Введение в язык Python

Лекция 4: Операторы отношений

Операции сравнения:

>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Не равно

Сравним 2 числа:

```
>>> 6>5
True
>>> 7<1
False
>>> 7==7 # не путайте == и =
True
>>> 7 != 7
False
>>>
```

Python возвращает `True` (Истина == 1), когда сравнение верное и `False` (Ложь == 0) – в ином случае. `True` и `False` относятся к логическому (булевому) типу данных `bool`.

```
>>> type(True)
<class 'bool'>
>>>
```

Сравнение вещественных чисел:

```
>>> 0.1 + 0.1 == 0.2
True
>>> 0.1 + 0.1 + 0.1 == 0.3
False
>>>
```

Логические выражения

Логическим высказыванием (предикатом) называется любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Рассмотрим комбинацию из высказываний:

```
>>> 2>4
False
>>> 45>3
True
>>> 2>4 and 45>3 # комбинация False and (И) True вернет False
False
>>> 2>4 or 45>3 # комбинация False or (ИЛИ) True вернет True
True
>>>
```

Комбинация логических высказываний:

X Y	and	or
0 0	0	0
0 1	0	1
1 0	0	1
1 1	1	1

- Для Python истинным или ложным может быть не только логическое высказывание, но и объект.
- В Python любое число, не равное нулю, или непустой объект интерпретируется как истина.
- Числа, равные нулю, пустые объекты и специальный объект `None` интерпретируются как ложь.

Пример:

```
>>> '' and 2 # False and True
''
>>> '' or 2 # False or True
2
>>>
```

В Python есть три логических оператора `and`, `or`, `not`.

- Результатом применения логического оператора `not` (НЕ) произойдет отрицание операнда, т.е. если операнд истинный, то `not` вернет – ложь, если ложный, то – истину.

```
>>> y = 6>8
>>> y
False
>>> not y
True
>>> not None
True
>>> not 2
False
>>>
```

Логический оператор `and` (И) вернет `True` (истину) или `False` (ложь), если его операндами являются логические высказывания.

```
>>> 2>4 and 45>3 # комбинация False and (И) True вернет False
False
>>>
```

Если операндами оператора `and` являются объекты, то в результате Python вернет объект:

```
>>> '' and 2 # False and True
''
>>>
```

Для вычисления оператора `and` Python вычисляет операнды слева направо и возвращает первый объект, имеющий ложное значение.

Если среди операндов (X,Y) есть ложный, то получим ложное значение, но вместо ложного значения для операндов-объектов Python возвращает первый ложный операнд, встретившийся в выражении, и дальше вычисления НЕ производит. Это называется вычислением по короткой схеме.

```
>>> 0 and 3 # вернет первый ложный объект-операнд
0
>>> 5 and 4 # вернет крайний правый объект-операнд
4
>>>
```

Если Python не удается найти ложный объект-операнд, то он возвращает крайний правый операнд.

Логический оператор or

Логический оператор `or` действует похожим образом, но для объектов-операндов Python возвращает первый объект, имеющий истинное значение. Python прекратит дальнейшие вычисления, как только будет найден первый объект, имеющий истинное значение.

```
>>> 2 or 3 # вернет первый истинный объект-операнд
2
>>> None or 5 # вернет второй объект-операнд, т.к. первый всегда ложный
5
>>> None or 0 # вернет оставшийся объект-операнд
0
>>>
```

Логические выражения можно комбинировать:

```
>>> 1+3 > 7 # приоритет + выше, чем >
False
>>> (1+3) > 7 # скобки способствуют наглядности и избавляют от ошибок
False
>>> 1+(3>7) # подумайте, что вернет выражение и почему
1
>>>
```

В Python можно проверять принадлежность интервалу:

```
>>> x=0
>>> -5< x <10 # эквивалентно: x > -5 and x<10
True
>>>
```

Пример:

```
>>> x = 5 < 10 # True
>>> y = 2 > 3 # False
>>> x or y
True
>>> (x or y) + 6
7
>>>
```

Пример:

Как вычислить `1/x`, чтобы не возникало ошибки деления на нуль. Для этого достаточно воспользоваться логическим оператором.

```
>>> x=1
>>> x and 1/x
1.0
>>> x=0
>>> x and 1/x
0
>>>
```

- Символы, как и все остальное, представлено в компьютере в виде чисел. Есть специальная таблица, которая ставит в соответствие каждому символу некоторое число.
- Определить, какое число соответствует символу можно с помощью функции `ord()`:

```
>>> ord ('L')
76
>>> ord ('Ф')
1060
>>> ord ('A')
65
>>>
```

Теперь сравнение символов сводится к сравнению чисел, которые им соответствуют:

```
>>> 'A' > 'L'
False
>>>
```

Для сравнения строк Python их сравнивает посимвольно:

