**Тема 6. Условный оператор if-else**

**Аннотация.** Урок посвящен условному оператору if-else.

Мы познакомились с базовыми строительными блоками программ, научились писать программы, обеспечивающие ввод, обработку и вывод данных. Более того, умеем работать со строками и числами, как мы делаем это в математике. Теперь научимся управлять ходом выполнения программы.

Программы должны уметь выполнять разные действия в зависимости от введенных данных. Для принятия решения программа проверяет, истинно или ложно определенное условие.

В Python существует несколько способов проверки, и в каждом случае возможны два исхода: истина (**True**) или ложь (**False**).

Проверка условий и принятие решений по результатам этой проверки называется **ветвлением**. Программа таким способом выбирает, по какой из возможных ветвей ей двигаться дальше.

В Python проверка условия осуществляется при помощи ключевого слова if.

Рассмотрим следующую программу:

answer = input('Какой язык программирования мы изучаем?')

if answer == 'Python':

print('Верно! Мыботаем Python =)')

print('Python - отличный язык!')

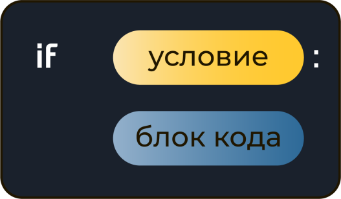
Программа просит пользователя ввести текст и проверяет результат ввода. Если введенный текст равен строке «Python», то выводит пользователю текст:

Верно! Мы ботаем Python =)

Python - отличный язык!

**Двоеточие (:)** в конце строки с инструкцией if сообщает интерпретатору Python, что дальше находится **блок команд**. В блок команд входят все строки с отступом под строкой с инструкцией if, вплоть до следующей строки без отступа.

Если условие истинно, выполняется весь расположенный ниже блок. В предыдущем примере блок инструкций составляет третья и четвёртая строки программы.

**Блоком кода** называют объединённые друг с другом строки. Они всегда связаны с определённой частью программы (например, с инструкцией if). В Python блоки кода формируются при помощи **отступов**.

Предыдущая программа выводит текст в случае, если условие истинно. Но если условие ложно, то программа ничего не выводит. Для того, чтобы обеспечить возможность выполнять что-либо, если условие оказалось ложным, мы используем ключевое слово else.

answer = input('Какой язык программирования мы изучаем?')

if answer == 'Python':

print('Верно! Мыботаем Python =)')

print('Python - отличный язык!')

else:

print('Не совсем так!')

В новой программе мы обрабатываем сразу два случая: если условие истинно (пользователь ввел «Python»), и если условие ложно (пользователь ввел что угодно, кроме «Python»).

**Отступы**

В некоторых языках программирования отступы — дело личного вкуса, и можно вообще обходиться без них. Однако в Python они неотъемлемая часть кода. Именно отступ сообщает интерпретатору Python, где начинается и где заканчивается блок кода.

Некоторым инструкциям в Python (например, инструкции if) именно блок кода сообщает, какие действия следует предпринять. После if блок кода информирует интерпретатор Python, как действовать, если условие истинно, и как — если оно ложно.

**Операторы сравнения**

Можно заметить, что в проверке условия мы использовали двойное равенство (==), вместо ожидаемого одиночного (=). Не стоит путать **оператор присваивания** (=) с **условным оператором** (==).

Операция неравенства (!=)

num = 1992

s = 'I love Python'

Для проверки двух элементов на равенство Python использует удвоенный знак равно (==). Вот так: if answer == 'Python':

if name == 'Gvido':

if temperature == 40:

Путаница с операторами == и = является одной из самых распространенных ошибок в программировании. Эти символы используются не только в Python, и каждый день множество программистов используют их неправильно.

