## 

## Логические выражения и операторы

Логические выражения — любые конструкции, результатом выполнения которых является True или False.

в программировании оценка выражений всегда сводится к одному из этих двух вариантов. Для представления истинности и ложности используются ключевые слова True и False соответственно.

***Операции сравнения***

Операции сравнения используются для создания условий на основании сравнения элементов. Элементы, к которым применяется операция, называют операндами. Python поддерживает следующие операции сравнения:

== — «равно»: истинно (True), если оба операнда равны, иначе — ложно False;

!= — «не равно»: истинно (True), если оба операнда не равны, иначе — ложно False;

> — «больше чем»: истинно (True), если первый операнд больше второго, иначе — ложно False;

< — «меньше чем»: истинно (True), если первый операнд меньше второго, иначе — ложно False;

>= — «больше или равно»: истинно (True), если первый операнд больше или равен второму, иначе — ложно False;

<= — «меньше или равно»: истинно (True), если первый операнд меньше или равен второму, иначе — ложно False.

M\_2\_1 str – 1-20

***Использование значений в качестве условий***

В Python существуют правила, согласно которым можно привести любое значение любого типа к логическому. Так, если вместо условия написать строку или число, программа заменит его на True или False.

Значения, которые будут заменены на False:

■структуры, не содержащие элементы:

* строки без символов;
* пустые списки, словари, массивы и т.д.

■нули любых численных типов:

* 0;
* 0.0.

■константа None.

Значения, которые будут заменены на True:

■структуры, содержащие элементы:

* строки любой длины, отличной от нуля;
* списки, словари, массивы и т.д. с элементами;

■ненулевые значения любых численных типов:

* 1, 2, 3…; •0.1, 1.2, 2.3… .

С помощью функции bool() можно проверить результат любого такого преобразования:

M\_2\_1 str – 22-26

**Сравнение строк**

Чтобы определить, что одна строка больше другой, Python использует «алфавитный» или «лексикографический» порядок. Другими словами, строки сравниваются посимвольно. Например:

**print**('Я' > 'А' )

**print**( 'Кот' > 'Код' )

**print**( 'Сонный' > 'Сон' )

Алгоритм сравнения двух строк довольно прост:

1. Сначала сравниваются первые символы строк.
2. Если первый символ первой строки больше (меньше), чем первый символ второй, то первая строка больше (меньше) второй.
3. Если первые символы равны, то таким же образом сравниваются уже вторые символы строк.

Сравнение продолжается, пока не закончится одна из строк. Если обе строки заканчиваются одновременно, и все их соответствующие символы равны между собой, то строки считаются равными. Иначе, большей считается более длинная строка.

В примерах выше сравнение 'Я' > 'А' завершится на первом шаге, тогда как строки "Кот" и "Код" будут сравниваться посимвольно:

1. К равна К.
2. о равна о.
3. т больше чем д.

На этом сравнение заканчивается. Первая строка больше.

M\_2\_1 str – 28-30

## Понятие «блока» выполнения

Ранее был рассмотрен общий случай использования оператора ветвления, в котором упоминался «набор инструкций». Однако, каким образом определить, что инструкция принадлежит к этому набору или, другими словами, «блоку выполнения»?

***Определение блока выполнения***

В Python, в отличие от многих других языков, инструкции, принадлежащие к одному блоку выполнения, достаточно выделить одинаковым отступом. Концом блока выполнения считается последняя инструкция, которая будет выделена соответствующим отступом.

Отступы должны состоять из пробелов и, согласно [руководству по оформлению кода,](https://www.python.org/dev/peps/pep-0008/) их количество желательно делать кратным 4 (т. е. 4 пробела, 8, 16…). Хотя если блок будет выделен отступом в 2 пробела, программа также будет работать. Кроме того, в последних версиях языка для отступа нельзя использовать табуляцию, однако современные среды разработки (в сокращенном варианте «IDE») автоматически размещают 4 пробела при нажатии на клавиатуре кнопки «Tab».

