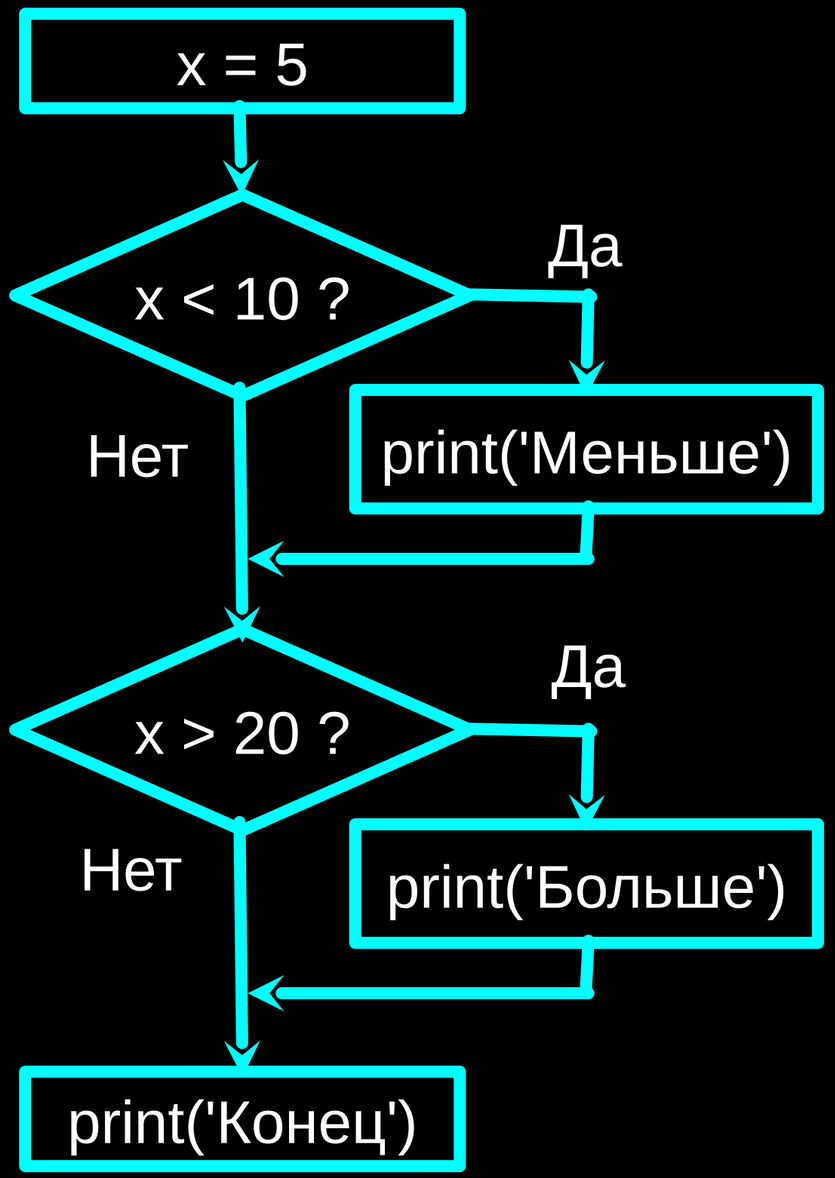
Обработка условий

Глава 3

Пайтон для всех

www.py4e.com



Условные шаги

Программа:

```
x = 5
if x < 10:
    print('Меньше')
if x > 20:
    print('Больше')

print('Конец')
```

Результат:

```
Меньше
Конец
```

Операторы сравнения

- Булево / логическое выражение — выражение, результатом вычисления которого является либо «Да», либо «Нет». Этот результат мы затем используем для управления ходом программы

- Булевы выражения используют операторы сравнения, чтобы вычислить результат: «Истина» / «Ложь» или «Да» / «Нет»

- Операторы сравнения оценивают значения переменных, но не меняют их

Пайтон	Значение
<	Меньше
<=	Меньше или равно
==	Равно
>=	Больше или равно
>	Больше
!=	Не равно

Запомните: "=” — знак присвоения

https://ru.wikipedia.org/wiki/Буль,_Джордж

Операторы сравнения

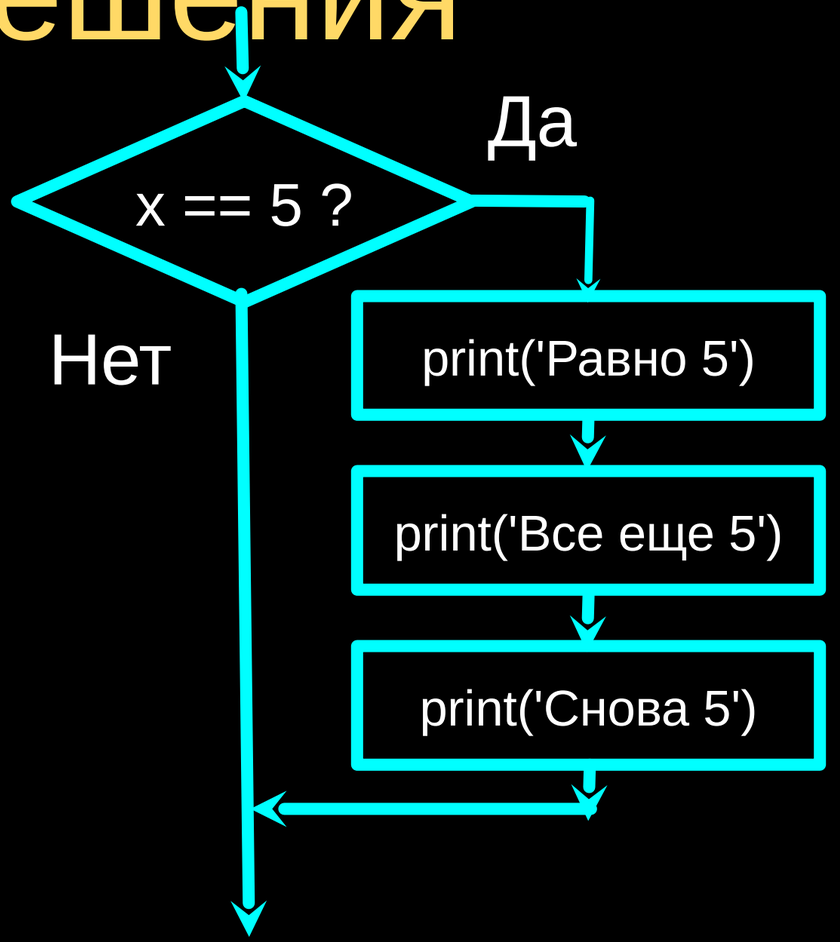
```
x = 5
if x == 5 :
    print('Равно 5')
if x > 4 :
   print('Больше, чем 4')
if  x >= 5 :
    print('Больше или равно 5')
if x < 6 : print('Меньше, чем 6')
if x <= 5 :
    print('Меньше или равно 5')
if x != 6 :
    print('Не равно 6')
```

```
Равно 5
Больше, чем 4
Больше или равно 5
Меньше, чем 6
Меньше или равно 5
Не равно 6
```

Односторонние решения

```
x = 5
print('Перед 5')
if  x == 5 :
    print('Равно 5')
    print('Все еще 5')
    print('Снова 5')
print('Потом 5')
print('Перед 6')
if x == 6 :
    print('Равно 6')
    print('Все еще  6')
    print('Снова 6')
print('Потом 6')
```

```
Перед 5
Равно 5
Все еще 5
Снова 5
Потом 5
Перед 6
Потом 6
```