В Python существует 6 основных операторов сравнения.

| **Выражение** | **Описание** |
| --- | --- |
| if x > 7 | если x больше 7 |
| if x < 7 | если x меньше 7 |
| if x >= 7 | если x больше либо равен  7 |
| if x <= 7 | если x меньше либо равен  7 |
| if x == 7 | если x равен  7 |
| if x != 7 | если x не равен  7 |

Рассмотрим пример:

num1 = int(input())

num2 = int(input())

if num1 < num2:

print(num1, 'меньшечем', num2)

if num1 > num2:

print(num1, 'большечем', num2)

if num1 == num2: # используем двойное равенство

print(num1, 'равно', num2)

if num1 != num2:

print(num1, 'неравно', num2)

**Цепочки сравнений**

Операторы сравнения в Python можно объединять в цепочки (в отличии от большинства других языков программирования, где для этого нужно использовать логические связки), например, a == b == c или 1 <= x <= 10. Следующий код проверяет, находится ли значение переменной age в диапазоне от 3 до 6:

age = int(input())

if3<= age <= 6:

print('Вы ребёнок')

Код, проверяющий равенство трех переменных, может выглядеть так:

if a == b == c:

print('числаравны')

else:

print('числа не равны')

**Транзитивность**

Операция равенства является [транзитивной](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C). Это означает, что если a == b и b == c, то из этого следует, что a == c. Именно поэтому предыдущий код, проверяющий равенство трёх переменных, работает как полагается.

Из курса математики вам могут быть знакомы другие примеры транзитивных операций:

* **Отношение порядка:** если a > bи b > c, то a > c;
* **Параллельность прямых:** если a∥*b* и b∥*c*, то a∥*c*;
* **Делимость:** если a делится на b и b делится на c, то a делится на c.

Наглядно транзитивность отношения порядка можно понять на таком примере: если сосед слева старше вас (a > b) а вы старше соседа справа (b > c), то сосед слева точно старше соседа справа (a > c).

Операция неравенства (!=), в отличие от операции равенства (==), является [нетранзитивной](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C). То есть из того, что a != b и b != c,вовсе не следует, что a != c. Действительно, если вас зовут не так, как соседа слева и не так, как соседа справа, то нет гарантии, что у обоих соседей не окажутся одинаковые имена.

Таким образом, следующий код вовсе не проверяет тот факт, что все три переменные различны:

if a != b != c:

print('числа не равны')

else:

print('числа равны')

Код, проверяющий, что значения трёх переменных различны, мы научимся писать немного позже.

## Примеры решения задач

**Задача 1.** Напишите программу, которая считывает одну строку. Если это строка «Python», программа выводит «ДА», в противном случае программа выводит «НЕТ».

**Решение.** Программа, решающая поставленную задачу, может иметь вид:

word = input()

if word == 'Python':

print('ДА')

else:

print('НЕТ')

**Задача 2.** Напишите программу, которая определяет, состоит ли двузначное число, введенное с клавиатуры, из одинаковых цифр. Если состоит, то программа выводит «ДА», в противном случае программа выводит «НЕТ».

**Решение.** Программа, решающая поставленную задачу, может иметь вид:

num = int(input())

last\_digit = num % 10# последняяцифрачисла

first\_digit = num // 10# первая цифра числа

if last\_digit == first\_digit:

print('ДА')

else:

print('НЕТ')

**Задача 3.** Напишите программу, которая считывает три числа и подсчитывает количество чётных чисел.

**Решение.** Программа, решающая поставленную задачу, может иметь вид:

num1, num2, num3 = int(input()), int(input()), int(input())

counter = 0# переменная счётчик

if num1 % 2 == 0:

counter = counter + 1# увеличиваем счётчик на 1

if num2 % 2 == 0:

counter = counter + 1# увеличиваем счётчик на 1

if num3 % 2 == 0:

counter = counter + 1# увеличиваем счётчик на 1

print(counter)

**Тест. Проверь себя!**

Работа каких операторов дает верный ответ при любом значении переменной i?

1. if i / 2:

print(i, 'чётное')

else:

print(i, 'нечётное')

1. if i // 2:

print(i, 'чётное')

else:

print(i, 'нечётное')

1. if i % 2 == 0:

print(i, 'чётное')

else:

print(i, 'нечётное')

1. if i // 2 == 0:

print(i, 'чётное')

else:

print(i, 'нечётное')

1. if i % 2 != 0:

print(i, 'нечётное')

else:

print(i, 'чётное')

1. if i // 2 != 0:

print(i, 'нечётное')

else:

print(i, 'чётное')

ответ 3 и 5

## Задача 6\_1. Пароль

При регистрации на сайтах требуется вводить пароль дважды. Это сделано для безопасности, поскольку такой подход уменьшает возможность неверного ввода пароля.

