



Python. Числовые типы данных. Условный оператор. Логический тип.

Шухман Александр Евгеньевич,
Заведующий кафедрой
геометрии и компьютерных наук Оренбургского
государственного университета

СПИСОК ИСТОЧНИКОВ

Книги

- Тонни Гэддис. Начинаем программировать на Python
- Н. А. Прохоренок, В. А. Дронов. Python 3. Самое необходимое
- <https://pythonworld.ru/>

Онлайн-курсы

- [Пайтонтьютор https://pythontutor.ru/](https://pythontutor.ru/)
- "Поколение Python"
<https://stepik.org/course/58852/syllabus>
<https://stepik.org/course/68343/syllabus>

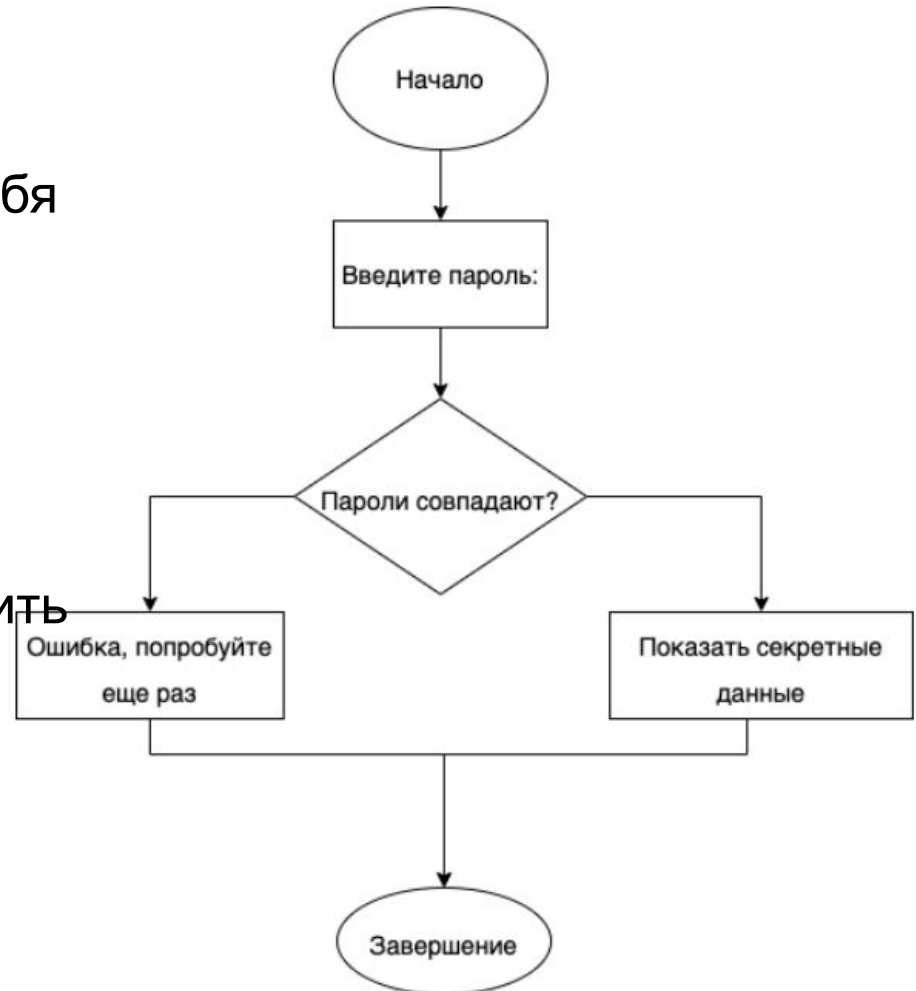
Программирование

Программирование – это деятельность по созданию программного обеспечения.

Программирование включает в себя разработку алгоритмов решения различных практических задач и их реализацию в виде компьютерных программ.

Алгоритм - понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленной цели.

Программа - это алгоритм, переведенный на понятный компьютеру язык, который будет выполнен на компьютере.