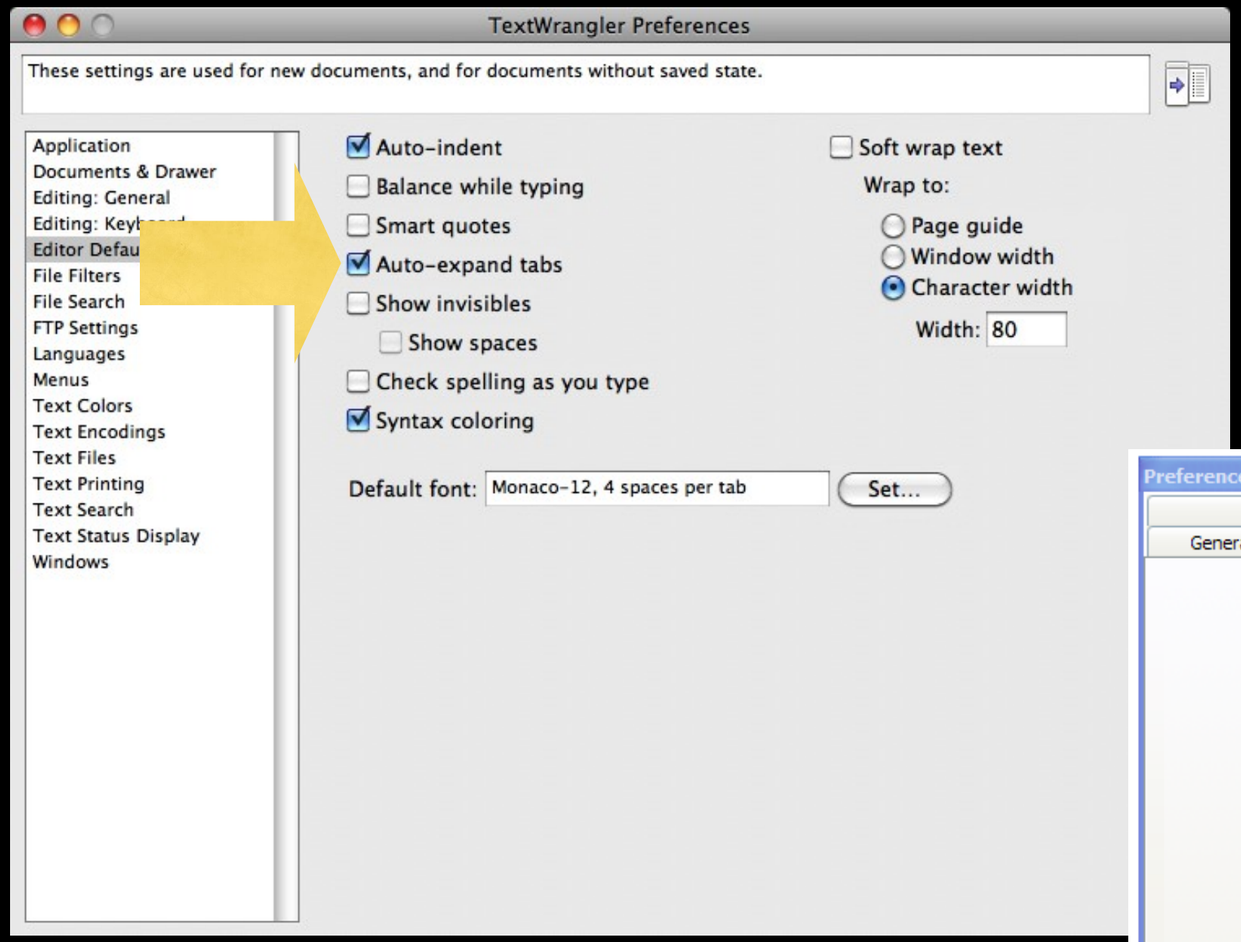


Отступы

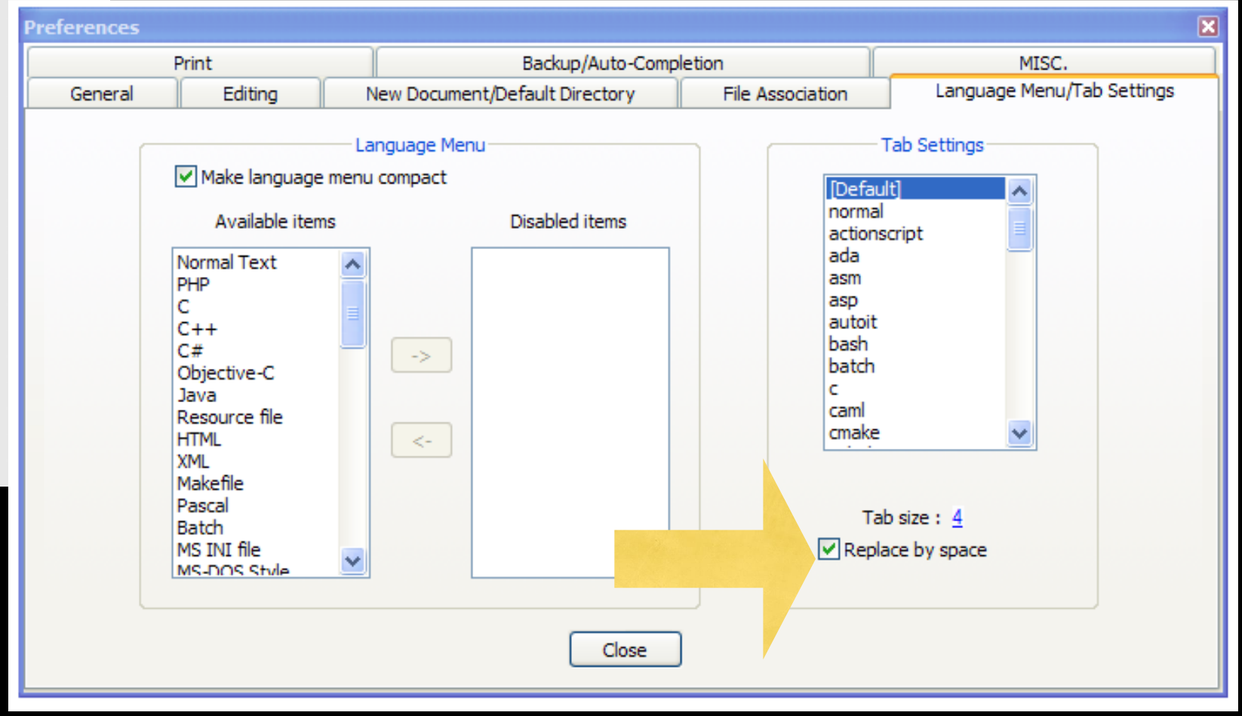
- Увеличивайте отступ после операторов if или for (после символа " : ")
- Сохраняйте отступ, чтобы обозначить блок кода (строки, на которые влияют операторы if/for)
- Сократите отступ, чтобы вернуться на уровень оператора if или for, и обозначить конец данного блока кода
- Пустые строки игнорируются и никак не влияют на отступы
- Комментарии к строке сами по себе игнорируются с учетом отступа

Внимание: Не используйте клавишу Tab!

- Редактор Atom автоматически использует пробелы для файлов с расширением ".py" (Приятно!)
- Большинство текстовых редакторов могут преобразовывать таб-отступы в пробелы. Обязательно включите эту функцию!
  - NotePad++: Настройки -> Предпочтения -> Языковое меню/Настройка Отступов
  - TextWrangler: TextWrangler -> Настройки -> Настройки редактора по умолчанию
- Пайтон !Очень чувствителен к отступам. Одновременное использование таб-отступов и пробелов, может привести к «ошибкам отступов». Причем визуально все может выглядеть правильно

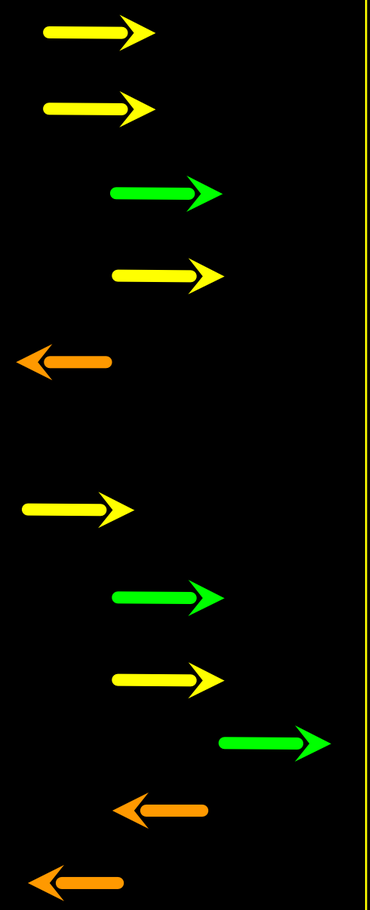


Это сбережет вам много нервов



Увеличивайте / поддерживайте отступ после if и for.

Сокращайте отступ, чтобы обозначить конец блока.