Напишите программу, которая сравнивает пароль и его подтверждение. Если они совпадают, то программа выводит: «Пароль принят», иначе: «Пароль не принят».

**Формат входных данных**  
На вход программе подаются две строки.

**Формат выходных данных**  
Программа должна вывести одну строку в соответствии с условием задачи.

**Sample Input 1:**

qwerty

qwerty

**Sample Output 1:**

Пароль принят

**Sample Input 2:**

qwerty

Qwerty

**Sample Output 2:**

Пароль не принят

## Задача 6\_2. ****Четное или нечетное?****

Напишите программу, которая определяет, является число четным или нечетным.

**Формат входных данных**  
На вход программе подаётся одно целое число.

**Формат выходных данных**  
Программа должна вывести «Четное», если число четное, и «Нечетное» — если число нечетное.

**Sample Input 1:**

10

**Sample Output 1:**

Четное

**Sample Input 2:**

17

**Sample Output 2:**

Нечетное

## Задача 6\_3. ****Соотношение****

Напишите программу, которая проверяет, что для заданного четырехзначного числа выполняется следующее соотношение: сумма первой и последней цифр равна разности второй и третьей цифр.

**Формат входных данных**  
На вход программе подаётся одно целое положительное четырёхзначное число.

**Формат выходных данных**  
Программа должна вывести «ДА», если соотношение выполняется, и «НЕТ» — если не выполняется.

**Sample Input 1:**

1614

**Sample Output 1:**

ДА

**Sample Input 2:**

1234

**Sample Output 2:**

НЕТ

**Sample Input 3:**

7911

**Sample Output 3:**

ДА

## Задача 6\_4. Роскомнадзор

Напишите программу, которая определяет, разрешен пользователю доступ к интернет-ресурсу или нет.

**Формат входных данных**  
На вход программе подаётся целое число — возраст пользователя.

**Формат выходных данных**  
Программа должна вывести текст «Доступ разрешен» если возраст не менее 18, и «Доступ запрещен» в противном случае.

**Sample Input 1:**

16

**Sample Output 1:**

Доступ запрещен

**Sample Input 2:**

18

**Sample Output 2:**

Доступ разрешен

## Задача 6\_5. Арифметическая прогрессия

Напишите программу, которая определяет, являются ли три заданных числа (в указанном порядке) последовательными членами [арифметической прогрессии](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F).

**Формат входных данных**  
На вход программе подаются три числа, каждое на отдельной строке.

**Формат выходных данных**  
Программа должна вывести «YES» или «NO» (без кавычек) в соответствии с условием задачи.

**Sample Input 1:**

1

2

3

**Sample Output 1:**

YES

**Sample Input 2:**

1

2

4

**Sample Output 2:**

NO

## Задача 6\_6. ****Наименьшее из двух чисел****

Напишите программу, которая определяет наименьшее из двух чисел.

**Формат входных данных**  
На вход программе подаётся два различных целых числа.

**Формат выходных данных**  
Программа должна вывести наименьшее из двух чисел.

**Sample Input 1:**

8

11

**Sample Output 1:**

8

**Sample Input 2:**

20

5

**Sample Output 2:**

5

## Задача 6\_7. ****Наименьшее из четырёх чисел****

Напишите программу, которая определяет наименьшее из четырёх чисел.

**Формат входных данных**  
На вход программе подаётся четыре целых числа.

**Формат выходных данных**  
Программа должна вывести наименьшее из четырёх чисел.

**Sample Input 1:**

1

2

3

4

**Sample Output 1:**

1

**Sample Input 2:**

10

9

11

12

**Sample Output 2:**

9

**Задача 6\_8. Возрастная группа**

Напишите программу, которая по введённому возрасту пользователя сообщает, к какой возрастной группе он относится:

* до 13 включительно – детство;
* от 14 до 24 – молодость;
* от 25 до 59 – зрелость;
* от 60 – старость.

