

Условное исполнение

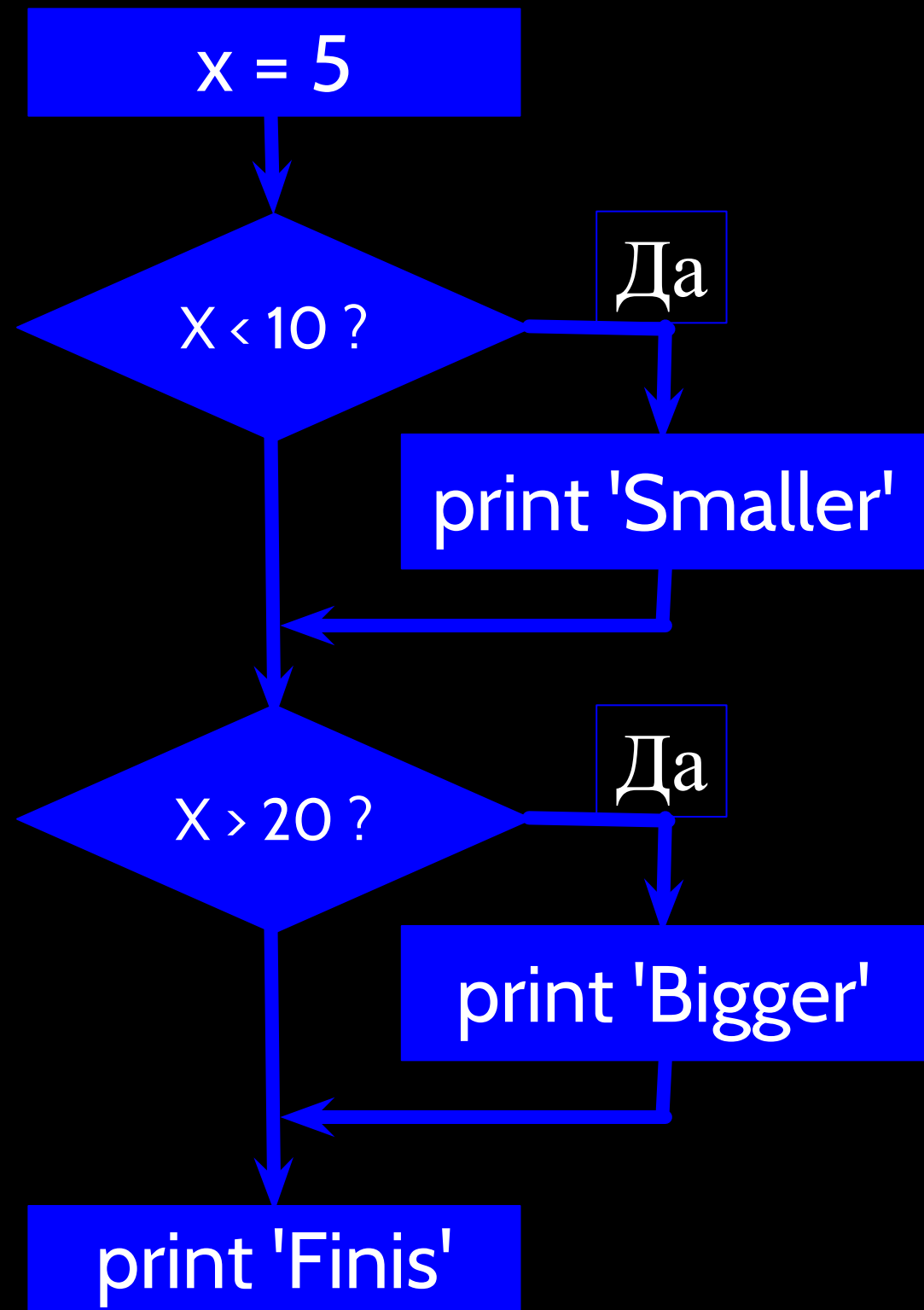
Глава 3



Python for Informatics: Exploring Information
www.pythonlearn.com



Условные шаги



Программа:

```
x = 5
if x < 10:
    print 'Smaller'
if x > 20:
    print 'Bigger'
print 'Finis'
```

Результат:

Smaller
Finis

Операторы сравнения

- **Логические выражения** задают вопрос и производят результат “Да” или “Нет”, с помощью которого мы контролируем ход выполнения программы
- **Логические выражения** с использованием **операторов сравнения** производят True / False - да/нет
- **Операторы сравнения оценивают** переменные **без изменения** значений переменных

Python	Значение
<	Меньше чем
<=	Меньше или равно
==	Равно
>=	Больше или равно
>	Больше чем
!=	Не равно


Запомните! “=” используется для присваивания.

http://en.wikipedia.org/wiki/George_Boole

Операторы сравнения

```
x = 5
if x == 5 :
    print 'Equals 5'
if x > 4 :
    print 'Greater than 4'
if x >= 5 :
    print 'Greater than or Equals 5'
if x < 6 : print 'Less than 6'
if x <= 5 :
    print 'Less than or Equals 5'
if x != 6 :
    print 'Not equal 6'
```

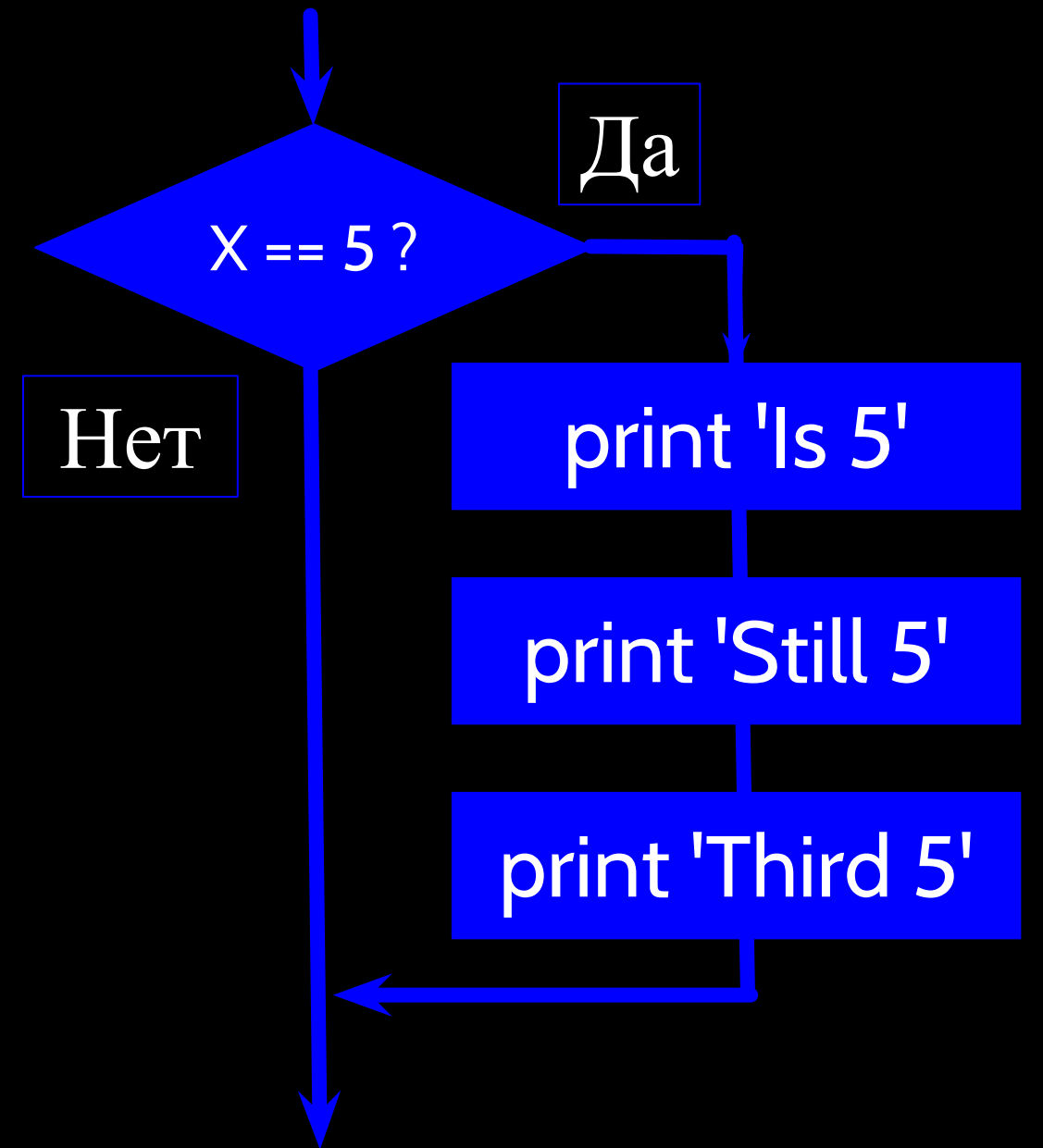
Equals 5
Greater than 4
Greater than or Equals 5
Less than 6
Less than or Equals 5
Not equal 6



