Условный оператор if, ветвления

Оператор if

Этот оператор используется для составления условных блоков, которые нужны для управления потоком выполнения кода. Он проверяет условие и выполняет разные части кода в зависимости от того, соблюдается условие или нет.

Синтаксис использования:

```python
if x > 0:
    # блок кода
```

Сначала идёт сам оператор `if`. Затем условное выражение, результат которого можно трактовать как правду или ложь (`x > 0`).

Условное выражение может быть как простым, так и сложным.

Примеры простых условий

- `if True:` — `True` или `False` могут использоваться напрямую.
- `if 1:` — единица считается эквивалентом `True` в Python.
- `if 0:` — ноль считается эквивалентом `False`.

Пример проверки выражения
Проверим, чтобы число после произведённых с ним операций оказалось не равным 0.

```python
x = 0
if x:  # Условие не сработает, так как x сейчас равен 0.
...
x += 1
if x:  # Теперь сработает, так как x равен 1.
```

Оператор else

Оператор `else` позволяет обработать все остальные возможные варианты, если условие в `if` не выполнено. Например, если `x` равен случайному числу от 1 до 100:

```python
if x == 1:
    # Код, если x равен 1.
else:
    # Код, если x равен всем остальным числам, кроме 1.
```

При написании кода операторы сравнения можно записывать следующим образом:

Операции сравнения		Примеры использования
>	Больше	if a > b:
<	Меньше	if a < b:
>=	Больше либо равно	if a >= b:
<=	Меньше либо равно	if a <= b:
==	Равно	if a == b:
!=	Не равно	if a != b:

Анастасия Верхорубова для Skillbox

Не допускайте следующих ошибок

- Оператор `else` не нуждается в условии.

  Пример 1:

```python
if x == 1:
    # код
else x == 2:  # Такой код будет ошибкой!
    # код
```

Пример 2:

```python
if x == 1:
    # код
else:
    x == 2:   # Это условие никак не влияет на блок else или алгоритм в целом.
```

- Переменные должны быть созданы **до** условного блока.

```python
start = 0
if start > 0:  # Условие не выполнится, так как start равен 0.
    finish = 5 # Переменная не будет создана.

print(finish)  # Выдаст ошибку, так как переменной finish не существует, ведь код finish = 5 не был выполнен.
```

Следуйте этим рекомендациям, чтобы писать понятный и правильный код.