**Формат входных данных**  
На вход программе подаётся одно целое число – возраст пользователя.

**Формат выходных данных**  
Программа должна вывести название возрастной группы.

**Sample Input 1:**

4

**Sample Output 1:**

детство

**Sample Input 2:**

91

**Sample Output 2:**

старость

**Sample Input 3:**

40

**Sample Output 3:**

зрелость

## Задача 6\_9. ****Только +****

Напишите программу, которая считывает три числа и подсчитывает сумму только положительных чисел.

**Формат входных данных**  
На вход программе подаются три целых числа.

**Формат выходных данных**  
Программа должна вывести одно число – сумму положительных чисел.

**Примечание.** Если положительных чисел нет, то следует вывести 0.

**Sample Input 1:**

4

-22

1

**Sample Output 1:**

5

**Sample Input 2:**

33

55

63

**SampleOutput 2:**

151

**Тема 7. Логические операторы**

**Логические операторы**

Как быть в ситуации, когда у нас есть несколько условий? В Python есть три логических оператора, которые позволяют создавать сложные условия:

* and — логическое умножение;
* or — логическое сложение;
* not — логическое отрицание.

**Оператор and**

Предположим, мы написали программу для учеников от двенадцати лет, которые учатся по крайней мере в 7 классе. Доступ к ней тем, кто младше, надо запретить. Следующий код решает поставленную задачу:

age = int(input('Сколько вам лет?: '))

grade = int(input('В каком классе вы учитесь?: '))

if age >= 12 and grade >= 7:

print('Доступразрешен.')

else:

print('Доступ запрещен.')

Мы объединили два условия при помощи оператора and. Оно означает, что в этом ветвлении блок кода выполняется только при выполнении **обоих условий** **одновременно**!

Оператор and может объединять произвольное количество условий:

age = int(input('Сколько вам лет?: '))

grade = int(input('В каком классе вы учитесь?: '))

city = input('В каком городе вы живете?: ')

if age >= 12and grade >= 7and city == 'Москва':

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Это таблица истинности для оператора and. В ней перечислены выражения, соединённые оператором and, показаны все возможные комбинации истинности и ложности и приведены результирующие значения выражений.

| **a** | **b** | **a and b** |
| --- | --- | --- |
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

Как показывает таблица, чтобы значение выражения с оператором and было истинным, должны быть истинными **оба (все)** объединенные им условия.

**Оператор or**

Оператор orтакже применяется для объединения условий. Однако, в отличие от and, для выполнения блока кода достаточно выполнения **хотя бы одного из условий**.

city = input('В каком городе вы живете?: ')

if city == 'Москва' or city == 'Санкт-Петербург' or city == 'Екатеринбург':

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Доступ будет разрешен в случае, если хотя бы одно из условий выполнится.

Это таблица истинности для оператора or. В ней перечислены выражения, соединённые оператором or, показаны все возможные комбинации истинности и ложности и приведены результирующие значения выражений.

| **a** | **b** | **a or b** |
| --- | --- | --- |
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

Для того, чтобы выражение or было истинным, требуется, чтобы **хотя бы одно** условие оператора or было истинным. При этом не имеет значения, истинным или ложным является второе выражение.

Логическое выражение X and Y истинно, если **оба**значения X и Y истинны.

Логическое выражение X or Y истинно, если **хотя бы одно** из значений X и Y истинно.

Мы можем использовать оба логических оператора одновременно:

age = int(input('Сколько вам лет?: '))

grade = int(input('В каком классе вы учитесь?: '))

city = input('В каком городе вы живете?: ')

if age >= 12 and grade >= 7 and (city == 'Москва'or city == 'Санкт-Петербург'):

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Такой код проверяет, что возраст учеников от двенадцати лет и учатся они по крайней мере в 7 классе и живут в Москве или Санкт-Петербурге.

**Оператор not**

Оператор not позволяет инвертировать (т.е. заменить на противоположный) результат логического выражения. Например, следующий код:

age = int(input('Сколько вам лет?: '))

if not (age <12):

print('Доступразрешен.')

else:

print('Доступ запрещен.')

полностью эквивалентен коду:

age = int(input('Сколько вам лет?: '))

if age >= 12:

print('Доступразрешен.')

else:

print('Доступ запрещен.')