Односторонние решения

```
x = 5
print 'Before 5'
if x == 5 :
    print 'Is 5'
    print 'Is Still 5'
    print 'Third 5'
print 'Afterwards 5'
print 'Before 6'
if x == 6 :
    print 'Is 6'
    print 'Is Still 6'
    print 'Third 6'
print 'Afterwards 6'
```

Before 5
Is 5
Is Still 5
Third 5
Afterwards
5
Before 6
Afterwards
6



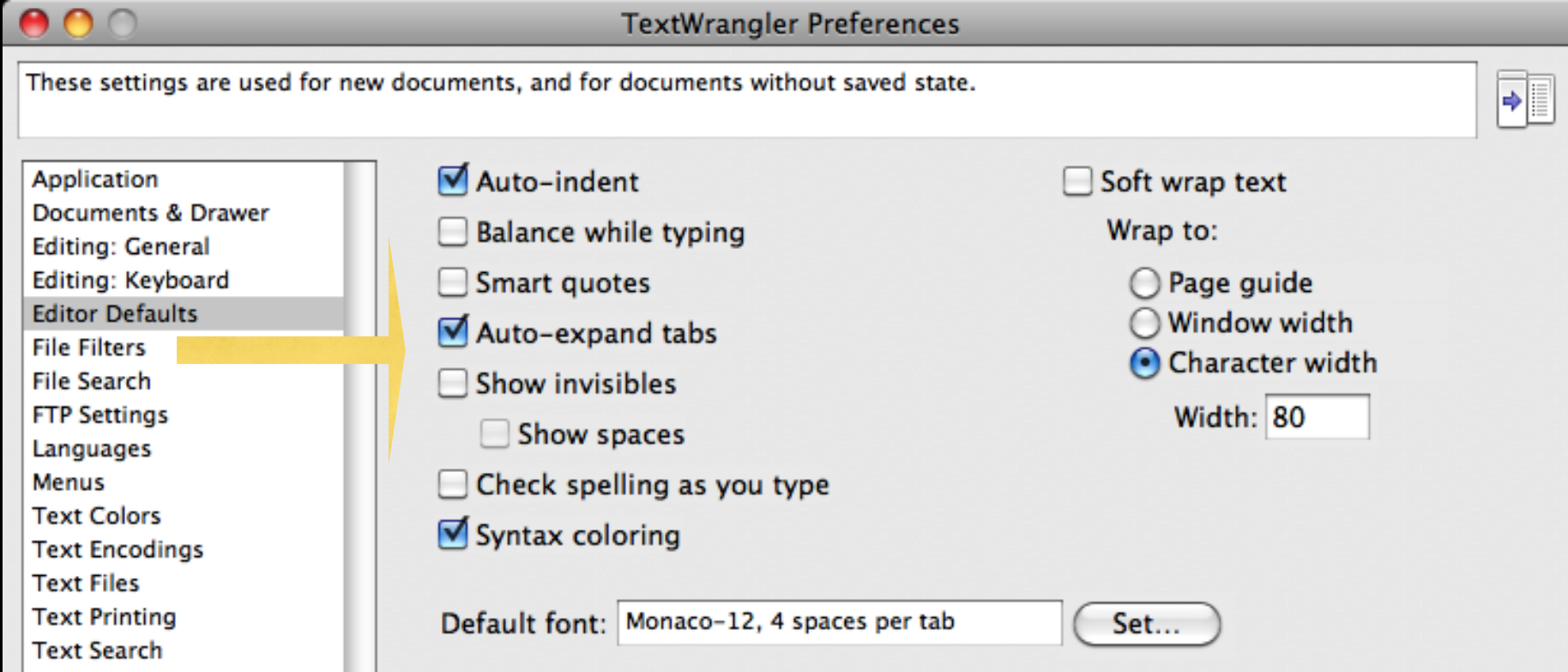
Отступ

- **Увеличение отступа** после инструкции **if** или **for** (после знака “:”)
- **Сохранение отступа** для указания **области** действия блока (строки, относящиеся к инструкциям **if/for**)
- **Уменьшение отступа** для **возврата к** уровню инструкции **if** или **for** для указания окончания действия блока
- **Пустые строки** игнорируются, не оказывая влияния на **отступ**
- **Комментарии** в отдельной строке к **отступу** не относятся

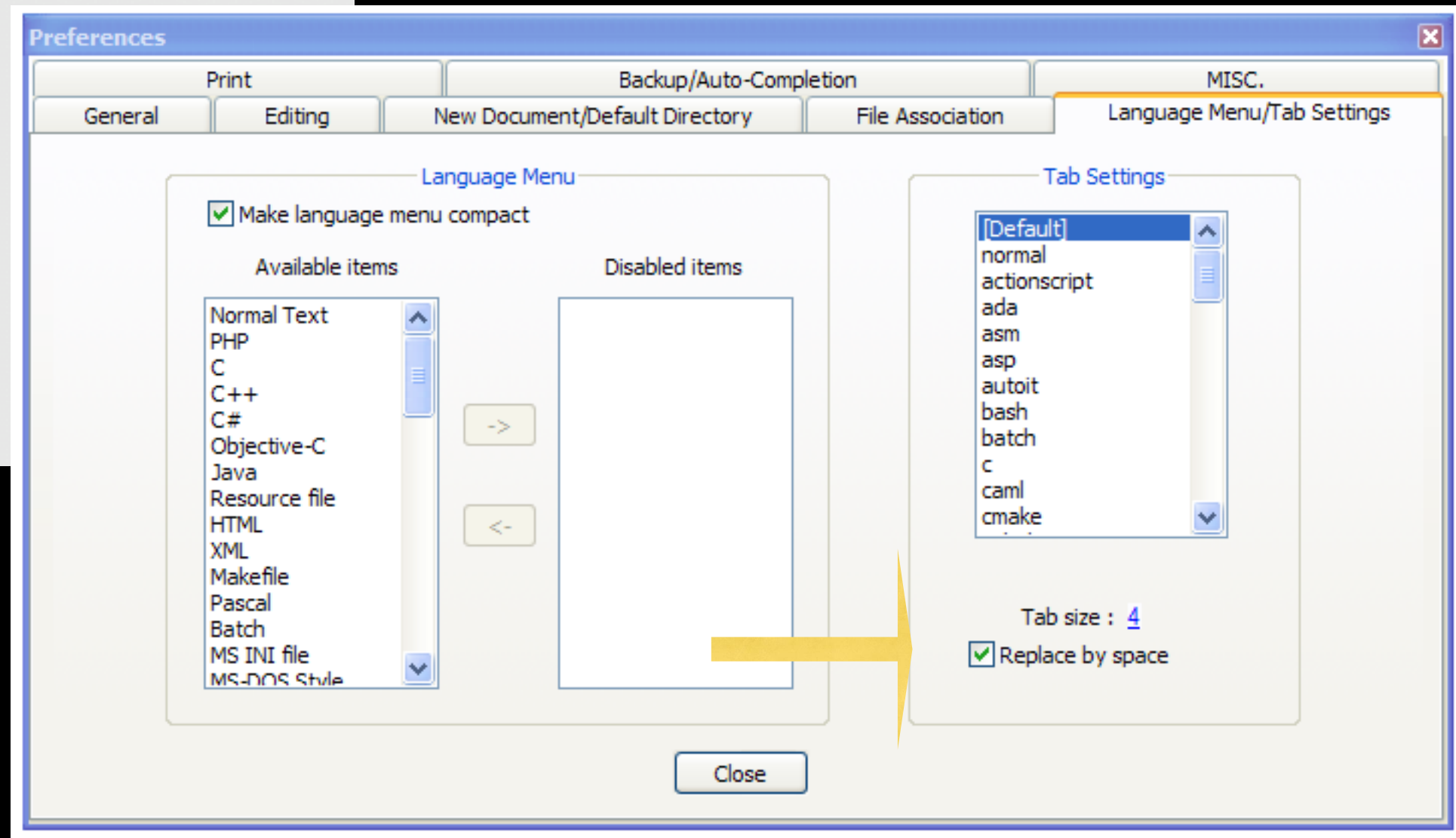
Внимание! Отключите табуляцию!

