Условия и булева алгебра

Логический тип данных

В число встроенных типов данных языка Python входит также **логический**, или же **булевый** тип данных - `bool`. Он может принимать только 2 значения:

- `True` - истина
- `False` - ложь
  Многие методы и функции языка возвращают этот тип данных, и именно он используется для **условий**.
  Пример объявления такого типа данных:

```
a = True
b = False

print(a, b) # выведет True, False
```

В отличие от других типов данных, у логического типа нет операций сложения, умножения, и деления. Вместо них присутствуют три оператора: `and`, `or` и `not`:

- `and` применяется к двум логическим значениям и возвращает `True`, если оба значения - `True`. В остальных случаях - `False`.
- `or` применяется к двум логическим значениям и возвращает `True`, если хотя бы одно из значений - `True`. В остальных случаях - `False`.
- `not` применяется к одному логическому значению и возвращает `True`, если исходное значение `False`, и `False` - если исходное `True` (то есть инвертирует значение).

Пример использования операторов:

```
a = True
b = False

print(a and b)
print(a or b)
print(not a)
print(not b)
```

Вывод программы:

```
False
True
False
True
```

Помимо функций и методов, возвращающих `bool`, операторы сравнения также возвращают этот тип данных:

```
a = 10

print(a == 10)        # проверка на равенство
print(a != 20)        # проверка на неравенство
print(a > 3)          # проверка на то, что число a больше трех
print(a < 100)        # проверка на то, что число a меньше 100
print(a >= 0 or a == 100) # число a больше или равно нуля, либо a равно 100
```

```
print(a != 10 and a < 20) # число a не равно 10 и a меньше 20
print(a % 2 == 0)     # остаток от деления a на 2 равен нулю (проверка на четность)
```

Вывод программы:

```
True
True
True
True
True
False
True
```

Пример с реальными функциями:

```
text = "cat dog fox"

print(len(text.split()) == 3 and text.endswith("fox"))
# проверка на то, что строка text содержит 3 слова, разделенных пробелом и оканчивается на `fox`
# выведет True
```

Условия

Некоторые утверждения и операции необходимо выполнять только в определенных условиях. Условия в языке Python - это части кода, которые выполнятся только тогда, когда само условие - логический тип данных - равен `True`.

Условие состоит из обязательной части `if`, опционального дополнительного условия `elif`, и опционального *прочего случая* `else`.

- Часть `if` выполнится, если условие внутри нее равно `True`.
- Часть `elif` выполнится, если часть `if` и предыдущие `elif` не выполнились, и условие внутри нее равно `True`.
- Часть `else` выполнится, если все прочие части этого условия не выполнились. Часть `else` не содержит в себе условия.

Синтаксис условия таков:

```
if условие:
    действие
elif другое условие:
    другое действие
else:
    действие в прочих случаях
```

Части `elif` и `else` - необязательные части условия.

Пример с получением числа вводом от пользователя и проверками:

```
num = int(input())                    # получаем число от пользователя

if num > 0:                           # если число больше нуля, то...
    print("Число положительное")
elif num == 0:                        # или если число равно нулю, то...
    print("Число равно нулю")
else:                                 # если ни то, ни другое...
```

```
    print("Число отрицательное")

if num % 2 == 0:                        # если число делится на 2 без остатка, то...
    print("Число четное")
else:                                   # иначе
    print("Число нечетное")
```

Ввод и вывод программы:

```
Input: 10
Output:
Число положительное
Число четное

Input: -19
Output:
Число отрицательное
Число нечетное
```

В условии могут находиться любые функции и выражения, дающие в результате `True` или `False`.
Более комплексный пример со строками:

```
text = "cat fox dog chicken cuckoo"

if len(text) < 10 and text.startswith("cat") and text.lower() == text:
    print("Строка меньше 10 символов, начинается на 'cat' и состоит только из строчных букв")
else:
    print("Условие не выполнено!")

if text[::-1] == text:
    print("Строка является палиндромом!")
```

Вывод:

```
Строка меньше 10 символов, начинается на 'cat' и состоит только из строчных букв
```