**Конструкция if – elif – else**

предположим, что хотим определить знак числа в переменной x. Конечно, проверку можно записать вот так:

x = -5

**if** x < 0 : **print**("x отрицательное число")

**if** x >= 0 : **print**("x неотрицательное число")

Но можно сделать лучше. Смотрите, мы здесь имеем дело со взаимоисключающими условиями, то есть, они не могут произойти одновременно: либо первое, либо второе. Для таких ситуаций можно использовать ключевое слово else – иначе, чтобы ускорить процесс проверки:

x = 5

**if** x < 0:

**print**("x отрицательное число")

**else**:

**print**("x неотрицательное число")

Теперь, у нас здесь всего одно условие. Если оно истинно, то выполнится первый print, а иначе – второй. Такая программа будет работать быстрее. И обратите внимание на синтаксис записи функции print: перед ней необходимо записать хотя бы один пробел (обычно ставится 4 пробела или символ табуляции). Эти пробелы в Python означают блок кода, который выполняется по некоторому условию.

Если же мы хотим по такому принципу выполнить три проверки: x > 0; x < 0; x == 0, то предыдущую программу можно записать так:

**if** x < 0:

**print**("x отрицательное число")

**elif** x > 0:

**print**("x положительное число")

**else**:

**print**("x равен 0")

И вообще таких конструкций

if – elif – elif –…– elif – else

может быть много. Далее, обратим внимание на такой факт: во всех наших примерах по условию шел один оператор – print.

Но что если нужно выполнить несколько операторов по некоторому условию? Для этого их нужно записывать по такому синтаксису:Например:

x = -10; sgn = 0

**if** x < 0:

    sgn = -1

**print**("x отрицательное число", sgn)

**elif** x > 0:

    sgn = 1

**print**("x положительное число", sgn)

**else**:

**print**("x равен 0", sgn)

Здесь по первым двум условиям выполняется два оператора: присвоение значения переменной sgn и вывод результата в консоль.

В ряде случаев конструкцию if-else удобнее записывать через **тернарный условный оператор**, который имеет такой синтаксис:

**result = значение1 if <условие> else значение2**

При истинности условия возвращается значение1, в противном случае – значение2. Например:

age = 18

accessAllowed = True **if** age >= 18 **else** False

**print**(accessAllowed)

Получим True, если возраст (age) больше или равен 18, иначе – False. Кстати, проверку из данного примера можно сделать короче, просто прописав

accessAllowed = age >= 18

здесь оператор >= вернет True при возрасте больше или равен 18 и False – в противном случае.

***Логические операции***

Логические операции позволяют комбинировать несколько логических выражений в одно. В Python имеются следующие логические операторы:

■ and

«Логическое умножение»: возвращает True, если оба выражения равны True.

■ or

«Логическое сложение»: возвращает True, если хотя бы одно из выражений равно True.

■ not

«Логическое отрицание»: возвращает значение, обратное операнду.

Обратите внимание, что если один из операндов оператора and возвращает False, то другой операнд не оценивается, так как результатом выполнения оператора в любом случае будет False. Аналогично, если один из операндов оператора or возвращает True, то второй операнд не оценивается, так как результатом выполнения оператора в любом случае будет True.

 Приоритет операций: сначала выполняется not, затем and, затем or. Как и с математическими операциями, для переопределения порядка выполнения можно использовать скобки.

Теперь, когда мы разобрались с базовыми моментами проверки условий, сделаем следующий шаг и попробуем реализовать проверку попадания переменной x в диапазон [2; 7], то есть, условие должно быть истинным, когда x принимает значения в этом диапазоне чисел. Очевидно, что здесь должно быть две проверки: первая – мы проверяем, что x >= 2 и вторая – проверяем, что x <= 7. Если оба этих условия выполняются одновременно, то x попадает в наш диапазон. Реализовать такую проверку на Python можно так:

x = 4

**if** x >= 2 **and** x <= 7 : **print**("x попадает в [2; 7]")

**else**: **print**("x не попадает в [2; 7]")

Смотрите, здесь записано два условия, объединенных по И (and – это И). В результате, общее составное условие будет считаться истинным, если истинно и первое и второе условие. Если хотя бы одно из этих условий ложно, то ложно и все составное условие. В результате мы корректно реализуем проверку на вхождение значения переменной в диапазон [2; 7].

А теперь давайте реализуем противоположное условие, что x не принадлежит диапазону [2; 7]. Условие будет таким:

x = 40

**if**(x < 2 **or** x > 7): **print**("x не попадает в [2; 7]")

**else**: **print**("x попадает в [2; 7]")

Здесь в составном условии используется связка по ИЛИ (or – это ИЛИ) и оно будет истинно, если истинно или первое, или второе условие. То есть, в нашем случае, если x < 2 или x > 7, то делается вывод о не вхождении переменной x в указанный диапазон.

