



Урок 8. Python: Границы условий и таблица тестов

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 8 из 33 · 45 минут · Модуль «Ветвления. Условный оператор. Сложные условия»

Уровень: начинающий программист; Python 3.

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

Программа верно обработала 20 и 90, но ошиблась ровно на 40. Научимся выбирать проверки возле границ, где один знак меняет результат.

Результат урока: Разбить допустимый диапазон на непересекающиеся части и составить таблицу тестов «до, на, после» каждой границы.

01. Проблемная ситуация

0–5 минут. Учебная игра задаёт уровни: 0...39 start, 40...69 middle, 70...100 advanced. Автор написал первую проверку `points <= 40`. Найдите вход, на котором программа нарушает правило, и объясните, почему тест 20 его не обнаружит. Это вымышленные игровые уровни, не школьные оценки.

02. Понятия и разобранный пример

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

Диапазоны без пробелов и пересечений

Граничные тесты берут значение непосредственно перед порогом, сам порог и после него. Для порога 40 это 39, 40, 41; для 70 — 69, 70, 71. Проверяем также край допустимого диапазона и выход за него. Тесты получают из условия задачи до чтения результата программы.

Цепочка `if/elif` выбирает первую истинную ветвь. После `points < 40` ложного уже известно `points ≥ 40`, поэтому в следующей ветви достаточно `points < 70`. Раннее исключение `points < 0` или `points > 100` отделяет недопустимый вход от нижнего и верхнего уровней.

Полнота означает, что каждому допустимому входу назначен результат. Непересечение означает единственное назначение. Неверные диапазоны могут иметь пробел, например 0...39 и 41...69, или пересечение на 40. Правильный порядок иногда скрывает пересечение, но не исправляет неясное условие.

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 40 проверка ошибки ложна; `points < 40` ложна; `points < 70` истинна: middle. Для 70 первые три проверки ложны и выбирается advanced. Для -1 срабатывает error до классификации. Код ниже печатает восемь пар из контрольного набора.

```
def band(points):
    """Учебные уровни: целое points, допустимо 0...100."""
    if points < 0 or points > 100:
        return "error"
    elif points < 40:
        return "start"
    elif points < 70:
        return "middle"
    else:
        return "advanced"

for p in [-1, 0, 39, 40, 69, 70, 100, 101]:
    print(p, band(p))
```



Нижний диапазон	0...39 → start
Средний диапазон	40...69 → middle
Верхний диапазон	70...100 → advanced
Недопустимые	<0 или >100 → error

Непересекающиеся целочисленные диапазоны: 0...39 start, 40...69 middle, 70...100 advanced; снаружи error.

03. Практика 1. Проверяем материал

Мутация знака и минимальный контрпример

14–25 минут. Выполните все 12 тестов таблицы, записав ожидания до запуска. Затем по очереди внесите две ошибки: <40 заменить на <=40; >100 заменить на >=100.

points	Ожидаемый результат
–1; 0; 1	error; start; start
39; 40; 41	start; middle; middle
69; 70; 71	middle; advanced; advanced
99; 100; 101	advanced; advanced; error

2 минуты — таблица; 4 — исходный запуск; 3 — поиск контрпримеров для двух отдельных изменений; 2 — восстановление. Продукт: таблица и две записи «изменение → обнаруживающий вход». Критерии: 12 ожиданий, минимальные контрпримеры 40 и 100, окончательно правильный код.

04. Практика 2. Перенос способа

Новое разбиение на временные интервалы

25–33 минуты. Учебный таймер принимает целые seconds: 0...59 short, 60...119 medium, 120...180 long; вне 0...180 error. Создайте timer_band. За 2 минуты запишите тесты 59,60,61,119,120,121,180,181; за 4 код; за 2 проверка. Критерии: границы включены по условию, недопустимое отделено, все восемь результатов.

05. Обсуждение и границы вывода

33–40 минут. Что полезнее: десять случайных входов внутри одного диапазона или значения вокруг всех границ? Предложите компромисс для проверки. Объясните, почему удачный запуск на 90 не защищает от ошибки на 100.

06. Индивидуальная проверка

40–45 минут. Для правил 0...9 A, 10...19 B, 20...30 C назовите ответы на 9,10,20 и граничный тест, отличающий >30 от >=30. По баллу за каждый элемент.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Первую ошибку обнаруживает 40: получается start вместо middle. Вторую — 100: error вместо advanced. Вносите ошибки отдельно, затем восстанавливайте. Полный драйвер ниже делает 12 проверок правильной функции.

```
def band(points):
    """Учебные уровни: целое points, допустимо 0...100."""
    if points < 0 or points > 100:
        return "error"
    elif points < 40:
        return "start"
    elif points < 70:
        return "middle"
    else:
        return "advanced"

for p in [-1,0,1,39,40,41,69,70,71,99,100,101]:
    print(p, band(p))
```

Практика 2: разбор и критерии

short,medium,medium,medium,long,long,long,error. Дополнительно 0 → short и −1 → error. На 120 проверка <120 ложна, поэтому long. Иные имена интервалов не меняют принцип выбора тестов.

```
def timer_band(seconds):
    """Классифицировать целое число секунд; 0...180."""
    if seconds < 0 or seconds > 180:
        return "error"
    elif seconds < 60:
        return "short"
    elif seconds < 120:
        return "medium"
    else:
        return "long"

for s in [59,60,61,119,120,121,180,181]:
    print(s, timer_band(s))
```

Обсуждение: возможный вывод

Граничные тесты целенаправленно проверяют знаки сравнений и смену ветвей; обычные внутренние случаи тоже нужны для основного поведения. Тест 90 не выполняет равенство верхней границе, поэтому не различает >100 и >=100. Хороший набор покрывает границы, типичные значения и ошибочные входы.

Индивидуальная проверка: ответ и критерии

А,В,С; вход 30: он допустим и должен дать С. 4 балла. Вход 31 не различает эти две ошибочно сравниваемые проверки: обе его отвергнут.

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: if/elif/else, сравнения и логические операции. 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
Ветвления. Условный оператор. Сложные условия	Разбирает пример и создаёт две подпрограммы	Исходники, прогнозы и протокол проверки
Составление и отладка программ	Сравнивает фактический результат с контрактом	Объяснение границ и самостоятельный ответ

Источники и права на изображения

Исходная программа курса

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

Python: функции и 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.