Python: ветвления | Я.Шпора

Условный оператор `if`

Используется для объявления ветвлений. Он задаёт определённое поведение программы в зависимости от установленных условий:

```
if <условие>:
    ...  # Тело отбивается четырьмя пробелами, чтобы Python знал,
    ...  # что этот код относится к оператору ветвления.
```

Пример:

```
if 5 < 10:
    print('Пять меньше десяти!')

# Вывод в терминал: Пять меньше десяти!
```

Конструкция `if...else`

Используется для обработки случаев, когда условие ложно:

```
if <условие>:
    ...  # Тело оператора if: код, который выполнится,
    ...  # если условие возвращает True.
else:
    ...  # Тело оператора else: код, который выполнится,
    ...  # если условие возвращает False.
```

Пример:

```
if 5 < 10:
    print('Пять меньше десяти!')
else:
    print('Пять больше или равно десяти!')

# Вывод в терминал: Пять меньше десяти!
```

Вложенные условия

```
if <условие_1>:
    ...
    if <условие_2>:
        ...
        if <условие_3>:
            ...
    else:
        ...
else:
    ...
```

Конструкция `if...elif...else`

Применяется для проверки дополнительных условий в тех случаях, когда условие в `if` ложно:

```
if <условие>:
    ...  # Тело оператора if: код, который выполнится,
    ...  # если условие возвращает True.
elif <условие_1>:
    ...  # Тело оператора elif: этот код выполнится,
    ...  # если условие в if возвращает False, а условие_1 в elif возвраща
elif <условие_2>:
    ...  # Тело оператора elif: этот код выполнится,
    ...  # если <условие> в if возвращает False,
    ...  # <условие 1> в elif тоже возвращает False,
    ...  # а <условие_2> в elif возвращает True.
else:
    ...  # Тело оператора else: этот код выполнится, если
    ...  # условие в if возвращает False и условия во всех блоках elif
    ...  # тоже возвращают False.
```

Пример:

```
if 5 < 10:
    print('Пять меньше десяти!')
```

```
elif 5 == 10:
    print('Пять равно десяти!')
else:
    print('Пять больше или равно десяти!')

# Вывод в терминал: Пять меньше десяти!
```

> Блок `elif` — «синтаксический сахар» для конструкции `else` с вложенным `if`.

Тернарный оператор

Используется, когда на основе условия нужно выбрать одно из двух значений, вместо стандартного `if...else`. Это более короткий и компактный способ записи условных выражений:

```
<действие_если_True> if <условие> else <действие_если_False>
```

Такой код…

```
total_harvest_value = 120

if total_harvest_value:
    print('Урожай получен!')
else:
    print('Урожая нет!')

print('Проверка окончена.')
```

…благодаря тернарному оператору можно сократить до такого:

```
total_harvest_value = 120

print('Урожай получен!') if total_harvest_value else print('Урожая нет!')
print('Проверка окончена.')
```

Логические операторы

Оператор `and`

В выражении `операнд_1 and операнд_2` сначала вычисляется `операнд_1`.

- Если `операнд_1` — это `False`, то выражение сразу вернёт `операнд_1`, а второй операнд даже не будет рассматриваться.
- Если первый операнд — это `True`, логическое выражение с `and` вернёт второй операнд, даже не оценивая его.

Например, выражение `5 > 3 and 8 < 15` вернёт `8 < 15`, ведь `5 > 3` — это `True`, и выражение должно просто вернуть второй операнд.

Оператор `or`

- В выражении `операнд_1 or операнд_2` сначала вычисляется `операнд_1`. Если `операнд_1` равен `True`, то выражение сразу вернёт `операнд_1`, а второй операнд даже не будет рассматриваться.
- Если первый операнд равен `False`, логическое выражение с `or` вернёт второй операнд, даже не оценивая его.

Например, выражение `5 > 3 or 8 < 15` вернёт `True`, ведь `5 > 3` — это `True`, `8 < 15` — это `True`.

Оператор `not`

Работает с одним операндом; единственный операнд ставится справа от оператора: `not значение_или_выражение`.

Оператор `not` инвертирует булевы значения: `not True` вернёт `False`; `not False` вернёт `True`.

```
value = False
result = not value  # В переменной будет храниться значение True.
```