- Большинство текстовых редакторов могут преобразовать **табуляцию** в **пробелы**. Проверьте, выбрана ли у вас эта опция.
 - › Notepad++: Опции -> Правка -> Синтаксис/**Табуляция**
 - › TextWrangler: TextWrangler -> Preferences -> Editor Defaults
- В Python **отступ** строк имеет *большое* значение. При одновременном использовании **табуляции** и **пробелов** вы можете получить сообщение об ошибке отступа - “**indentation errors**”, даже если на вид все отступы кажутся правильными

А сейчас проверьте настройки табуляции вашего текстового редактора



Это избавит вас от ненужных ошибок



увеличение / сохранение отступа после if
или for

уменьшение отступа для указания
окончания блока

```
→ x = 5
→ if x > 2 :
→     print 'Bigger than 2'
→     print 'Still bigger'
← print 'Done with 2'

→ for i in range(5) :
→     print i
→     if i > 2 :
→         print 'Bigger than 2'
←     print 'Done with i', i
← print 'All Done'
```

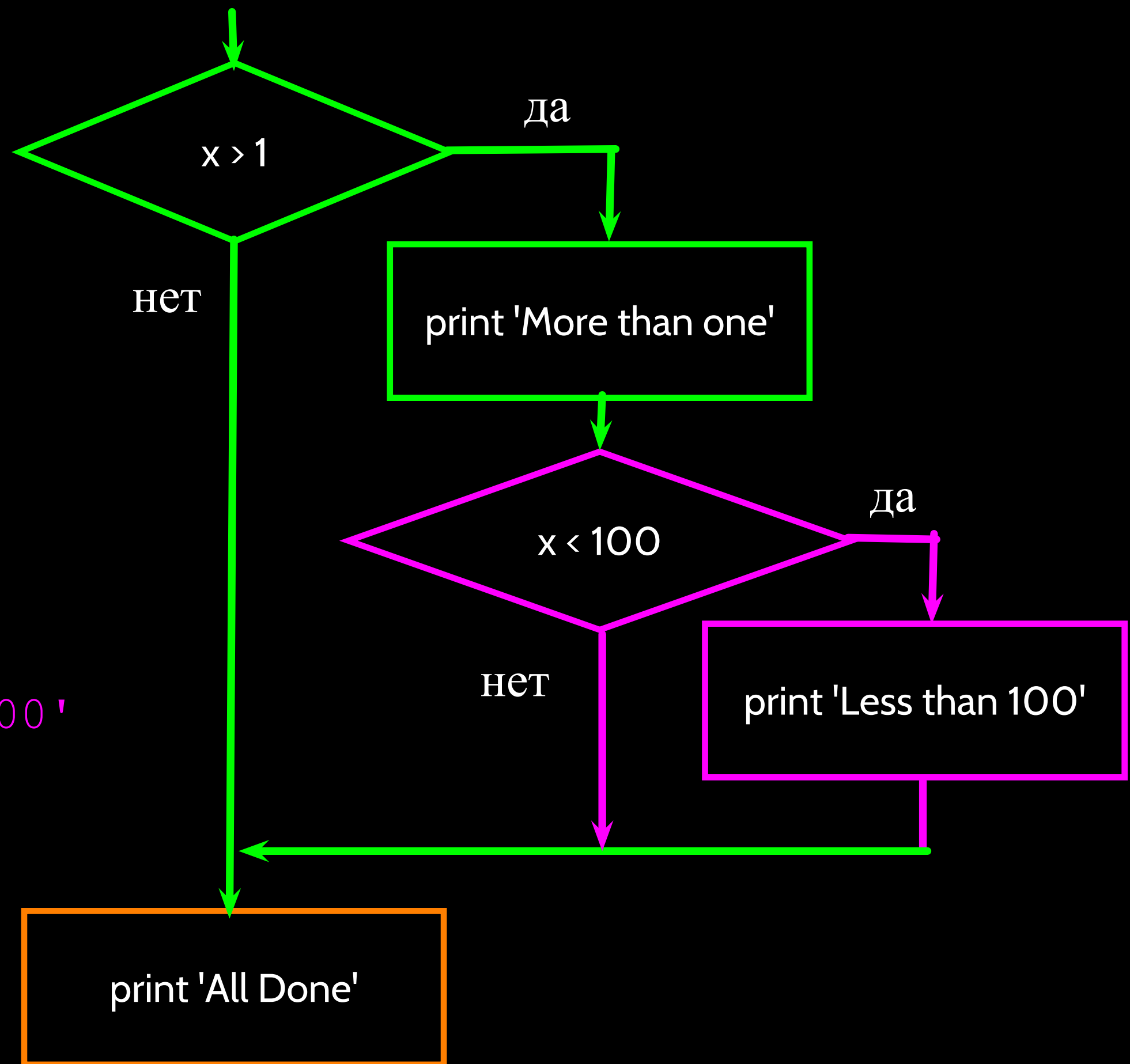
Помните о начале/окончании блоков

```
x = 5
if x > 2 :
    print 'Bigger than 2'
    print 'Still bigger'
print 'Done with 2'
```

```
for i in range(5) :
    print i
    if i > 2 :
        print 'Bigger than 2'
    print 'Done with i', i
print 'All Done'
```

