Логические операции

Булевы Операции — and, or, not

Это логические операции, упорядоченные по возрастанию приоритета:

* **x or y** - если x равно true, то x, иначе y
* **x and y** - если x равно false, то x, иначе y
* **not x** - если x равно false, то True, иначе False

Сравнения

В Python существует восемь операций сравнения. Все они имеют одинаковый приоритет (который выше, чем у логических операций). Сравнения могут быть скомпонованы произвольно; например, x < y <= z эквивалентно x < y and y <= z, за исключением того, что y вычисляется только один раз (но в обоих случаях z вообще не вычисляется, когда x < y оказывается равным false).

**В этой таблице обобщены операции сравнения:**

**<** строго меньше, чем

**<=** меньше или равно

**>** строго больше, чем

**>=** больше или равно

**==** равные

**!=** не равны

**is** идентификация объекта

**is not** отрицаемая идентификация объекта

Числовые типы

Существует три различных числовых типа: целые числа, числа с плавающей запятой и комплексные числа. Кроме того, логические значения являются подтипом целых чисел. Целые числа обладают неограниченной точностью. Числа с плавающей запятой обычно реализуются с использованием double в C; информация о точности и внутреннем представлении чисел с плавающей запятой для компьютера, на котором выполняется ваша программа, доступна в sys.float\_info. Комплексные числа имеют действительную и мнимую части, каждая из которых является числом с плавающей запятой. Чтобы извлечь эти части из комплексного числа z, используйте z.real и z.imag.

**Все числовые типы (кроме сложных) поддерживают следующие операции:**

**x + y** сумма x и y

**x – y** разница в x и y

**x \* y** продукт x и y

**x / y** коэффициент x и y

**x // y** коэффициент заполнения x и y

**x % y** оставшаяся часть x / y

**-x** x отклонено

**+x** x без изменений

**abs(x)** абсолютное значение или величина x

**int(x)** x преобразовано в целое число

**float(x)** x преобразовано в формат с плавающей запятой

**complex(re, im)** комплексное число с вещественной частью re, мнимая часть im. im значение по умолчанию равно нулю.

**c.conjugate()** сопряженное комплексное число c

**divmod(x, y)** пара (x // y, x % y)

**pow(x, y)** x для обеспечения y

**x \*\* y** x для обеспечения y

**math.trunc(x)** x сокращено до Integral

**round(x[, n])** x округлено до n цифры, округленные вдвое до четных. Если n значение опущено, по умолчанию равно 0.

**math.floor(x)** величайший Integral <= x

**math.ceil(x)** наименьший Integral >= x

Побитовые операции над целочисленными типами

**x | y** побитовый или из x и y

**x ^ y** побитовый эксклюзивный или из x и y

**x & y** побитовый и из x и y

**x << n** x сдвинут влево на n биты

**x >> n** x сдвинут вправо на n биты

**~x** фрагменты x перевернутый

Дополнительные методы для целочисленных типов

**int.bit\_length()** - возвращает количество битов, необходимое для представления целого числа в двоичном формате, исключая знак и начальные нули.

**int.bit\_count()** - возвращает количество единиц в двоичном представлении абсолютного значения целого числа. Это также известно как подсчет населения.

**int.to\_bytes()** - возвращает массив байтов, представляющий целое число.

**int.from\_bytes()** - возвращает целое число, представленное заданным массивом байтов.

**int.as\_integer\_ratio()** - возвращает пару целых чисел, отношение которых в точности равно исходному целому числу и с положительным знаменателем. Целочисленное отношение целых чисел всегда является целым числом в числителе и 1 в знаменателе.

Дополнительные методы для Float

**float.as\_integer\_ratio()** - возвращает пару целых чисел, отношение которых в точности равно исходному значению с плавающей точкой и с положительным знаменателем. Обращается OverflowError к бесконечностям и ValueError к NANS.

**float.is\_integer()** - возвращает, True если экземпляр float конечен с целым значением, и False в противном случае.

**float.hex()** - возвращает представление числа с плавающей запятой в виде шестнадцатеричной строки. Для конечных чисел с плавающей запятой это представление всегда будет включать начальную 0x и конечную p единицы и показатель степени.

**float.fromhex(s)** - метод класса для возврата значения с плавающей точкой, представленного шестнадцатеричной строкой s. Строка s может содержать начальные и конечные пробелы.