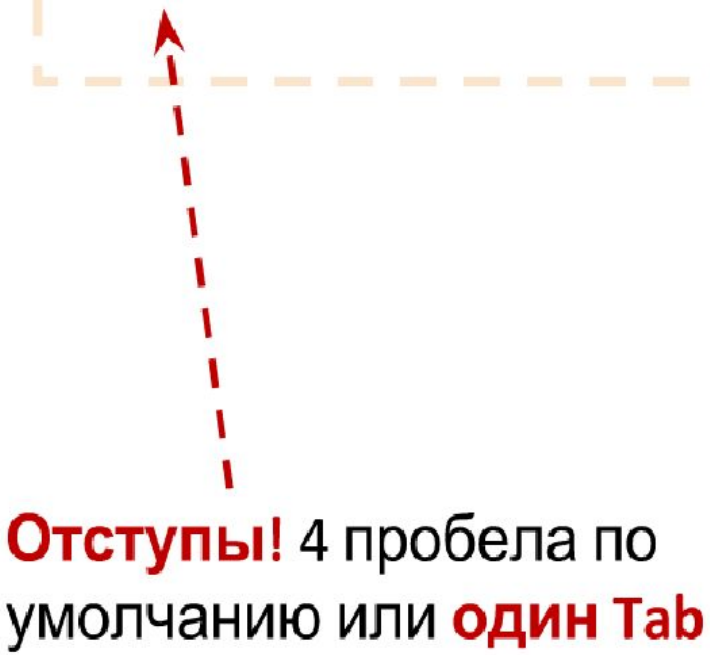
```
>>> 'Aa' > 'Ll'
False
>>>
```

Оператор `in` проверяет наличие подстроки в строке:

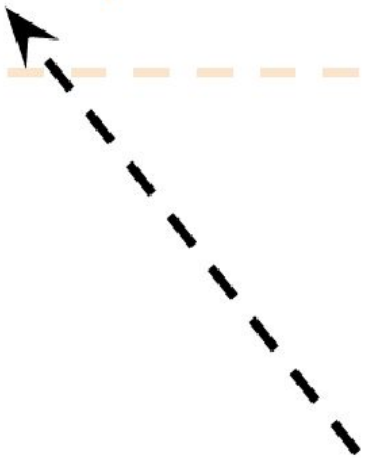
```
>>> 'a' in 'abc'
True
>>> 'A' in 'abc' # большой буквы A нет
False
>>> "" in 'abc' # пустая строка есть в любой строке
True
>>> '' in ''
True
>>>
```

Условная инструкция if

```
if << условие >>:
        << блок выражений >>
```



Отступы! 4 пробела по умолчанию или **один Tab**



Блок выражений выполняется, если условие *True*

Пример: таблица с водородными показателями для различных веществ.

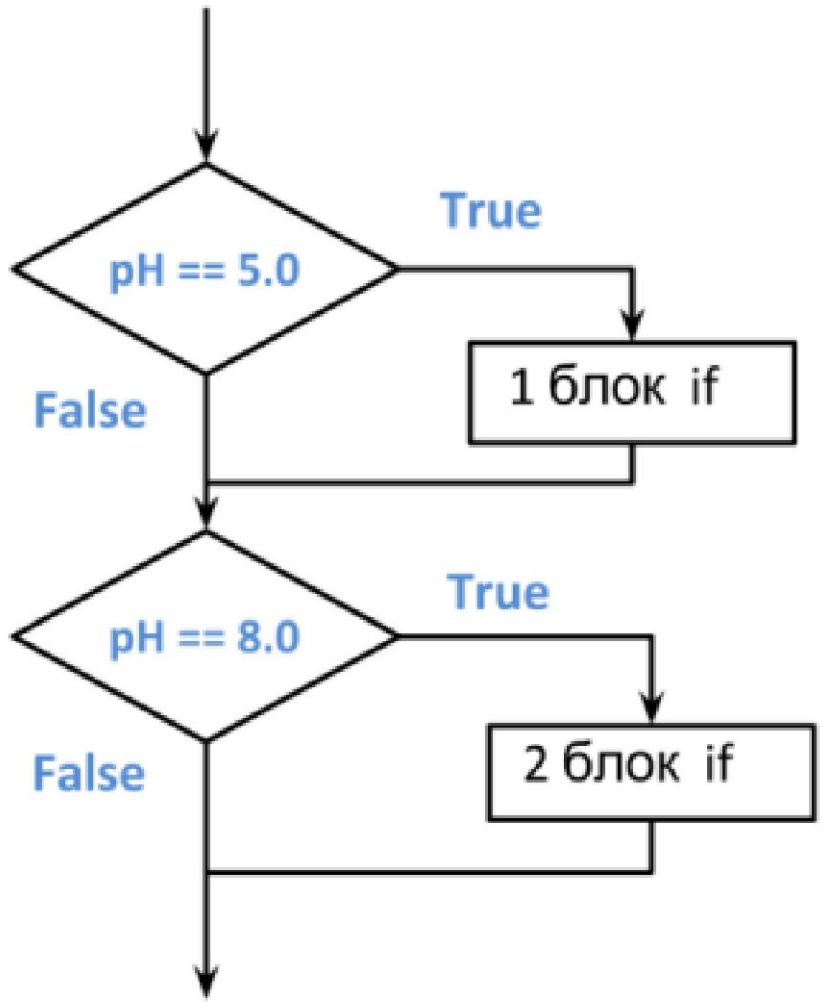
Вещество	pH
Электролит в свинцовых аккумуляторах	<1,0
Желудочный сок	1,0–2,0
Лимонный сок (5 % р-р лимонной кислоты)	2,0±0,3
Пищевой уксус	2,4
Кока-кола	3,0±0,3
Яблочный сок	3,0
Пиво	4,5
Кофе	5,0
Шампунь	5,5
Чай	5,5
Кожа здорового человека	5,5
Кислотный дождь	< 5,6
Питьевая вода	6,5–8,5
Слюна	6,8–7,4 [1]
Молоко	6,6–6,93
Чистая вода при 25 °C	7,0
Кровь	7,36–7,44
Морская вода	8,0
Мыло (жировое) для рук	9,0–10,0
Нашатырный спирт	11,5
Отбеливатель (хлорная известь)	12,5
Концентрированные растворы щелочей	>13

