

# Execução Condicional

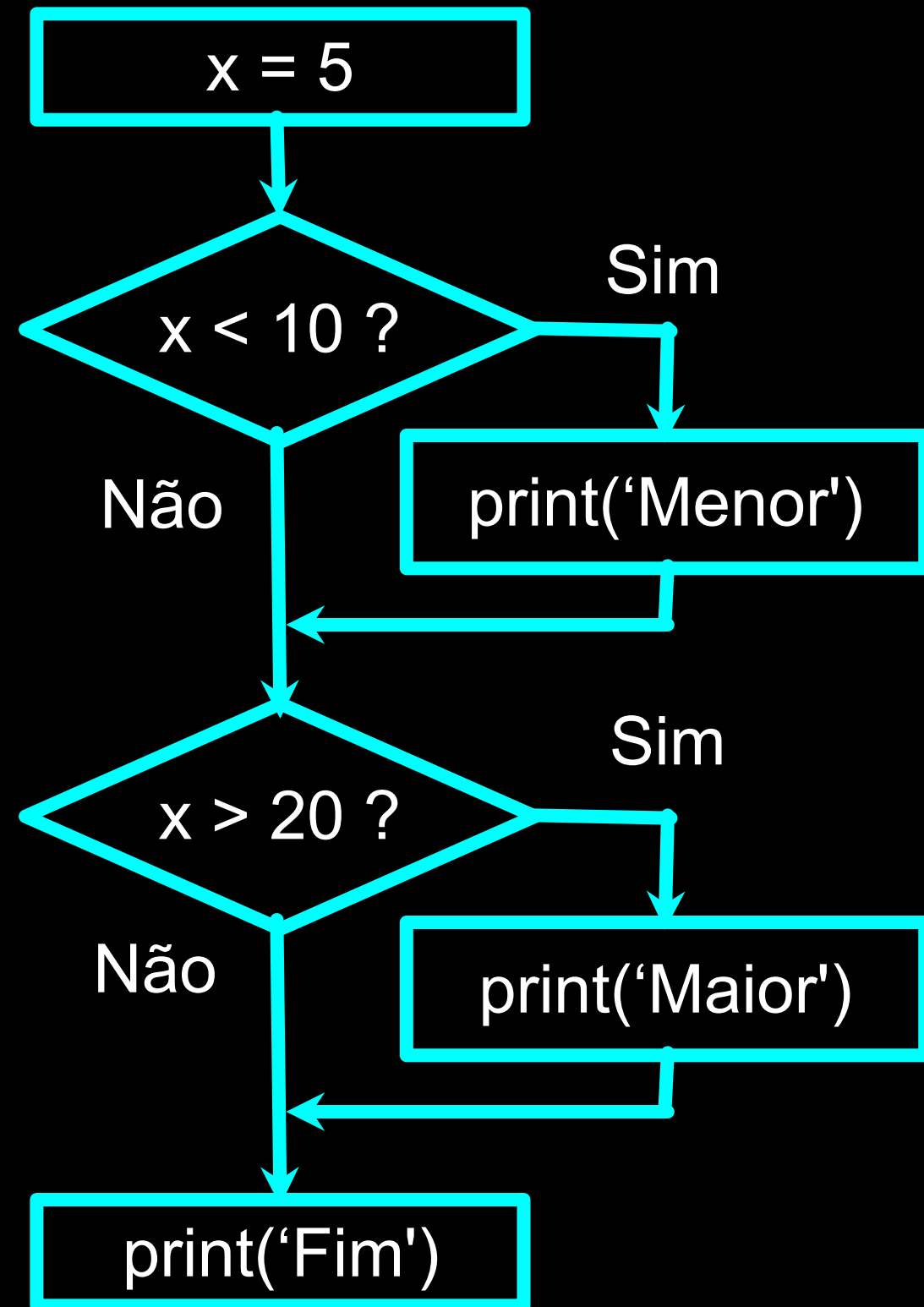
## Capítulo 3



Python for Everybody  
[www.py4e.com](http://www.py4e.com)



# Passos Condicionais



Programa:

```
x = 5
if x < 10:
    print('Menor')
if x > 20:
    print('Maior')

print('Fim')
```

Saída:

Menor  
Fim

# Operadores de Comparação

- **Expressões booleanas** fazem uma pergunta e produzem um resultado Sim ou Não, que usamos para controlar o fluxo do programa
- **Expressões booleanas** usando **operadores de comparação** avaliam como True / False ou Sim / Não
- Operadores de comparação examinam variáveis, mas não alteram as variáveis

Python	Significado ou Semântica
<	Menor que
<=	Menor ou igual a
==	Igual a
>=	Maior ou igual a
>	Maior que
!=	Diferente de

Lembrar: “=” é usado para atribuição.