Особенности Python

- Интерпретируемый язык программирования высокого уровня
- Динамическая типизация
- Высокоуровневые структуры данных
- Поддерживает структурное, объектно-ориентированное, функциональное программирование
- Активно развивающийся (последняя версия 05.10.21, python 3.10.0)
- Большое количество различных библиотек
- Области применения – анализ данных, веб-разработка, системное программирование

Среды разработки

- Интерпретатор + IDLE

<https://www.python.org/>

- Wing IDE,

<http://wingware.com/>

- PyCharm,

<https://www.jetbrains.com/ru-ru/pycharm/>

Сравнение языков



Python

```
print('Hello, World!')
```

```
#include <iostream>
```

C++

```
int main() {  
    std::cout << "Hello, World!";  
    return 0;  
}
```

Pascal

```
begin  
    writeln('Hello, World!')  
end.
```

```
class Program
```

C#

```
{  
    Ссылка: 0  
    static void Main()  
    {  
        System.Console.WriteLine("Hello, World!");  
    }  
}
```

Структура программы

- **Программа** на Python – текст, содержащий последовательность команд (операторов).
- **Оператор** – предложение языка, описывающее определенное действие. Обычно каждый оператор записывается в отдельной строке программы.
- В программе могут быть также определения функций и классов, которые начинаются с ключевого слова `def` (будут изучаться позже).
- В операторах могут использоваться ключевые слова. **Ключевые слова** – английские слова, имеющие специальные значения. Эти слова зарезервированы и не могут использоваться в другом качестве, например в качестве имен. Среда разработки автоматически их выделяет в тексте программы.

```
>>> import keyword
```

```
>>> keyword.kwlist
```

```
['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
```

```
'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',
```

Константы

- Операторы языка работают с данными (числами, текстами, множествами и пр.), которые хранятся в памяти компьютера
- **Константы** - это данные, которые зафиксированы в тексте программы и не изменяются в процессе ее выполнения
- Примеры констант:
 - **целые** числа: 12, -23, 0b1010001
 - **действительные** числа: 1.0, -7.8
 - **логические**: True (истина), False (ложь)
 - **строковые**: "I'm study Python", 'информатика'

Переменные

- **Переменные** – это данные, которые могут изменять свои значения в ходе выполнения программы
- Переменная имеет
 - **Имя (идентификатор)** - как обращаться к переменной
 - **Значение** – что лежит в переменной
- **Оператор присваивания** устанавливает связь между именем и значением переменной

имя = выражение

- **Имя** переменной может содержать буквы, цифры, знак `_` и не может начинаться с цифры

```
>>> x = 2 + 2
>>> x
4
>>> name = "Вася"
>>> name
'Вася'
>>> |
```

Выражения

- **Выражения** могут включать константы, переменные, вызовы функций, соединенные знаками операций. Вычисление значения выражение выполняется в соответствии с приоритетами операций
- Каждое выражение имеет **значение**, которое относится к одному из **типов данных**
- **Тип данных** определяет
 - Область допустимых значений
 - Допустимые операции
 - Объём и структуру памяти для хранения значения
- Python использует динамическую типизацию (тип переменной определяется ее значением) и строгий контроль типов

```
>>> (x + 2) * x - 10
14
>>> name * 2
'ВасяВася'
```

Простые типы в Python

Целое число

```
>>> a=5
>>> type(a)
<class 'int'>
```

Строка

```
>>> s="Hello"
>>> s='Hello'
>>> type(s)
<class 'str'>
```

Действительное

```
>>> x=3.5
>>> type(x)
<class 'float'>
```

число

Комплексное

```
>>> z=3+5j
>>> type(z)
<class 'complex'>
```

число

Логический тип

```
>>> b=True
>>> type(b)
<class 'bool'>
```

Арифметические операции

+, -, *

/ - частное от деления

// - целая часть от деления

% - остаток от деления

```
>>> -10//3
-4
>>> -10%3
2
```

```
>>> -10//(-3)
3
>>> -10%(-3)
-1
>>> 10//(-3)
-4
>>> 10%(-3)
-2
```

```
>>> -10.5//3
-4.0
>>> -10.5%3
1.5
>>> -10.5//3.5
-3.0
>>> -10.5%3.5
0.0
```

Арифметические операции

** возведение в степень

```
>>> 3**4  
81
```

Все арифметические операции можно сокращать с присваиванием: +=, -=, ...

```
>>> a=5  
>>> a**=2  
>>> a  
25
```

```
>>> b=7  
>>> b*=3  
>>> b  
21
```