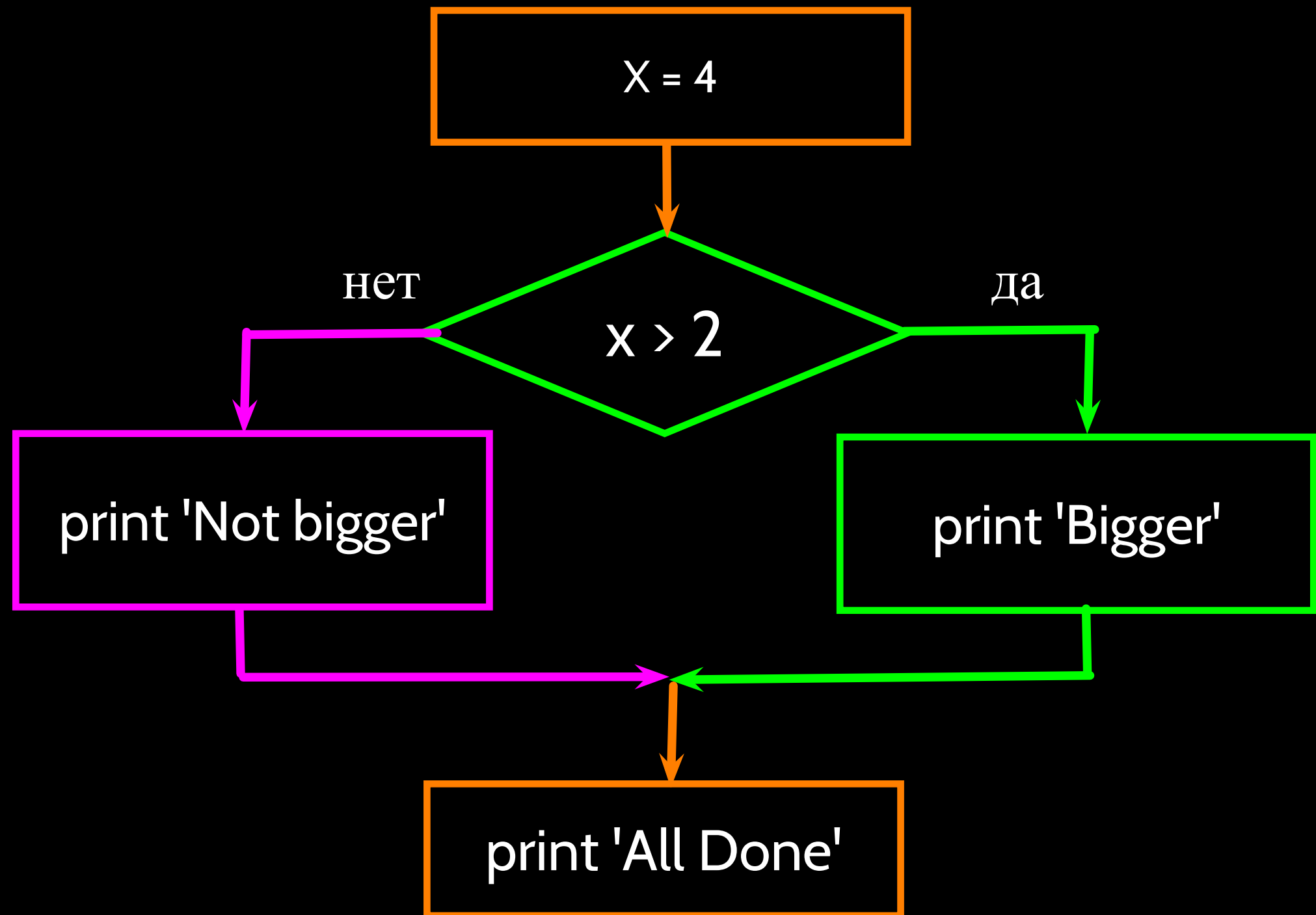
Вложенные решения

```
x = 42
if x > 1 :
    print 'More than one'
    if x < 100 :
        print 'Less than 100'
print 'All done'
```



Двусторонние решения

- Иногда необходимо выполнить одну операцию, если логическое выражение истинно, и другую, если оно ложно
- Это подобно развилке на дороге, когда нужно выбрать **один** из двух путей

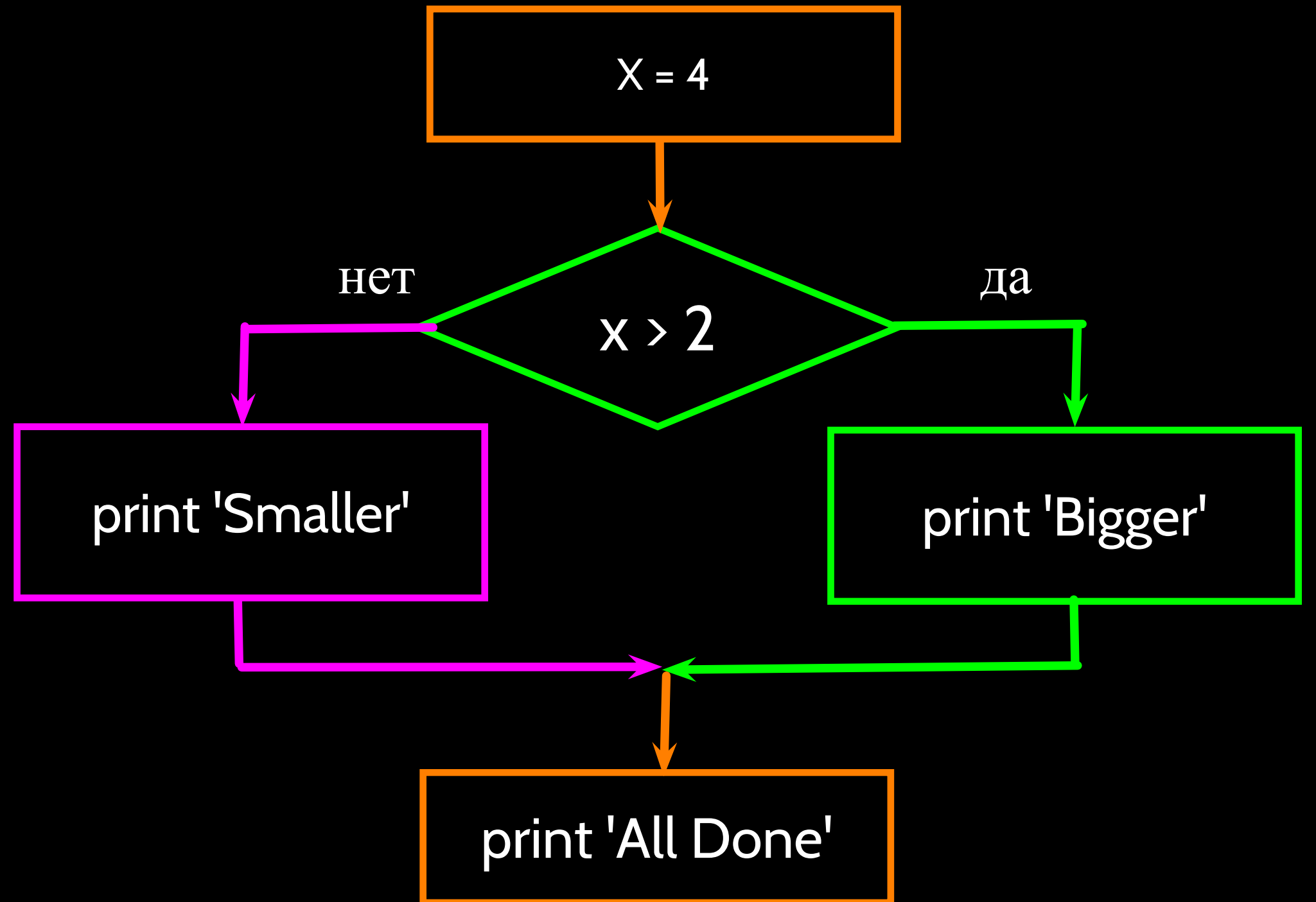


Двустороннее решение с ИСПОЛЬЗОВАНИЕМ else :

x = 4

```
if x > 2 :  
    print 'Bigger'  
else :  
    print 'Smaller'
```

```
print 'All done'
```

