Шпаргалка: логические выражения и ветвления

Кратко

Условия в коде

Что	Как	О чём помнить
Написать одно условие	`if password_symbols < 8:` `    print('Пароль слишком короткий!')`	Код после if нужно отделить от начала строки четырьмя пробелами

Написать два условия	`if password_symbols < 8:`
	`    print('Пароль слишком короткий!')`
	`else:`
	`    print('Всё в порядке!')`	

Написать много условий	`if password_symbols < 0:`
	`    print('Ошибка!')`
	`elif password_symbols == 0:`
	`    print('Введите пароль')`
	`elif password_symbols < 8:`
	`    print('Пароль слишком короткий!')`
	`elif password_symbols > 15:`
	`    print('Пароль слишком длинный!')`
	`else:`
	`    print('Всё в порядке!')`	

Логические операторы

Оператор	Пример	Истинно	Ложно
и, `and`	`(0 < 1) and (2 < 3)`	Если **все** операнды истинны	Если **хотя бы один** операнд ложный
или, `or`	`(0 < 1) or (2 < 3)`	Если **хотя бы один** из операндов истинный	Если **оба** операнда ложные
не, `not`, `!=`	`not (0 < 1)`	Если операнд — ложь	Если операнд — истина

Переменная типа bool

Что	Как	Что учитывать
Как задать	`a = 5 > 4`, значение True `a = 5 < 4`, значение False	Как называть переменные типа bool **Со слова `is`**: • `is_even` — чтобы проверить чётность; • `is_positive` — чтобы проверить, положительное число или отрицательное. **Со слова `has`**. Например, `has_access` — имеет ли пользователь доступ к чему-либо.

Аргумент → bool	Со значением True
	`bool(-5)`
	`bool(5)`
	`bool('строка')`
	`bool('  ')` — строка из пробелов
	
	Со значением False
	`bool(0)`
	`bool()`
	`bool(' ')` — пустой
	`bool('''''')` — пустой	

Подробно

Что такое логические выражения

Логическим называют выражение, про которое можно сказать, истинно оно или ложно. А если проще — правда это или нет. Например, «5 > 4» или «идёт дождь».

Чтобы было проще понять, посмотри на разницу между логическими выражениями и просто действиями:

Логические выражения

Арифметические выражения

`В слове ‘пять’ четыре буквы`. Это выражение логическое, потому что можно сказать, истинно оно или ложно.

`1 > 0`. Это выражение логическое, потому что можно сказать, истинно оно

`5 + 4`. Это действие: нельзя сказать, истинно оно или нет.

или ложно.

`1 + 5 > 0`. Это выражение логическое, потому что можно сказать, истинно оно или ложно.

`1 - 2`. Это действие: нельзя сказать, истинно оно или нет.

`1 - 2 + 10`. Это действие: нельзя сказать, истинно оно или нет.

Значения логических выражений

Есть выражение «идёт дождь». Если на улице и правда дождливо, оно **истинно.** То есть его значение — **True**. Если дождя нет, оно **ложно:** его значение **False**.

Условно говоря, обычный разговор проходит так:
— Идёт дождь.
— Ой, и правда идёт.

На языке логики этот звучит как:
— Идёт дождь.
— Значение этого выражения — истина.

Логические операторы `==` , `<` и `>`

Равно `==`. Выражение `4 == 5` означает: «а правда ли, что четыре равно пяти?». Его значение `False`.

Выражение `3 * 3 == 0` означает: «правда ли, что трижды три — ноль?». Его значение тоже `False`.

Не путай со знаком `=`. Это не одно и тоже!

Меньше `<` и **больше `>`**. Выражение `5 < 3`: «а правда ли, что пять меньше трёх?». Ответ: `False`.

Выражение `3 * 3 > 1`: «действительно ли трижды три больше одного?». Ответ: `True`.

Логические операторы `<=` , `>=`

Больше или равно `>=`. Например, `3 >= 3`: «а правда ли, что три больше или равно трём?». Ответ: `True`.

Меньше или равно `<=`. Например, `5 <= 5`: «действительно ли пять меньше или равно пяти?». Ответ: `True`.

Нужно проверить, истинно ли выражение «дважды два больше шести» — `2 * 2 > 6`.

Создадим переменную `check`. В ней будет храниться ответ — истина или ложь:

```
check = (2 * 2 > 6)
```

Попробуем напечатать значение этой переменной:

```
check = (2 * 2 > 6)
print(check)

# Будет напечатано: False
```

Python вернул значение `False`.

Значения типа `bool`

Тип `bool` может принимать одно из двух значений:

- **True** — истина.
- **False** — ложь.