Математические функции

Модуль `math` нужно подключить командой `import math`

`sqrt(x)` – квадратный корень из x

`fabs(x)` – модуль x

`sin(x)`, `cos(x)`, `tan(x)` – тригонометрические функции

`floor(x)` – округление вниз

`ceil(x)` – округление вверх

```
>>> import math
>>> math.sqrt(16)
4.0
```

Перенос выражения

Перенос можно делать
внутри скобок или с
помощью знака \

```
>>> (3+5-  
7)
```

```
1
```

```
>>> 3+5\  
-7
```

```
1
```

```
>>> 3+5
```

```
8
```

Комментарии

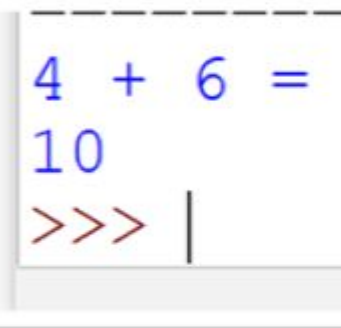
- # комментирует весь текст до конца строки;
- ''' закоментированный текст '''
- """ закоментированный текст """

```
a=3+5#-4
''' комментарий
многострочный
'''
```


Вывод данных

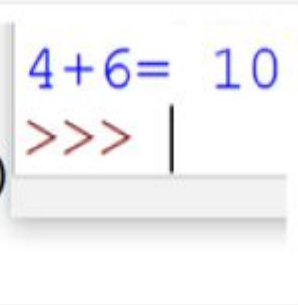
- Используется функция `print(выр.1, выр.2...)`
- По умолчанию между выводимыми объектами ставится пробел, а в конце перевод строки

```
a = 4
b = 6
print(a, '+', b, '=')
print(a+b)
```



- Можно изменить разделитель в параметре **sep** и последний символ в параметре **end**

```
a = 4
b = 6
print(a, '+', b, '=', sep='', end=' ')
print(a+b)
```



ФОРМАТНЫЙ ВЫВОД

```
a = 4
b = 6
print (' {}+{}={}' .format (a,b,a+b) )
```

```
4+6=10
```

```
>>> |
```

```
a = 4
b = 6
print (f"{a} + {b} = {a + b}")
4 + 6 = 10
|
```

```
import math
print ('{:10.3f}' .format (math.pi) )
```

```
=== RESTART: 
```

```
3.142
```

```
>>> |
```

Ввод данных

- Вводится целая строка!

```
a = input() | 36  
print(type(a)) | <class 'str'>
```

- При необходимости её надо преобразовывать в число:

```
a = int(input()) | 35  
print(a+1) | 36
```

```
print("Как тебя зовут?")  
s = input()  
print(f"Привет, {s}")
```

Ввод данных

- Можно выводить строку с вопросом перед вводом переменной:

```
a = int(input('Enter '))  
print(a+1)
```

