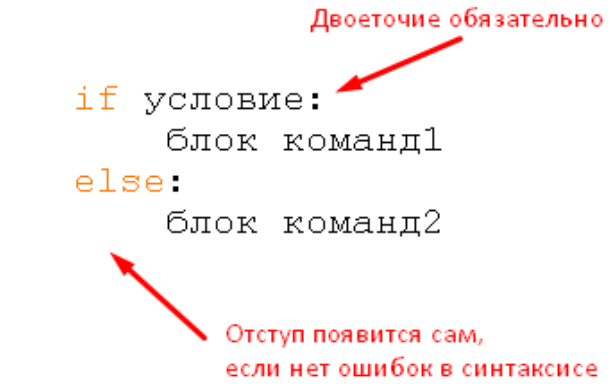
Условный оператор

Условный оператор позволяет выполнять некоторые команды только в том случае, если какое-то Условие выполнится или НЕ выполнится.



If условие:
Делай А
else:
Делай Б
Переводится как «Если условие верно, то делай А, иначе делай Б»

Условие – логическое выражение, на него можно ответить только «Да» или «Нет» (True or False)
Блок команд1 – отделяется отступом (кнопка Tab). Выполняется, если Условие истинно.
Блок команд2 – также отделяется отступом. Выполняется, если Условие Ложно.
else – необязательная конструкция. Её может не быть:

```
if условие:
    блок команд1
```

Если условие истинно, то выполнится блок команд1, иначе не произойдёт ничего
Если тебе нужно несколько условий и несколько блоков команд, используй конструкцию **elif** (else if – иначе если)

```
if условие1:
    блок команд1
elif условие2:
    блок команд2
elif условие3:
    блок команд3
else:
    блок команд4
```

Это единый условный оператор. В нём Питон будет проверять условия по очереди, пока не найдёт верное, выполнит соответствующий блок команд и другие условия проверять уже не будет.
То есть:
Проверяем Условие1. Если оно верно, то выполним блок команд1 и завершаем этот условный оператор (Условие2 и Условие3 проверяться не будут)
Если Условие1 ложно, то проверим условие2 и т.д.
Если все условия ложны, то приходим в else (все другие случаи) и выполняем блок команд4
Конструкций **elif** может быть сколько угодно
В else не может быть никакого условия. Else – все остальные случаи.

Условия

Конструкции сравнения:

a > b **a** больше **b**
a < b **a** меньше **b**
a == b **a** равно **b** (не путать с **a = b** – в **a присвоить** значение **b**. **a == b** – **просто сравнивает**)
a != b **a** не равно **b**
a >= b **a** больше или равно **b**
a <= b **a** меньше или равно **b**
a < b < c **b** больше **a,** но меньше **c**

Условие может состоять из нескольких небольших условий, которые могут связываться логическими операциями:
Условие1 **and** условие2 – верно, когда **оба** условия верны
Условие1 **or** условие2 – верно, когда **хотя бы одно** условие верно
not условие1 – верно, когда условие1 **ложно**

Пример большого условного оператора

```
x = int(input("Введи результат ЕГЭ"))
if x > 100 or x < 0:
    print('Таких баллов на ЕГЭ не бывает')
    atestat = "Cheater"
elif x == 100:
    print("Приглашаем тебя в Google на работу")
    atestat = "5+ и наклейка со звёздочкой"
elif x > 35:
    print("Ты сдал ЕГЭшку")
    atestat = "норрмально"
else:
    print("Стране нужны паровозы. Стране нужен металл")
    atestat = "Not found"
y = input("Ты доволен своим результатом?")
input("Поделись своим мнением. Оно очень важно для нас")
```

Циклы

Конструкция, позволяющая выполнить один и тот же блок кода несколько раз.
Итерация – один полный круг цикла
Циклы бывают **условными** и **безусловными**

Условный цикл (while)

Блок кода выполняется, пока условие истинно. Как только условие становится ложным, цикл завершается.

```
while условие:
    блок кода
```

Двоеточие обязательно

Отступ тоже

Важно: цикл завершается тогда, когда условие перестанет быть истиной. То есть после завершения цикла условие гарантированно будет ложным.

Безусловный цикл (for)

Блок кода выполняется **конкретное количество раз**, пока переменная счётчик не пройдёт все необходимые значения.
Немного о функции **range:**
range(a, b, c) – по очереди возвращает числа от **a** до **b** с шагом **c**. Причём **a** включительно, **b – не включительно.**
range(5, 13, 2)
Вернёт числа: 5, 7, 9, 11
Частный случай:
range(a, b)- если пишем 2 параметра, значит **шаг c** будет по умолчанию **= 1**
range(2, 6)
Вернёт числа: 2, 3, 4, 5
Второй частный случай:
range(b) – если пишем 1 параметр, значит начальное значение **a = 0, шаг c = 1**
range(3)
Вернёт числа: 0, 1, 2
range(6)
Вернёт числа: 0, 1, 2, 3, 4, 5

Возвращаемся к циклу **for:**

```
for i in range(a, b, c):
    блок кода
```

Переменная-счётчик i

Не забываем про двоеточие

И про отступ

В **переменную-счётчик i** по очереди присваиваются значения из функции **range**. Когда числа закончатся, цикл завершится
Цикл **for** используется, когда обрабатывается какой-либо диапазон или известно количество итераций.
Цикл **while** используется, когда есть какое-либо условие выполнения цикла