Типы итераторов

**container.\_\_iter\_\_() -** возвращает объект iterator. Объект необходим для поддержки протокола iterator, описанного ниже. Если контейнер поддерживает различные типы итераций, могут быть предоставлены дополнительные методы для конкретного запроса итераторов для этих типов итераций. (Примером объекта, поддерживающего несколько форм итерации, может быть древовидная структура, которая поддерживает как обход в ширину, так и в глубину.) Этот метод соответствует tp\_iter слоту структуры типов для объектов Python в Python/C API.

**iterator.\_\_iter\_\_() -** возвращает сам объект iterator. Это требуется для того, чтобы разрешить использование как контейнеров, так и итераторов с инструкциями for and in. Этот метод соответствует tp\_iter слоту структуры типов для объектов Python в Python/C API.

**iterator.\_\_next\_\_() -** возвращает следующий элемент из итератора. Если других элементов нет, вызовите StopIteration исключение. Этот метод соответствует tp\_iternext слоту структуры типов для объектов Python в Python/C API.

Типы последовательностей — list, tuple, range

**x in s** True если элемент из s равен x, иначе False

**x not in s** False если элемент из s равен x, иначе True

**s + t** объединение s и t

**s \* n или n \* s** эквивалентно добавлению s к самому себе n раз

**s[i]** i-й элемент s, источник 0

**s[i:j]** фрагмент s от i до j

**s[i:j:k]** фрагмент s от i до j с шагом k

**len(s)** длина s

**min(s)** наименьший элемент из s

**max(s)** самый большой элемент s

**s.index(x[, i[, j]])** индекс первого вхождения x в s (в начале или после индекса i и перед индексом j)

**s.count(x)** общее количество вхождений x в s

Изменяемые типы последовательностей

**s[i] = x** элемент i в s заменен на x

**s[i:j] = t** фрагмент s от i до j заменяется содержимым итерируемого t

**del s[i:j]** то же, что s[i:j] = []

**s[i:j:k] = t** элементы s[i:j:k] заменяются элементами t

**del s[i:j:k]** удаляет элементы s[i:j:k] из списка

**s.append(x)** добавляет x в конец последовательности (такой же, как s[len(s):len(s)] = [x])

**s.clear()** удаляет все элементы из s (такие же, как del s[:])

**s.copy()** создает поверхностную копию s (такую же, как s[:])

**s.extend(t) или s += t** расширяет s содержимым t (по большей части таким же, как s[len(s):len(s)] = t)

**s \*= n** обновляет s с повторением его содержимого n раз

**s.insert(i, x)** вставляет x в s с индексом, заданным i (таким же, как s[i:i] = [x])

**s.pop() или s.pop(i)** извлекает элемент из i, а также удаляет его из s

**s.remove(x)** удалите первый элемент из s, где s[i] равно x

**s.reverse()** возвращает элементы s на место

Списки

**Списки** - это изменяемые последовательности, обычно используемые для хранения коллекций однородных элементов.

**sort()** - Этот метод сортирует список на месте, используя только < сравнения между элементами. Исключения не подавляются - если какие-либо операции сравнения завершатся неудачей, вся операция сортировки завершится неудачей.

Кортежи

**Кортежи** - это неизменяемые последовательности, обычно используемые для хранения коллекций разнородных данных (таких как 2-кортежи, созданные enumerate() встроенные). Кортежи также используются в случаях, когда необходима неизменяемая последовательность однородных данных.

Диапазоны

**range** Тип представляет неизменяемую последовательность чисел и обычно используется для выполнения цикла определенное количество раз в for циклах.

Тип последовательности текста — str

Текстовые данные в Python обрабатываются с помощью str объектов, или строк. Строки представляют собой неизменяемые последовательности кодовых точек Unicode. Строковые литералы записываются различными способами:

* Одинарные кавычки: 'allows embedded "double" quotes'
* Двойные кавычки: "allows embedded 'single' quotes"
* В тройных кавычках: '''Three single quotes''', """Three double quotes"""

**Строковые методы:**

**str.capitalize() -** Возвращает копию строки, первый символ которой написан заглавной буквой, а остальные - в нижнем регистре.

**str.casefold()** - Возвращает копию строки, свернутую в регистре. Строки с заглавной буквой могут использоваться для сопоставления без заглавной буквы.

**str.center()** - Возвращает значение с центром в строке длиной ширина. Заполнение выполняется с использованием указанного fillchar (по умолчанию используется пробел ASCII). Исходная строка возвращается, если ширина меньше или равно len(s).

**str.count()** - Возвращает количество неперекрывающихся вхождений substring SUB в диапазоне [начать, завершение]. Необязательные аргументы начать и завершение интерпретируются как в нотации slice.