В первом примере мы поместили выражение age < 12 в скобки для того, чтобы было чётко видно, что мы применяем оператор not к значению выражения age < 12, а не только к переменной age.

Таблица истинности для оператора not:

| **a** | **not a** |
| --- | --- |
| False | True |
| True | False |

**Приоритеты логических операторов**

Логические операторы, подобно арифметическим операторам (+, -, \*, /), имеют приоритет выполнения. Приоритет выполнения следующий:

* в первую очередь выполняется логическое отрицание not;
* далее выполняется логическое умножение and;
* далее выполняется логическое сложение or.

Для **явного указания порядка** выполнения условных операторов **используют скобки**.

**Примечания:**

**Примечание 1.** Частой ошибкой у начинающих программистов является путаница логических операторов and и or. Рассмотрим два условия:

If x>1 and x<100:

If x>1 or x<100:

Верным является только первое условие. В нём проверяется, что число x находится в диапазоне от 1 до 100, другими словами,  *x* ∈(1;100). Второе условие проверяет, что число x или больше 1, или меньше 100. Однако такому условию удовлетворяет любое число!

**Примечание 2.** Другую частую ошибку видим в следующем примере кода:

if age >= 7 and <= 9:

Запуск такого кода приведет к появлению ошибки во время выполнения программы. Необходимо явно записывать условия:

if age >= 7 and age <= 9:

**Примечание 3.** Не забывайте, что в Python есть удобный способ для проверки принадлежности диапазону. Например, следующий код:

if age >= 7 and age <= 9:

полностью эквивалентен коду:

if 7 <= age <= 9:

Последний код предпочтительнее.

**Примечание 4.** Оба оператора, and и or, вычисляются по **укороченной схеме**.

Вот как это работает с оператором and. Если условие слева от оператора and является ложным, то условие справа от него не проверяется, так как результат выражения будет гарантированно ложным и проверка оставшегося условия — пустая трата процессорного времени.

Аналогично работает оператор or. Если условие слева от оператора or истинное, то условие справа от него не проверяется. Действительно, результат будет гарантированно истинным и проверка оставшегося условия станет пустой тратой процессорного времени.

## Примеры решение задач

**Задача 1.** Напишите программу, которая определяет, является ли заданное натуральное число трёхзначным.

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

if 100<= num <= 999: # num >= 100 and num <= 999

print('Число является трёхзначным')

else:

print('Число не является трёхзначным')

**Задача 2.** Напишите программу, которая проверяет, что все три цифры натурального трёхзначного числа различны.

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

num = int(input())

d3 = num % 10

d2 = num % 100 // 10

d1 = num // 100

if d3 != d2 and d3 != d1 and d2 != d1:

print ('Цифры различны')

else:

print('Цифры не различны')

**Задача 3.** Напишите программу, которая по координатам точки, не лежащей на осях координат, определяет номер координатной четверти, в которой она находится.

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

x = int(input())

y = int(input())

if x >0and y >0:

print('1 четверть')

if x <0and y >0:

print('2 четверть')

if x <0and y <0:

print('3 четверть')

if x >0and y <0:

print('4 четверть')

## Задача 7\_1. Красивое число

Назовем число красивым, если оно является четырехзначным и делится нацело на 7 или на 17. Напишите программу, определяющую, является ли введённое число красивым. Программа должна вывести «YES», если число является красивым, или «NO» в противном случае.

**Формат входных данных**  
На вход программе подаётся натуральное число.

**Формат выходных данных**  
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

1043

**Sample Output 1:**

YES

**Sample Input 2:**

1045

**Sample Output 2:**

NO

## Задача 7\_2. Неравенство треугольника

Напишите программу, которая принимает три положительных числа и определяет, существует ли невырожденный треугольник с такими сторонами (проверяем неравенство треугольника).

**Формат входных данных**  
На вход программе подаётся три положительных целых числа.

**Формат выходных данных**  
Программа должна вывести «YES» или «NO» в соответствии с условием задачи.

**Примечание.** Треугольник существует, если выполняется неравенство треугольника.

**Sample Input 1:**

5

2

3

**Sample Output 1:**

NO

**Sample Input 2:**

3

4

6

**Sample Output 2:**

YES

## Задача 7\_3. Високосный год

Напишите программу, которая определяет, является ли год с данным номером високосным. Если год является високосным, то выведите «YES», иначе выведите «NO».