[http://en.wikipedia.org/wiki/George\\_Boole](http://en.wikipedia.org/wiki/George_Boole)

# Operadores de Comparação

```
x = 5
```

```
if x == 5 :
```

```
    print('Igual a 5')
```

Igual a 5

```
if x > 4 :
```

```
    print('Maior que 4')
```

Maior que 4

```
if x >= 5 :
```

```
    print('Maior ou Igual a 5')
```

Maior ou Igual a 5

```
if x < 6 : print('Menor que 6')
```

Menor que 6

```
if x <= 5 :
```

```
    print('Menor ou Igual a 5')
```

Menor ou Igual a 5

```
if x != 6 :
```

```
    print('Diferente de 6')
```

Diferente de 6

# One-Way Decisions

```
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

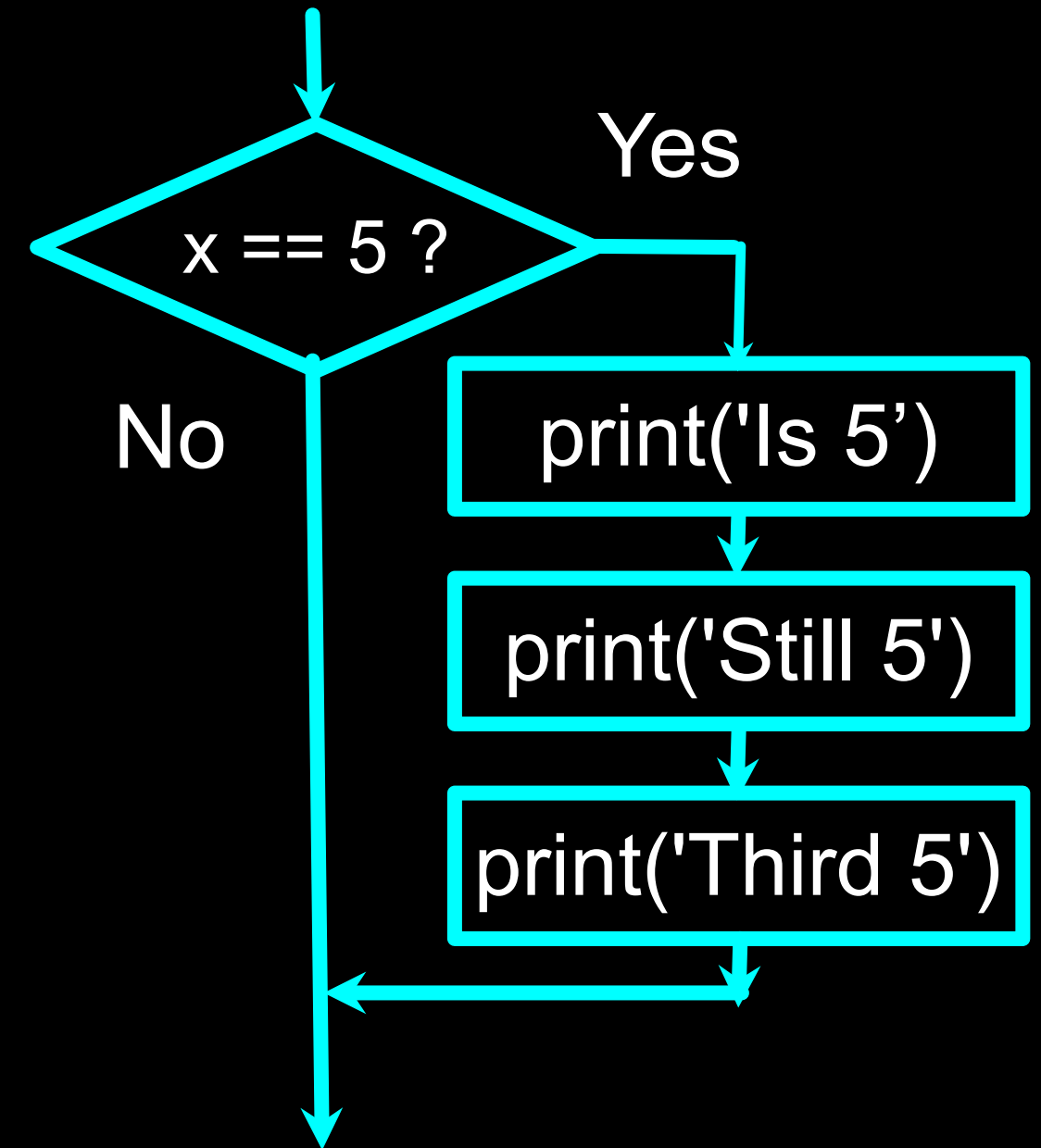
Is Still 5

Third 5

Afterwards 5

Before 6

Afterwards 6

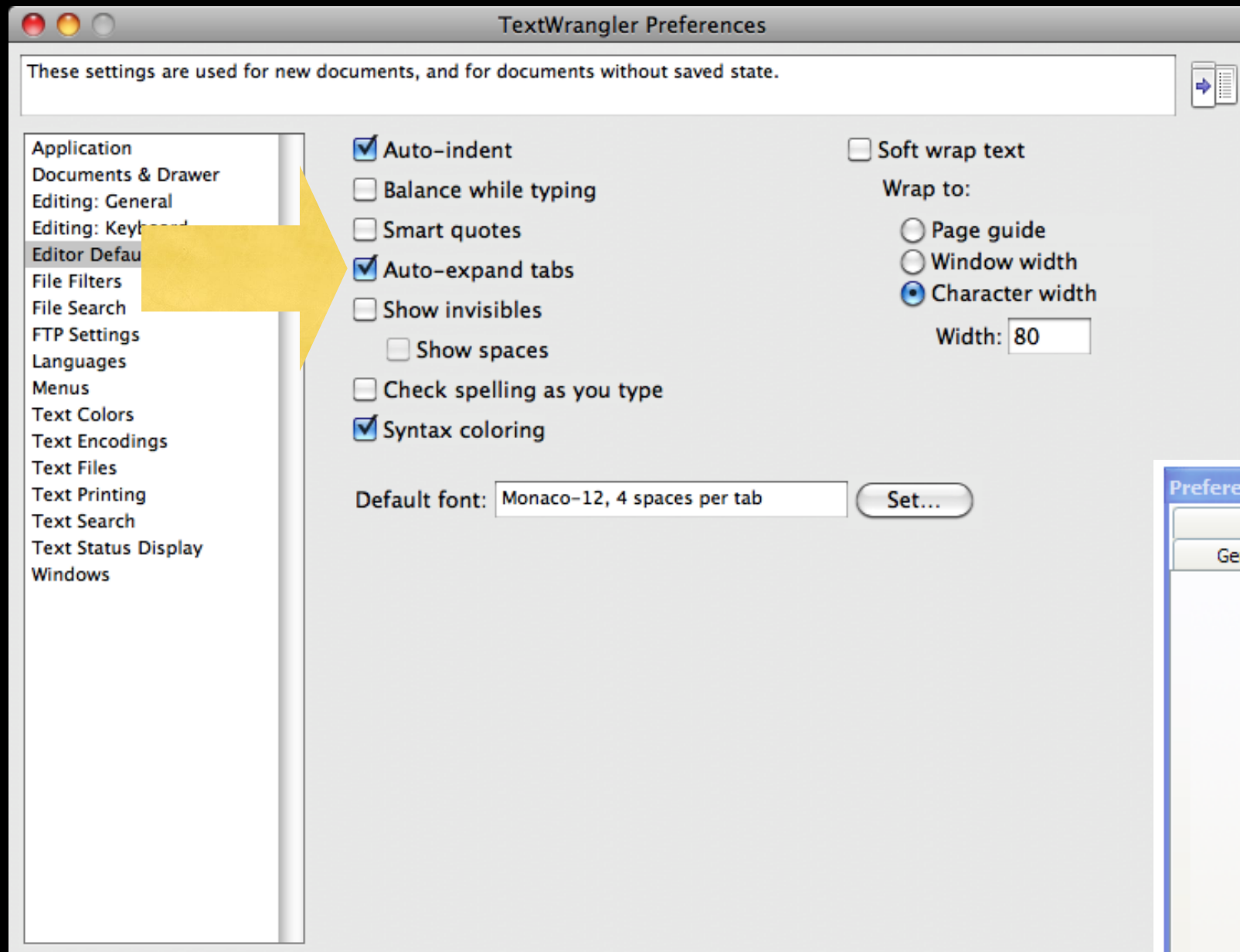


# Indentação

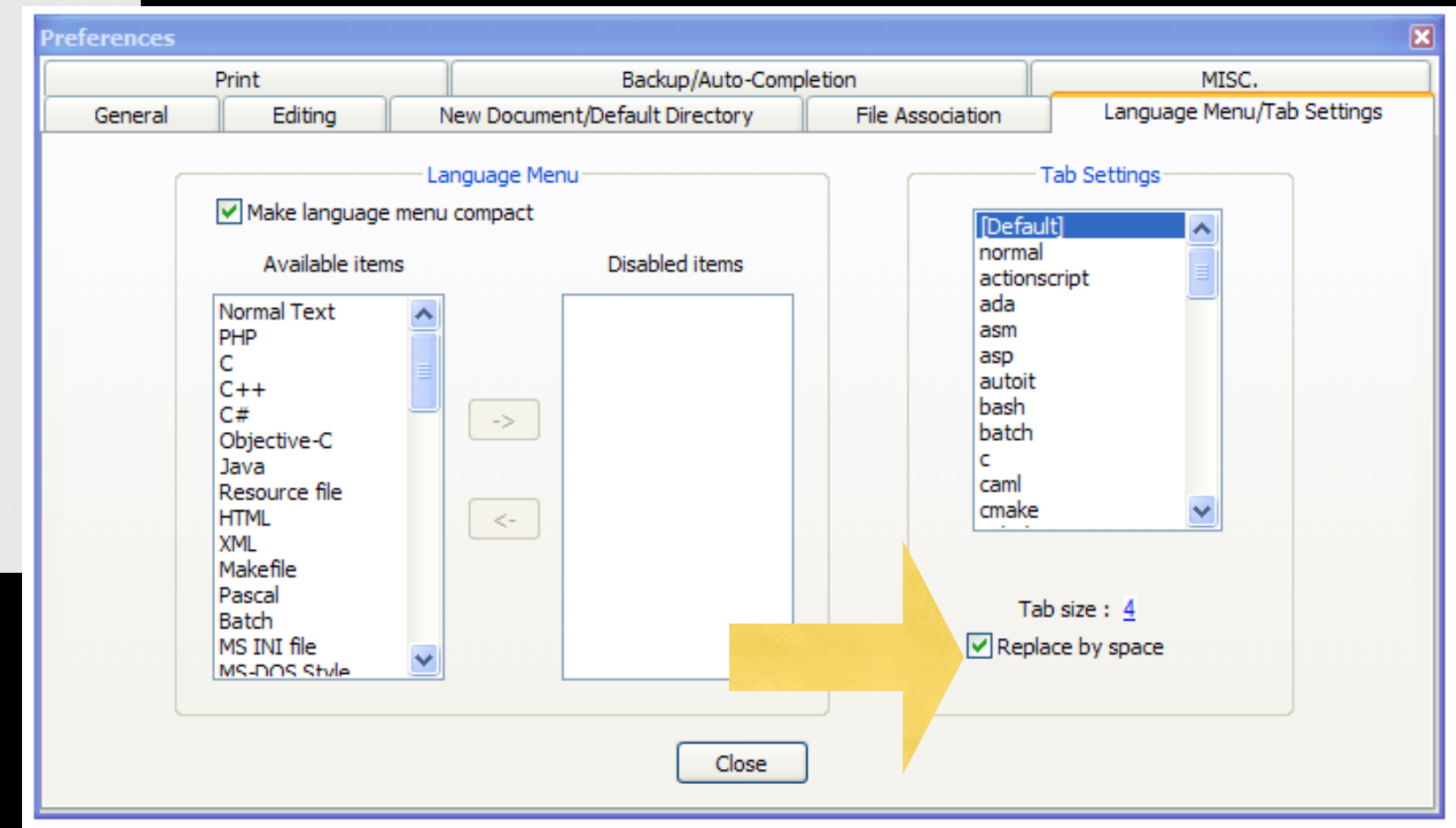
- **Aumente a indentação** depois de uma declaração de um **if** ou um **for** (depois dos : )
- **Mantenha a indentação** para indicar o **escopo** do bloco (quais linhas são afetadas pelo **if/for**)
- **Reduza a indentação** ao nível do **if** ou do **for** para indicar o fim do bloco
- **Linhas em branco** são ignoradas - elas não afetam a **indentação**