Например, `a = 5 > 4` . Тип данных — `bool`, значение — True.

Оба этих значения пишутся с заглавной буквы. А ещё оба этих слова — ключевые: их нельзя использовать как имена переменных. Вот так написать не получится:

```
True = 'abc'
```

Выйдет ошибка:

```
SyntaxError: cannot assign to True
```

Как называть переменные типа `bool`

Имена таких переменных принято начинать с определённых слов. Это не жёсткие правила, но с ними проще понимать код.

Со слова `is`. С английского это переводится как «является». Например:

- `is_even` — чтобы проверить чётность;
- `is_positive` — чтобы проверить, положительное число или отрицательное.

Со слова `has`. Переводится как «имеет». Например, `has_access` — имеет ли пользователь доступ к чему-либо.

Например, нужно проверить, отрицательное ли число 50. Можно написать логическое выражение `50 < 0`, присвоить его переменной `is_negative` и попросить программу напечатать значение:

```
is_negative = 50 < 0
print('Переменная is_negative = ', is_negative)
```

В результате программа выведет:

```
Переменная is_negative = False # потому что 50 положительное: оно больше 0
```

Функция `bool()`

Она превращает аргумент в логический тип: либо `True`, либо `False`.

Вот какие аргументы превращаются в истину, а какие — в ложь:

True	False
Любое число, отличное от `0`. Например, `bool(5)` или `bool(-5)`.	**Пустой аргумент.** Например, `bool()`.
Не пустая строка. Например, `bool(' ')`. Строка из пробелов не считается пустой: `bool('  ')` — True.	**Пустая строка.** Например, `bool('')` или `bool('''''')`. **Число 0:** `bool(0)`.

Выражение может быть и сложнее. Например, такое:

```
is_bool = (bool() == bool('  ')) == True

print(is_bool)
```

Как это распутать:

1. Первое значение в скобках `bool()` — `False`
2. Второе значение в скобках `bool('  ')` — `True`.
3. Одно не равно другому, поэтому все скобки — `False`.
4. Если скобки схлопываются в ложь, остаётся выражение `False == True`, это `False`. Получается, is_bool = `False`.

Как написать ветвление

Ветвление объявляют оператором `if`. С английского это переводится как «если».

После `if` пишут логическое выражение, результатом которого может быть `True` или `False`. Это **условие**.

Если выражение в условии истинно, выполняется код после двоеточия. Если ложно, код после условия не сработает.

```
if <условие>:
    <код, который выполнится, если условие вернуло True>
```

Чтобы задать условие, нужно:

- написать `if`;
- обозначить условие — например, `password_symbols < 8`;
- поставить двоеточие `:` и перейти на другую строку;
- сделать четыре отступа с начала строки и написать, какое действие нужно выполнить. Например, `print('Пароль слишком короткий!')`.

Вот как в коде выглядит условие для пароля. Если меньше 8 символов, выводим «Пароль слишком короткий».

Попробуй заменить значение переменной на другое число и посмотри, что получится.

```
# Допустим, пользователь ввёл 7 символов
password_symbols = 7
if password_symbols < 8:
    print('Пароль слишком короткий!')
```

📌 Код после if нужно отделить от начала строки четырьмя пробелами.

Ветвление с else

Пригодится конструкция `if-else` — «если-иначе».

Ты добавляешь к `if` ещё один блок кода. Он выполнится, если условие ложно:

```
if <условие>:
    <код выполнится, если условие истинно>

else:
    <код выполнится, если условие ложно>
```

Вот как будет выглядеть код для пароля:

```
# Допустим, пользователь ввёл 7 символов
password_symbols = 7
if password_symbols < 8:
    print('Пароль слишком короткий!')
else:
    # Если не сработало условие в if - выполняется код в блоке else
    print('Всё в порядке!')
```

Ветвление с `elif`

Код можно упростить. Пригодится **конструкция** `elif`: в ней можно проверить ещё одно условие.

Это вложенная конструкция для `if`. Читается она так: «Если условие для `if` не выполнено, но выполняется условие для `elif` — выполнить код в блоке `elif`».

В `elif` нужно написать условие, которое вернёт `True` или `False`.

```
if password_symbols < 0:
    print('Ошибка!')
elif password_symbols == 0:
    print('Введите пароль')
elif password_symbols < 8:
    print('Пароль слишком короткий!')
elif password_symbols > 15:
    print('Пароль слишком длинный!')
else:
    # Если не сработало ни одно условие в предыдущем коде - выполняется код в блоке else
    print('Всё в порядке!')
```

Читается это так: «Если пароль меньше 0, вывести ошибку. Если 0 — надо ввести пароль, если меньше 8, слишком короткий. Если больше 15 — слишком длинный. Если ничего из этого — всё в порядке».

