Условный оператор

01 Оформление условного оператора

Условие — это выражение, которое должно быть оценено как истинное или ложное

Условный оператор — это конструкция, которая позволяет программе принимать решение о выполнении определенного блока кода на основе условия

В Python, условный оператор **if** используется для проверки определенного условия и выполнения блока кода, только если условие истинно. Дополнительно к оператору if, также используются операторы **else**(необязателен) и **elif**(необязателен) для обработки альтернативных условий

Оператор **if** — основной условный оператор, который выполняет блок кода, если условие истинно

Оператор **else** — дополнительный блок кода, который выполняется, если условие в операторе if ложно

Оператор **elif** — дополнительный блок кода, который проверяется, если предыдущие условия в операторе if и операторе elif ложны. Может быть несколько блоков elif

Синтаксис условного оператора

if условие_1

```
ooo
блок_кода_1  # блок кода выполняется, если условие_1 истинно
```

elif условие_2

```
ooo
блок_кода_2  # блок кода выполняется, если
условие_1 ложно и условие_2 истинно (необязателен)
```

else

```
блок_кода_3  # блок кода выполняется, если
оба условия ложны (необязателен)
```

Пример

```
x = 10
if x > 10:
    print("x больше 10")
elif x == 10:
    print("x равно 10")
else:
    print("x меньше 10")
```

Вывод

```
x равно 10
```

Отступы играют важную роль при использовании условных операторов. Отступы определяют, какой код будет выполнен внутри блока, связанного с условным оператором. В Python отступы создаются с помощью пробелов или символов табуляции (обычно 4 пробела). Используется однородный тип отступов в рамках одной программы

Обратите внимание, что блок кода, который должен быть выполнен внутри условия, должен быть сдвинут вправо, чтобы создать отступ от начала строки. Все строки с тем же уровнем отступа считаются частью блока кода и будут выполнены, только если условие истинно

Если отступы не будут использованы правильно, то возникнет ошибка синтаксиса. Поэтому важно следить за правильным форматированием кода с использованием отступов при работе с условными операторами

02 Составление условий. Логические операторы и операции

В Python составление условий в условных операторах является важной частью программирования. Условие представляет собой выражение, которое оценивается как истинное или ложное, и определяет, будет ли соответствующий блок кода выполнен или нет

Операции сравнения

Операции сравнения используются для сравнения значений и проверки, удовлетворяют ли они определенному отношению

Пример

```
x == 9 # x равен 9
x != 4 # x не равен 4
x < 10 # x меньше 10
x > 5    # x больше 5
z <= 3   # z меньше или равно 3
a  >= 6 # a больше или равно 6
```

Сложные условия. Логические операторы

В Python сложные условия создаются путем комбинирования нескольких условий с помощью логических операторов **and, or и not**. Это позволяет программе проверять истинность или ложность нескольких условий одновременно

Примеры

01 Использование оператора and (логического И)

```
x = 5
y = 10
if x > 0 and y > 0:
    print("Оба числа положительные")
```

Вывод

Оба числа положительные

В этом примере кода, блок кода внутри if будет выполнен только в том случае, если оба условия x > 0 и y > 0 истинны

02 Использование оператора and (логического И)

```
x = 5
y = -2
if x > 0 or y > 0:
    print("Хотя бы одно число положительное")
```

Вывод

Хотя бы одно число положительное

В этом примере кода, блок кода внутри if будет выполнен, если хотя бы одно из условий x > 0 или y > 0 истинно

03 Использование оператора not (логического НЕ)

```
x = 5
if not x > 10:
    print("x не больше 10")
```

Вывод

x не больше 10

В этом примере кода, блок кода внутри if будет выполнен только в том случае, если условие x > 10 ложно. Использование оператора not противоречит результирующему значению условия

04 Приоритет логический операторов:

В Python операции логических операторов имеют определенный приоритет, который определяет порядок вычислений при комбинировании условий

Вот приоритет логических операторов в порядке возрастания

01 Оператор not имеет самый высокий приоритет. Он выполняется первым

02 Оператор and выполняется после оператора not, но перед оператором or

03 Оператор or имеет самый низкий приоритет и выполняется последним

Приоритет логических операторов можно изменить с помощью использования скобок ((и)). Операции, заключенные в скобки, будут выполнены первыми, а затем продолжится выполнение оставшейся части выражения

Пример

```
x = 5
y = 10
if (x > 0 and y > 0) or x + y > 15:
    print("Хотя бы одно из условий истинно")
```

Вывод

Хотя бы одно из условий истинно

В этом примере кода, блок кода внутри if будет выполнен, если одно из условий - x > 0 и y > 0, или x + y > 15 - истинно

Сложные условия позволяют программистам создавать более гибкие и мощные проверки, которые учитывают несколько факторов сразу

03 Вложенные условные операторы

В Python вложенный условный оператор, также известный как вложенный if, это ситуация, когда один условный оператор находится внутри другого. Вложенные условные операторы позволяют программе делать более точные проверки и принимать решения на основе множества условий

Пример

```
x = 5
if x > 0:
    print("x положительное число")
    if x % 2 == 0:
        print("x - четное число")
    else:
        print("x - нечетное число")
else:
    print("x отрицательное или ноль")
```

Вывод

x положительное число
x - нечетное число

В этом примере внешний условный оператор if проверяет, является ли x положительным числом. Если это истинно, будет выполнено внутреннее условие, которое проверяет, является ли x четным числом. Если внешнее условие ложно, программа выполняет блок кода после внешнего оператора else.

Вложенные условные операторы позволяют программам делать более сложные проверки и управлять выполнением кода в зависимости от множества условий. Важно правильно выравнивать блоки кода с отступами для обозначения вложенности и облегчения понимания кода

04 Каскадный условный оператор

Каскадные условные операторы в Python используются, когда нужно проверить несколько условий последовательно до нахождения соответствующего блока кода или выполнения действия. В таких случаях может применяться оператор elif (сокращение от "else if") после оператора if

Пример

```
x = 5
if x > 0:
    print("x положительное число")
elif x < 0:
    print("x отрицательное число")
else:
    print("x равно нулю")
```

Вывод

x положительное число

В этом примере оператор if проверяет, является ли x положительным числом. Если это истинно, будет выполнен соответствующий блок кода, и выполнение программы продолжится после каскада. Если условие ложно, то передается выполнение оператору elif (или в конечном итоге оператору else, если все предыдущие условия ложны)

В каскаде условий с помощью оператора elif проверяются дополнительные условия, и первое истинное условие (имеющее истинное значение) будет действовать, а каждый последующий не будет даже рассматриваться

Каскадные условные операторы полезны для проверки нескольких вариантов или условий в последовательном порядке с выполнением соответствующего блока кода только для первого истинного условия

Итог

В этой теме мы рассмотрели понятие условного оператора, его структуру, изучили способы оформления условного оператора, операции сравнения, логические операторы, разобрали оформление вложенных и каскадных условных операторов