- **Comentários** sozinhos em uma linha são ignorados em relação à **indentação**

# Warning: Turn Off Tabs!!

- Atom automatically uses spaces for files with ".py" extension (nice!)
- Most text editors can turn **tabs** into **spaces** - make sure to enable this feature
  - Notepad++: Settings -> Preferences -> Language Menu/**Tab** Settings
  - TextWrangler: TextWrangler -> Preferences -> Editor Defaults
- Python cares a \*lot\* about how far a line is indented. If you mix **tabs** and **spaces**, you may get "**indentation errors**" even if everything looks fine

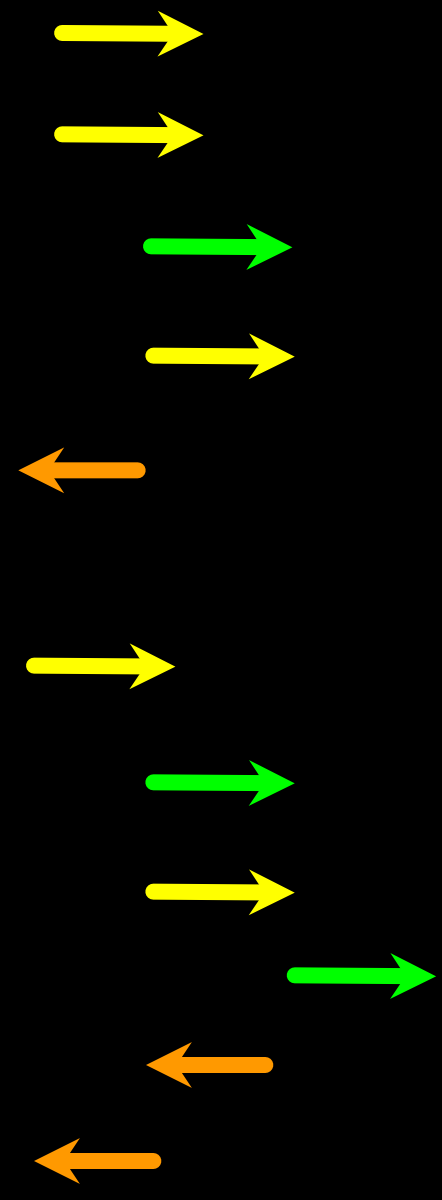


This will save you  
much unnecessary  
pain.





**aumente / mantenha** depois do **if** ou **for**  
**diminua** para indicar o fim do bloco



The diagram illustrates the execution flow of the provided Python code. It uses colored arrows to show the sequence of execution and the nesting of blocks. Yellow arrows indicate the main flow of the program. Green arrows indicate the execution of code inside an if or for block. Orange arrows indicate the exit from a block and the continuation of the main flow.