Enter 45
46

A + B

```
a = int(input())  
b = int(input())  
c = a+b  
print(c)
```

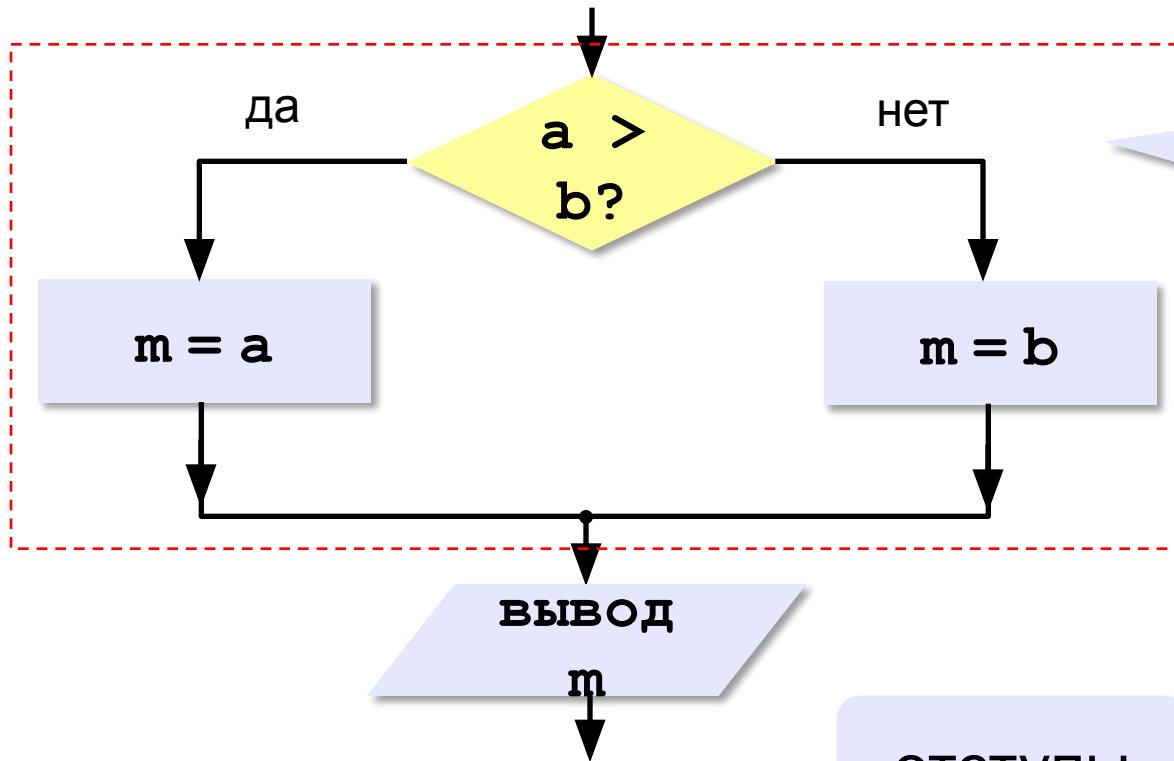
12
15
27
>>>

```
print(int(input())+int(input()))
```

