Python.
Логические выражения. Логические операции.

В правой части инструкции присваивания может стоять не только арифметическое выражение, но и выражение другого типа, например, логического.

Логическое выражение — это совокупность арифметических выражений, объединенная логическими операциями и операциями отношения и принимающее только 2 значения — ***True*** (правда) или ***False*** (ложь) (0 или 1). ***True*** и ***False*** пишутся только с большой буквы.

Наименование "булевский" выбрано в честь английского математика Джорджа Буля, заложившего основы математической логики. Термины булевский и логический обычно употребляются как синонимы. В Python этот тип имеет название ***bool***.

Пример 1.

```
a = True
print(a)
```

В этом примере переменная ***a*** будет типа ***bool***, а в результате работы программы будет выведено ***True***.

Как уже говорилось, логическое выражение может включать в себя: арифметические выражения, операции отношения и логические операции. Что такое оперции отношения и логические операции?

Операции отношения

Операции отношения предназначены для сравнения двух величин. Результат сравнения имеет значение ***True*** или ***False***.

==	—	равно
!=	—	не равно
<	—	меньше
<=	—	меньше или равно
>	—	больше
>=	—	больше или равно

Пример 2.

```
x = 2
ok = x > 0
exist = x == 3 - 27
```

В результате выполнения этой программы переменная ***ok*** примет значение ***True***, а переменная ***exist*** — значение ***False***.

Логические операции

Логические операции применяются к величинам логического типа, результатом выполнения операции тоже является величина логического типа.

Рассмотрим следующие логические операции:

- ***not*** (отрицание, унарная операция)
- ***and*** (и) (логическое умножение)
- ***or*** (или) (логическое сложение).

Таблица значений логических операций

X	Y	not X	X and Y	X or Y
False	*False*	*True*	*False*	*False*
False	*True*	*True*	*False*	*True*
True	*False*	*False*	*False*	*True*
True	*True*	*False*	*True*	*True*

Значение выражения вычисляется в определенном порядке.

Таблица приоритета выполнения операций

Тип действий	Операции
Вычисления в круглых скобках	*( )*
Вычисления значений функций	*функции*
Операция возведения в степень	**
Унарные операции	*унарный “-”*
Операции типа умножения	**, /, //, %*
Операции типа сложения	*+, -*
Операции отношения	*==, !=, <, >, <=, >=, is, is not*
Логическое отрицание	*not*
Логическое умножение	*and*
Логическое сложение	*or*

Операции одинакового приоритета выполняются слева направо в порядке их следования в выражении.

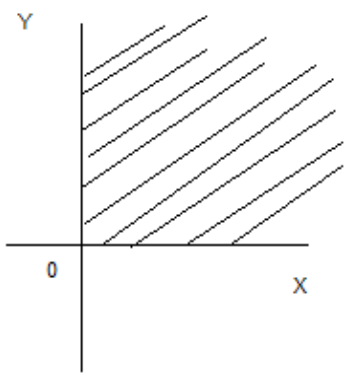
Пример 3.

```
a = 17 // 5 * 5 + (17 % 5)
b = 4 + abs(-5 + 9)
c = a < 3
d = a * 2 > b or not a - 1 > 10 and c
print(d)
```

В результате работы программы будет выведено ***True***.

В качестве примера рассмотрим еще одну задачу. Пусть заданы координаты точки (x, y) на плоскости. Определить, попадает ли точка в заштрихованную область. Если попадает, то вывести ***True***, в противном случае – ***False.***

```
x = float(input('Введите координату X: '))
y = float(input('Введите координату Y: '))
c = (x > 0) and (y > 0)
print(c)
```



Математическая запись -4 < x ≤ 18.3 на языке Python запишется в виде: `x > -4 and x<=18.3`.

Допустимо и такое описание `-4 < x <= 18.3`.

Например, следующая программа

```
x = 5
y = x < 10 < x*10 < 100
print(y)
```

выведет ***True***

Хоть множественные неравенства и допустимы в языке Python, лучше не злоупотреблять этой возможностью.

Приведение типов.

В некоторых случаях python автоматически умеет приводить типы данных, но для явного приведения данных к типу *bool* необходимо использовать функцию *bool()*.

При применении функции *bool()* к любому числу отличному от нуля (как целому, так и вещественному) будет получен результат ***True***, и только при нулевом значении аргумента будет получен результат ***False***.

Например, при выполнении следующей программы

```
print(bool(1))
print(bool(27/45))
print(bool(2**17))
print(bool(-0.00001))
```

все 4 раза будет выведено ***True***.

При выполнении же программы

```
print(bool(0))
print(bool(0.00))
print(bool(2-2))
print(bool(101.1-101.1))
```