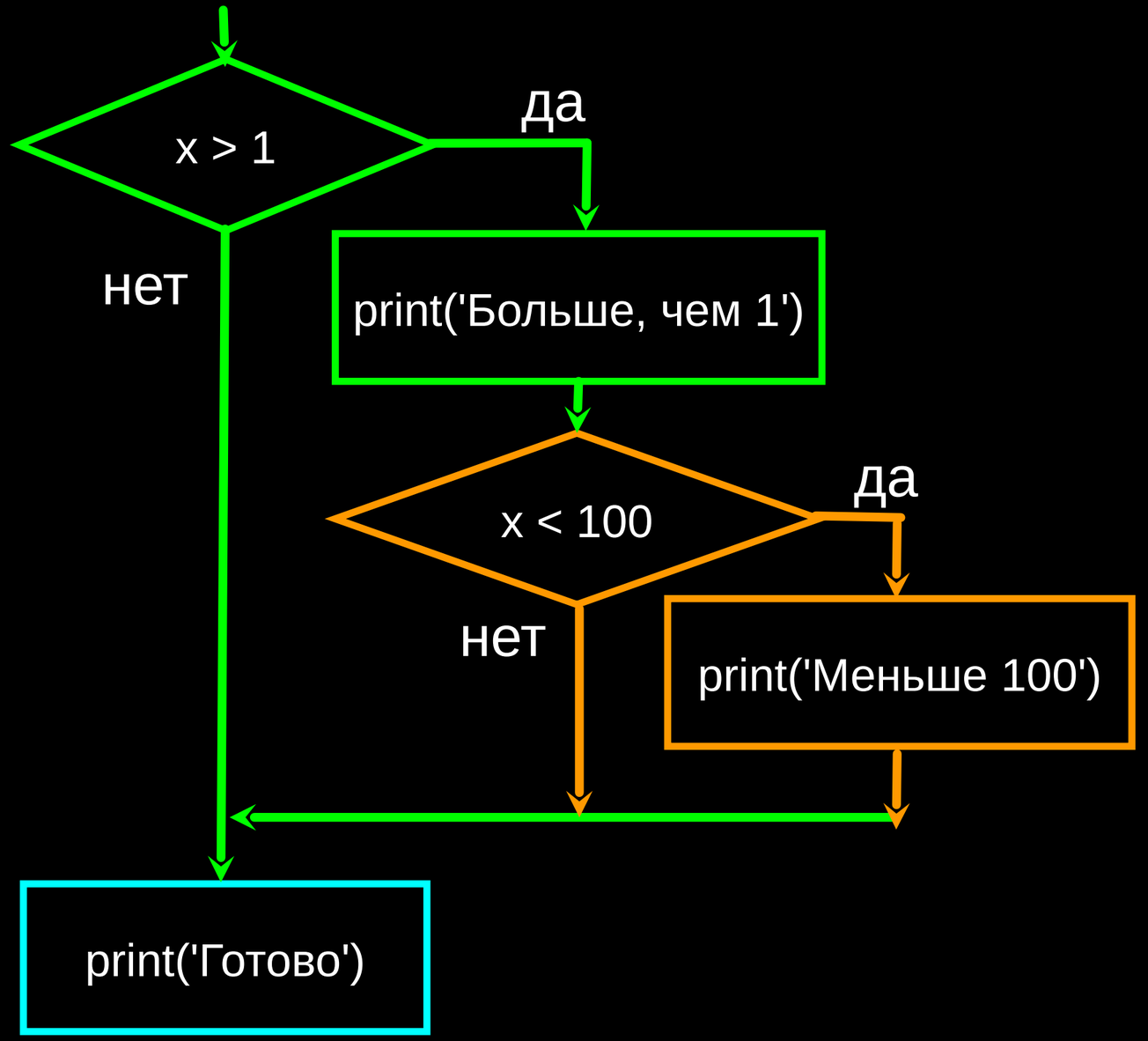
```
x = 5
if x > 2 :
    print('Больше, чем 2')
    print('Все еще больше')
print('Закончил с 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Больше, чем 2')
    print('Закончил с i', i)
print('Готово')
```

Помните о начале/конце блока

```
x = 5
if x > 2 :
    print('Больше, чем 2')
    print('Все еще больше')
print('Закончил с 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Больше, чем 2')
    print('Закончил с i', i)
print('Готово')
```

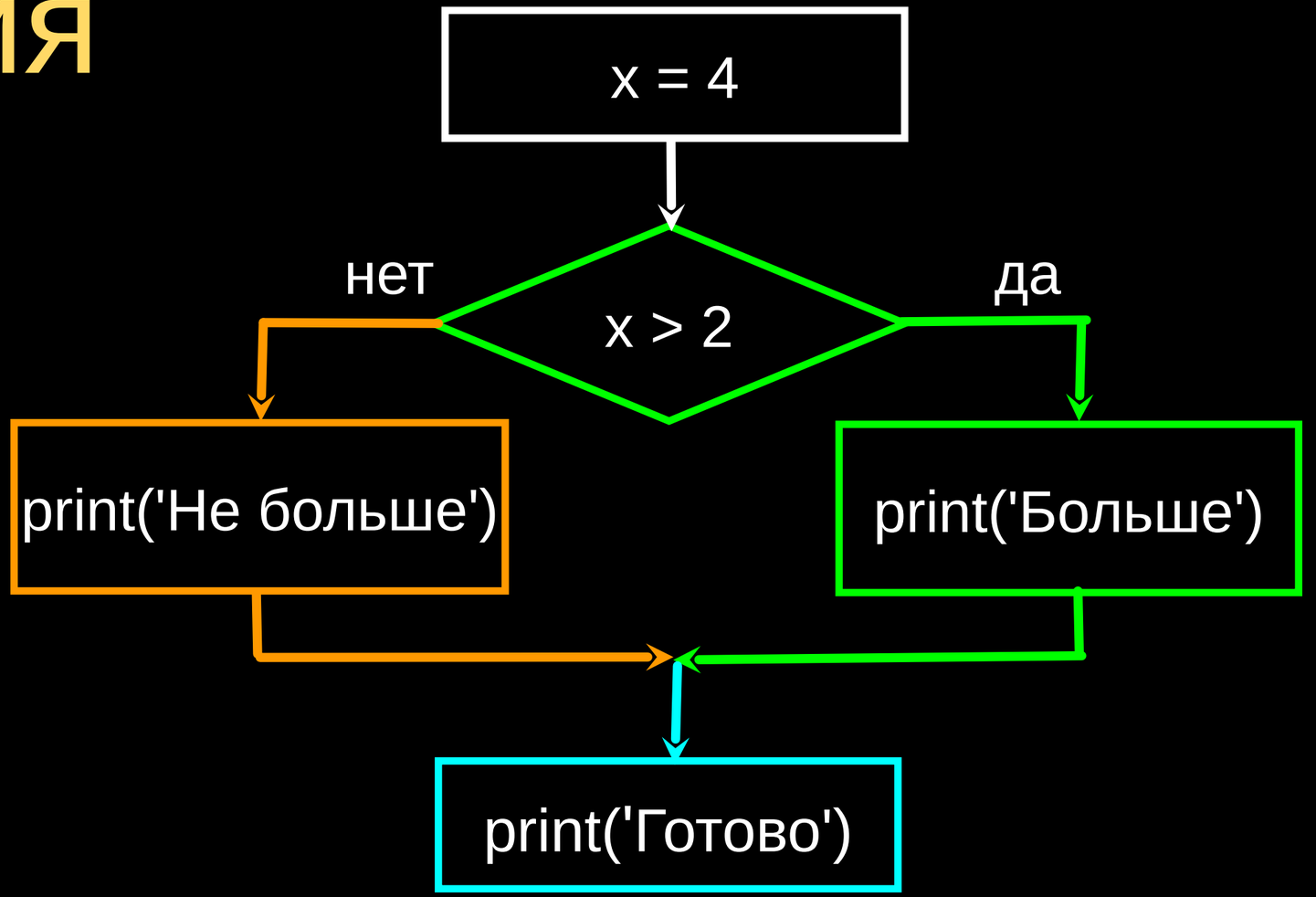
Вложенные решения

```
x = 42
if x > 1 :
    print('Больше, чем 1')
    if x < 100 :
        print('Меньше 100')
print('Готово')
```