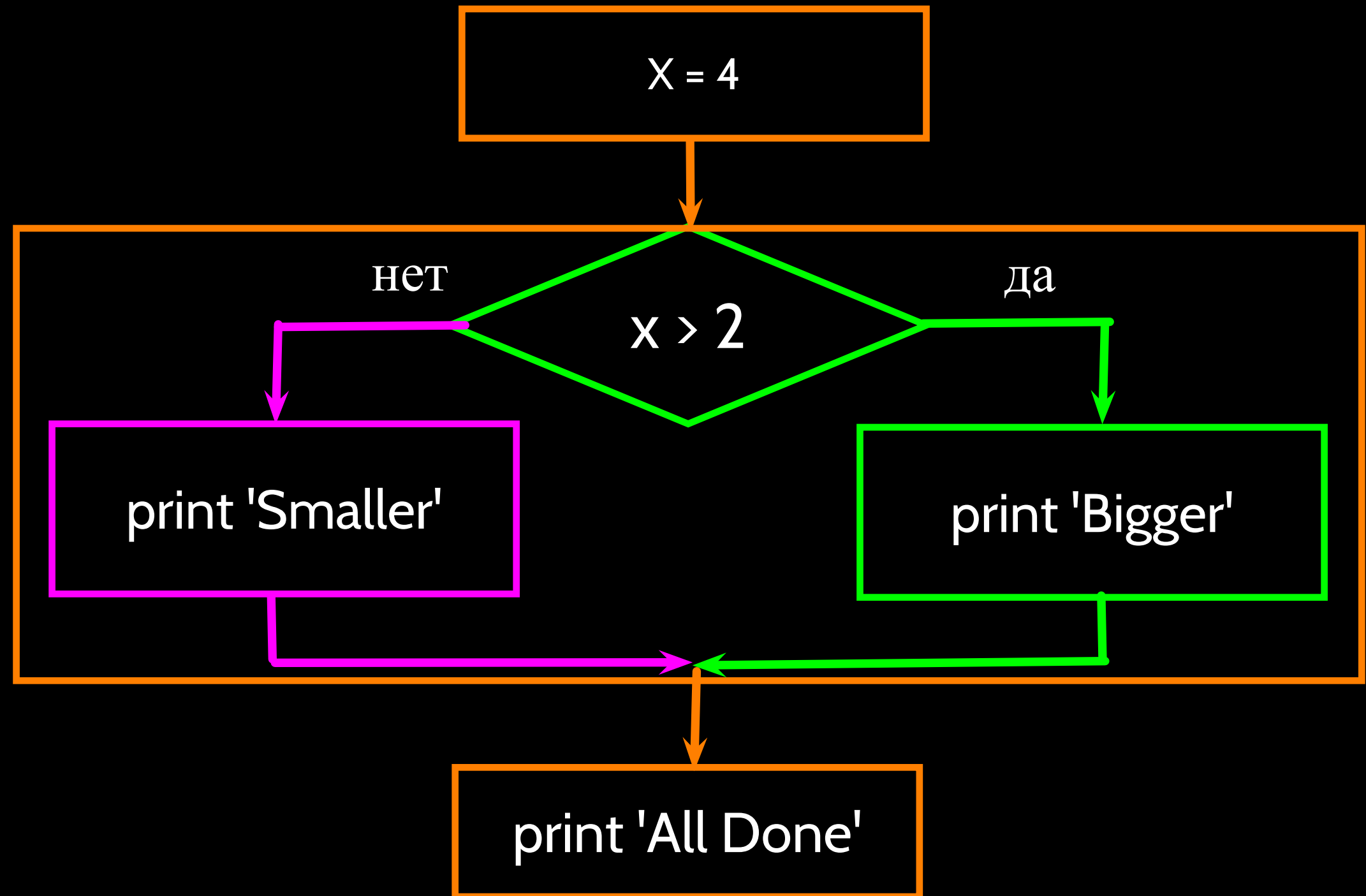


Двустороннее решение с ИСПОЛЬЗОВАНИЕМ else :

x = 4

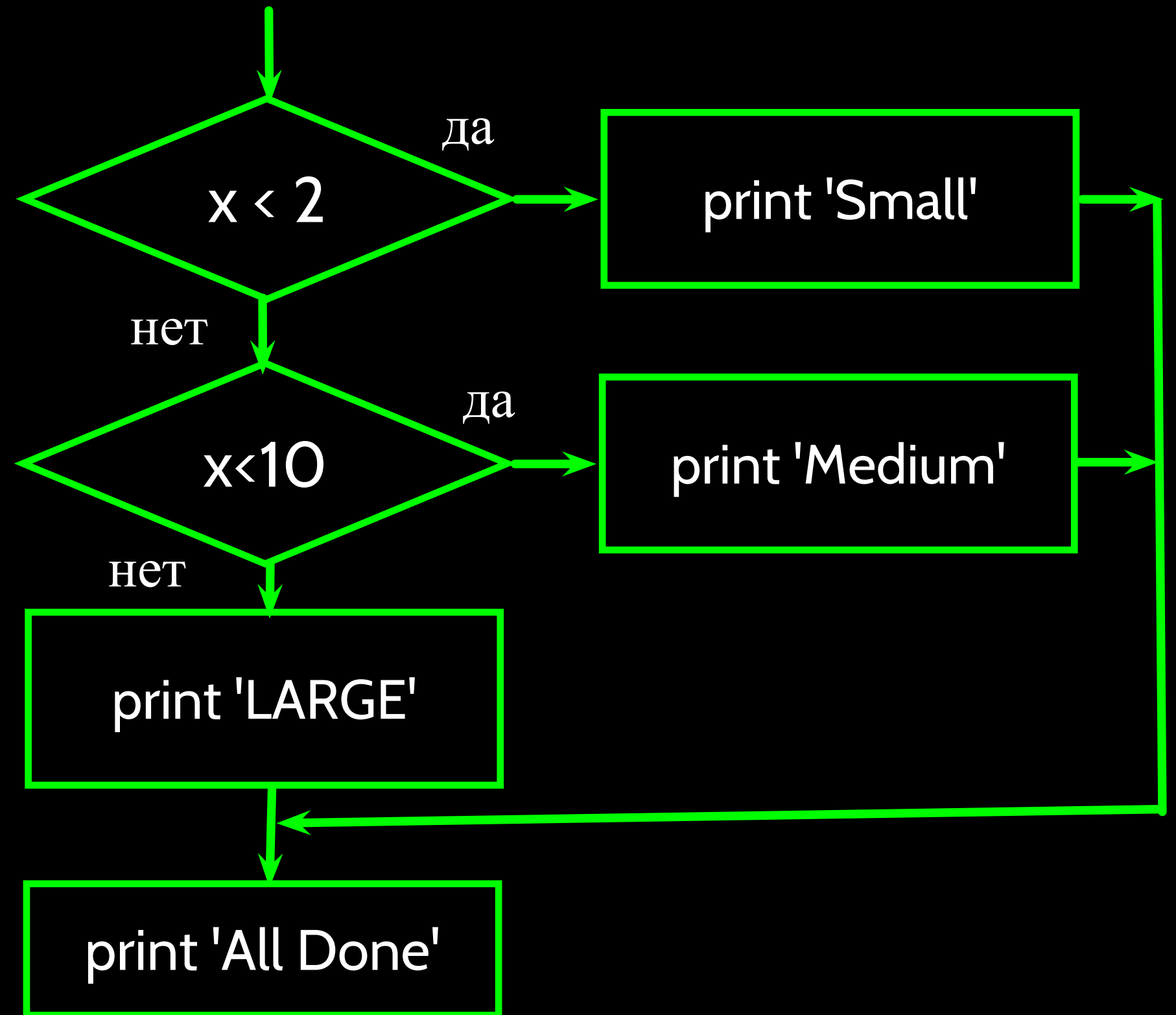
```
if x > 2 :  
    print 'Bigger'  
else :  
    print 'Smaller'
```

```
print 'All done'
```