Как только выполнится одно из условий, все следующие `elif` и `else` игнорируются. Например, длина пароля 0. Программа напечатает «Введите пароль», а остальные блоки кода пропустит.

В логическом выражении может быть несколько частей. Например, «**на улице тепло** и **солнечно**». Эти части называются **операндами**.

Логические операторы

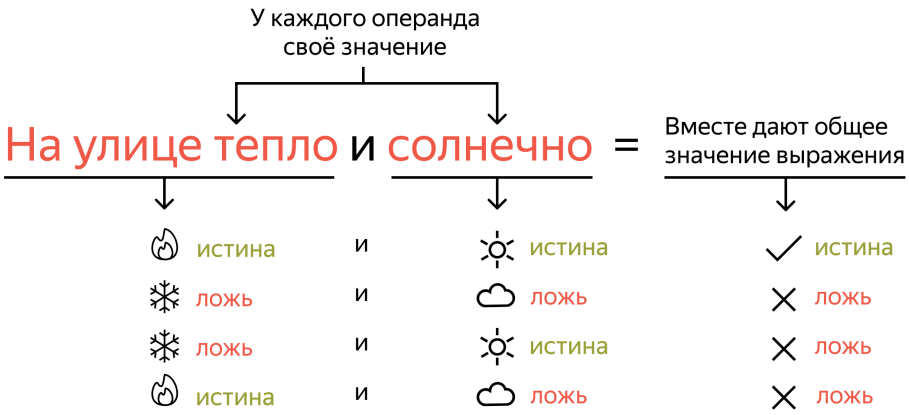
Операнды чем-то соединяются: например, словом «и». На улице тепло **И** солнечно». Слово «и» тут называется **логическим оператором.**

Чаще всего ты будешь встречать логические операторы И, ИЛИ, НЕ. Есть и другие, но этих пока достаточно.

Операторы влияют на то, истинно выражение целиком или ложно.

Выражение «(на улице тепло) И (солнечно)» будет истинным, только если на улице **и** тепло, **и** солнечно. Обе части должны быть правдой.

Если тепло и пасмурно, всё выражение — ложно.



Операторы «И» и «ИЛИ»

Разные логические операторы работают по-разному:

И

`(На улице тепло) И (солнечно)`

Чтобы выражение было истинным, оба операнда должны быть истинны.

Если на улице тепло, но пасмурно, выражение — ложь.

ИЛИ

`(На улице тепло) ИЛИ (солнечно)`

Чтобы выражение было истинным, достаточно чего-то одного.

Может быть либо тепло, либо солнечно. В любом из этих случаев выражение целиком будет истинно.

Apple выпускает (айфоны) И (эйрподсы). Это выражение истинно, потому что истинны оба операнда.

Котики умеют (летать) И (дышать под водой). Это выражение ложно, потому что оба операнда ложны. Истины тут вообще нет.

Котики умеют (летать) ИЛИ (мурчать). Это выражение истинно, потому что мурчать-то котики умеют. Этого достаточно.

Оператор НЕ

Этот оператор — отрицание. Представь, что НЕ — это минус. Он переворачивает значение того, что идёт за ним.

«НЕ (в году 12 месяцев)» — это ложь. В скобках написана правда: в году 12 месяцев. Перед ней как бы поставили знак «минус» — отзеркалили. Получилась ложь.

«НЕ (в году 10 месяцев)» — это истина. В скобках написана ложь: на самом деле в году 12 месяцев. Перед ложью поставили «минус» — получилась истина.

Код с логическими операторами

Чтобы использовать логические операторы в коде, понадобятся специальные символы:

- И — `and`;

- ИЛИ — `or`;
- НЕ — `not` или `!=`. Можно и так, и так.

Например, есть логическое выражение `(3 < 4) and (2 < 3)`. Ты можешь попросить программу определить, истинно оно или ложно.

Для этого нужно написать логическое выражение в скобках после `print`. Код выведет ответ — `true` или `false`:

```
print((3 < 4) and (2 < 3));
# программа выведет true: оба операнда истинны
```

Приоритет операторов

В выражении может быть несколько логических операторов. Например, `x > 40 or x < 45 and x != 42`.

Тогда операторы выполняются не в порядке записи, а в порядке приоритета:

- первый — `not`;
- второй — `and`;
- последний — `or`.

Например, `10 > 40 or 5 < 45 and 1 != 42` выполнится так:

- not: `1 != 42` — True;
- and: `5 < 45 and True` — True;
- or: `10 > 40 or True` — True.