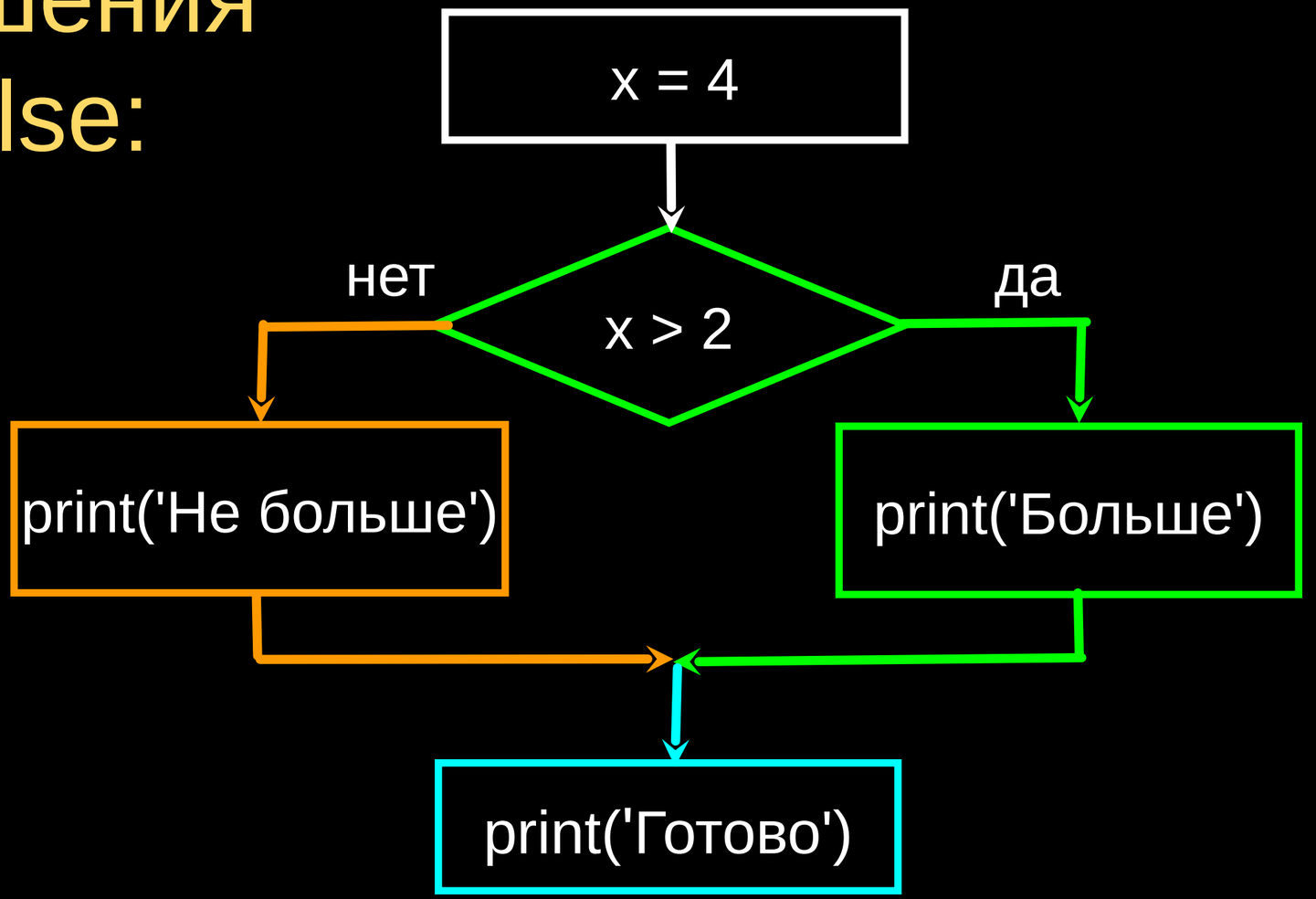
Двусторонние решения

- Иногда мы хотим сделать одну вещь, если логическое выражение правдиво, и что-то другое, если оно ложно
- Как на развилке дорог, нужно выбрать один из путей, но не оба (не все разом)



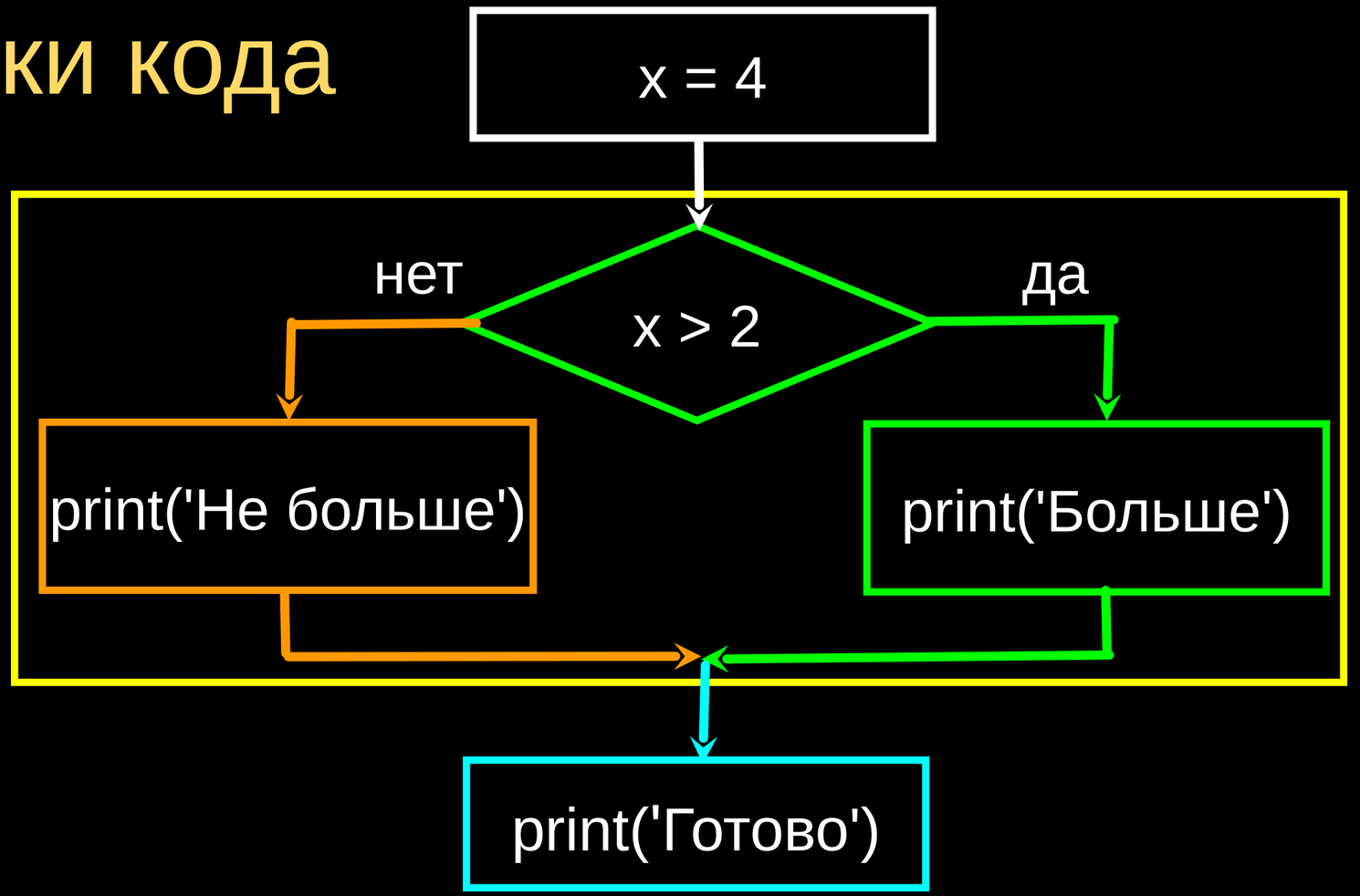
Двусторонние решения с оператором else:

```
x = 4

if x > 2 :
    print('Больше')
else :
    print('Меньше')

print('Готово')
```