Год является високосным, если его номер кратен 4, но не кратен 100, или если он кратен 400.

**Формат входных данных**  
На вход программе подаётся натуральное число.

**Формат выходных данных**  
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

2020

**Sample Output 1:**

YES

**Sample Input 2:**

2012

**Sample Output 2:**

YES

**Тема 8.Условный оператор**

**Аннотация.** Изучим вложенный и каскадный условный оператор.

**Вложенный оператор**

Внутри условного оператора можно использовать любые инструкции языка 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 следующего уровня).

if условие1:

блок кода

else:

if условие2:

блок кода

else:

if условие3:

блок кода

...

В предыдущем уроке мы разбирали задачу об определении координатной четверти точки. Программу можно переписать с использованием вложенного оператора:

x = int(input())

y = int(input())

if x >0:

if y >0:

print('Первая четверть')

else:

print('Четвертая четверть')

else:

if y >0:

print('Вторая четверть')

else:

print('Третья четверть')

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

Рассмотрим программу, которая переводит стобалльную оценку в пятибалльную. Для её реализации нужно воспользоваться вложенным условным оператором:

grade = int(input('Введите вашу отметку по 100-балльной системе: '))

if grade >= 90:

print(5)

else:

if grade >= 80:

print(4)

else:

if grade >= 70:

print(3)

else:

ifgrade>= 60:

print(2)

else:

print(1)

В этом примере уровень вложенности настолько глубок, что код становится трудно понять.

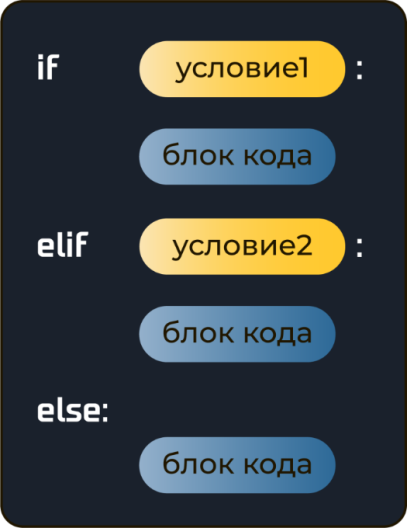
Выбор из нескольких альтернатив – это обычное дело, здесь имеет смысл избегать глубокого вложения. Для этого в Python есть **каскадный условный оператор**.

Мы не могли написать 5 независимых if-ов, поскольку в таком случае было бы напечатано сразу несколько значений пятибалльной оценки.

**Каскадный условный оператор**

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

Синтаксис каскадного условного оператора имеет следующий вид:

if условие1:

блок кода

elif условие2:

блок кода

...

else:

блок кода

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.

grade = int(input('Введите вашу отметку: '))

if grade >= 90:

print(5)

elif grade >= 80:

print(4)

elif grade >= 70:

print(3)

elif grade >= 60:

print(2)

else:

print(1)

Обратите внимание на выравнивание и выделение отступом, которые применены в инструкции if – elif - else: выражения if, elif и else выравнены и исполняемые по условию блоки выделены отступом.

Инструкция if – elif - else не является обязательной, потому что ее логика может быть запрограммирована вложенными инструкциями if - else. Однако длинная серия вложенных инструкций if - else имеет два характерных недостатка:

* программный код может стать сложным и трудным для восприятия;
* из-за необходимого выделения отступом продолжительная серия вложенных инструкций if - else может стать слишком длинной, чтобы целиком уместиться на экране монитора без горизонтальной прокрутки.

Логика инструкции if – elif - else обычно прослеживается легче, чем длинная серия вложенных инструкций if - else. И поскольку в инструкции if – elif - else все выражения выравнены, длина строк в данной инструкции, как правило, короче.

**Запомните.** Заключительный блок else в операторе if – elif - else является необязательным.

## Примеры решения задач

**Задача 1.** Даны три целых числа. Определите, сколько среди них совпадающих. Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадает) или 0 (если все числа различны).