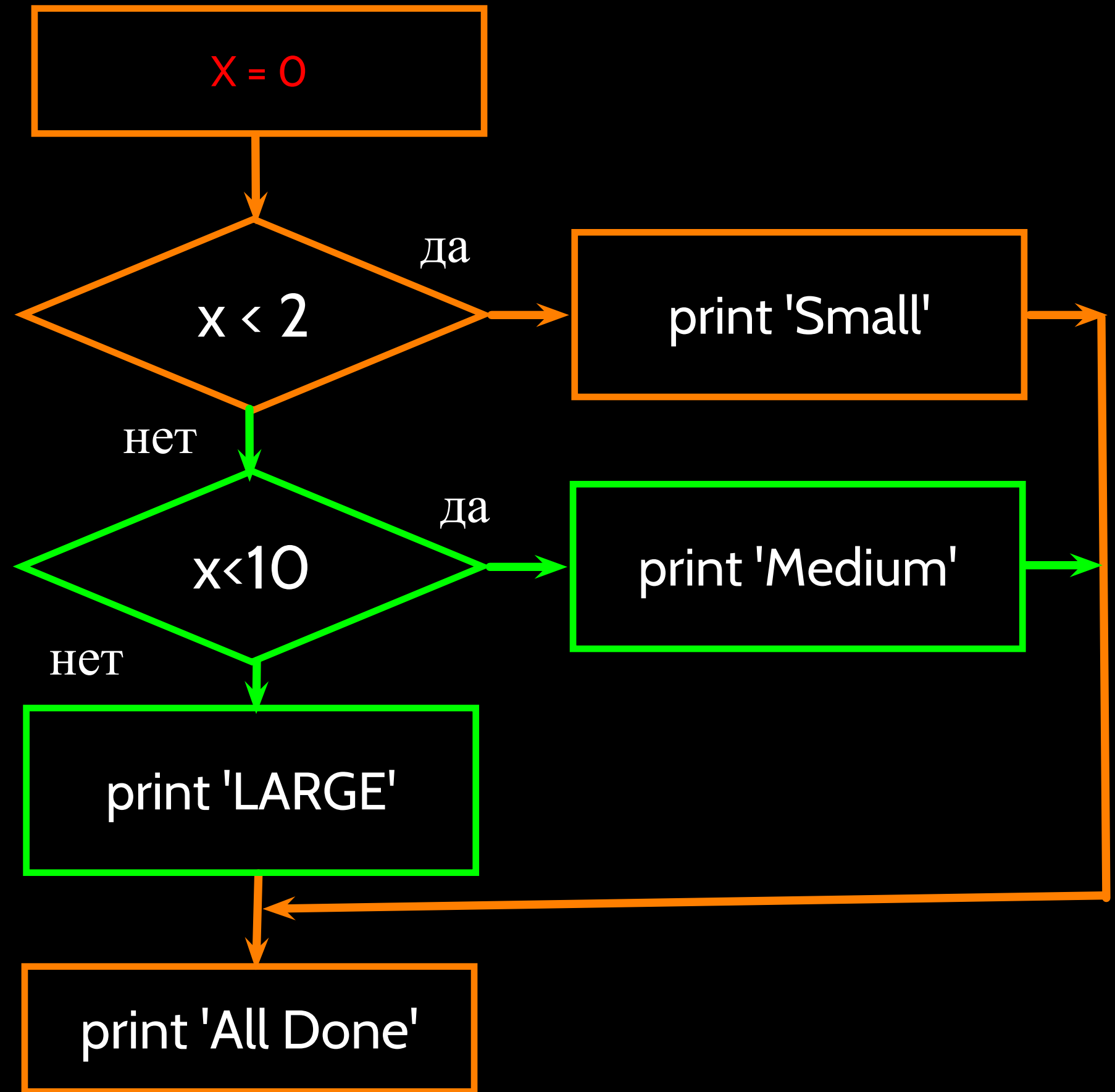
Многостороннее решение

```
if x < 2 :  
    print 'Small'  
elif x < 10 :  
    print 'Medium'  
else :  
    print 'LARGE'  
print 'All done'
```



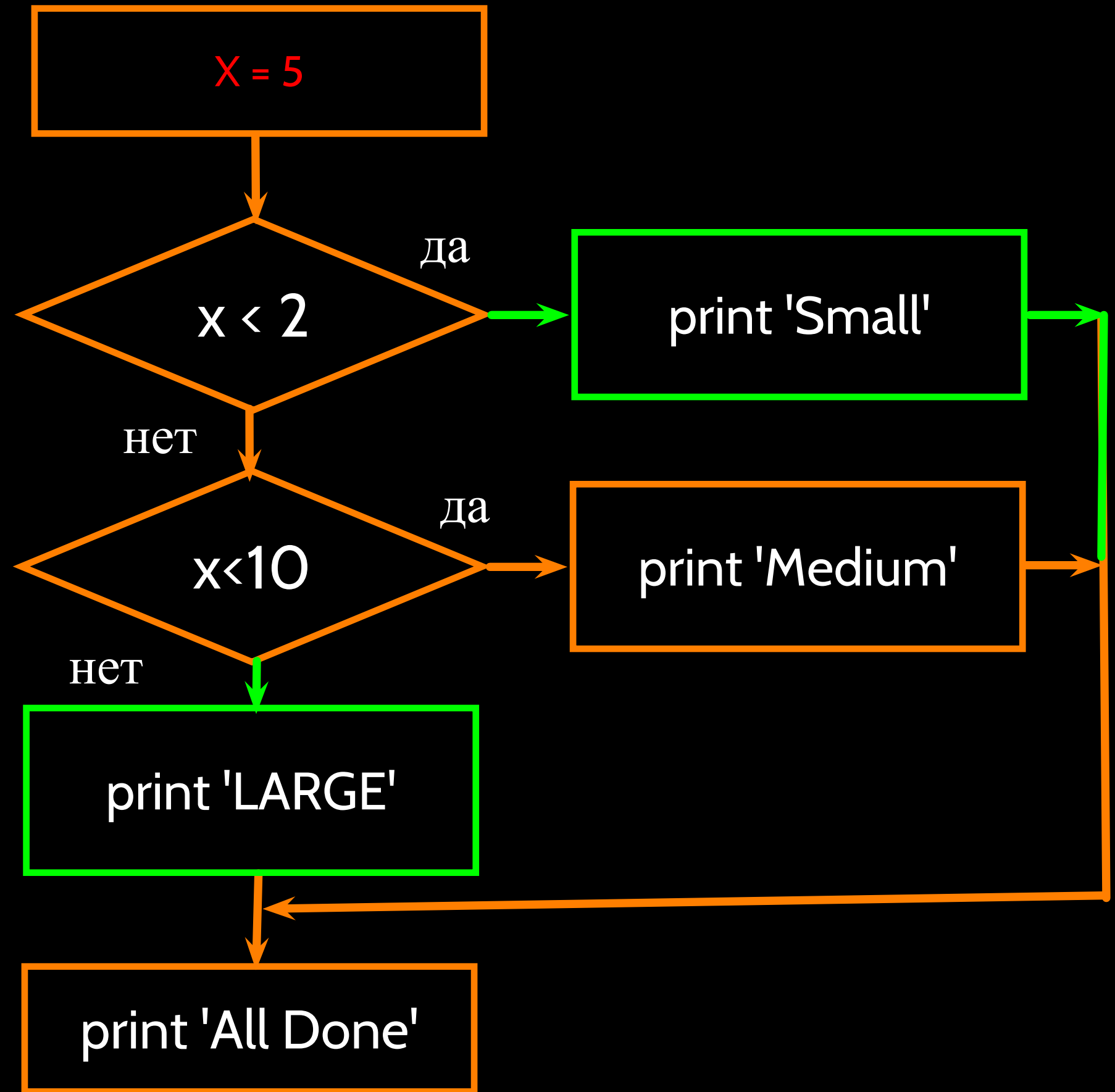
Многостороннее решение

```
x = 0
if x < 2 :
    print 'Small'
elif x < 10 :
    print 'Medium'
else :
    print 'LARGE'
print 'All done'
```