Старайтесь визуально представить блоки кода

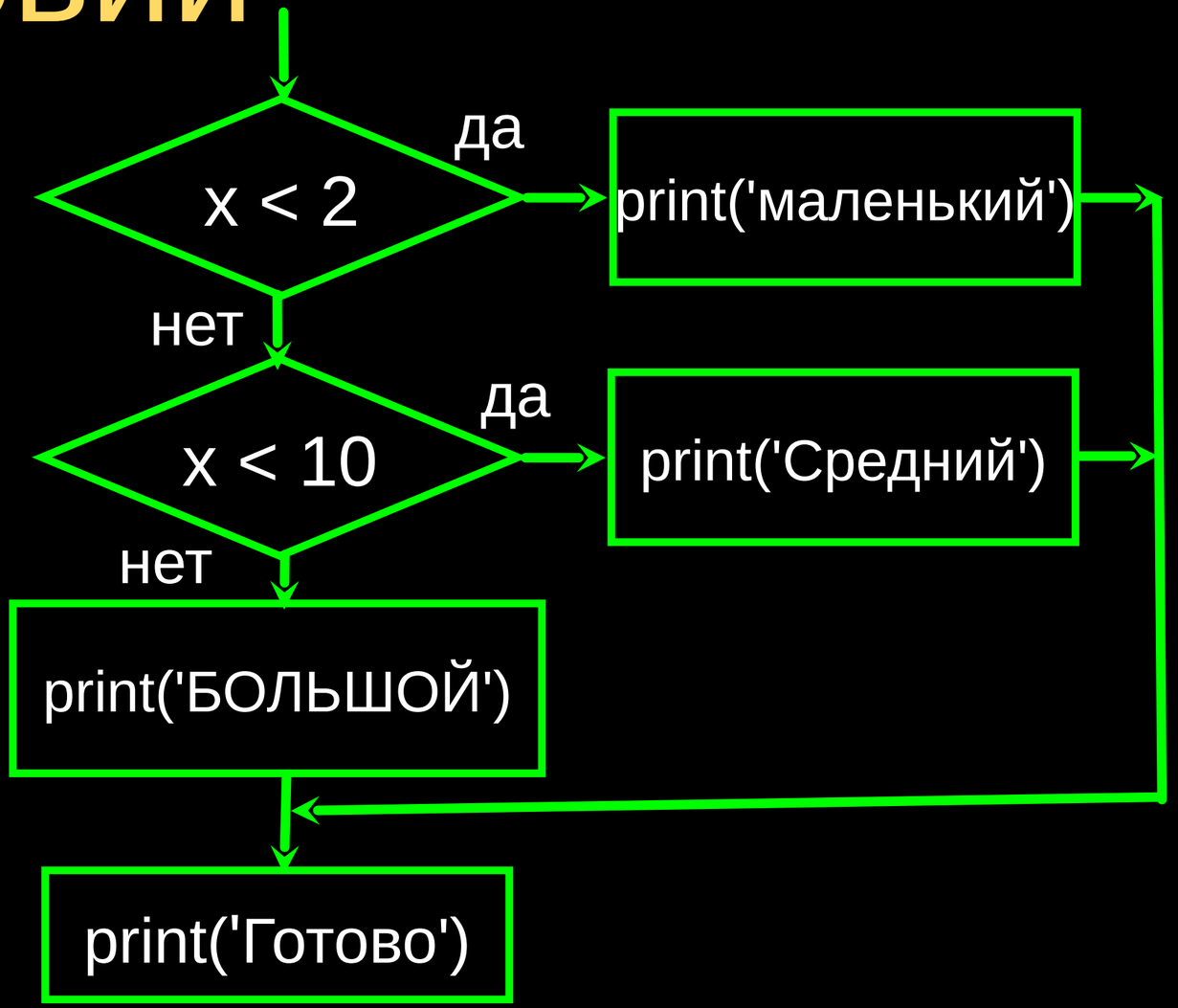
```
x = 4

if x > 2 :
    print('Больше')
else :
    print('Меньше')

print('Готово')
```



Другие условные структуры…

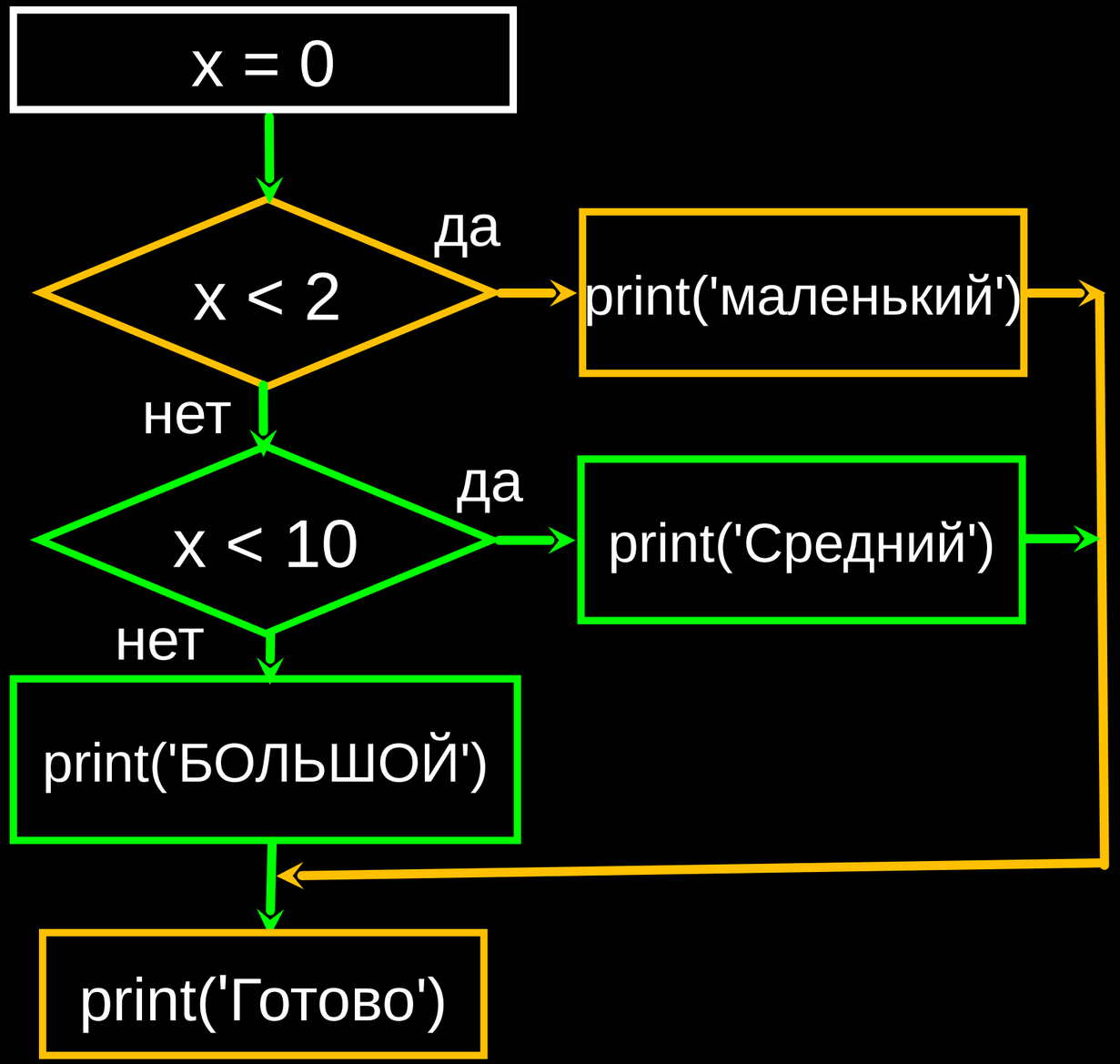
Несколько условий

```
if x < 2 :
    print('маленький')
elif x < 10 :
    print('Средний')
else :
    print('БОЛЬШОЙ')
print('Готово')
```



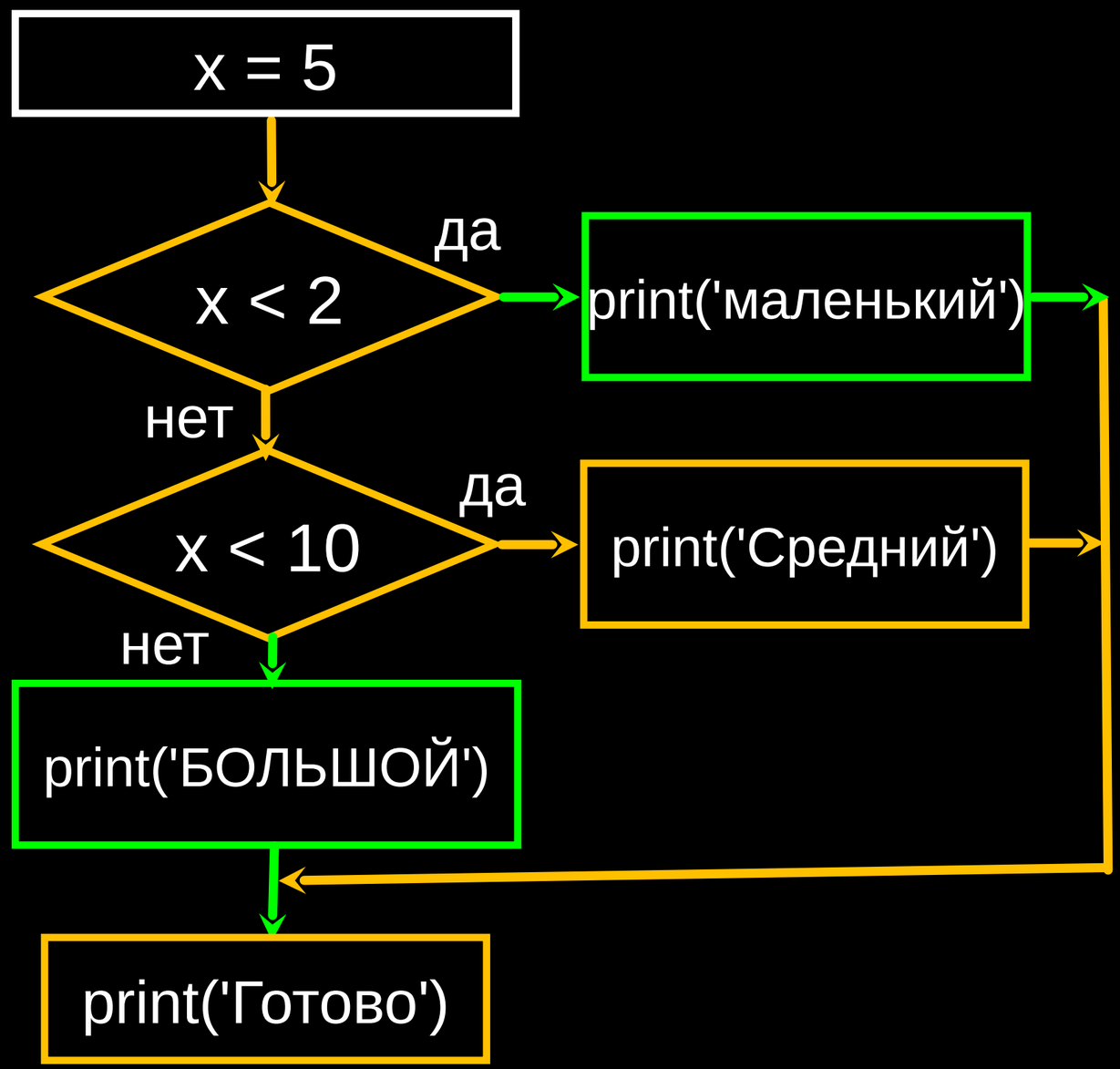
Несколько условий

```
x = 0
if x < 2 :
    print('маленький')
elif x < 10 :
    print('Средний')
else :
    print('БОЛЬШОЙ')
print('Готово')
```



