а не 7,9,11.

```
target = int(input())
if not 0 <= target <= 1000:
    print("error")
else:
    total = 7
    day = 0
    amount = 4
    while total < target:
        total += amount
        day += 1
        amount += 2
    print(day, total)
```

Обсуждение: возможный вывод

Один большой тест не проверяет нулевую цель и точное попадание. Нужны случаи 0, первый взнос, сумма нескольких взносов и число на 1 больше такой суммы. Доказательство минимальности опирается на предыдущую и новую суммы. Для иных правил взносов, например отрицательных, доказательство остановки нужно строить заново.

Индивидуальная проверка: ответ и критерии

Вывод 4 38, предыдущая сумма 24. При цели 24 условие $\leq$ истинно после третьего дня и ошибочно добавляет 14. По 1 баллу за каждый критерий.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, рабочую папку с правом записи, редактор, распечатку ученической части PDF, бумагу и ручку. Python 3.11+; внешние библиотеки не нужны. Все входы и ожидаемые результаты даны. Установку и проверку запуска учитель выполняет до занятия; детские аккаунты и интернет не нужны.

Уровень — начинающий программист, 7 класс. Предпосылки: while, накопитель, сравнения и целочисленная арифметика. В паре автор вводит и объясняет, проверяющий сопоставляет с прогнозом, затем роли меняются. Без компьютера трассируйте полный код на бумаге, отмечайте координаты по схеме и проверяйте готовые результаты; учитель демонстрирует заранее собранный пример. Бумажная трассировка не выдаётся за реальную компиляцию или запуск ученика. План 45 минут включает чтение, ввод, сборку и обсуждение. В графических задачах сначала прогноз, затем сравнение. 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
Циклы с условием и обработка целых чисел	Разрабатывает цикл накопления по состоянию	Минимальный шаг и доказательство остановки
Марафон по программированию	Выбирает граничные тесты самостоятельно	Таблица ожидаемых и фактических выводов

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-7-klass-programma>

Python: условия, циклы и функции

<https://docs.python.org/3.11/tutorial/controlflow.html>

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: 17.10.2026.