

Модуль #2

Построение таблиц истинности и
преобразование логических выражений.



В данном модуле изучим задания
КЕГЭ #2 и #15, содержащие
основные понятия алгебры логики.

Алгебра логики – раздел
математический логики,
находящийся на стыке алгебры, и
логики.

Встречается также название булева
алгебра в честь её
основоположника, *Джорджа Буля*.

В школьной алгебре у нас есть операции над переменными, которые обозначаются строчными буквами латинского алфавита (a, b, x, y). В логике же, в свою очередь, есть основные понятия об истине и лжи (*true and false*). На стыке именно этих дисциплин и находится алгебра логики.

Объектами алгебры логики являются высказывания — какие-то утверждения, относительно которых можно сказать, истинны они или же ложны.

Высказывания обозначают заглавными английскими буквами и называют логическими переменными.

Если высказывание истинно, то значение соответствующей ему логической переменной обозначают единицей, если ложно – нулём.
0 и 1 – логические значения.



Рассмотрим основные операции над
логическими высказываниями:

Дизъюнкция

Дизъюнкция – логическое «или»
(логическое сложение).

Обозначается « \vee ». Выражение $A \vee B$
читается как «A или B»

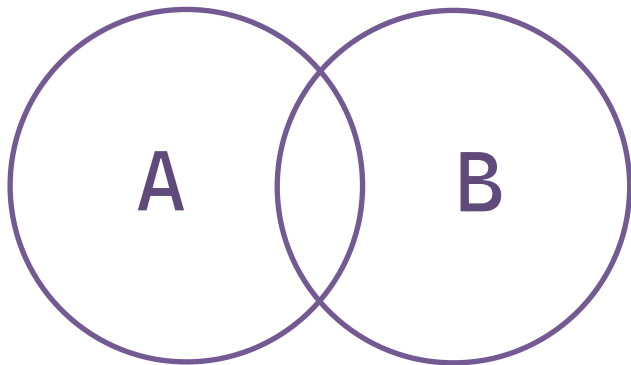


Таблица истинности:

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

Таблица истинности логического выражения F – таблица, где расписаны все возможные комбинации логических значений и результат выражения, соответствующий им.

Конъюнкция

Конъюнкция – логическое «и»
(логическое умножение).

Обозначается « \wedge ». Выражение $A \wedge B$
читается как «A и B»

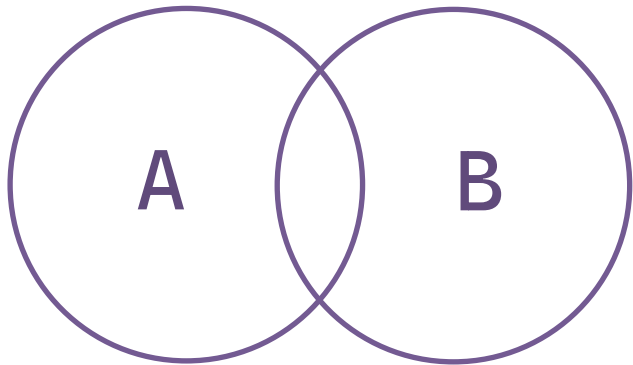


Таблица истинности:

A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

Отрицание и эквивалентность

При отрицании/инверсии (логическое «не»)
заменяем логическое значение на
противоположное.

Эквивалентность даёт истину тогда и только
тогда, когда значение логических выражений
совпадают.

Закон двойного отрицания:

$$\neg(\neg A) = A$$

(минус на минус даёт плюс)

A	$\neg A$
1	0
0	1

A	B	$A \equiv B$
1	1	1
1	0	0
0	1	0
0	0	1

Импликация

Импликация (следование) – более сложная операция, близкая по своему значению к высказыванию «Если А, то В»

Обозначается « \rightarrow ». Выражение $A \rightarrow B$ читается как «А импликация В».

Импликация ложна тогда и только тогда, когда из истинного выражения следует ложное.

Таблица истинности:

A	B	$A \rightarrow B$
1	1	1
1	0	0
0	1	1
0	0	1

Основные законы алгебры логики

Закон	Дизъюнкция	Конъюнкция
Переместительный		
Сочетательный		
Распределительный		
Де Моргана		
Идемпотентности		
Исключения 3-го		
Операции с константами		
Поглощения		
Склеивания		



Задание #2

Построение таблиц истинности

Пример задания #2

2

Миша заполнял таблицу истинности логической функции F

$$\neg(y \rightarrow x) \vee (z \rightarrow w) \vee \neg z,$$

но успел заполнить лишь фрагмент из трёх различных её строк, даже не указав, какому столбцу таблицы соответствует каждая из переменных w, x, y, z .