```
x = 5
if x > 2 :
    print('Maior que 2')
    print('Continua maior')
print('Terminado com 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Maior que 2')
    print('Terminado com i', i)
print('Tudo terminado')
```

# Sobre começo/fim dos blocos

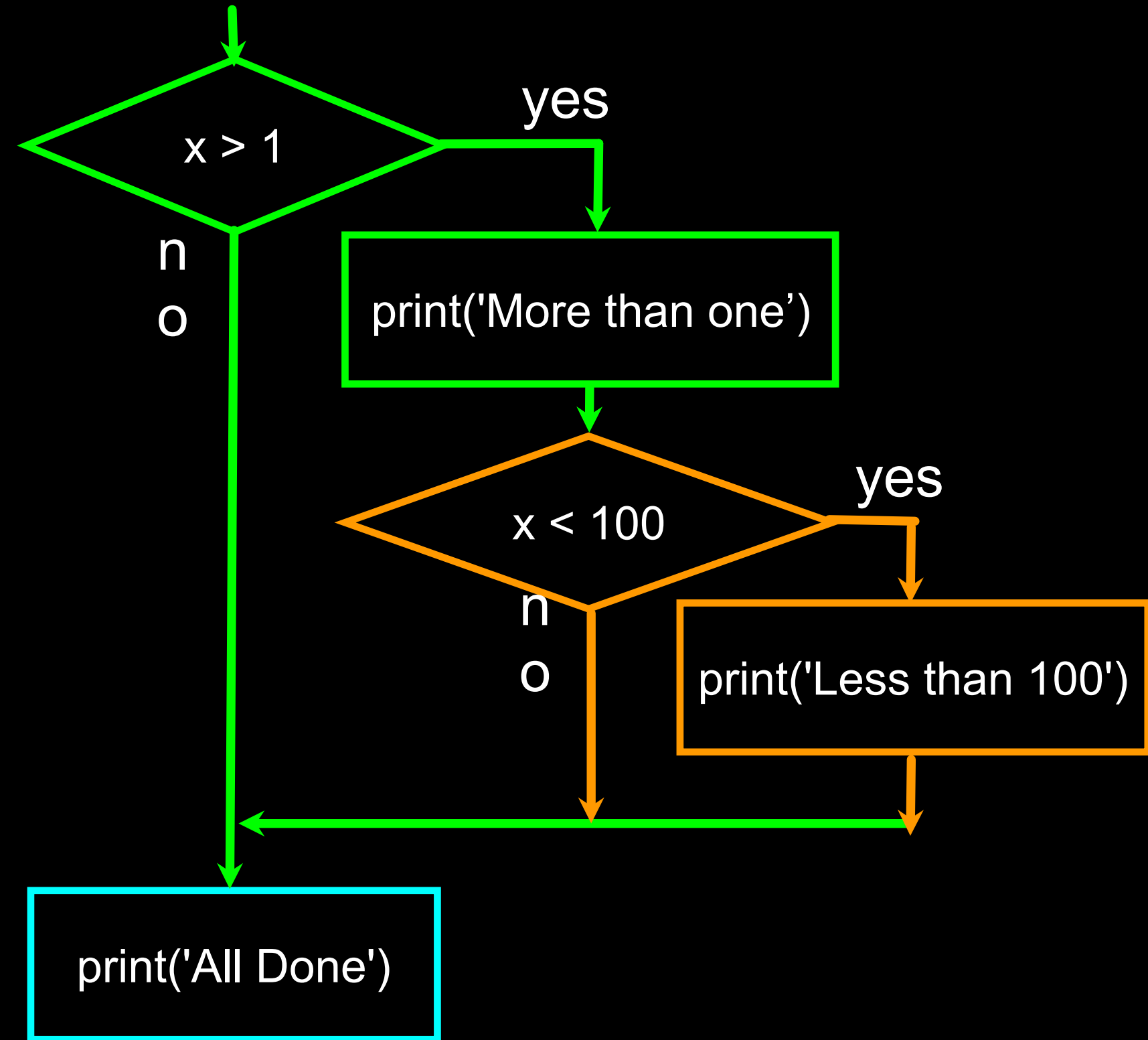
```
x = 5
```

