УНИВЕРСИТЕТ СИНЕРГИЯ

РАЗРАБОТКА НА PYTHON

ЛОГИЧЕСКИЕ И УСЛОВНЫЕ ОПЕРАТОРЫ

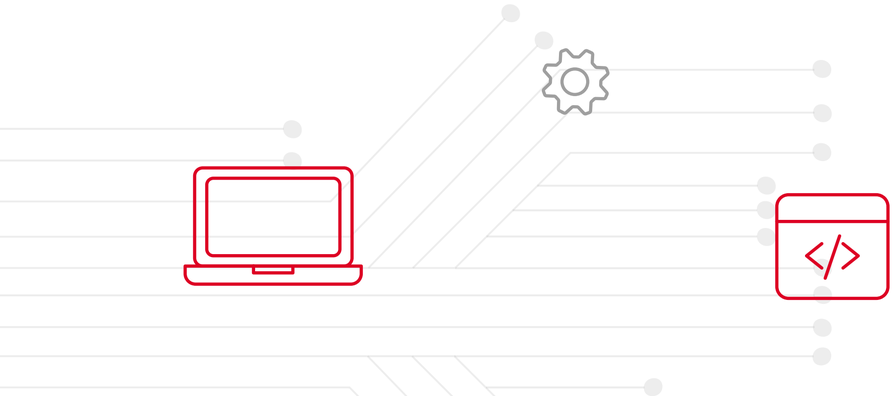
О ЧЕМ ПОГОВОРИМ СЕГОДНЯ

01 Что такое логический оператор и зачем он нужен

02 Виды логических операторов

Определение логического оператора

Логический оператор
это инструмент в программировании, который позволяет сравнивать и комбинировать логические значения и выражения. Он используется для создания условий и проверки истинности или ложности выражений



Логические операции в python

В языках программирования используются специальные знаки, подобные тем, которые используются в математике:

> — больше

> = — больше или равно

< — меньше

== — равно # не путать с =

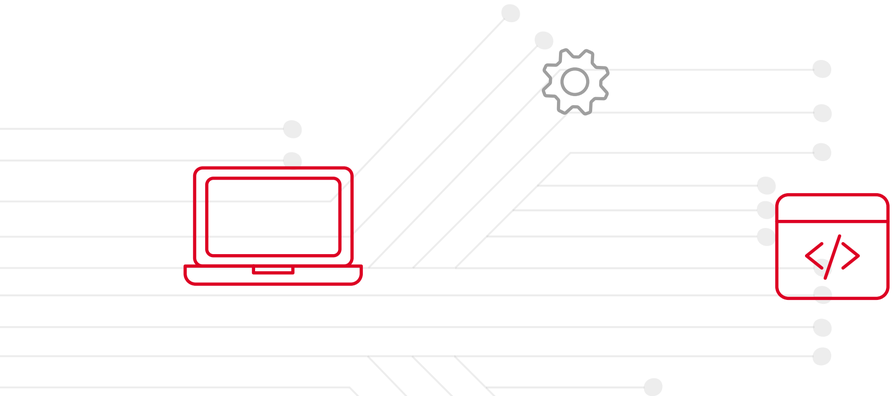
>= — больше или равно

!= — не равно

Определение логического оператора

Пример использования логического оператора для принятия решения

```
x = 5
y = 10
result = (x > 0 and y < 20) # Сохраняет результат логического выражения
if result:
    print("Оба условия выполняются") # Выводит "Оба условия выполняются"
```



Приоритет логических операторов в Python

При использовании нескольких логических операторов в выражении, важно понимать порядок их выполнения. В Python порядок выполнения логических операторов определяется следующим образом (от наивысшего приоритета к наименьшему):

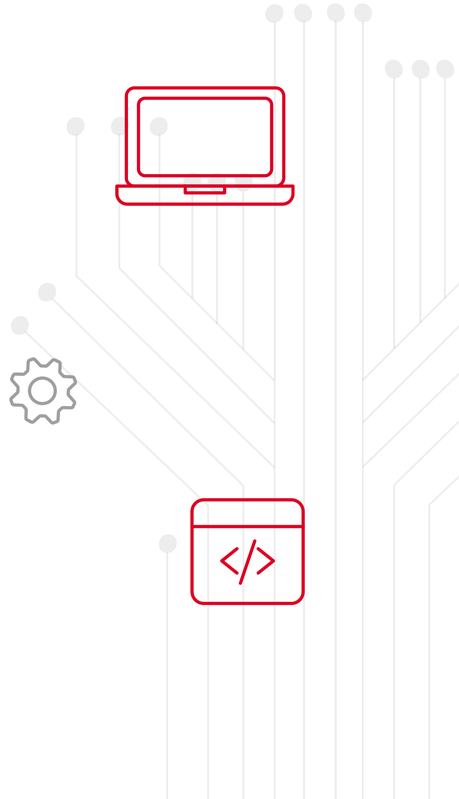
`not` `and` `or`

Это значит, что оператор not имеет наивысший приоритет, а оператор or имеет наименьший приоритет