**str.encode()** - Возвращает строку, закодированную в bytes.

**str.endswith()** - Возвращайте, True если строка заканчивается указанным суффиксом, в противном случае возвращайте False. суффикс также может быть набором суффиксов для поиска. При необязательном запуске тест начинается с этой позиции. С необязательным окончанием прекратите сравнение в этой позиции.

**str.expandtabs()** - Возвращает копию строки, в которой все символы табуляции заменены одним или несколькими пробелами, в зависимости от текущего столбца и заданного размера табуляции. Позиции табуляции отображаются через каждые символы tabsize (по умолчанию равно 8, что указывает позиции табуляции в столбцах 0, 8, 16 и так далее). Чтобы развернуть строку, текущему столбцу присваивается нулевое значение, и строка проверяется посимвольно. Если символ представляет собой табуляцию (\t), в результат вставляется один или несколько пробелов до тех пор, пока текущий столбец не сравняется со следующей позицией табуляции. (Сам символ табуляции не копируется.) Если символ является символом перевода строки (\n) или возврата (\r), он копируется, а текущий столбец сбрасывается на ноль. Любой другой символ копируется без изменений, а текущий столбец увеличивается на единицу независимо от того, как символ представлен при печати.

**str.find()** - Возвращает наименьший индекс в строке, где подстрока sub находится внутри фрагмента s[start:end]. Необязательные аргументы start и end интерпретируются как в нотации slice. Верните, -1 если sub не найден.

**str.format()** - Выполните операцию форматирования строки. Строка, для которой вызывается этот метод, может содержать буквенный текст или поля для замены, разделенные фигурными скобками {}. Каждое поле замены содержит либо числовой индекс позиционного аргумента, либо имя ключевого аргумента. Возвращает копию строки, в которой каждое заменяющее поле заменяется строковым значением соответствующего аргумента.

**str.format\_map()** - Аналогично str.format(\*\*mapping), за исключением того, что mapping используется напрямую, а не копируется в dict.

**str.index()** - Нравится find(), но возникает ValueError, когда подстрока не найдена.

**str.isalnum()** - Возвращает, True если все символы в строке являются буквенно-цифровыми и присутствует хотя бы один символ, False в противном случае. Символ c является буквенно-цифровым, если возвращается одно из следующих значений True: c.isalpha(), c.isdecimal(), c.isdigit() или c.isnumeric().

**str.isalpha()** - Возвращает, True если все символы в строке являются алфавитными и присутствует хотя бы один символ, False в противном случае. Буквенные символы - это символы, определенные в базе данных символов Unicode как “Буква”, то есть символы с общим свойством категории, являющимся одним из “Lm”, “Lt”, “Lu”, “Ll” или “Lo”. Обратите внимание, что это отличается от свойства “Alphabetic”, определенного в стандарте Unicode.

**str.isascii()** - Возвращает, True если строка пуста или все символы в строке являются ASCII, False в противном случае. Символы ASCII имеют кодовые точки в диапазоне U + 0000-U + 007F.

**str.isdecimal()** - Возвращает, True если все символы в строке являются десятичными символами и присутствует хотя бы один символ, False в противном случае. Десятичные символы - это те, которые могут использоваться для формирования чисел в базе 10, например U + 0660, АРАБСКО-ИНДИЙСКАЯ ЦИФРА НОЛЬ. Формально десятичный символ - это символ в общей категории Unicode “Nd”.

**str.isdigit()** - Возвращает, True если все символы в строке являются цифрами и присутствует хотя бы один символ, False в противном случае. Цифры включают десятичные символы и разряды, которые требуют специальной обработки, такие как цифры надстрочного знака совместимости. Сюда входят цифры, которые нельзя использовать для формирования чисел в базе 10, например, чисел Харостхи. Формально цифра - это символ, который имеет значение свойства Numeric\_Type=Цифра или Numeric\_Type=десятичное число.

**str.isidentifier()** - Возвращает, True если строка является допустимым идентификатором в соответствии с определением языка, идентификаторами разделов и ключевыми словами.

**str.islower()** - Возвращает, True если все символы в регистре 4 в строке являются строчными и присутствует хотя бы один символ в регистре, False в противном случае.

**str.isnumeric()** - Возвращает, True если все символы в строке являются числовыми символами, и в них присутствует хотя бы один символ, False в противном случае. Числовые символы включают в себя символы цифр и все символы, обладающие свойством числового значения в Юникоде, например U + 2155, ОБЫЧНАЯ ДРОБЬ В ОДНУ ПЯТУЮ. Формально числовыми символами являются символы со значением свойства Numeric\_Type=Digit, Numeric\_Type=Decimal или Numeric\_Type=Numeric.

