**Лекция 2**

**Логические выражения в Python**

Логический тип данных (или Boolean) – это примитивный тип, который принимает одно из двух возможных значений: истину (True) или ложь (False). Этот тип используется во многих языках программирования для построения алгоритмов и управления поведением программ.

**Операторы сравнения**

В программировании операторы сравнения используются при оценке и сравнении значений для последующего сведения их к одному логическому значению (True или False).

Операторы сравнения Python 3 представлены в этой таблице:

| Оператор | Значение |
| --- | --- |
| == | Проверяет равенство между компонентами; условие истинно, если компоненты равны. |
| != | Проверяет равенство между компонентами; условие истинно, если компоненты НЕ равны. |
| < | Оценивает значение левого компонента; условие истинно, если он меньше, чем правый. |
| > | Оценивает значение левого компонента; условие истинно, если он больше, чем правый. |
| <= | Оценивает значение левого компонента; условие истинно, если он меньше или равен правому компоненту. |
| >= | Оценивает значение левого компонента; условие истинно, если он больше или равен правому компоненту. |

Пример.

Создайте пару переменных:

x = 5  
y = 8

Теперь сравните значения переменных с помощью вышеперечисленных операторов.

x = 5  
y = 8  
print("x == y:", x == y)  
print("x != y:", x != y)  
print("x < y:", x < y)  
print("x > y:", x > y)  
print("x <= y:", x <= y)  
print("x >= y:", x >= y)  
x == y: False  
x != y: True  
x < y: True  
x > y: False  
x <= y: True  
x >= y: False

Python оценивает соотношения между значениями переменных так:

* 5 равно 8? Ложь
* 5 не равно 8? Истина
* 5 меньше 8? Истина
* 5 больше 8? Ложь
* 5 меньше или равно 8? Истина
* 5 больше или равно 8? Ложь

Также операторы сравнения можно применять к числам с плавающей точкой и строкам.

Строки чувствительны к регистру; чтобы отключить такое поведение, нужно использовать специальный метод.

Попробуйте сравнить две строки:

Hello = "Hello"  
hello = "hello"  
print("Hello == hello: ", Hello == hello)  
Hello == hello:  False

Строки Hello и hello содержат одинаковый набор символов, однако они не равны, поскольку одна из них содержит символы верхнего регистра. Попробуйте добавить ещё одну переменную, которая также будет содержать символы верхнего регистра, а затем сравните их.

Hello = "Hello"  
hello = "hello"  
Hello\_there = "Hello"  
print("Hello == hello: ", Hello == hello)  
print("Hello == Hello\_there", Hello == Hello\_there)  
Hello == hello:  False  
Hello == Hello\_there:  True

Также для сравнения строк можно использовать операторы > и <. Python выполнит лексикографическое сравнение строк на основе значений символов ASCII.

Операторы сравнения можно применять к логическим значениям True и False:

t = True  
f = False  
print("t != f: ", t != f)  
t != f:  True

Обратите внимание на разницу между операторами = и ==.

x = y   # Оператор присваивания. Устанавливает равенство между x и y (то есть присваивает x значение y).  
x == y  # Оператор сравнения. Проверяет равенство между x и y и оценивает выражение как истинное или ложное. Выражение истинно, если x и y равны.

**Логические операторы**

Для сравнения значений используется три логических оператора, которые сводят результат к логическому значению True или False.

| Оператор | Значение |
| --- | --- |
| and | Оператор «и»: выражение истинно, если оба его компонента истинны. |
| or | Оператор «или»: выражение истинно, если хотя бы один из его компонентов истинен. |
| not | Оператор «не»: изменяет логическое значение компонента на противоположное. |

Логические операторы обычно используются для оценки двух или больше выражений. Например, их можно использовать в программе, которая проверит:

* сдал ли студент экзамен
* и зарегистрирован ли он.

Если оба значения истинны, студент будет переведён на следующий курс.

Другой пример: программа с логическими операторами может проверять активность пользователя в онлайн-магазине:

* использовал ли он кредит магазина
* или заказывал ли он товары в течение последних 6 месяцев.