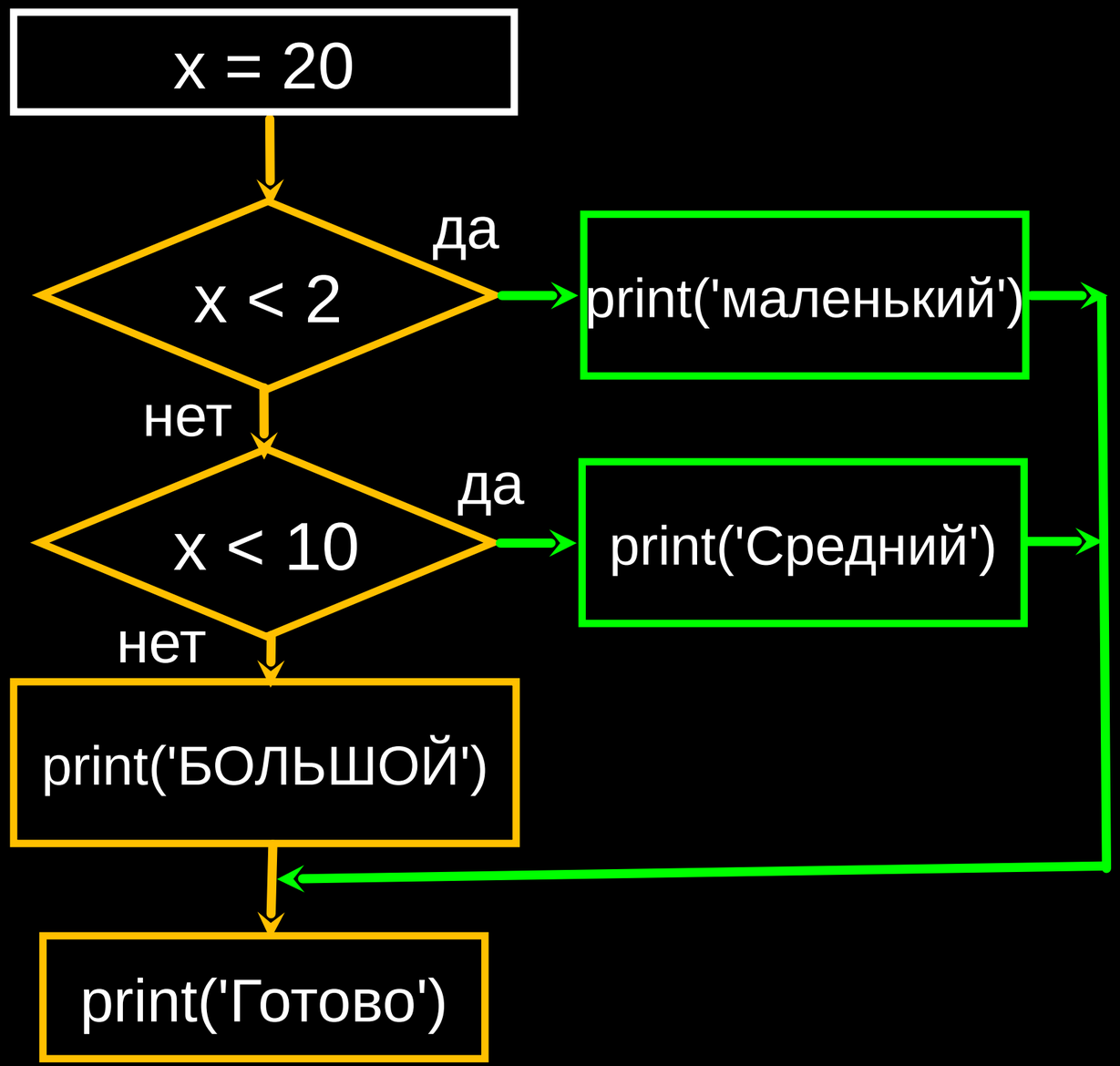
Несколько условий

```
x = 5
if x < 2 :
    print('маленький')
elif x < 10 :
    print('Средний')
else :
    print('БОЛЬШОЙ')
print('Готово')
```



Несколько условий

```
x = 20
if x < 2 :
    print('маленький')
elif x < 10 :
    print('Средний')
else :
    print('БОЛЬШОЙ')
print('Готово')
```



Несколько условий

```
# Без оператора Else
x = 5
if x < 2 :
    print('Маленький')
elif x < 10 :
    print('Средний')

print('Готово')
```

```
if x < 2 :
    print('Маленький')
elif x < 10 :
    print('Средний')
elif x < 20 :
    print('Большой')
elif x < 40 :
    print('Внушительный')
elif x < 100:
    print('Огромный')
else :
    print('Гигантский')
```

Запутанность множества условий

Что никогда не будет выведено
вне зависимости от значения x?

```
if x < 2 :
    print('Меньше 2')
elif x >= 2 :
    print('Два или более')
else :
    print('Что-то еще')
```

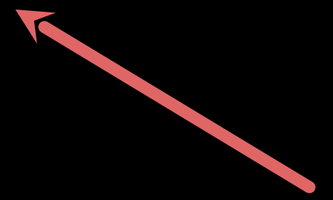
```
if x < 2 :
    print('Меньше 2')
elif x < 20 :
    print('Меньше 20')
elif x < 10 :
    print('Меньше 10')
else :
    print('Что-то еще')
```

Конструкция try / except

- Окружайте рискованные блоки кода конструкцией try (попытаться) и except (за исключением)
- Если код внутри блока try работает, выполнение кода внутри except пропускается
- Если код внутри try терпит неудачу, программа переходит к выполнению кода из блока except

```
$ cat notry.py
astr = 'Привет, Боб'
istr = int(astr)
print('Первый', istr)
astr = '123'
istr = int(astr)
print('Второй', istr)
```

```
$ python3 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: 'Привет, Боб'
```



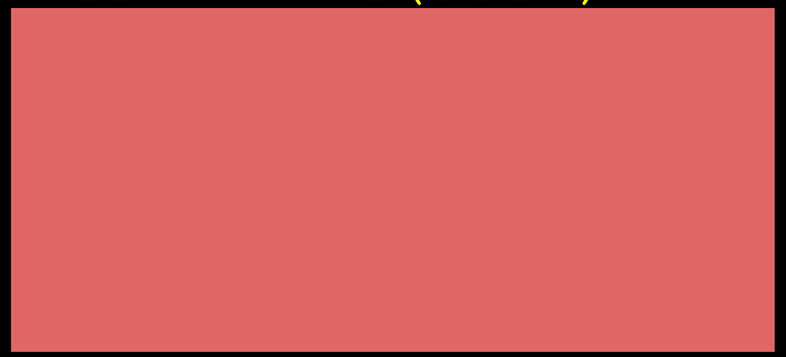
Готово

```
$ python3 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: 'Привет, Боб'
```