**Решение.** Программа, решающая поставленную задачу, может иметь следующий вид:

**1 способ.** Использование вложенного условного оператора.

a, b, c = int(input()), int(input()), int(input())

if a == b:

if b == c:

print(3)

else:

print(2)

else:

if a == c:

print(2)

else:

if b == c:

print(2)

else:

print(0)

**2 способ.** Использование каскадного условного оператора.

a, b, c = int(input()), int(input()), int(input())

if a == b == c:

print(3)

elif a == b != c:

print(2)

elif a != b == c:

print(2)

elif a == c != b:

print(2)

else:

print(0)

**3 способ.** Использование каскадного условного оператора и логического оператора or.

a, b, c = int(input()), int(input()), int(input())

if a == b == c:

print(3)

elif a == b != c or a != b == c or a == c != b:

print(2)

else:

print(0)

## Задача 8\_1. Гонка спидстеров

Зум бросил вызов Флэшу и предложил ему честный поединок в виде гонки вокруг магнетара. В случае проигрыша эта нейтронная звезда зарядится и уничтожит мир, поэтому Флэш решил не рисковать без причины, и узнать у своего друга Циско Рамона есть ли смысл принимать вызов. Циско получил данные, что скорость Зума равна n, а скорость Флэша равна k.

Напишите программу, которая должна вывести ответ Циско на вопрос Флэша.

**Формат входных данных**  
На вход программе подаётся два целых числа n и k, скорость Зума и Флэша.

**Формат выходных данных**  
Если Зум быстрее Флэша нужно вывести «NO», если Флэш быстрее Зума нужно вывести «YES», если их скорости равны нужно вывести "Don't know".

**Sample Input 1:**

2204

1505

**Sample Output 1:**

NO

**Sample Input 2:**

2344

4324

**Sample Output 2:**

YES

## Задача 8\_2. Вид треугольника

Напишите программу, которая принимает три положительных числа и определяет вид треугольника, длины сторон которого равны введенным числам.

**Формат входных данных**  
На вход программе подаются три числа – длины сторон существующего треугольника.

**Формат выходных данных**  
Программа должна вывести на экран текст – вид треугольника («Равносторонний», «Равнобедренный» или «Разносторонний»).

**Sample Input 1:**

145

145

139

**Sample Output 1:**

Равнобедренный

**Sample Input 2:**

59

59

59

**Sample Output 2:**

Равносторонний

## Задача 8\_3. Среднее число

Даны три различных целых числа. Напишите программу, которая находит среднее по величине число.

**Формат входных данных**  
На вход программе подаётся три различных целых числа, каждое на отдельной строке.

**Формат выходных данных**  
Программа должна вывести среднее число.

**Примечание.** Средним называется число, которое будет вторым, если три числа отсортировать в порядке возрастания.

**Sample Input 1:**

1

2

3

**Sample Output 1:**

2

**Sample Input 2:**

10

30

20

**Sample Output 2:**

20

**Sample Input 3:**

67

100

54

**Sample Output 3:**

67

## Задача 8\_4. Количество дней

Дан порядковый номер месяца (1,2,…, 12). Напишите программу, которая выводит на экран количество дней в этом месяце. Принять, что год является невисокосным.

**Примечание**. Постарайтесь написать программу, так чтобы в ней было не более трех условий.

**Формат входных данных**  
На вход программе подаётся одно целое число – порядковый номер месяца.

**Формат выходных данных**  
Программа должна вывести количество дней в этом месяце.

**Sample Input 1:**

12

**Sample Output 1:**

31

**Sample Input 2:**

5

**Sample Output 2:**

31

## Задача 8\_5. Церемония взвешивания

Известен вес боксера-любителя (целое число). Известно, что вес таков, что боксер может быть отнесён к одной из трех весовых категорий:

1. Легкий вес – до 60 кг;
2. Первый полусредний вес – до 64 кг;
3. Полусредний вес – до 69 кг.

Напишите программу, определяющую, в какой категории будет выступать данный боксер.