**str.isprintable()** - Возвращает, True если все символы в строке доступны для печати или строка пуста, False в противном случае. Непечатаемые символы - это те символы, которые определены в базе данных символов Unicode как “Other” или “Separator”, за исключением ASCII-пробела (0x20), который считается доступным для печати. (Обратите внимание, что печатаемые символы в этом контексте - это те, которые не должны экранироваться при repr() вызове в строке. Это не имеет никакого отношения к обработке строк, записанных в sys.stdout или sys.stderr.)

**str.isspace()** - Возвращает, True если в строке есть только пробельные символы и есть хотя бы один символ, False в противном случае.

**str.istitle()** - Возвращает True, если строка является строкой с заголовком и содержит хотя бы один символ, например, символы верхнего регистра могут следовать только за символами без кавычек, а символы нижнего регистра - только за символами с кавычками. Возвращайте False в противном случае.

**str.isupper()** - Возвращает, True если все символы в регистре 4 в строке написаны заглавными буквами и присутствует хотя бы один символ в регистре, False в противном случае.

**str.join()** - Возвращает строку, которая является конкатенацией строк в iterable. TypeErrorБудет выдан запрос A, если в iterable есть какие-либо нестроковые значения, включая bytes объекты. Разделителем между элементами является строка, предоставляющая этот метод.

**str.ljust()** - Возвращает строку, выровненную по левому краю, в строке длиной width. Заполнение выполняется с использованием указанного символа заполнения (по умолчанию используется пробел ASCII). Исходная строка возвращается, если ширина меньше или равна len(s).

**str.lower()** - Возвращает копию строки со всеми символами в регистре 4, преобразованными в нижний регистр.

**str.lstrip()** - Возвращает копию строки с удаленными начальными символами. Аргумент chars представляет собой строку, определяющую набор символов, которые необходимо удалить. Если опущено или None, в аргументе chars по умолчанию используется удаление пробелов. Аргумент chars не является префиксом; скорее, все комбинации его значений удалены.

**str.maketrans()** - Этот статический метод возвращает таблицу перевода, пригодную для str.translate().

**str.partition()** - Разделите строку при первом вхождении sep и верните 3-кортеж, содержащий часть перед разделителем, сам разделитель и часть после разделителя. Если разделитель не найден, верните 3-кортеж, содержащий саму строку, за которой следуют две пустые строки.

**str.removeprefix()** - Если строка начинается с префикс строка, возвращаемая string[len(prefix):]. В противном случае верните копию исходной строки.

**str.removesuffix()** - Если строка заканчивается на суффикс строка и это суффикс не является пустым, возвращает string[:-len(suffix)].

**str.replace()** - Возвращает копию строки со всеми вхождениями substring Старый заменено на новое. Если необязательный аргумент количество приведен только первый количество вхождения заменены.

**str.rfind()** - Возвращает самый высокий индекс в строке, где substring SUB найден такой, что SUB содержится в s[start:end]. Необязательные аргументы начать и завершение интерпретируются как в нотации slice. Возврат -1 при сбое.

**str.rindex()** - Нравится rfind() но возникает ValueError когда подстрока SUB не найден.

**str.rjust()** - Возвращает строку, выровненную по правому краю в строке длиной ширина. Заполнение выполняется с использованием указанного fillchar (по умолчанию используется пробел ASCII). Исходная строка возвращается, если ширина меньше или равно len(s).

**str.rpartition()** - Разделите строку при последнем появлении sep и возвращает 3-кортеж, содержащий часть перед разделителем, сам разделитель и часть после разделителя. Если разделитель не найден, верните 3-кортеж, содержащий две пустые строки, за которыми следует сама строка.

**str.rsplit()** - Возвращает список слов в строке, используя sep в качестве строки-разделителя. Если maxsplit приводится, не более maxsplit разделение выполнено, самый правый единицы. Если сентябрь не указан или None любая строка пробела является разделителем. За исключением разделения справа, rsplit() ведет себя как split() которая подробно описана ниже.

**str.rstrip()** - Возвращает копию строки с удаленными конечными символами. The символы аргумент - это строка, определяющая набор символов, которые необходимо удалить. Если опущено или None, the символы аргумент по умолчанию удаляет пробелы. The символы аргумент не является суффиксом; скорее, все комбинации его значений удаляются

**str.split()** - Возвращает список слов в строке, используя sep в качестве строки-разделителя. Если задан maxsplit, выполняется не более разбиения на maxsplit (таким образом, список будет содержать не более maxsplit+1 элементов). Если maxsplit не указан или -1, то количество разделений не ограничено (выполняются все возможные разделения).