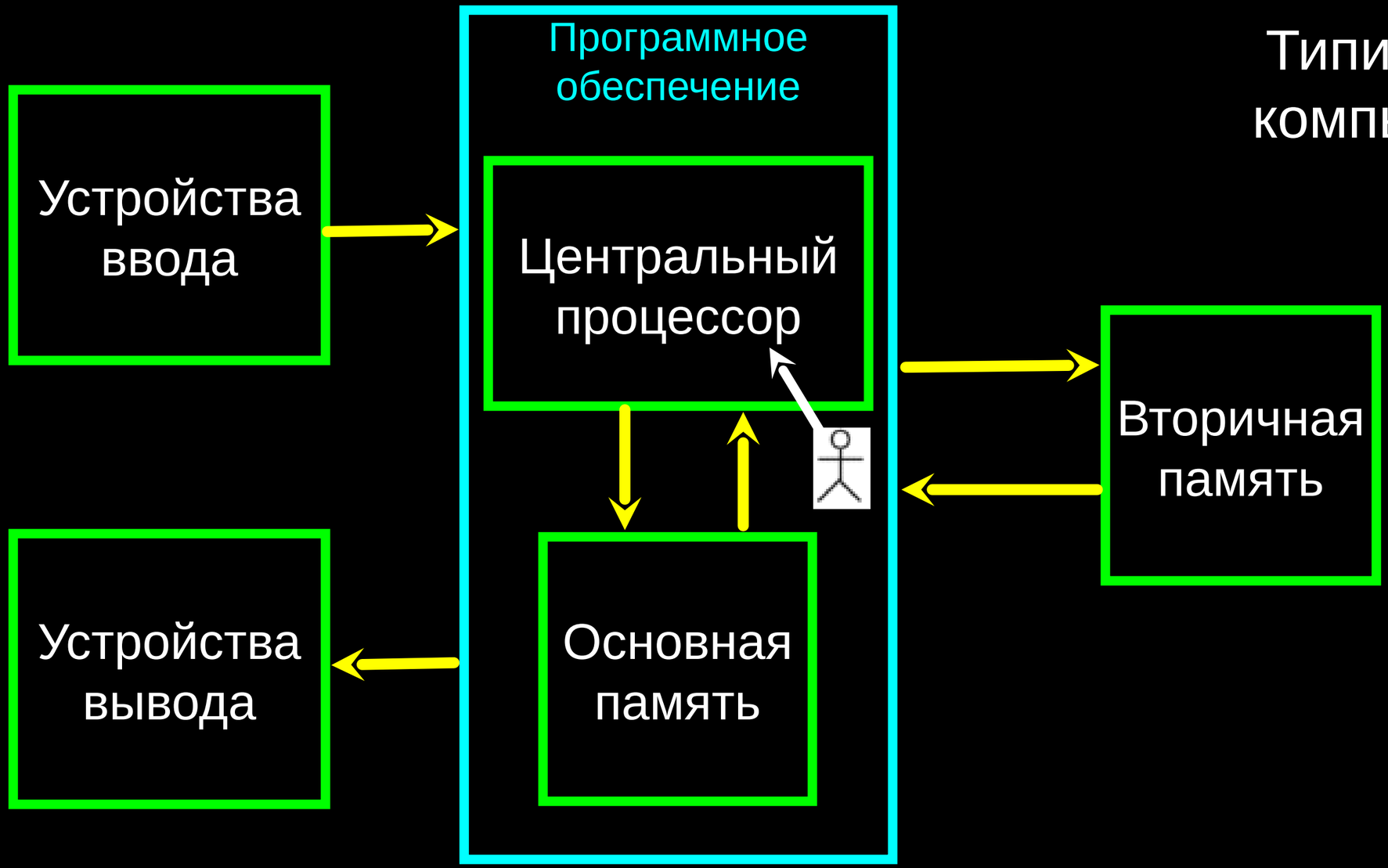
Готово

Программа останавли-вается здесь

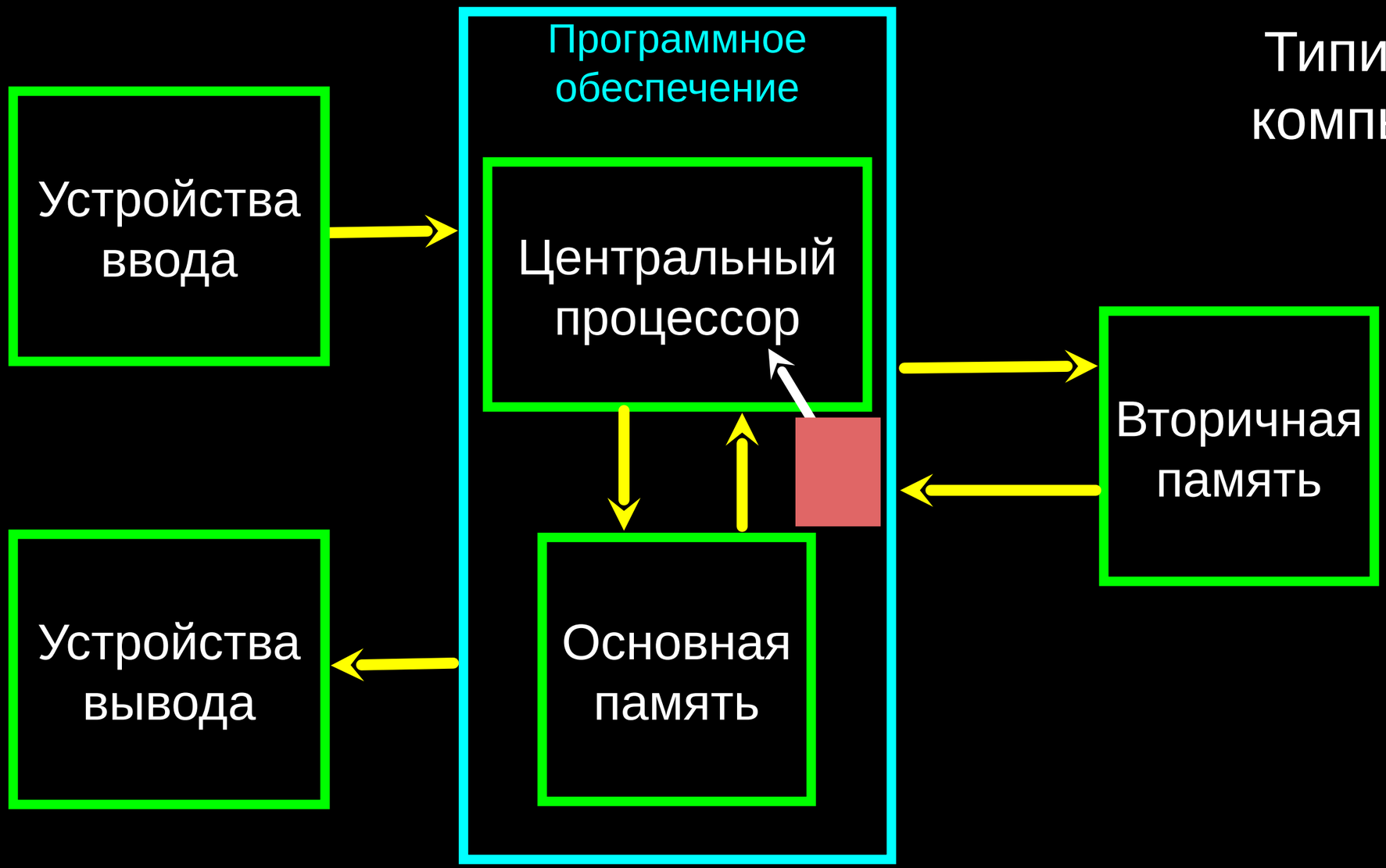
```
$ cat notry.py
astr = 'Привет, Боб'
istr = int(astr)
```

