



Урок 13. Календарные условия

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 13 из 33 · 45 минут · Модуль «Повторение и обобщение: марафон по ветвлениям»

Уровень: начинающий программист; Python 3.

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

Для 1900 и 2000 проверка «год делится на 4» отвечает одинаково. Но в принятой календарной модели февраль имеет разную длину. Как учесть исключение из исключения?

Результат урока: Проверить високосность по григорианскому правилу и определить допустимость дня февраля с обработкой ошибочного ввода.

01. Проблемная ситуация

0–5 минут. Используем пролептический григорианский календарь: одно правило применяется к годам 1...9999, независимо от исторических дат перехода стран. Год високосный, если делится на 4, но столетний — только если делится на 400. Запишите прогноз для 1900, 2000, 2024, 2025.

02. Понятия и разобранный пример

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

Исключение для столетия и возврат по 400

Условие записывается как $y \% 400 == 0$ or $(y \% 4 == 0 \text{ and } y \% 100 != 0)$. Скобки показывают смысл: достаточно делимости на 400 либо совместного выполнения делимости на 4 и неделимости на 100. Год 0 в контракт не входит.

По остаткам получаются четыре случая: не делится на 4; делится на 4, но не на 100; делится на 100, но не на 400; делится на 400. Каждый должен иметь свой тест. Проверяем календарную модель, а не историческую дату введения календаря.

Отделяйте тип входа от календарного правила. В основной функции предполагается целый год; функция возвращает True/False либо None для года вне 1...9999. None означает ошибку, а False — корректный невисокосный год. Нельзя смешивать эти ответы.

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

$1900 \% 4 = 0, 1900 \% 100 = 0, 1900 \% 400 = 300$: False. $2000 \% 400 = 0$: True. $2024 \% 4 = 0$ и $2024 \% 100 = 24$: True. $2025 \% 4 = 1$: False.

```
def leap(y):  
    """Целый год 1..9999; None вне диапазона."""  
    if y < 1 or y > 9999:  
        return None  
    return y % 400 == 0 or (y % 4 == 0 and y % 100 != 0)
```

```
for y in [1, 1900, 2000, 2024, 2025, 0, 10000]:  
    print(y, leap(y))
```

Не делится на 4	2025 → 28 дней февраля
На 4, но не на 100	2024 → 29 дней
На 100, но не на 400	1900 → 28 дней
Делится на 400	2000 → 29 дней

Четыре случая григорианского правила; годы рассматриваются в явно заданной календарной модели.



03. Практика 1. Проверяем материал

Четыре класса годов

14–25 минут. Составьте журнал из десяти входов таблицы, объясняя остатки, а не вспоминая календарь.

Годы	Ответ leap
1; 4	False; True
100; 400	False; True
1900; 2000	False; True
2024; 2025	True; False
0; 10000	None; None

2 минуты — прогноз; 4 — запуски; 3 — временно замените всё условие на $y \% 4 == 0$ и найдите два контрпримера; 2 — восстановите. Критерии: 10 ответов, различение False и None, контрпримеры 100/1900, объяснение 400.

04. Практика 2. Перенос способа

Проверяем дату февраля

25–33 минуты. Напишите `valid_feb(y,d)` для целых года и дня: вернуть True, если дата февраля существует в нашей модели; иначе False.

Проверить(1900,29),(2000,29),(2024,30),(2025,28),(0,1),(2024,0). За 2 минуты выберите предел, за 4 напишите код, за 2 проверьте. Критерии: год, нижняя граница дня, верхняя граница, правильная високосность.

05. Обсуждение и границы вывода

33–40 минут. Почему пример 2000 не проверяет исключение для 1900? За 2 минуты распределите тесты по четырём случаям, за 3 обсудите утверждение «код проверил два современных года, значит календарь верен», за 2 сформулируйте ограничение модели.

06. Индивидуальная проверка

40–45 минут. Определите `leap(2100)`, `leap(2400)`, `valid_feb(2100,29)`, `valid_feb(2400,29)`. Для каждого ответа укажите причину. По баллу за верный ответ с причиной.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Ошибку простого правила обнаруживают 100 и 1900: оба невисокосные. Год 2000 нужен для проверки исключения из исключения: запрет всех столетних годов тоже неверен. Полная правильная функция приведена выше.

Практика 2: разбор и критерии

False, True, False, True, False, False. Дню недостаточно быть меньше 30: нужны допустимый год и $d \geq 1$. Полный автономный файл ниже.

```
def valid_feb(y, d):
    if y < 1 or y > 9999:
        return False
    is_leap = y % 400 == 0 or (y % 4 == 0 and y % 100 != 0)
    limit = 29 if is_leap else 28
    return 1 <= d <= limit
```

```
for y, d in [(1900, 29), (2000, 29), (2024, 30),
             (2025, 28), (0, 1), (2024, 0)]:
    print(y, d, valid_feb(y, d))
```

Обсуждение: возможный вывод

2000 делится на 400, а 1900 нет: они идут по разным основаниям. Современные соседние годы обычно не проверяют столетние исключения. Модель не описывает исторические переходы стран и не определяет допустимость дат других месяцев.

Индивидуальная проверка: ответ и критерии

False, True, False, True. 2100 делится на 100, но не 400; 2400 делится на 400. Поэтому пределы февраля 28 и 29 соответственно.

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: остаток от деления, Python `if/elif`, функции. 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
Повторение и обобщение: марафон по ветвлениям	Проверить високосность по григорианскому правилу и определить допустимость дня февраля с обработкой ошибочного ввода.	Исходник, таблица прогнозов и фактических результатов
Составлять и отлаживать программы; использовать условные операторы	Проверяем дату февраля	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 calendar: пролептический григорианский календарь и isleap

<https://docs.python.org/3/library/calendar.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: 18.10.2026.