



Урок 15. Рефакторинг сложного условия

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 15 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>

Две команды записали правило допуска к учебному тренажёру по-разному. Как доказать, что более короткое условие не меняет поведение?

Результат урока: Сопоставить логически эквивалентные условия по полной таблице и отделить рефакторинг от изменения требования.

01. Проблемная ситуация

0–5 минут. Тренажёр закрыт, если `closed=True`. Иначе вход возможен при `member=True` или `ticket=True`. Предскажите результат для `(False,False,True)` и `(False,True,False)` в порядке `member,ticket,closed`. Почему нельзя убрать проверку `closed`?

02. Понятия и разобранный пример

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

Отрицание группы и смысл правила

Исходное условие `not(closed or not(member or ticket))` означает: не закрыт и есть хотя бы одно основание для входа. По закону де Моргана оно равно `not closed and (member or ticket)`. Предполагаются именно логические значения, а не строки "False" или произвольные числа.

Рефакторинг сохраняет ответы на всех допустимых входах. Три независимых булевых признака образуют $2^3=8$ сочетаний. Полный перебор этих восьми строк вместе с ясным контрактом проверяет равенство двух данных функций на всём их входном множестве.

Важно не только получить True на сравнении версий, но и сверить обе с человеческим правилом. Две одинаково ошибочные функции тоже совпадут. В коде функции чистые, без побочных эффектов; рассуждаем о значениях и не переносим автоматически преобразования на вызовы, меняющие состояние.

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

`member=False,ticket=True,closed=False`: `member or ticket=True`, `not closed=True`, итог True. При `closed=True` обе версии обязаны вернуть False даже при двух основаниях.

```
def old(member, ticket, closed):
    return not (closed or not (member or ticket))

def clear(member, ticket, closed):
    return not closed and (member or ticket)

for m in [False, True]:
    for t in [False, True]:
        for c in [False, True]:
            print(m, t, c, old(m,t,c), clear(m,t,c))
```

not(A or B)	(not A) and (not B)
A = closed	B = not(member or ticket)
Двойное отрицание	not not X = X для bool
Итог	not closed and (member or ticket)

Преобразование условия для трёх булевых признаков; правило допуска учебное, не система контроля доступа.



03. Практика 1. Проверяем материал

Восемь сочетаний вместо догадки

14–25 минут. Прогнозируйте все восемь ответов таблицы до запуска; сравните оба столбца программы.

member,ticket,closed	Правильный допуск
False,False,False / False,False,True	False / False
False,True,False / False,True,True	True / False
True,False,False / True,False,True	True / False
True,True,False / True,True,True	True / False

2 минуты — правило; 4 — запуск полного перебора; 3 — замените `or` на `and` внутри скобок `clear`, найдите различающую строку; 2 — восстановите и объясните. Критерии: восемь прогнозов, сравнение, контрпример, различие «хотя бы одно» и «оба».

04. Практика 2. Перенос способа

Новое требование меняет таблицу

25–33 минуты. Правило меняется: теперь при незакрытом тренажёре требуется ровно одно основание, `member` и `ticket` не должны совпадать. За 2 минуты отметьте изменившуюся строку таблицы, за 4 напишите `exact_one`, за 2 проверьте все 8. Критерии: ровно одно, сохранённый `closed`, полный перебор, объяснение изменения.

05. Обсуждение и границы вывода

33–40 минут. Одна команда предлагает всегда сокращать до самой короткой записи, другая сохраняет именованные логические признаки. За 2 минуты подготовьте примеры, за 3 сопоставьте читаемость и доказательство, за 2 сформулируйте критерий принятия правки.

06. Индивидуальная проверка

40–45 минут. Для `(True,True,False)` найдите `clear` и `exact_one`. Назовите количество допустимых сочетаний трёх `bool` и одну причину, почему равенства двух функций на одном тесте недостаточно. По баллу за пункт.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Различающие строки(False,True,False) и(True,False,False): правильный допускTrue, ошибочныйFalse. При(True,True,False) версии совпадут, поэтому такой единственный тест слаб. Все правильные исходники приведены выше.

Практика 2: разбор и критерии

По сравнению с исходным правилом меняется только(True,True,False): теперьFalse. Это изменение требований, не рефакторинг. Для двух bool неравенство $m \neq t$ выражает «ровно одно».

```
def exact_one(member, ticket, closed):  
    return not closed and (member != ticket)  
  
for m in [False, True]:  
    for t in [False, True]:  
        for c in [False, True]:  
            print(m, t, c, exact_one(m,t,c))
```

Обсуждение: возможный вывод

Короткая запись полезна, если сохраняет смысл и понятна читателю. Имена member,ticket,closed объясняют предметную область. При рефакторинге требуем неизменность всех 8 ответов и соответствие исходному правилу; изменение таблицы должно быть согласованным новым требованием.

Индивидуальная проверка: ответ и критерии

True;False;8;остаются непроверенные сочетания, на которых ответы могут отличаться. При полном переборе фиксированного булева контракта непроверенных сочетаний нет.

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: Python: bool, not/and/or, функции и циклы. 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	Сопоставить логически эквивалентные условия по полной таблице и отделить рефакторинг от изменения требования.	Исходник, таблица прогнозов и фактических результатов
Составлять и отлаживать программы; использовать условные операторы	Новое требование меняет таблицу	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: управление выполнением

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