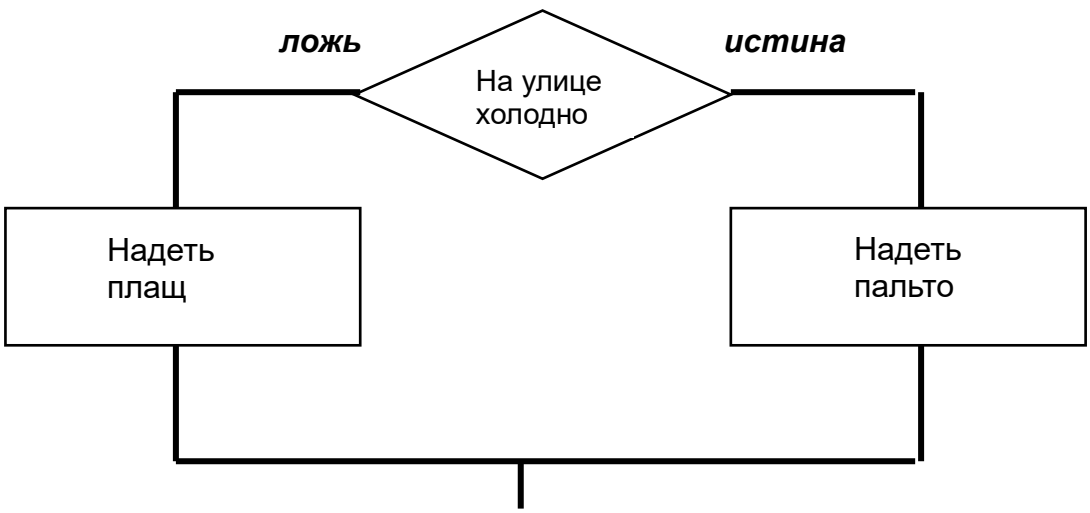
четыре раза будет выведено ***False***.

При преобразовании логического значения в целое число получим ***1***, если логическое значение ***True,*** в противном случае - ***0.***

При преобразовании логического значения в вещественное число получим соответственно ***1.0*** и ***0.0.***

Организация ветвлений в программе

В жизни мы часто встречаемся с проблемой выбора. Например, если на улице холодно, мы надеваем пальто, иначе — надеваем плащ. Эту ситуацию можно представить в виде следующей схемы:



В зависимости от истинности утверждения в ромбе “на улице холодно” выполняется либо действие “надеть пальто”, либо действие “надеть плащ”.

Такая структура называется структурой ветвления

Ситуация, связанная с выбором одной из двух альтернатив, встречается в программировании довольно часто. В языке Python структуре выбора соответствует инструкция ***if ... : ...else : …*** .

Формат описания:

```
if <логическое выражение> :
    <инструкция 1.1>
    <инструкция 1.2>
…
    <инструкция 1.n>
else:
    <инструкция 2.1>
    <инструкция 2.2>
…
    <инструкция 2.m>
```

Порядок выполнения инструкции ***if ... : ...else : …***

- сначала вычисляется значение логического выражения,
- если значение выражения ***True***, то выполняются *инструкции с 1.1 по 1.n*,
- иначе (если значение выражения ***False***), выполняются *инструкции с 2.1 по 2.m*,.

Обратите внимание на синтаксис написания оператора. ***if*** и ***else*** должны иметь одинаковый отступ. После логического выражения и слова ***else*** должно стоять двоеточие. Инструкции долдны иметь отступ вправо на 4 позиции относительно ***if*** (правда многие современные среды допускают отступ и на другое количество позиций, но в любом случае все инструкции внутри ***if*** или ***else*** должны иметь одинаковый отступ).

Иногда структура выбора выглядит несколько иначе:

Такой структуре в языке Python соответствует краткая инструкция ***if***....

Формат описания:

if <логическое выражение> :
    <инструкция 1.1>

    <инструкция 1.2>

…

    <инструкция 1.n>

Краткая инструкция выполняется аналогично полной, то есть вычисляется значение логического выражения, если его значение ***True***, то выполняются *инструкции с 1.1 по 1.n*, иначе выполняется следующая за блоком ***if***... инструкция.

Инструкции, находяциеся внутри веточки ***if*** или ***else***, сами могут быть инструкциями выбора, в этом случае инструкция ***if***... называется вложенной. Например,

```
if <логическое выражение 1> :
    <инструкция 1>
else :
    if <логическое выражение 2> :
        <инструкция 2>
    else:
        <инструкция 3>
```

Каждая из инструкций (инструкция 1, инструкция 2, инструкция 3) может быть также составной.

В некоторых случаях вместо вложенных инструкций ***if*** удобно использовать какскадные операторы **if**.

```
if <логическое выражение 1> :
    <инструкция 1>
elif <логическое выражение 2> :
    <инструкция 2>
elif <логическое выражение 3> :
    <инструкция 3>
        …
elif <логическое выражение n> :
        <инструкция n>
else:
    <инструкция x>
```

Логика работы данного оператора следующая.

Если, логическое выражение 1 истинно, то выполняется инструкция 1, в противном случа вычисляется значение логического выражения 2 и т.д. Ели все логические выражения ложь, то выполняется инструкция x. Веточка ***else*** может отсутствовать, тогда не выполняется никакая инструкция из каскадного оператора ветвления, а упраление переходит на первую строчку после этого оператора. И, конечно, на месте любой из инструкций может стоять любое количество инструкций (в том числе и операторов выбора).

Тернарная условная операция

Иногда бывают ситуации, когда некая переменная в зависимости от условий может принимать только одно из двух значений.Например, если ученик набрал более 50 баллов, то он сдал зачет, и переменная *itog* должна получить значение 5, в противном случае в переменную *itog* должно быть записано число 2.

Программа могла бы выглядеть следующим образом.

```
x=int(input('Введите количество баллов'))
if x>50:
    itog = 5
else:
    itog = 2
print(itog)
```

Однако существует более короткая запись

```
x=int(input('Введите количество баллов '))
itog = 5 if x>50 else 2
print(itog)
```

В последней программе применена тернарная условная операция 5 if x>50 else 2. Любая тернарная условная операия может быть частью выражения, а так же может быть вложена в другую тернарную условную операцию.