				F
	0			0
0	1			0
1			0	0

Определите, какому столбцу таблицы соответствует каждая из переменных w, x, y, z .

В ответе напишите буквы w, x, y, z в том порядке, в котором идут соответствующие им столбцы (сначала буква, соответствующая первому столбцу; затем буква, соответствующая второму столбцу, и т.д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Пример задания #2

Пример. Функция F задана выражением $\neg x \vee y$, зависящим от двух переменных, а фрагмент таблицы имеет следующий вид.

		F
0	1	0

В этом случае первому столбцу соответствует переменная y , а второму столбцу – переменная x . В ответе следует написать: yx .

Варианты выполнения задания #2:

- Построение полной таблицы истинности выражения

Крайне долгий способ, который не только тратит время, но и не даёт понимания. Категорически не рекомендую использовать на практике, но попробовать стоит хотя бы раз.

- Аналитическое решение

Аналитическое решение является оптимальным вариантом, однако подходит не для всех заданий. Этим способом решения овладеть крайне желательно.

- Программное решение

Этот метод является минимальным, овладеть им необходимо, но сделать это нужно в последнюю очередь, для того, чтобы не действовать бездумно.

Обозначение логических операций в python

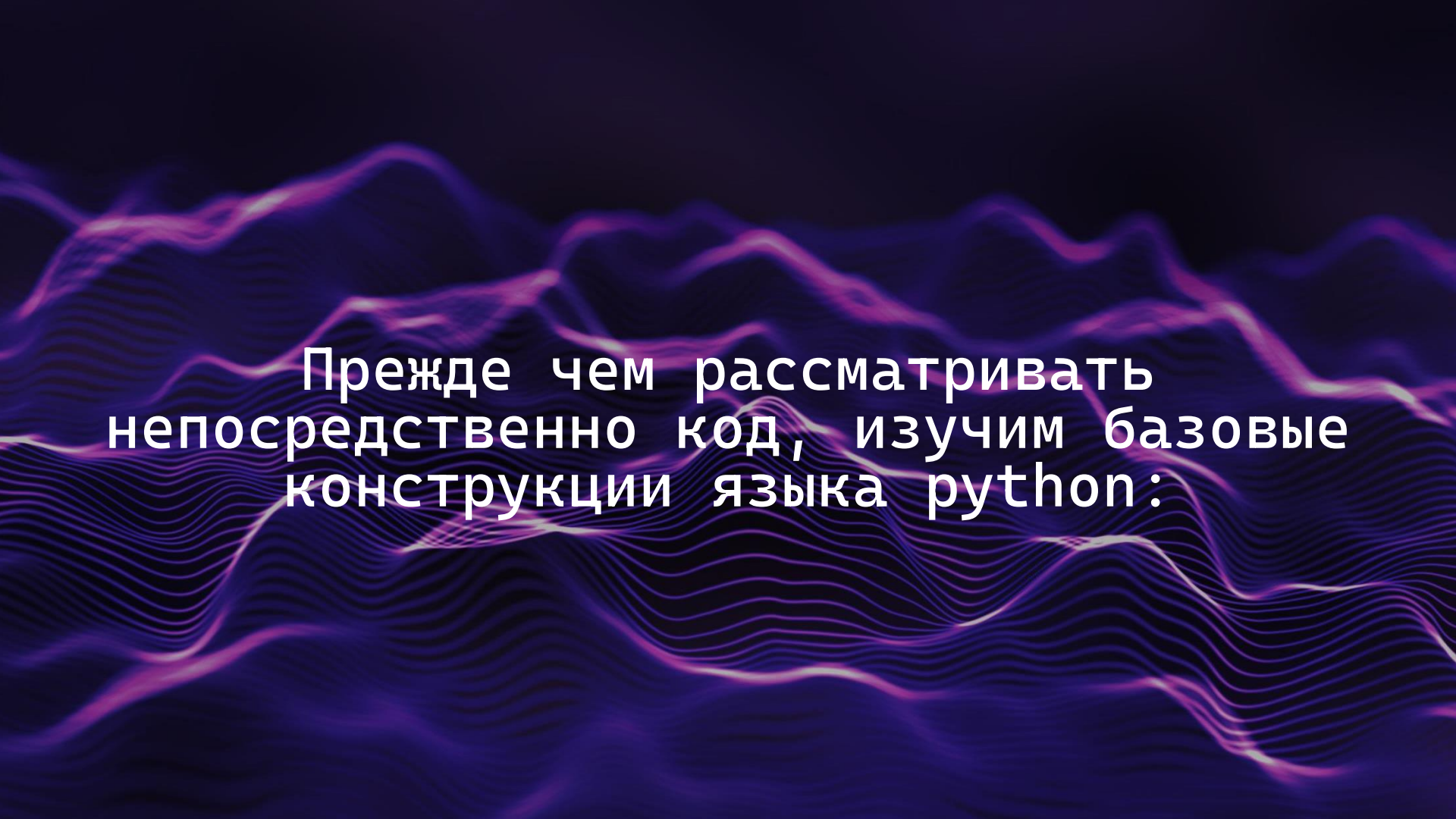
Выражение	Запись в python
$A \vee B$	<code>A or B</code>
$A \wedge B$	<code>A and B</code>
$\neg A$	<code>not(A)</code>
$A \equiv B$	<code>A == B</code>
$A \rightarrow B$	<code>A <= B</code>

