Числовые данные целого типа:

Правило	Например

Числовые данные целого типа в Python представлены типом данных "int". Он используется для хранения целых чисел, как положительных, так и отрицательных	x = 10 # целое число положительное
	y = -5 # целое число отрицательное

Набор операций над данными целого типа:

Операция	Пример	Результат
Сложение: +	4 + 3	7
Вычитание: -	4 -3	1
Умножение: *	4 * 3	12
Деление: /	4 / 3	1.(3)
Целочисленное деление: //	4 // 2	2
Взятие остатка: %	4 % 3	1
Возведение в степень: **	4 ** 3	64
Операции сравнения: ==, !=, >, <, >=, <=	4 == 3	False
Присваивание: =	var = 3	-
Модуль числа: abs(x)	-4	4
Смена знака числа	-(-4)	4

Над целыми числами также можно производить битовые операции

Побитовое или:	x | y
Побитовое исключающее или:	x ^ y
Побитовое и:	x & y
Битовый сдвиг влево:	x << n
Битовый сдвиг вправо:	x >> y
Инверсия битов:	~x

Числовые данные вещественного типа:

Python предоставляет три типа значений с плавающей точкой:

Правило	Например
float (двойная точность)	5.7
complex (комплексные числа)	3.5 + 5j
decimal.Decimal (большая точность, по умолчанию 28 знаков после запятой).	0.14285714285714285714285714291

Набор операций над данными вещественного типа:

Операция	Пример	Результат
Сложение: +	4.2 + 3.6	7.8
Вычитание: -	4.2 -3.6	0.6
Умножение: *	4.2 * 3.6	15.12
Деление: /	4.2 / 3.6	1.1(6)
Целочисленное деление: //	4.2 // 2.6	1

Взятие остатка: %	4.2 % 3.6	0.(6)
Возведение в степень: **	4.2 ** 3.6	175.2659073103862
Операции сравнения: ==, !=, >, <, >=, <=	4.2 == 3.6	False
Присваивание: =	var = 3.6	-
Модуль числа: abs(x)	-4.2	4.2
Смена знака числа	-(-4.2)	4.2

Логические типы:

Правило	Например
Логический тип представлен типом bool и позволяет хранить 2 значения: True (Истина / Да / 1) False (Ложь / Нет / 0)	True

Набор операций над данными логического типа:

Операция	Пример	Результат
not	not True	False
and	True and False	False
or	True or False	True

Последовательности:

Правило	Например

str, list, tuple и range	x = 10 # целое число положительное
	y = -5 # целое число отрицательное

Операция над последовательностями:

Операция	Пример	Результат
Длина: len(s)	len((1,2,3,4,5,6,7)) t = (1,2)	7
s + t: ‘ ’ + ’ ’	len((1,2,3,4,5,6,7)) t = (1,2)	(1,2,3,4,5,6,7,1,2)
Дублирование: ' ' * ' '	t = (1,2) * 3	(1,2,1,2,1,2)
Индексация и срезы: []	(1,2,3,4,5,6,7)[0]	1
Минимальное значение: min()	min((1,2,3,4,5,6,7))	1
Максимальное значение: max()	max((1,2,3,4,5,6,7))	7
Проверка на вхождение: in	1 in (1,2,3,4,5)	True
Количество повторений: s.count(x)	(1,2,3,4,5,1).count(1)	2
sorted(s, key=None, reverse=False) Возвращает отсортированный объект в виде списка. Исходный объект при этом не изменяется.	sorted((4,3,2,1))	[1, 2, 3, 4]
Индекс (положение) элемента Возвращает первое вхождение элемента x в последовательность s (между индексами start и end, если они заданы).	(1,2,3,4).index(1)	0

Символ и строка:

Правило	Например
Строка (str) - это упорядоченная неизменяемая последовательность символов Юникода	s1 = “string” s2 = “python”

Операция конкатенации (сцепления) над данными символьного и строкового типа:

Операция	Пример	Результат
Сложение: +	“string” + “python”	'stringpython'
Умножение: *	“string” * 2	'stringstring'
Операции сравнения: ==, !=, >, <, >=, <=	“string” == “python”	False
Проверка вхождения: in	"str" in "string"	True

А так большое количество характерных операций:

chr(i)

Возвращает символ № i из таблицы Unicode.

ord(c)

Возращает номер символа c из таблицы Unicode.

upper()

Возвращает копию строки s в верхнем регистре.

lower()

Возвращает копию строки s в нижнем регистре.

capitalize()

Возвращает копию строки с первым символом в верхнем регистре.

title()

Возвращает копию строки, в которой первые символы каждого слова преобразованы в верхний регистр, а все остальные - в нижний регистр.

count(t[, start[, end]])

Возвращает число вхождений строки t в строку s (или в срез s[start:end]).

find(t[, start[, end]])

Возвращает позицию самого первого (крайнего слева) вхождения подстроки t в строку s (или в срез s[start:end]); если подстрока t не найдена, возвращается -1.

index(t[, start[, end]])

Аналогично str.find(), но генерируется исключение ValueError, если подстрока не найдена.

replace(old, new[, count])

Возвращает копию строки s, в которой каждое (но не более count, если этот аргумент определен) вхождение подстроки old замещается подстрокой new.

split(sep=None, maxsplit=- 1)

Возвращает список строк, разбитых по строке sep.

join(seq)

Возвращает строку-«склейку» элементов seq, используя s в качестве разделителя.