12
15
27

Условный оператор в Python

Пример – определение максимума из двух чисел



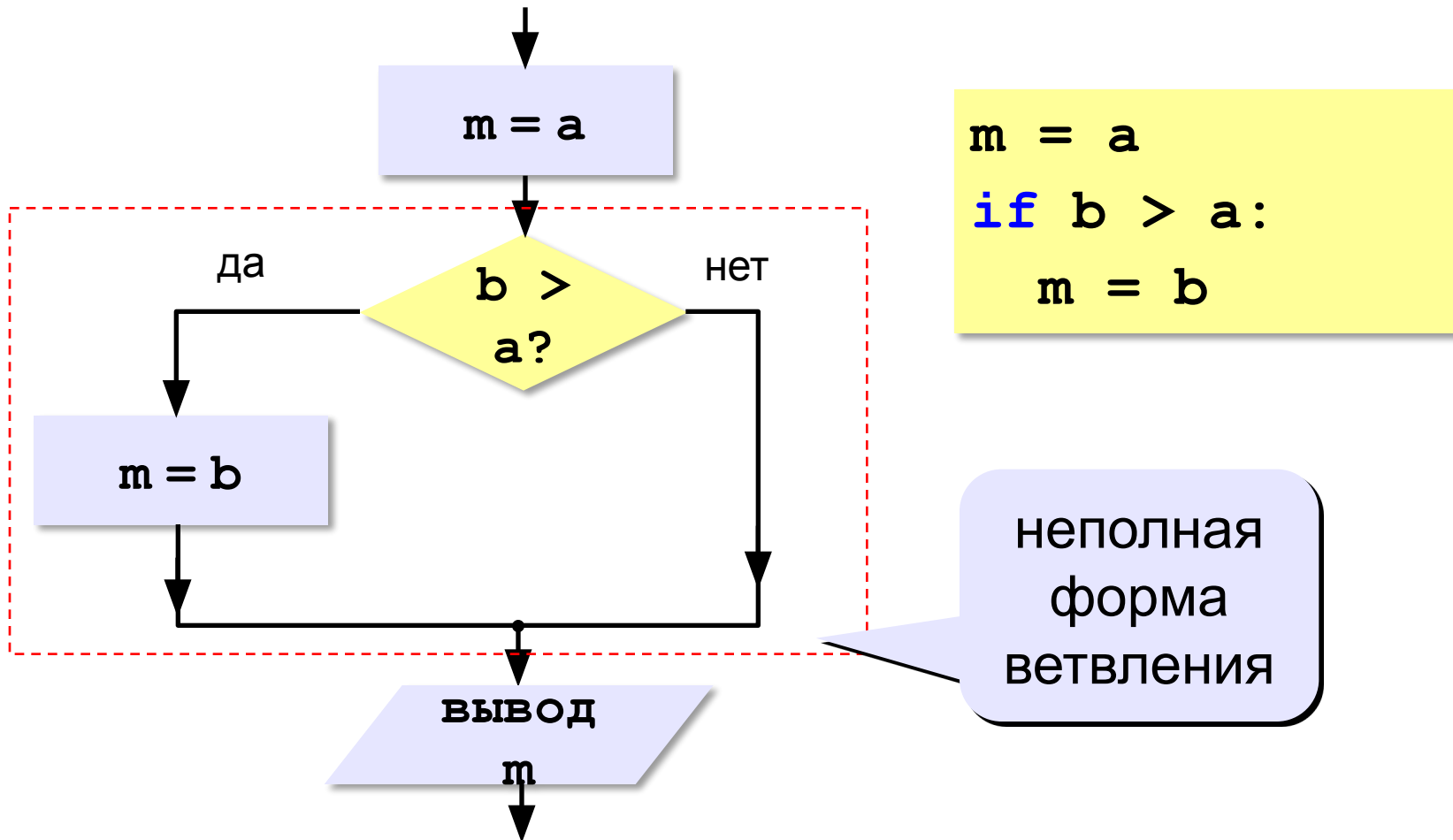
полная
форма
ветвления

? Если $a = b$?

```
if a > b:  
    m = a  
else:  
    m = b
```

ОТСТУПЫ

Неполная форма



Решение в стиле Python:

```
m = max(a, b)
```

```
m = a if a > b else b
```

Операции сравнения

> **<** больше, меньше

>= больше или равно

<= меньше или равно

== равно

!= не равно

Вложенные условия

Задача: определить оценки студента на основе введенных баллов

Пользователь вводит оценку

Если оценка ≥ 80

Вывод: "отлично"

Иначе если оценка ≥ 60

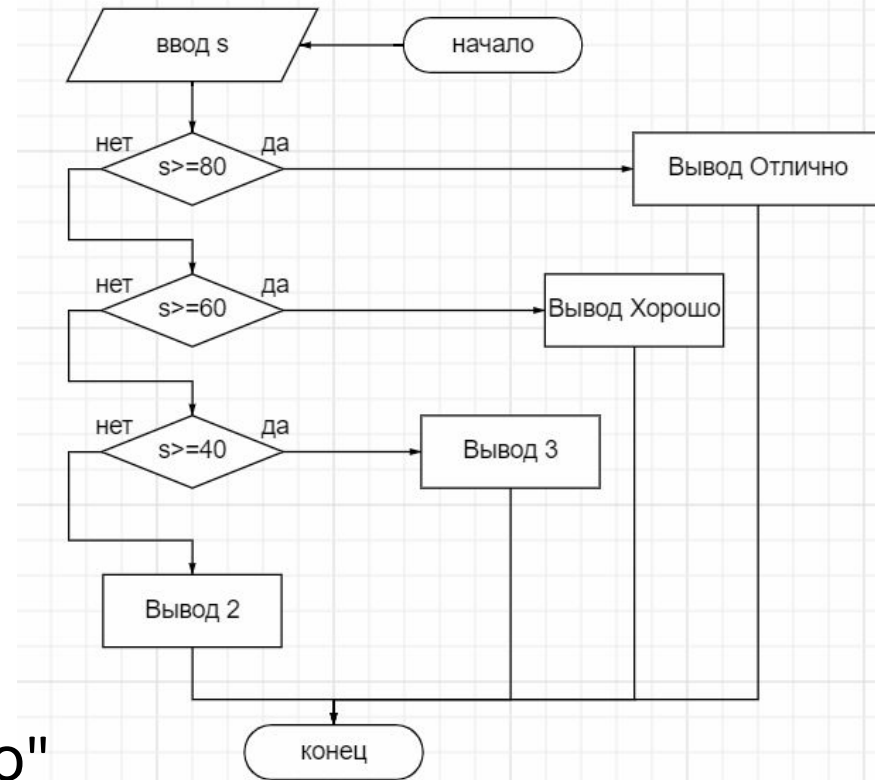
Вывод: "хорошо "