Комбинирование логических операторов в Python

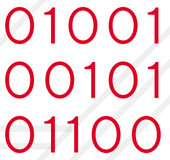
В Python вы можете комбинировать логические операторы для создания сложных условий. Для задания порядка выполнения логических операций можно использовать скобки

```
x = 5
y = 10
z = 15
print((x > 0 and y < 0) or z == 15) # True
print(not (x > 0 and y < 20)) # False
print( x > 0 and (y < 20 or z == 15)) # True
```



ВИДЫ ЛОГИЧЕСКИХ ОПЕРАТОРОВ

Короткое замыкание
Short-circuit evaluation



В Python, при использовании логических операторов and и or, выполнение выражения может быть остановлено на определенном этапе, если результат уже ясен. Это называется "коротким замыканием" и может быть полезным для повышения эффективности и избежания ненужных вычислений

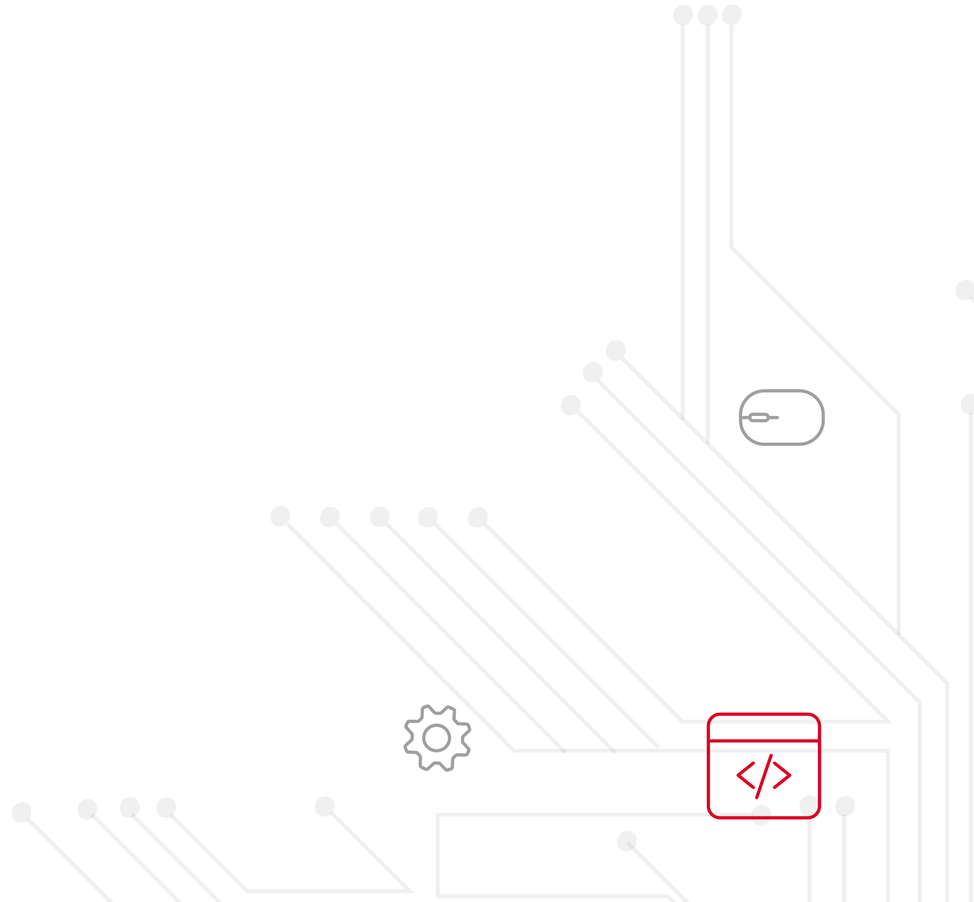
```
x = 5
y = 10
result = (x > 0) and (y < 0)
# y < 0 не вычисляется
```

Приведение к логическому типу

Возможно и обратное. Можно преобразовать какое-либо значение к булевому типу

И здесь работает правило:
всё, что не **0** и не пустота, является **правдой**

```
bool(3.14)
>> True
bool(-15)
>> True
bool(0)
>> False
bool(' ') # тут в кавычках символ пробела
>> True
bool('')
>> False
```



Операторы сравнения с цепочкой

В Python, операторы сравнения могут быть использованы для сравнения более чем двух значений в цепочке. Это позволяет проверять, находится ли значение между двумя другими значениями

```
x = 5
y = 10
z = 15
result = x < y < z
# сравнивает что x меньше y и y меньше z
```