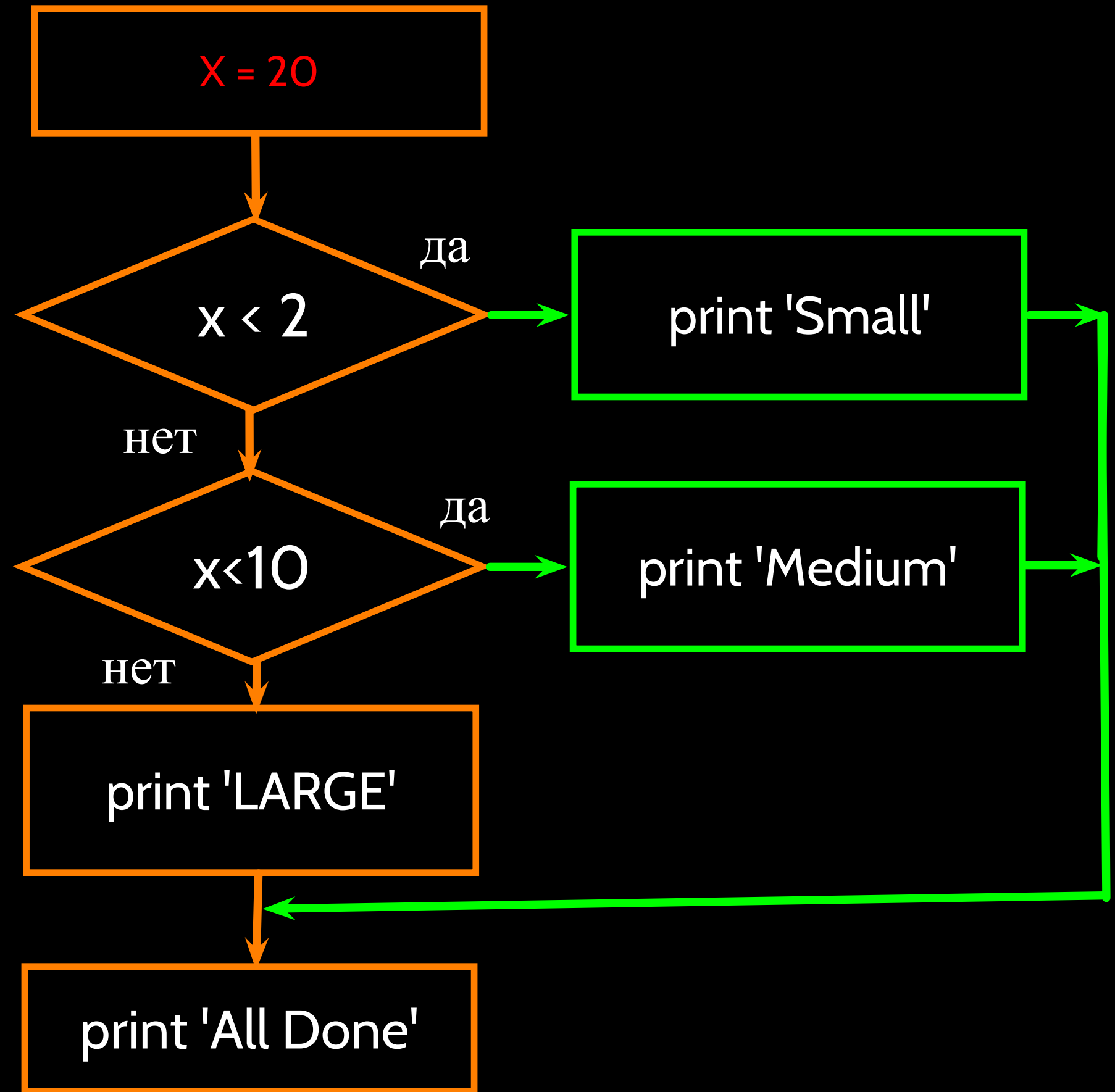
Многостороннее решение

```
x = 5
if x < 2 :
    print 'Small'
elif x < 10 :
    print 'Medium'
else :
    print 'LARGE'
print 'All done'
```



Многостороннее решение

```
x = 20
if x < 2 :
    print 'Small'
elif x < 10 :
    print 'Medium'
else :
    print 'LARGE'
print 'All done'
```



Многостороннее решение

```
# Без else
x = 5
if x < 2 :
    print 'Small'
elif x < 10 :
    print 'Medium'

print 'All done'
```

```
if x < 2 :
    print 'Small'
elif x < 10 :
    print 'Medium'
elif x < 20 :
    print 'Big'
elif x < 40 :
    print 'Large'
elif x < 100:
    print 'Huge'
else :
    print 'Ginormous'
```

Задача

Какая из инструкций никогда не будет выполнена?

```
if x < 2 :  
    print 'Below 2'  
elif x >= 2 :  
    print 'Two or more'  
else :  
    print 'Something  
else'
```

```
if x < 2 :  
    print 'Below 2'  
elif x < 20 :  
    print 'Below 20'  
elif x < 10 :  
    print 'Below 10'  
else :  
    print 'Something else'
```

Структура `try` / `except`

- Опасная часть кода помещается в блок `try` и `except`
- Если код в части `try` успешно выполняется, то часть `except` пропускается
- Если код в части `try` дает сбой, то выполняется часть `except`

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
print 'First', istr
astr = '123'
istr = int(astr)
print 'Second', istr
```

```
$ python notry.py
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```

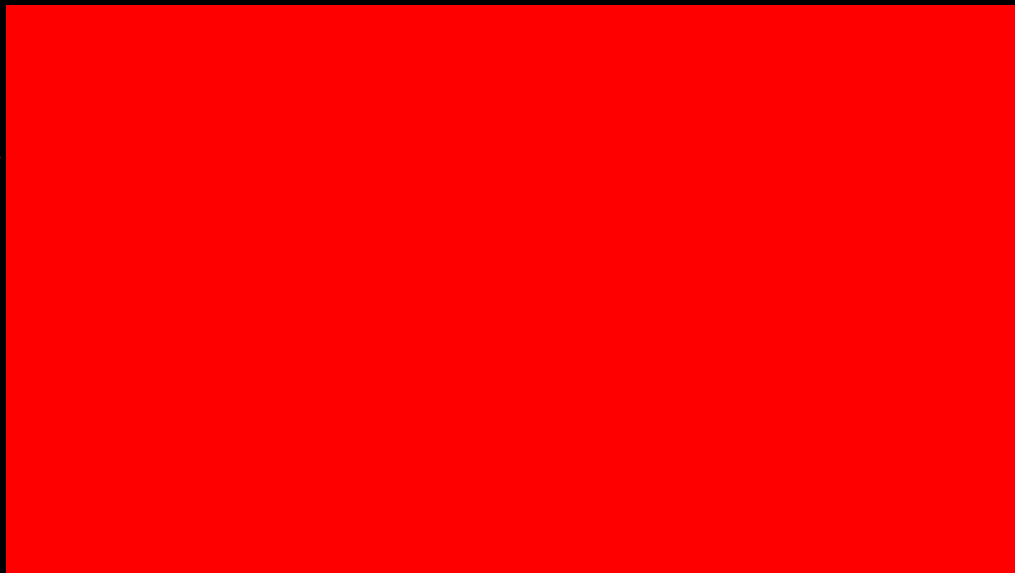


Программа
выполнена

```
$ cat notry.py
astr = 'Hello Bob'
istr = int(astr)
```



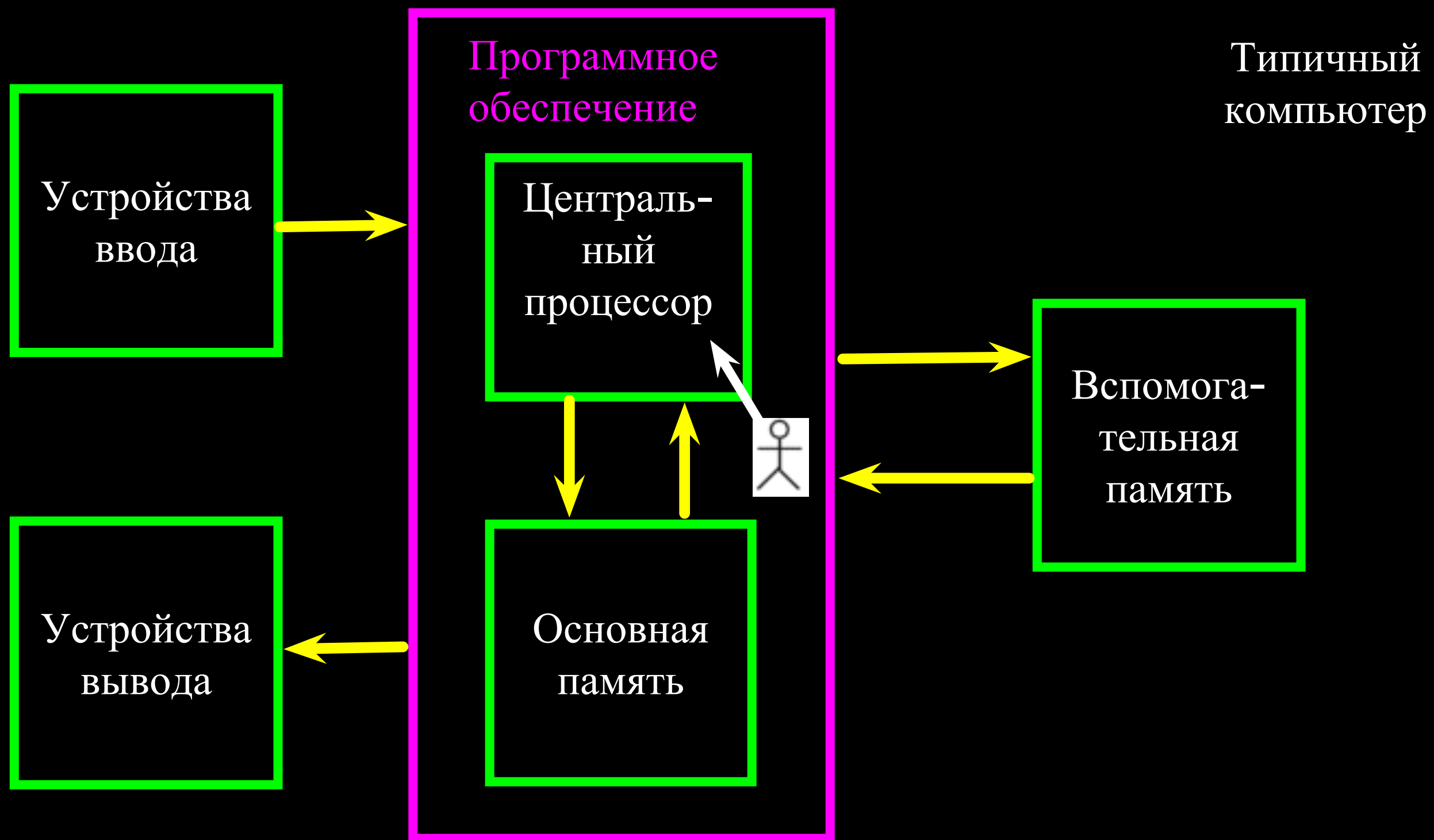
Здесь программа
остановится



```
$ python notry.py
Traceback (most recent call last):
File "notry.py", line 2, in <module>
istr = int(astr)ValueError: invalid literal
for int() with base 10: 'Hello Bob'
```