Иначе если оценка ≥ 40

Вывод: "удовлетворительно"

Иначе

Вывод: "неудовлетворительно"



Вложенные условия в Python

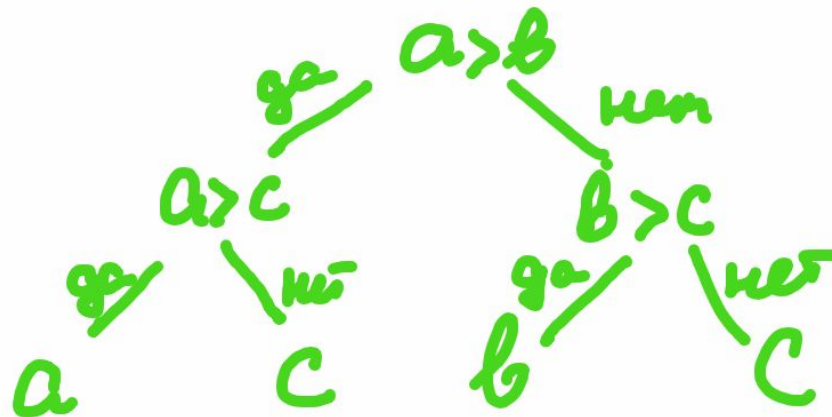
```
score = int(input())
if score >= 80:
    print("отлично")
else:
    if score >= 60:
        print("хорошо")
    else:
        if score >= 30:
            print("удовлетворительно")
        else:
            print("неудовлетворительно")
```

Каскадные условия в Python

```
score = int(input())
if score >= 80:
    print("отлично")
elif score >= 60:
    print("хорошо")
elif score >= 30:
    print("удовлетворительно")
else:
    print("неудовлетворительно")
```

Пример: максимум из трех чисел

```
a = int(input())
b = int(input())
c = int(input())
if a > b:
    if a > c:
        print(a)
    else:
        print(c)
elif b > c:
    print(b)
else:
    print(c)
```



```
a = int(input())
b = int(input())
c = int(input())
if a <= b <= c:
    print(c)
elif b <= a <= c:
    print(c)
elif a <= c <= b:
    print(b)
elif c <= a <= b:
    print(b)
else:
    print(a)
```

Логический тип данных

- Выражения логического типа в Python принимают одно из двух значений True (истина) и False (ложь)
- Логический тип называется bool в честь Джорджа Буля
- Условия ==, !=, <, >, <=, >= вычисляют значение логического типа
- Для логического типа можно использовать специальные логические операции

Логическое умножение (and, и)

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

- Логическое выражение a and b **ИСТИННО**, только если **оба** значения a и b **ИСТИННЫ**
- В общем случае значение выражения с оператором and истинно, если истинны **все** объединенные им условия

Пример на логическое умножение

Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

Пример:

5

Весна

Пример:

15

Неверный номер месяца

```
m = int(input())  
if m >= 6 and m <= 8:  
    print('Лето')
```

Логическое сложение (or, или)

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

- Логическое выражение $a \text{ or } b$ **ИСТИННО**, если **ХОТЯ БЫ ОДНО** значение a и b **ИСТИННО**
- В общем случае значение выражения с оператором or истинно, если истинно **ХОТЯ БЫ ОДНО** условие

Пример на логическое сложение

```
m = int(input())  
if m == 12 or m <= 2:  
    print('Зима')  
|
```

Логическое отрицание (not, не)

a	not a
False	True
True	False

- Логическое выражение not a **ИСТИННО**, если a ложно и наоборот

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно)

сложное условие

```
if v >= 25 and v <= 40 :  
    print ("подходит")  
else:  
    print ("не подходит")
```

Приоритет :

- 1) отношения (<, >, <=, >=, ==, !=)
- 2) **not** («НЕ»)
- 3) **and** («И»)
- 4) **or** («ИЛИ»)