**Условный оператор if. Синтаксис условной инструкции.**

**Условная инструкция**

Все ранее рассматриваемые программы имели линейную структуру: все инструкции выполнялись последовательно одна за одной, каждая записанная инструкция обязательно выполнялась.

Допустим, мы хотим по данному числу xx определить его абсолютную величину (модуль). Программа должна напечатать значение переменной xx, если x>0x>0, или же величину −x−x в противном случае. Линейная структура программы нарушается: в зависимости от справедливости условия x>0x>0 должна быть выведена одна или другая величина. Соответствующий фрагмент программы на Python имеет вид:

x = int(input())

**if** x > 0:

**print**(x)

**else**:

**print**(-x)

В этой программе используется условная инструкция **if** (если). После слова **if** указывается проверяемое условие x > 0, завершающееся двоеточием. После этого идет блок (последовательность) инструкций, который будет выполнен, если условие истинно. В нашем примере это вывод на экран величины x. Затем идет слово **else** (иначе), также завершающееся двоеточием, и блок инструкций, который будет выполнен, если проверяемое условие неверно, то есть будет выведено значение -x.

Итак, условная инструкция в Python имеет следующий синтаксис:

**if** Условие:

Блок инструкций 1

**else**:

Блок инструкций 2

Блок инструкций 1 будет выполнен, если Условие истинно. Если Условие ложно, будет выполнен Блок инструкций 2.

Для выделения блока инструкций, относящихся к инструкции **if** или **else**, в языке Python используются отступы. Все инструкции, которые относятся к одному блоку, должны иметь равную величину отступа, то есть одинаковое число пробелов в начале строки. Рекомендуется использовать отступ в 44 пробела и **не рекомендуется** использовать в качестве отступа символ табуляции.

Это одно из существенных отличий синтаксиса Python от синтаксиса большинства языков, в которых блоки выделяются специальными словами или фигурными скобками.

**Неполное ветвление**

В условной инструкции может отсутствовать слово **else** и последующий блок. Такая инструкция называется неполным ветвлением. Например, если дано число xx, и мы хотим заменить его на абсолютную величину xx, то это можно сделать следующим образом:

**if** x < 0:

x = -x

**print**(x)

В этом примере переменной x будет присвоено значение -x, но только в том случае, когда x < 0 . А вот инструкция **print**(x) будет выполнена всегда, независимо от проверяемого условия.

**Пример: Максимум из двух чисел**

Рассмотрим следующую задачу. Нам даны два числа, необходимо найти наибольшее из них и напечатать его на экран.

Эта задача решается с помощью условной инструкции:

a = int(input())

b = int(input())

**if** a > b:

m = a

**else**:

m = b

**print**(m)

Также эту задачу можно решить с помощью неполного ветвления. В этом случае нам не нужно использовать дополнительную переменную:

a = int(input())

b = int(input())

**if** b > a:

a = b

**print**(a)

**Тип данных bool, логические операции.**

### Операции сравнения

Как правило, в качестве проверяемого условия используется результат вычисления одной из следующих операций сравнения (или величин логического типа):

< — меньше. Условие верно, если первый операнд меньше второго.

> — больше. Условие верно, если первый операнд больше второго.

<= — меньше или равно. Условие верно, если первый операнд не больше второго.

>= — больше или равно. Условие верно, если первый операнд не меньше второго.

== — равенство. Условие верно, если два операнда равны. Обратите внимание, что просто знак равенства для сравнения двух значений использовать нельзя, так как он означает присваивание.

!= — неравенство. Условие верно, если операнды не равны.

**Пример: Автопробег**

Рассмотрим задачу из предыдущего модуля. За день машина проезжает nn километров. Сколько дней нужно, чтобы проехать маршрут длиной mm километров?

Данную задачу с использованием условной инструкции можно решить следующим образом:

n = int(input())

m = int(input())

**if** m % n == 0:

**print**(m // n)

**else**:

**print**(m // n + 1)

Также эту задачу можно было решить и одной формулой:

**print**((m + n - 1) // n)

### Тип данных bool

Операции сравнения возвращают значения специального логического типа bool. Выражения логического типа могут принимать одно из двух значений: True (истина) или False (ложь).

Если преобразовать логическое True к типу int, то получится 11, а преобразование False даст 00. При обратном преобразовании число 00 преобразуется в False, а любое ненулевое число — в True . При преобразовании str в bool пустая строка преобразуется в False, а любая непустая строка — в True .

Заметим, что задачу про автопробег теперь можно решить ещё одним способом:

**print**(m // n + (m % n != 0))

### Логические операции

Иногда нужно проверить одновременно не одно, а несколько условий. В этом помогают логические операции. В Python существуют стандартные логические операции: логическое И, логическое ИЛИ, логическое отрицание.

* **Логическое И** (конъюнкция) является бинарной операцией (то есть операцией с двумя операндами: левым и правым) и имеет вид **and**. Операция **and** возвращает True тогда и только тогда, когда оба её операнда имеют значение True.
* **Логическое ИЛИ** (дизъюнкция) является бинарной операцией и возвращает True тогда и только тогда, когда хотя бы один операнд равен True. Операция «логическое ИЛИ» имеет вид **or**.
* **Логическое НЕ** (отрицание) является унарной (то есть с одним операндом) операцией и имеет вид **not**, за которым следует единственный операнд. Логическое НЕ возвращает True, если операнд равен False, и наоборот.

**Примеры:**

Проверим, что число mm является корректным номером месяца (лежит в интервале от 11 до 1212):

**if** m >= 1 **and** m <= 12:

**print**("Корректно")

Также это можно проверить несколько иначе:

**if** 1 <= m <= 12:

**print**("Корректно")

Проверим, что число mm является номером зимнего месяца (декабрём, январём или февралём):

**if** m == 1 **or** m == 2 **or** m == 12:

**print**("Зима")

Проверим, что месяц с номером mm не зимний:

**if** **not**(m == 1 **or** m == 2 **or** m == 12):

**print**("Не Зима")

Кроме того, операции сравнения в Python можно объединять в цепочки (в отличие от большинства других языков программирования, где для этого нужно использовать логические связки), например,

x == 0 == y

или

10 <= n <= 100

**Вложенные условные инструкции**

### Вложенные условные инструкции

**Пример: Високосный год**

Дано натуральное число. Требуется определить, является ли год с данным номером високосным. Если год является високосным, то выведите YES, иначе выведите NO. Напомним, что в соответствии с григорианским календарем, год является високосным, если его номер делится на 44, но при этом не делится на 100100, или если он кратен 400400.

Решим задачу с помощью вложенных условных инструкций:

year = int(input())

**if** year % 100 == 0:

**if** year % 400 == 0:

**print**("YES")

**else**:

**print**("NO")

**else**:

**if** year % 4 == 0:

**print**("YES")

**else**:

**print**("NO")

Также эту задачу можно решить с помощью каскадных условных инструкций:

year = int(input())

**if** year % 400 == 0:

**print**("YES")

**elif** year % 100 == 0:

**print**("NO")

**elif** year % 4 == 0:

**print**("YES")

**else**:

**print**("NO")