Алгоритм выполнения задания #2 на python:

Решение этого задания на python будет сводиться к построению таблицы истинности – то есть перебору всех возможных комбинаций значений логических переменных, при которых выражение принимает заданное значение. Только в данном случае не придётся перебирать всё вручную – реализуем это в виде программы.

Алгоритм выполнения задания #2 на python:

1. Определим, какие переменные задействованы в выражении, исходя из условия.
2. Создаём несколько вложенных циклов **for** для логических переменных. Вспомним, что они могут принимать только значения 0 или 1. Следовательно, циклы нужны для перебора всевозможных комбинаций значений переменных.
3. Добавляем условие «если логическое выражение равно константе» в последний вложенный цикл, переписывая логическое выражение из задания и проверяя его равенство какому-то логическому значению, в зависимости от условия.
4. Если условие выполняется – печатаем значение всех переменных.



Прежде чем рассматривать
непосредственно код, изучим базовые
конструкции языка python:

Функция `print()`

Данная функция предназначена для вывода информации в консоль. В скобках указывается аргумент функции. Например, результатом выполнения `print(2)` будет вывод в консоль числа 2.

Также в качестве аргумента можно передать константу. Результатом выполнения следующего кода также будет вывод в консоль числа 2.

```
a = 2  
print(a)  
>>> 2
```

А вот если в скобках передать какой-то текст в кавычках, то будет напечатана строка. Кстати, аргументы функции можно передавать через запятую, чтобы напечатать всё необходимое.

```
a = 2  
print("Вывод числа", a)  
>>> Вывод числа 2
```

Управляющий цикл for

В основе цикла for лежат последовательности, и в примере ниже это последовательность чисел от 1 до 100. Мы записываем последовательность в диапазон (range). for поэлементно её перебирает и выполняет код, который записан в теле цикла.

```
for i in range(1, 100):
```

```
    print(i)
```

```
>>> 1
```

```
>>> 2
```

```
>>> 3
```

```
...
```

```
>>> 99
```

Условный оператор `if`

Условный оператор `if` позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. После оператора `if` записывается выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. В примере ниже это вывод сообщения, если число больше 5.

```
a = 10
if a > 5:
    print("Число", a, "больше", 5)
>>> Число 10 больше 5
```


Код задания #2 на python:

```
print("x y w z")
for x in range(2):
    for y in range(2):
        for w in range(2):
            for z in range(2):
                if (not(y <= x) or (z <= w) or not(z)) == 0:
                    print(x, y, w, z)
```

Вывод:

```
x y w z
0 0 0 1
1 0 0 1
1 1 0 1
```

Задание #15

Преобразование логических выражений

Виды заданий #15:

- Делимость

В таких заданиях в качестве логического выражения выступает целочисленная делимость: делится ли число n на число m нацело, то есть без остатка, или нет.

- Побитовая конъюнкция

Здесь используется операция умножения между разрядами чисел в двоичной записи.

- Числовые отрезки

В качестве логического выражения здесь выступает принадлежность числа отрезку.

- Множества

В качестве логического выражения здесь выступает принадлежность числа множеству.

- Координатная плоскость

Здесь в качестве переменной выступает не только число x , но и число y .

Задание #15 содержит логическую функцию с параметром A . Необходимо определить:

- наибольшее/наименьшее значение параметра A
- наибольшую/наименьшую длину отрезка A
- количество/сумму элементов множества A .

Помимо A , в функции содержится переменная x и опционально переменная y .