**Формат входных данных**  
На вход программе подаётся одно целое число.

**Формат выходных данных**  
Программа должна вывести текст – название весовой категории.

**Sample Input 1:**

55

**Sample Output 1:**

Легкий вес

**Sample Input 2:**

68

**Sample Output 2:**

Полусредний вес

## Задача 8\_6. Самописный калькулятор

Напишите программу, которая считывает с клавиатуры два целых числа и строку. Если эта строка является обозначением одной из четырёх математических операций (+, -, \*, /), то выведите результат применения этой операции к введённым ранее числам, в противном случае выведите «Неверная операция». Если пользователь захочет поделить на ноль, выведите текст «На ноль делить нельзя!».

**Формат входных данных**  
На вход программе подаются два целых числа, каждое на отдельной строке, и строка.

**Формат выходных данных**  
Программа должна вывести результат применения операции к введенным числам или соответствующий текст, если операция неверная, либо если происходит деление на ноль.

**Sample Input 1:**

6104

0

/

**Sample Output 1:**

На ноль делить нельзя!

**Sample Input 2:**

25

5

\*

**Sample Output 2:**

125

**Задача 8\_7. Цветовой микшер**

Красный, синий и желтый называются основными цветами, потому что их нельзя получить путем смешения других цветов. При смешивании двух основных цветов получается вторичный цвет:

* если смешать красный и синий, то получится фиолетовый;
* если смешать красный и желтый, то получится оранжевый;
* если смешать синий и желтый, то получится зеленый.

Напишите программу, которая считывает названия двух основных цветов для смешивания. Если пользователь вводит что-нибудь помимо названий «красный», «синий» или «желтый», то программа должна вывести сообщение об ошибке. В противном случае программа должна вывести название вторичного цвета, который получится в результате.

**Формат входных данных**  
На вход программе подаются две строки, каждая на отдельной строке.

**Формат выходных данных**  
Программа должна вывести полученный цвет смешения либо сообщение «ошибка цвета», если введён был не цвет.

**Примечание 1.** Если смешать красный и красный, то получится красный и т.д.

**Примечание 2.** Поиграйтесь с настоящим цветовым микшером 🙂

**Sample Input 1:**

красный

синий

**Sample Output 1:**

фиолетовый

**Sample Input 2:**

красный

блаблабла

**Sample Output 2:**

ошибка цвета

**Задача 8\_8.Цвета колеса рулетки**

На колесе рулетки карманы пронумерованы от 0 до 36. Ниже приведены цвета карманов:

* карман 0 зеленый;
* для карманов с 1 по 10 карманы с нечетным номером имеют красный цвет, карманы с четным номером – черный;
* для карманов с 11 по 18 карманы с нечетным номером имеют черный цвет, карманы с четным номером – красный;
* для карманов с 19 по 28 карманы с нечетным номером имеют красный цвет, карманы с четным номером – черный;
* для карманов с 29 по 36 карманы с нечетным номером имеют черный цвет, карманы с четным номером – красный.

Напишите программу, которая считывает номер кармана и показывает, является ли этот карман зеленым, красным или черным. Программа должна вывести сообщение об ошибке, если пользователь вводит число, которое лежит вне диапазона от 0 до 36.

**Формат входных данных**  
На вход программе подаётся одно целое число.

**Формат выходных данных**  
Программа должна вывести цвет кармана либо сообщение «ошибка ввода», если введённое число лежит вне диапазона от 0 до 36.

**Sample Input 1:**

0

**Sample Output 1:**

зеленый

**Sample Input 2:**

1

**Sample Output 2:**

красный

**Sample Input 3:**

37

**Sample Output 3:**

ошибка ввода

**Задача 8\_9. Пересечение отрезков**

На числовой прямой даны два отрезка: [*a*1​;  *b*1​] и [*a*2​; *b*2​]. Напишите программу, которая находит их пересечение.

Пересечением двух отрезков может быть:

* отрезок;
* точка;
* пустое множество.

**Формат входных данных**  
На вход программе подаются 4 целых числа a\_1, b\_1, a\_2, b\_2 каждое на отдельной строке. Гарантируется, что a\_1 < b\_1​​ и a\_2 < b\_2

**Формат выходных данных**  
Программа должна вывести на экран границы отрезка, являющегося пересечением, либо общую точку, либо текст «пустое множество».

**Sample Input 1:**

1

3

2

4

**Sample Output 1:**

2 3

**Sample Input 2:**

1

2

3

4

**Sample Output 2:**

пустое множество