Условный оператор

if

```
yesterday_temp, today_temp = int(input()), int(input())

if today_temp > yesterday_temp:
    print("Сегодня теплее, чем вчера.")
```

- Сначала идёт if
- Потом идёт условие, которое возвращает логическое значение True или False
- В конце строки двоеточие
- Далее идёт блок кода, который выполняется, если условие возвращает True

Блок кода – это то, что в паскале выделялось с помощью begin и end, а в питоне выделяется четырьмя пробелами в начале строки.

if if

```
yesterday_temp, today_temp = int(input()), int(input())

if today_temp > 0:
    print("Сегодня положительная температура")

    if today_temp > yesterday_temp:
        print("Сегодня теплее, чем вчера.")
```

Если блок кода находится внутри другого блока кода, то он выделяется уже не четырьмя, а восемью пробелами.

if else

```
yesterday_temp, today_temp = int(input()), int(input())
if today_temp > yesterday_temp:
    print("Сегодня теплее, чем вчера.")
else:
    print("Сегодня такая же температура, как вчера.")
```

- Сначала идёт if, условие, двоеточие
- Далее идёт блок кода, который выполняется, если условие возвращает True
- Потом идёт else, двоеточие
- Далее идёт блок кода, который выполняется, если условие возвращает False

if elif else

```
yesterday_temp, today_temp = int(input()), int(input())
if today_temp > yesterday_temp:
    print("Сегодня теплее, чем вчера.")
elif today_temp < yesterday_temp:
    print("Сегодня холоднее, чем вчера.")
else:
    print("Сегодня такая же температура, как вчера.")
```

- Сначала идёт `if`, условие, двоеточие
- Далее идёт блок кода, который выполняется, если условие возвращает `True`
- Потом идёт `elif`, условие, двоеточие
- Далее идёт блок кода, который выполняется, если первое условие возвращает `False`, а второе условие возвращает `True`
- Потом идёт `else`, двоеточие
- Далее идёт блок кода, который выполняется, если оба условия возвращают `False`

Операторов `elif` может быть несколько, а `else` может быть только последним.

bool

В качестве условия для `if` может выступать что угодно, что можно превратить в `bool`.

`bool` – это такой тип данных. У него есть два значения: `True` и `False`.
Значения типа `bool` можно превращать в значения других типов и наоборот:

```
print(bool(11), bool(0))        # True False
print(int(True), int(False))    # 1 0

print(21 > 11, 21 < 11)         # True False
print(11 == 11, 11 != 11)       # True False

print(bool('if'), bool(''))     # True False
print(str(True), str(False))    # True False
```

Почему в последнем примере результат `True False` ?
`str(True)` и `str(False)` - это строки `'True'` и `'False'`.

Операции сравнения

В качестве условия может выступать результат операции сравнения:

- > (больше)
- >= (больше или равно)
- < (меньше)
- <= (меньше или равно)
- == (равно)
- != (не равно)

Сложные условия

Для записи сложных условий можно применять логические операции:

- `and` — логическое «И» для двух условий.
- `or` — логическое «ИЛИ» для двух условий.
- `not` — логическое «НЕ» для одного условия.

Ниже приведена таблица истинности для логических операций.

x	y	`not` x	x `or` y	x `and` y
False	False	True	False	False
False	True	True	True	False
True	False	False	True	False
True	True	False	True	True

Сложные условия

Рассмотрим следующий пример. Пользователь должен ввести первую и последнюю буквы русского алфавита. Ввод производится в двух отдельных строках и в любом регистре.

```
print("Введите первую и последнюю буквы русского алфавита.")
first_letter = input()
last_letter = input()
if (first_letter == "а" or first_letter == "А") and (
        last_letter == "я" or last_letter == "Я"):
    print("Верно.")
else:
    print("Неверно.")

# Введите первую и последнюю буквы русского алфавита.
# а
# я
# Верно.
```

Двойное неравенство

В логическом операторе можно использовать двойное неравенство. Например, неравенство

```
if 0 <= x and x < 100:
    ...
```

лучше записать так:

```
if 0 <= x < 100:
    ...
```

Пару слов про строки

Строки тоже можно сравнивать

```
print('a' == 'b')       # False
print('a' != 'b')       # True
print('a' < 'b')        # True
print('a' <= 'b')       # True

print('aab' < 'abc')    # True
print('ABC' < 'aab')    # True
```

Пару слов про строки

Для проверки условия наличия подстроки в строке используется оператор in. Например, проверим, что во введённой строке встречается корень «добр» (для слов «добрый», «доброе» и подобных):

```
text = "Красота без доброты умирает невостребованной. (Бенджамин Франклин)"
if "добр" in text:
    print("Встретилось 'доброе' слово.")
else:
    print("Добрых слов не найдено.")

# Встретилось 'доброе' слово.
```

Некоторые полезные функции

- Для определения **длины** строки (и не только) используется функция `len()`.
- Для определения **максимального** и **минимального** из нескольких значений (не только числовых) используются функции `max()` и `min()` соответственно.
- Функция `abs()` используется для определения **модуля числа**.

```
m, n, k = 12, 19, 25
print(max(m, n, k)) # 25

line_1, line_2, line_3 = "m", "n", "k"
print(min(line_1, line_2, line_3))  # k

print(len(str(2 ** 2022)))  # 609
```

Все доступные в Python встроенные функции можно посмотреть тут:
https://docs.python.org/3/library/functions.html