**Вложенные и каскадные условия.**  
Внутри условного оператора можно использовать любые инструкции Python, в том числе и условный оператор. Получаем вложенное ветвление: после одной развилки в ходе исполнения программы появляется другая развилка. При этом вложенные блоки имеют больший размер отступа (+4 пробела для каждого уровня).

В данном случае уровень вложенности равен двум, так что программа одинаково читается как с помощью использования логического оператора and, так и с помощью вложенного оператора.

**M\_2\_1.2 str – 1 - 12**  
  
Давайте рассмотрим программу, которая переводит сто балльную оценку в пятибалльную. Для ее реализации нужно воспользоваться вложенным условным оператором:  
  
**M\_2\_1.2 str – 16 - 28**  
  
Выбор из нескольких альтернатив – это обычное дело, здесь имеет смысл избегать глубокого вложения. Для этого в Python есть каскадный условный оператор.

**Каскадный условный оператор**

Если требуется проверить несколько условий, в Python используется каскадный условный оператор.

При исполнении такого условного оператора сначала проверяется условие 1. Если оно является истинным, то исполняется блок кода, который следует сразу после него, вплоть до выражения elif. Остальная часть конструкции игнорируется. Однако если условие 1 является ложным, то программа перескакивает непосредственно к следующему выражению elif и проверяет условие 2. Если оно истинное, то исполняется блок кода, который следует сразу после него, вплоть до следующего выражения elif. И остальная часть условного оператора тогда игнорируется.

Этот процесс продолжается до тех пор, пока не будет найдено условие, которое является истинным, либо пока больше не останется выражений elif. Если ни одно условие не является истинным, то исполняется блок кода после выражения else.  
  
Приведенный ниже фрагмент кода является примером каскадного условного оператора if-elif-else. Этот фрагмент кода работает так же, как предыдущий код, использующий вложенный условный оператор.   
  
**M\_2\_1.2 str – 30 - 41**

Зачастую вложенные условия понижают читабельность кода. Для примера возьмем функцию, создающую пользователя. Перед созданием нам нужно проверить имя пользователя и его пароль:

**CODE**

Функция register\_user() выполняет свою задачу, но её тяжело отнести к категории читаемых - несколько вложенных условий, каждое из которых отвечает за какую-то проверку. Если потребуется добавить новые проверки, то придется каждый раз вчитываться во вложенные условия и стараться не задеть уже существующие условия и их функционал.

У этой проблемы есть два (не взаимоисключающих) решения.

Разворачивание

Если внутри условий есть места, где функция "досрочно" завершает выполнение, то можно попробовать "развернуть" эти условия. Такое условие:

if len(password) > 3:

create\_new\_user(username, password)

print("New user is created")

else:

print("The password is too short")

**CODE**

И так как при выполнении условия мы покидаем фунцию, мы можем убрать else (тем самым уменьшая вложенность):

**CODE**

Применив эту технику несколько раз, в конце получим изначальную функцию register\_user() в таком виде:

**CODE**

В таком виде код довольно линеен, добавление новых проверок (и действий в случае их нарушения) стало проще.

## Вынесение

Еще один способ "облегчения" условий - вынесение их в отдельные функции. Пример:

**CODE**

Здесь проверки вынесены в отдельные функции, что повысило читаемость - теперь по одним лишь названиям вызываемых функций точно видно, что register\_user() проверяет имя пользователя и пароль, и если они валидны, создает пользователя.

P.S. Для практики и лучшего понимания можно вернуться к своему старому коду и попробовать повысить его читаемость этими способами.

**Тернарный оператор**

Тернарный оператор в Python – это тип выражения условия в языке программирования Python, который позволяет разработчикам оценивать операторы. Тернарные операторы выполняют действие в зависимости от того, истинно это утверждение или нет. Таким образом, эти операторы более лаконичны, чем традиционный оператор if-else.

Тернарный оператор – это оператор, который используется для демонстрации какого-то условия, то есть вместо условной конструкции. Он состоит из значений True и False, а также выражения, которое должно быть вычислено для этих значений.

Синтаксис тернарного оператора:

[если истина] if [выражение] else [если ложь]

Также может быть реализован вложенный тернарный оператор. Давайте реализуем вложенный тернарный оператор в нашем скрипте Python.

# -----------------------------------TASK 1----------------------------------

Вводятся три целых числа.

Необходимо определить наименьшее среди них и вывести его на экран.

Реализовать программу, используя условный оператор, без использования функции min.

# -----------------------------------TASK 2----------------------------------

Вес боксера-любителя (в кг, в виде вещественного числа). Известно, что вес таков, что боксер может быть отнесен к одной из весовых категорий:

1) легкий вес – до 60 кг (включительно);  
2) первый полусредний вес – до 64 кг (включительно);  
3) полусредний вес – до 69 кг (включительно);  
4) остальные - более 69 кг.

Вывести на экран номер категории, в которой будет выступать боксер.

# -----------------------------------TASK 3----------------------------------

Вводится два вещественных числа.

Необходимо с помощью тернарного условного оператора наибольшее значение присвоить переменной d и вывести ее на экран.

x, y = 25, 50  
big = x if x < y else y

Вводится целое число. Необходимо переменной msg присвоить строку "кратно 3", если введенное число кратно 3, а иначе присвоить строку "не кратно 3".

Реализовать программу с использованием тернарного оператора. Переменную msg отобразить на экране.

Вводится целое число 0 или 1.

Необходимо преобразовать их в строки: 0 - в "False", 1 - в "True". Реализовать это с помощью тернарного условного оператора. Результат отобразить на экране.

# -----------------------------------TASK 4----------------------------------

Вводится текущее время (секунды) в диапазоне [0; 59].

Если значение равно 59, то следующее должно быть 0. И так по кругу.

Необходимо  вычислить следующее значение с проверкой граничного значения 59.

Реализуйте это с помощью тернарного условного оператора. Результат отобразите на экране.

print((a+1) % 60)

P.S. Попробуйте также реализовать эту же задачу с использованием только арифметических операций.

# -----------------------------------TASK 5----------------------------------

Пользователь вводит с клавиатуры номер месяца (1-12). Необходимо вывести на экран название месяца.

Например, если 1, то на экране надпись январь, 2 — февраль и т.д.

# -----------------------------------TASK 6----------------------------------

Вводятся четыре целых числа a, b, c, d.

Определить, войдет ли в конверт с внутренними размерами a и b мм прямоугольная открытка с размерами с и d мм.

Для размещения открытки в конверте необходим зазор в 1 мм с каждой стороны.

Открытку можно поворачивать на 90 градусов.

Вывести ДА, если входит и НЕТ - если не входит.