Произведем проверку:

```
>>> pH=5.0
>>> if pH==5.0:
print(pH, "Кофе")
5.0 Кофе
>>>
```

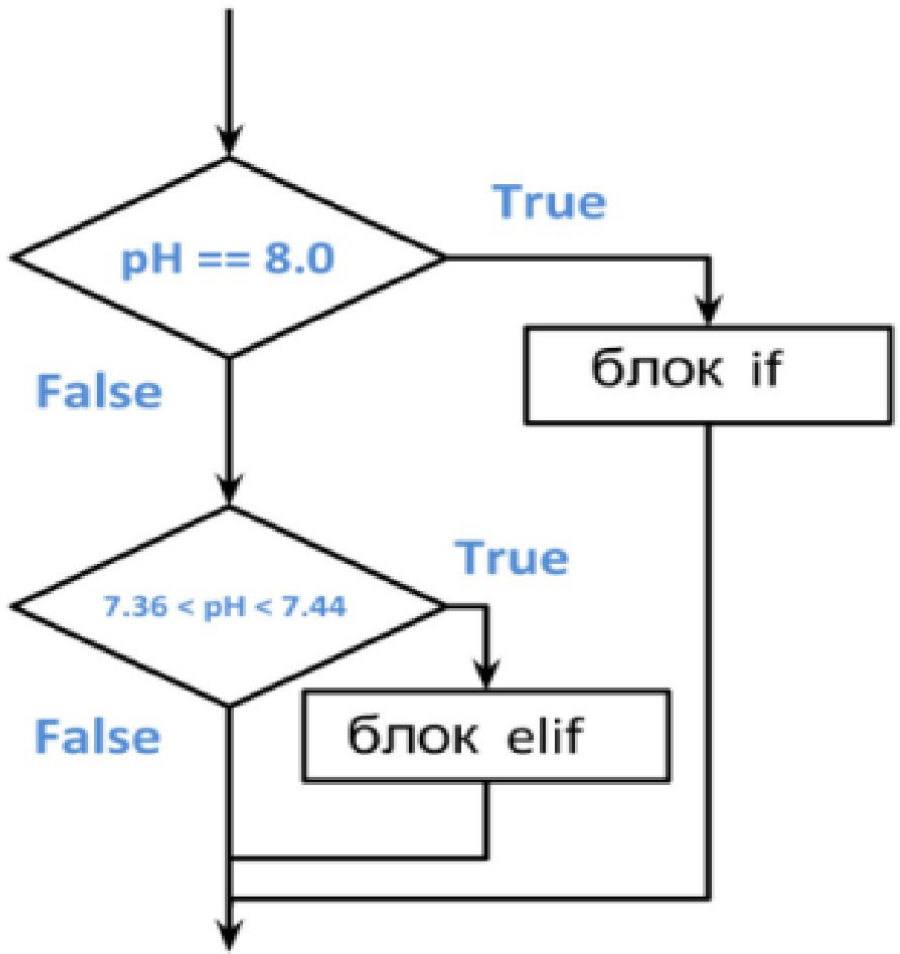
Можно производить несколько проверок подряд, и они выполнятся по очереди:

```
>>> pH=5.0
>>> if pH==5.0:
print(pH,"Кофе")
5.0 Кофе
>>> if pH==8.0:
print(pH,"Вода")
>>>
```


pH == 5.0
True
1 блок if
False
pH == 8.0
True
2 блок if
False

```
pH = 3.0
if pH == 8.0:
print(pH, "Вода")
elif 7.36 < pH < 7.44:
print(pH, "Кровь")
```

В этой программе используется ключевое слово `elif` (сокращение от else if), которое проверяет условие `7.36 < pH < 7.44`, если `pH == 8.0` оказалось ложным.


pH == 8.0
True
блок if
False
7.36 < pH < 7.44
True
блок elif
False

Условное выражение может включать множество проверок. Общий синтаксис у него следующий:

```
if << условие >>:
    << блок выражений >>
elif << условие >>:
    << блок выражений >>
else:
    << блок выражений >>
```

- Блок выражений, относящийся к `else`, выполняется, когда все вышестоящие условия вернули `False`.