**str.splitlines()** - Возвращает список строк в строке с разрывом на границах строк. Разрывы строк не включаются в результирующий список, если не задано значение keepends и значение true.

**\n** Перевод строки

**\r** Возврат каретки

**\r\n** Возврат каретки + перевод строки

**\v или \x0b** Табулирование строк

**\f или \x0c** Подача формы

**\x1c** Разделитель файлов

**\x1d** Разделитель групп

**\x1e** Разделитель записей

**\x85** Следующая строка (управляющий код C1)

**\u2028** Разделитель строк

**\u2029** Разделитель абзацев

**str.startswith()** - Возвращайте, True если строка начинается с префикса, в противном случае возвращайте False. префикс также может быть набором префиксов для поиска. С необязательным началом, тестовая строка начинается с этой позиции. С необязательным окончанием остановите сравнение строки в этой позиции.

**str.strip()** - Возвращает копию строки с удаленными начальными и конечными символами. Аргумент chars представляет собой строку, определяющую набор символов, которые необходимо удалить. Если опущено или None, в аргументе chars по умолчанию используется удаление пробелов. Аргумент chars не является префиксом или суффиксом; скорее, все комбинации его значений удалены

**str.swapcase()** - Возвращает копию строки с символами верхнего регистра, преобразованными в нижний регистр, и наоборот. Обратите внимание, что это не обязательно верно, что s.swapcase().swapcase() == s.

**str.title()** - Возвращает версию строки в формате title, где слова начинаются с символа верхнего регистра, а остальные символы - в нижнем регистре.

**str.translate()** - Возвращает копию строки, в которой каждый символ был сопоставлен с помощью данной таблицы перевода. Таблица должна быть объектом, который реализует индексирование с помощью \_\_getitem\_\_(), обычно сопоставления или последовательности. При индексации порядковым номером в Юникоде (целым числом) объект table может выполнять любое из следующих действий: возвращать порядковый номер в Юникоде или строку, чтобы сопоставить символ с одним или несколькими другими символами; возвращать None, чтобы удалить символ из возвращаемой строки; или вызывать LookupError исключение, чтобы сопоставить символ с самим собой.

**str.upper()** - Возвращает копию строки со всеми символами в регистре 4, преобразованными в верхний регистр. Обратите внимание, что это s.upper().isupper() может быть, False если s содержит символы без кавычек или если категория Юникода результирующих символов не “Lu” (буква в верхнем регистре), а, например, “Lt” (буква в верхнем регистре).

**str.zfill()** - Верните копию строки, оставленную заполненной '0' цифрами ASCII, чтобы создать строку длиной ширины. Начальный префикс знака ('+'/'-') обрабатывается путем вставки дополнения после символа знака, а не перед ним. Исходная строка возвращается, если ширина меньше или равна len(s).

Объекты Bytearray

**bytearray** объекты являются изменяемым аналогом bytes объектов.

**Операции с байтами и Bytearray:**

**bytes.count(sub[, start[, end]])**

**bytes.removeprefix(prefix, /)**

**bytes.removesuffix(suffix, /)**

**bytes.decode(encoding='utf-8', errors='strict')**

**bytes.endswith(suffix[, start[, end]])**

**bytes.find(sub[, start[, end]])**

**bytes.index(sub[, start[, end]])**

**bytes.join(iterable)**

**static bytes.maketrans(from, to)**

**bytes.partition(sep)**

**bytes.replace(old, new[, count])**

**bytes.rfind(sub[, start[, end]])**

**bytes.rindex(sub[, start[, end]])**

**bytes.rpartition(sep)**

**bytes.startswith(prefix[, start[, end]])**

**bytes.translate(table, /, delete=b'')**

**bytes.center(width[, fillbyte])**

**bytes.ljust(width[, fillbyte])**

**bytes.lstrip([chars])**

**bytes.rjust(width[, fillbyte])**

**bytes.rsplit(sep=None, maxsplit=- 1)**

**bytes.rstrip([chars])**

**bytes.split(sep=None, maxsplit=- 1)**

**bytes.strip([chars])**

**bytes.capitalize()**

**bytes.expandtabs(tabsize=8)**

**bytes.isalnum()**

**bytes.isalpha()**

**bytes.isdigit()**

**bytes.islower()**

**bytes.isspace()**

**bytes.istitle()**

**bytes.isupper()**

**bytes.lower()**

**bytes.splitlines(keepends=False)**

**bytes.title()**

**bytes.upper()**

**bytes.zfill(width)**