

# Условный оператор if. Продолжение

## Вложенные условия

Python строго следует вашим указаниям, поэтому важно научиться отслеживать процесс выполнения кода. В этом поможет функция `print()`, размещённая в ключевых местах.

Если одно условие нужно проверить только после выполнения другого, используйте вложенные условия:

```
if <условие_1>:  
    if <условие_2>:
```

- Если условие\_1 будет выполнено, Python заглянет внутрь блока с условием\_2 и проверит это условие.
- Если условие\_1 не выполнено, условие\_2 даже не будет проверяться, так как Python не заглянет внутрь блока с условием\_1.

## Оператор elif (иногда называют else-if)

Вы уже знаете, что с помощью `if` можно задать конкретное условие, а с помощью `else` — обработать остальные случаи. Но если нужно проверить два конкретных условия или более, используйте оператор `elif`.

### Синтаксис:

```
Python  
if <условие_1>:  
    ...  
elif <условие_2>:  
    ...  
else:  
    ...
```

Важно понимать, что `elif` и `else` необязательны. Вы можете использовать только `if`, или `if` с `elif`, или `if` с `else`, или всё вместе.

Также вы можете добавлять множество `elif`. Их количество неограниченно.

```
Python
if <условие_1>:
    ...
elif <условие_2>:
    ...
elif <условие_3>:
    ...
elif <условие_4>:
    ...
else:
    ...
```

## Логические операторы or и and

Иногда нужно, чтобы два условия приводили к одному действию. Здесь помогут операторы **or** и **and**.

**Оператор **or** (ИЛИ)** позволяет выбрать одно из условий.

Пример:

```
Python
if x > 0 or y < 0:
    # блок кода
```

В этом примере, если будет выполнено хотя бы одно из условий, Python выполнит и блок кода внутри. Если оба условия неверны (**False**), блок не выполнится.

**Оператор **and** (И)** позволяет соединить условия вместе, требуя выполнения каждого из них.

Пример:


```
Python
if x > 0 and y < 0:
    # блок кода
```

Такое условие сработает, только если  $x > 0$  и  $y < 0$ , то есть если оба условия истинны (**True**).

# Несколько операторов or/and

Как и операции вычисления, логические операции имеют свой приоритет выполнения. В таблице ниже представлен приоритет выполнения некоторых операций: чем выше операция, тем выше приоритет, то есть эти операции будут выполняться в первую очередь.

<, <=, >, >=, <>, !=, ==	Сравнение, принадлежность, тождественность
<u>not</u> x	Булево НЕ
<u>and</u>	Булево И
<u>or</u>	Булево ИЛИ



Анастасия Верхорубова для Skillbox

Чтобы условия проверялись в нужном порядке (так, как вы задумали), используйте скобки.

Пример:

```
Python
if <условие_1> and <условие_2> or <условие_3>:
```

При такой записи сначала проверится условие\_1 и условие\_2 с помощью **and**. Если оба вернут **True**, Python станет воспринимать их примерно так: **if True or <условие\_3>**. Далее будут сравниваться уже True и условие\_3.

Чтобы изменить порядок проверки, используйте скобки:

```
Python
if <условие_1> and (<условие_2> or <условие_3>):
```

Теперь Python сначала выполнит выражение в скобках: **if <условие\_1> and (True/False)**. После получения в скобках ответа True или False он сможет сравнить первое условие и этот ответ.

## Не допускайте следующих ошибок

- Если условия независимы друг от друга и могут выполняться одновременно, не объединяйте их в один условный блок (или делайте это осторожно).

Python

```
имя_1 = 'Иван'
имя_2 = 'Сеня'
x = 5
y = 1

if x == 5:
    print("Привет", имя_1)
elif y == 1:
    print("Привет", имя_2)
```

В этом примере выполнится только один блок — тот, который первый выдаст правильный ответ. То есть блок, который стоит выше.

Если нужно выполнить обе проверки, создавайте отдельные условные блоки:

Python

```
if x == 5:
    print("Привет", имя_1)
if y == 1:
    print("Привет", имя_2)
```

- Всегда явно прописывайте все логические операции.

Логические операции хоть и похожи на арифметические, но не во всём.

Python

```
if (x or y) > 10:
    ...
```

Внутри скобок будет выбрано только одно значение (первое, которое будет похоже на `True`), и только потом выполнится второе условие вне скобок.

Такой код **не будет** считаться аналогом этому:

Python

```
if (x > 10) or (y > 10):  
    ...
```