Функции в языке python

Мы уже использовали функции языка python в своих программах, например, функцию `print()`, выводящую сообщение на экран или функцию `range()`, создающую последовательность чисел. Эти функции были стандартными, они уже определены в стандартной библиотеке языка python. Мы можем понять, как они работают, но мы не знаем, как именно они реализованы на уровне языка – процесс их работы скрыт от нас, мы видим лишь результат.

Однако, мы можем написать свою функцию. У неё, как и у функции `print()` будет своё имя (только не `print`, а какое мы пожелаем) и набор аргументов, который она будет принимать. Будет у неё и результат работы – какое-то возвращаемое значение.

Функции в языке python

Функция – именованный фрагмент кода, часть программы, которая будет многократно повторяться.

Функция в python определяется оператором `def`, за которым следуют имя функции и круглые скобки. В круглых скобках пишутся аргументы, которые принимает функция.

Аргумент – это значение, которое передается функции при ее вызове.

Блок кода внутри каждой функции начинается с двоеточия и должен иметь отступ. Команда `return` указывает, какое значение нужно вернуть вызывающей функции.

Для примера рассмотрим функцию, принимающую число и возвращающую его квадрат:

```
def f(x):  
    return x * x
```

f – имя функции

x – аргумент функции

x^2 – возвращаемое значение

```
print(f(5))
```

Вызвав эту функцию с аргументом 5, получим число 25.

Решение с функцией all

Для начала напишем собственную функцию, переписав в неё формулу из задания

```
def f(x, a):  
    return ((72 % x == 0) <= (120 % x != 0)) or (a - x > 100)  
  
for a in range(0, 200):  
    #функция all возвращает истину, если некоторая булева функция от x верна для всех x  
    if all(f(x, a) == 1 for x in range(1, 1000)):  
        print(a)
```


Решение с использованием счётчика

```
def f(x, a):  
    return ((72 % x == 0) <= (120 % x != 0)) or (a - x > 100)  
  
for a in range(0, 200):  
    s = 0 # создаём счётчик для значений x, при которых функция истинна при данном a  
    for x in range(1, 1000):  
        if f(x, a) == 1:  
            s = s + 1  
    if s == 999:  
        print(a)
```

в этом диапазоне 999
значений (от 1 до 999
включительно)

Решение с использованием флага

Флаг – булева переменная, которая может иметь два значения: истина и ложь (True/False).

```
for a in range(1, 200):
    flag = True # флаг изначально поднят
    for x in range(1, 2000):
        f = (x % a == 0) <= ((x % a == 0) <= (x % 34 == 0) and (x % 51 == 0))
        if f == 0:
            flag = False # флаг опускается, если нашли хотя бы один x, при котором функция ложна
    # если прошли все x из диапазона (для данного a), а флаг остался поднят, значит ответ найден
    if flag == True:
        print(a)
```

Поразрядная конъюнкция

Поразрядная логическая операция «и» (обозначается символом $\&$) выполняется между соответствующими битами двоичной записи двух целых чисел. Её результат – целое число, а не истина или ложь, как у обычной конъюнкции.

Например, найдём результат операции $14 \& 5$:

$$\begin{array}{rcl} \& 14_{10} & = 1110_2 \\ \& 5_{10} & = 0101_2 \\ \hline & 0100_2 & = 4_{10} \end{array}$$

Поразрядная конъюнкция

Алгоритм поразрядной конъюнкции следующий:

- 1) Переводим два данных числа в двоичную СС
- 2) Записываем двоичные числа друг под другом при необходимости дописывая незначащие нули слева так, чтобы числа были одинаковой длины
- 3) Умножаем соответствующие разряды чисел попарно и записываем результат под исходными числами
- 4) Переводим результат обратно в десятичную СС – это и будет результат поразрядной конъюнкции

Поразрядная конъюнкция

В заданиях КЕГЭ в качестве логических операций, связанных с поразрядной конъюнкцией используется равенство её нулю, которое даёт истину или ложь.

Стоит помнить, что результаты операций $(a \& b = 0)$ и $(a \& b \neq 0)$ – это логические значения: истина или ложь, тогда как результаты самой поразрядной конъюнкции – целые числа.

Пример задания #15

15

Обозначим через $m \& n$ поразрядную конъюнкцию неотрицательных целых чисел m и n . Например, $14 \& 5 = 1110_2 \& 0101_2 = 0100_2 = 4$.

Для какого наименьшего неотрицательного целого числа A формула

$$x \& 73 = 0 \rightarrow (x \& 28 \neq 0 \rightarrow x \& A \neq 0)$$

тождественно истинна (т. е. принимает значение 1 при любом неотрицательном целом значении переменной x)?

Ответ: _____.