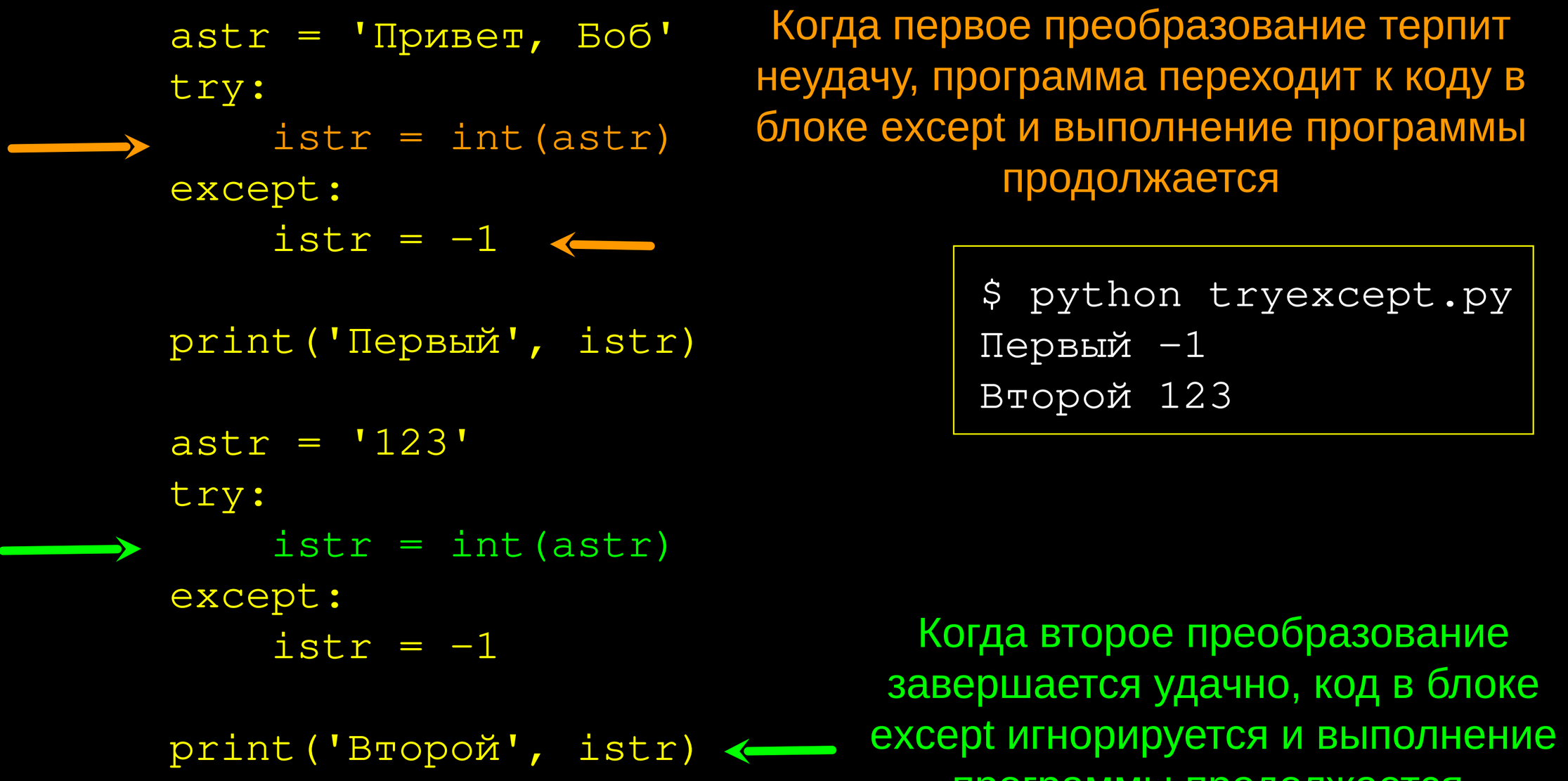


Типичный компьютер



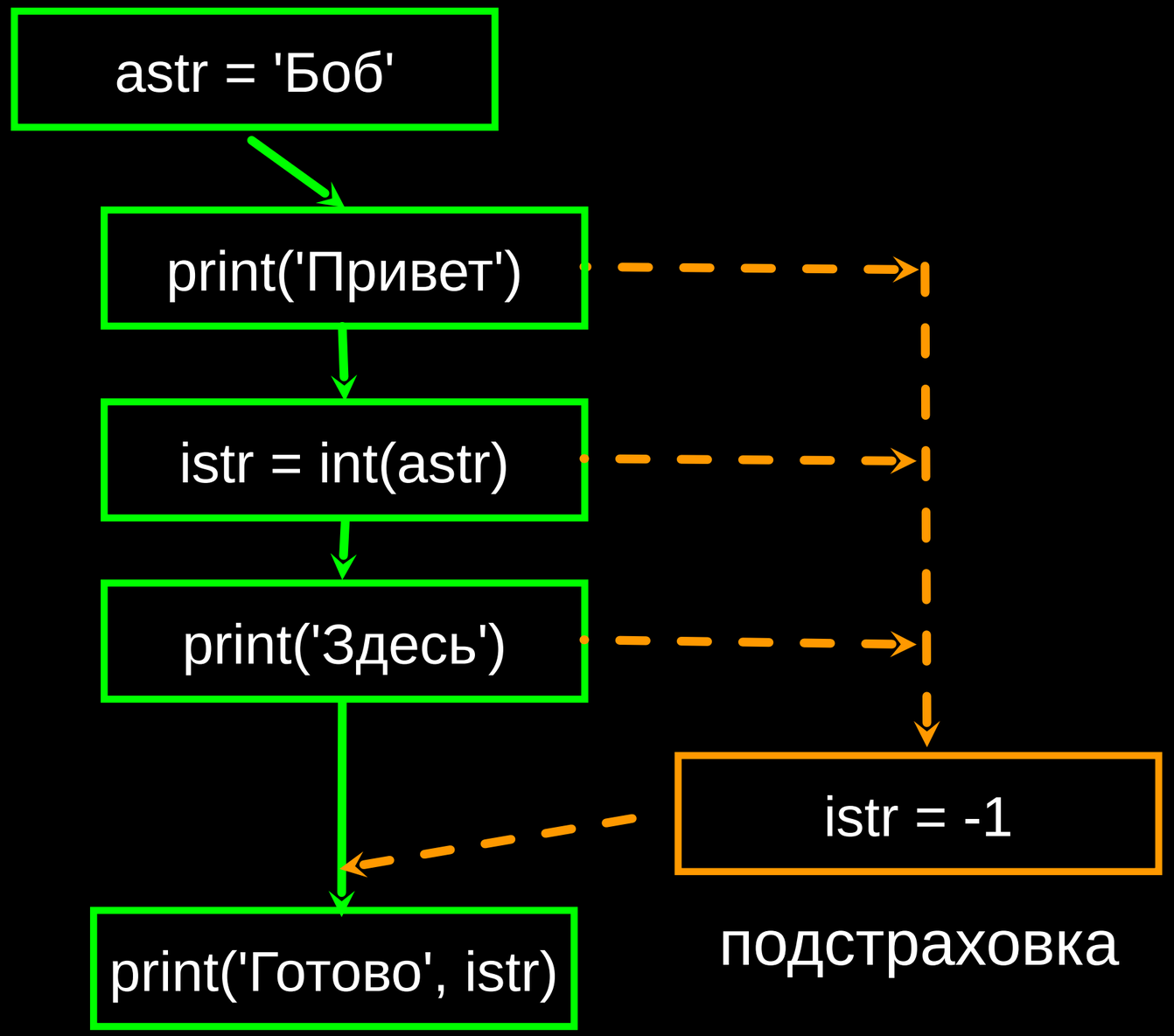
Типичный компьютер




astr = 'Привет, Боб'
try:
istr = int(astr)
except:
istr = -1
print('Первый', istr)
astr = '123'
try:
istr = int(astr)
except:
istr = -1
print('Второй', istr)
Когда первое преобразование терпит неудачу, программа переходит к коду в блоке except и выполнение программы продолжается
$ python tryexcept.py
Первый -1
Второй 123
Когда второе преобразование завершается удачно, код в блоке except игнорируется и выполнение программы продолжается.

try / except

```
astr = 'Боб'
try:
    print('Привет')
    istr = int(astr)
    print('Здесь')
except:
    istr = -1

print('Готово', istr)
```



Пример использования try / except

```
rawstr = input('Введите число:')
try:
    ival = int(rawstr)
except:
    ival = -1

if ival > 0 :
    print('Отличная работа!')
else:
    print('Не является числом')
```

```
$ python3 trynum.py
Введите число:42
Отличная работа!
$ python3 trynum.py
Введите число:forty-two
Не является числом
$
```

Резюме

- Операторы сравнения
  == <= >= > < !=
- Отступы
- Односторонние решения
- Двусторонние решения: операторы if: и else:
- Вложенные решения
- Многосторонние решения с использованием оператора elif
- try / except для компенсации ошибок

Задание

Перепишите программу расчета заработной платы, принимая в расчет, что за каждый сверхурочный час (после того, как отработал больше 40 часов) сотрудник получает 1,5-кратную ставку.

```
Введите количество часов: 45
Введите ставку: 10

Оплата: 475.0
```

475 = 40 * 10 + 5 * 15

Задание

Перепишите программу оплаты с использованием конструкции try / except, чтобы программа корректно обрабатывала нечисловой ввод.

```
Введите количество часов: 20
Введите ставку: девять
Ошибка, пожалуйста, введите числовое значение

Введите количество часов: сорок
Ошибка, пожалуйста, введите числовое значение
```

Авторы / Благодарности

Исходная разработка: Чарльз Северанс, Школа Информации Мичиганского Университета.

Перевод выполнила Фомкина Виолетта.

… Добавьте сюда новых авторов и переводчиков

...