Для примера попробуйте сравнить три выражения:

print((9 > 7) and (2 < 4))  # Оба выражения истинны (True)  
print((8 == 8) or (6 != 6)) # Одно из выражений истинно (True)  
print(not(3 <= 1))          # Выражение ложно (False)  
True  
True  
True

В первом случае оба выражения истинны, потому оператор and возвращает True.

Во втором случае истинно только значение 8 == 8. Поскольку хотя бы одно из предложенных условий истинно, оператор or возвращает True. Оператор and в таком случае выдал бы False.

В третьем случае выражение 3 <= 1 ложно. Оператор not изменяет полученное логическое значение на противоположное: not False = True.

Теперь попробуйте сравнить числа с плавающей точкой.

print((-0.2 > 1.4) and (0.8 < 3.1)) # Одно из выражений ложно (False)  
print((7.5 == 8.9) or (9.2 != 9.2)) # Оба выражения ложны (False)  
print(not(-5.7 <= 0.3))             # Выражение истинно (True)

* Поскольку в первом примере одно из выражений ложно, and вернёт False. Оператор and оценивает выражение как истинное только тогда, когда оба компонента истинны.
* Поскольку оба выражения ложны, оператор or выдаст False.
* Поскольку выражение истинно, оператор not вернёт False (not True = False).

Логические операторы можно объединять в составные выражения:

not((-0.2 > 1.4) and ((0.8 < 3.1) or (0.1 == 0.1)))

Выражение (0.8 < 3.1) or (0.1 == 0.1) истинно, поскольку оба математических выражения, из которых оно состоит, истинны. Оператор or вернёт True.

Полученное значение True становится компонентом следующего выражения: (-0.2 > 1.4) and (True). Оператор and выдаст False, потому что выражение -0.2 > 1.4 ложно. (False) and (True) = False.

Далее оператор not заменит полученное значение False на обратное ему логическое значение: not(False) = True. Значит, результат будет таким:

True

**Таблицы истинности**

Математическая логика – очень полезная в программировании область знаний. В данном руководстве мы ознакомимся с основными её аспектами.

Ниже представлены таблицы истинности для оператора сравнения == и всех логических операторов. Ими можно пользоваться при написании логических выражений. В таблицах перечислены общие случаи использования оператора, потому их рекомендуется выучить наизусть.

Таблица истинности оператора ==

| x | == | y | Результат |
| --- | --- | --- | --- |
| True | == | True | True |
| True | == | False | False |
| False | == | True | False |
| False | == | False | True |

Таблица истинности оператора AND

| x | and | y | Результат |
| --- | --- | --- | --- |
| True | and | True | True |
| True | and | False | False |
| False | and | True | False |
| False | and | False | False |

Таблица истинности оператора OR

| x | or | y | Результат |
| --- | --- | --- | --- |
| True | or | True | True |
| True | or | False | True |
| False | or | True | True |
| False | or | False | False |

Таблица истинности оператора NOT

| not | x | Результат |
| --- | --- | --- |
| not | True | False |
| not | False | True |

Таблицы истинности – общие математические таблицы, которые используются в логике. Их полезно выучить наизусть, чтобы затем применять при построении алгоритмов и написании программ.

**Использование логических операторов для управления потоком**

Для управления результатом и потоками данных программы можно использовать условные операторы (condition) с выражениями (clause).

Условные операторы оценивают значение как истинное или ложное.

Выражение – это блок кода, который идёт после условного оператора и определяет результат программы.

Ниже приведён блок кода, который показывает, как объединить условные операторы для управления потоком программы Python.

if grade >= 65:             # условие  
print("Passing grade")      # выражение  
else:  
print("Failing grade")

Эта программа оценивает результат каждого студента и определяет, сдал он экзамен или нет. К примеру, если студент набрал 83, первое условие будет иметь значение True, и программа выведет на экран строку:

Passing grade

Если же студент набрал 59 баллов, первое условие будет ложно, потому программа выдаст:

Failing grade



Элементы отсортированы по убыванию, с высокого приоритета к более низкому. В комбинациях с несколькими операторами оператор с большим приоритетом выполняется первым.

Например, в выражении 1 + 2 \*\* 2 \* 10 сначала выполнится возведение в степень, далее умножение, потом сложение. Итого результат: 41.