Итак, запомните следующие правила:

* условие x >= 2 and x <= 7 истинно, если истинно каждое из подусловий (x>=2 и x <= 7) и ложно, если ложно хотя бы одно из них;
* условие x < 2 or x > 7 истинно, если истинно хотя бы одно из подусловий (x < 2 или x > 7) и ложно, когда оба ложны.

Вот так можно записывать более сложные условия в условном операторе if. Причем они могут комбинироваться в любом сочетании, например:

x = 4; y = -2

**if** x >= 2 **and** x <= 7 **and** (y < 0 **or** y > 5):

**print**("x попадает в [2; 7], y не попадает в [0; 5]")

Здесь реализована проверка, что x должно принадлежать [2; 7], а y не принадлежать [0; 5].

## Вложенные конструкции

Блоки выполнения if, elif и else могут содержать другие условные конструкции, которые называют «вложенными». Блок выполнения вложенных конструкций также должен быть отделен отступом, образующим своего рода новую «ступень»:

С помощью вложенных конструкций можно наглядно продемонстрировать принцип работы блока elif. Допустим, пользователь проходит тест, и должен выбрать один из вариантов ответа, введя его номер:

|  |
| --- |
|  |
| number = int(input("Enter the answer number: ")) |
| if number == 1: |
| print("You’ve chosen answer A") |
| else: |
| if number == 2: |
| print("You’ve chosen answer B") |
| else: |
| if number == 3: |
| print("You’ve chosen answer C") |
| else: |
| if number == 4: |
| print("You’ve chosen answer D") |
| else: |
| print("There is no such answer.") |

Такой код, построенный с использованием вложенности, можно упростить, используя elif. При этом результат выполнения кода не изменится:

|  |
| --- |
|  |
| number = int(input("Enter the answer number: ")) |
| if number == 1: |
| print("You’ve chosen answer A") |
| elif number == 2: |
| print("You’ve chosen answer B") |
| elif number == 3: |
| print("You’ve chosen answer C") |
| elif number == 4: |
| print("You’ve chosen answer D") |
| else: |
| print("There is no such answer.") |

## Одиночные проверки

Внутри условия можно прописывать и такие одиночные выражения:

x = 4; y = True; z = False

**if**(x): **print**("x = ", x, " дает true")

**if**(**not** 0): **print**("0 дает false")

**if**("0"): **print**("строка 0 дает true")

**if**(**not** ""): **print**("пустая строка дает false")

**if**(y): **print**("y = true дает true")

**if**(**not** z): **print**("z = false дает false")

Вот этот оператор not – это отрицание – НЕ, то есть, чтобы проверить, что 0 – это false мы преобразовываем его в противоположное состояние с помощью оператора отрицания НЕ в true и условие срабатывает. Аналогично и с переменной z, которая равна false.

Из этих примеров можно сделать такие выводы:

1. Любое число, отличное от нуля, дает True. Число 0 преобразуется в False.
2. Пустая строка – это False, любая другая строка с символами – это True.
3. С помощью оператора not можно менять условие на противоположное (в частности, False превращать в True).

Итак, в условиях мы можем использовать три оператора: and, or и not. Самый высокий приоритет у операции not, следующий приоритет имеет операция and и самый маленький приоритет у операции or. Вот так работает оператор if в Python.

Задание 1

Пользователь вводит с клавиатуры число. Необходимо проверить его на четность и нечетность. Если число

четное требуется вывести на экран число и надпись Even

number. Если число нечетное выведите на экран число и

надпись Odd number.

Задание 2

Пользователь вводит с клавиатуры число. Необходимо проверить его на кратность 7. Если число кратно

требуется вывести на экран число и надпись Number is

multiple 7. Если число не кратно выведите на экран число

и надпись Number is not multiple 7.

Задание 3

Пользователь вводит с клавиатуры два числа. Необходимо найти максимум из двух чисел и показать его на

экран.

Задание 4

Пользователь вводит с клавиатуры два числа. Необходимо найти минимум из двух чисел и показать его на

экран.

Пользователь вводит с клавиатуры два числа. В зависимости от выбора пользователя нужно показать сумму двух чисел, разницу двух чисел, произведение двух чисел.