Программа
выполнена




```
$ cat tryexcept.py
astr = 'Hello Bob'
try:
    istr = int(astr)
except:
    istr = -1

print 'First', istr

astr = '123'
try:
    istr = int(astr)
except:
    istr = -1

print 'Second', istr
```

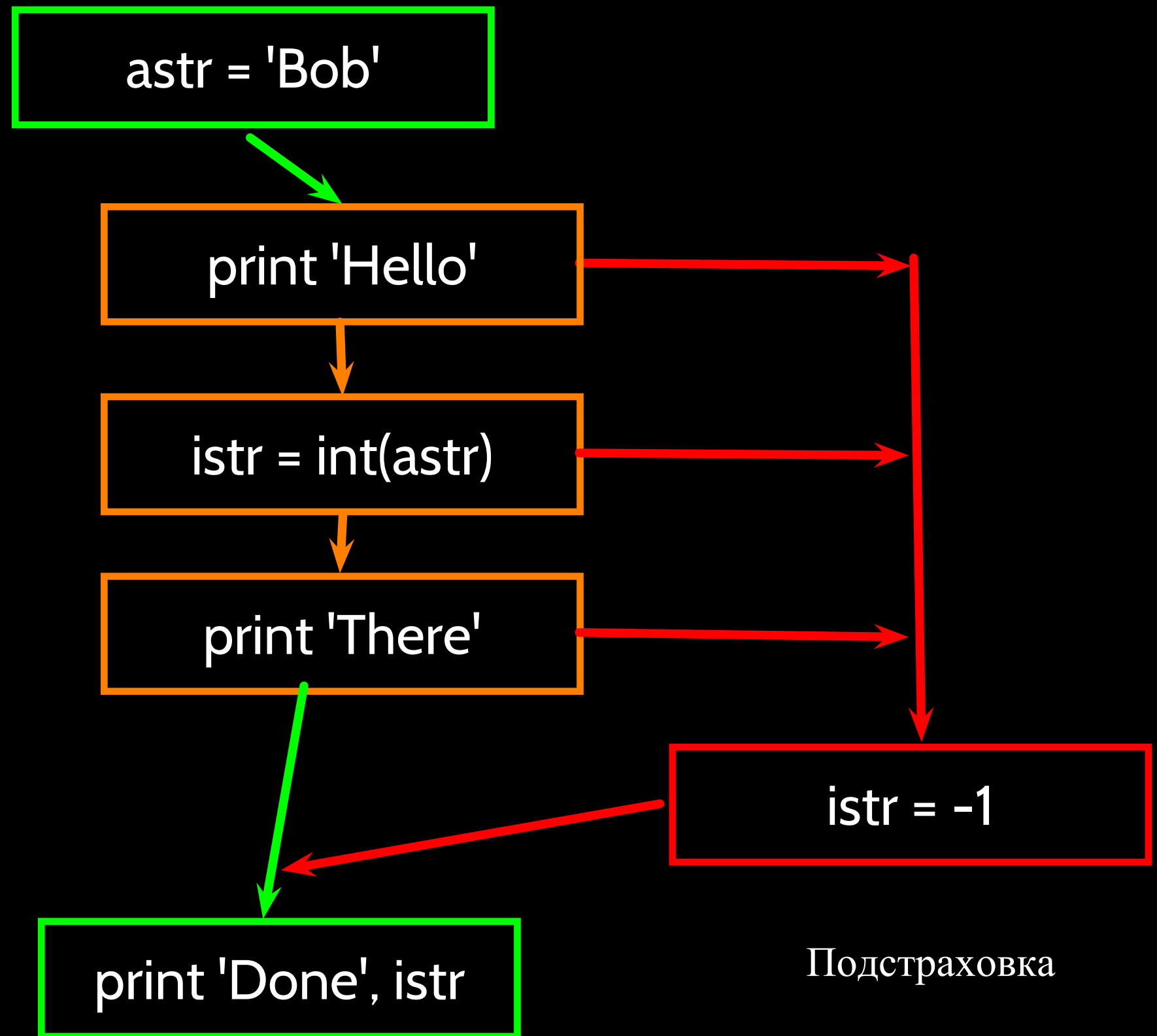
При сбое первого преобразования
программа переходит к
предложению **except** и выполняется
далее

```
$ python tryexcept.py
First -1
Second 123
```

При успешном выполнении второго
преобразования программа
выполняется, пропустив блок
except.

try / except

```
astr = 'Bob'  
try:  
    print 'Hello'  
    istr = int(astr)  
    print 'There'  
except:  
    istr = -1  
print 'Done', istr
```



Пример try / except

```
rawstr = raw_input('Enter a number:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print 'Nice work'
else:
    print 'Not a number'
```

```
$ python trynum.py
Enter a number:42
Nice work
$ python trynum.py
Enter a number:forty-two
Not a number
$
```

Упражнение

Измените код расчета заработной платы с учетом того, что ставка за сверхурочные часы в полтора раза выше обычной ставки.

Введите часы: 45

Введите ставку: 10

Зарплата: 475.0

$$475 = 40 * 10 + 5 * 15$$

Упражнение

Измените код расчета заработной платы с использованием конструкции `try ... except` на случай, если пользователь введет нечисловые значения.

Введите часы: 20

Введите ставку: девять

Ошибка! Введите числовые значения!

Введите часы: сорок

Ошибка! Введите числовые значения!

Обзор

- Операторы сравнения
`== <= >= > < !=`
- Логические операторы: `and` или `not`
- Отступ
- Односторонние решения
- `Ехсерт` для обработки ошибок
- Двусторонние решения
`if` : и `else` :
- Вложенные решения
- Многосторонние решения с использованием `elif`
- `Try / Ехсерт` для обработки ошибок



Благодарность / Содействие



Данная презентация охраняется авторским правом “Copyright 2010- Charles R. Severance (www.dr-chuck.com) University of Michigan School of Information” open.umich.edu и доступна на условиях лицензии 4.0 “С указанием авторства”. В соответствии с требованием лицензии “С указанием авторства” данный слайд должен присутствовать во всех копиях этого документа. При внесении каких-либо изменений в данный документ вы можете указать свое имя и организацию в список соавторов на этой странице для последующих публикаций.

Первоначальная разработка: Чарльз Северанс, Школа информации Мичиганского университета

Здесь впишите дополнительных авторов...