```
if x > 2 :  
    print('Maior que 2')  
    print('Continua maior')  
print('Terminado com 2')
```

```
for i in range(5) :  
    print(i)  
    if i > 2 :  
        print('Maior que 2')  
    print('Terminado com i', i)  
print('Tudo terminado')
```

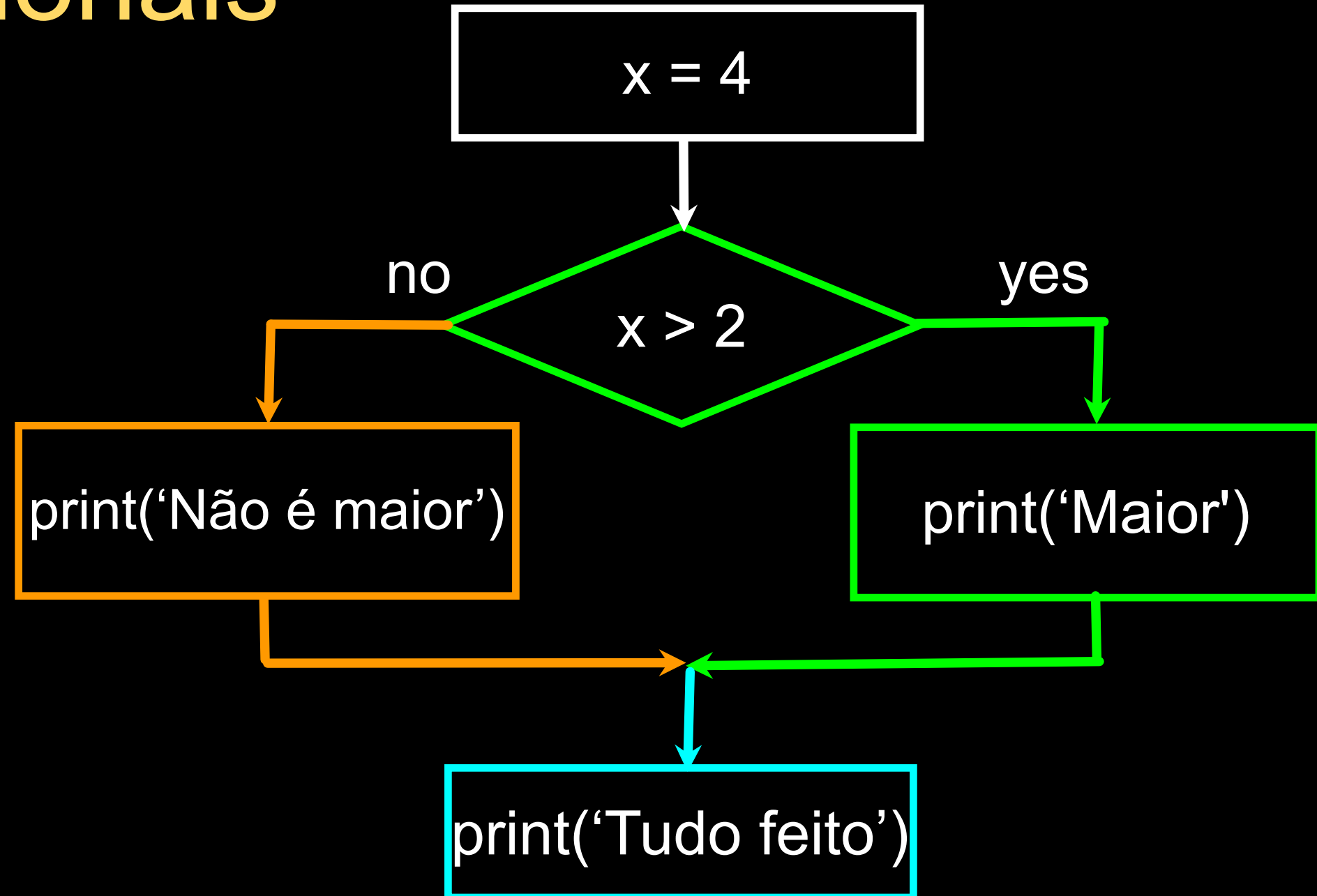
# Decisões Aninhadas

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



# Decisões Bidirecionais

- Às vezes, queremos fazer uma coisa se uma expressão lógica for verdadeira e outra coisa se a expressão for falsa
- É como uma bifurcação na estrada - precisamos escolher **um ou outro** caminho, mas não ambos

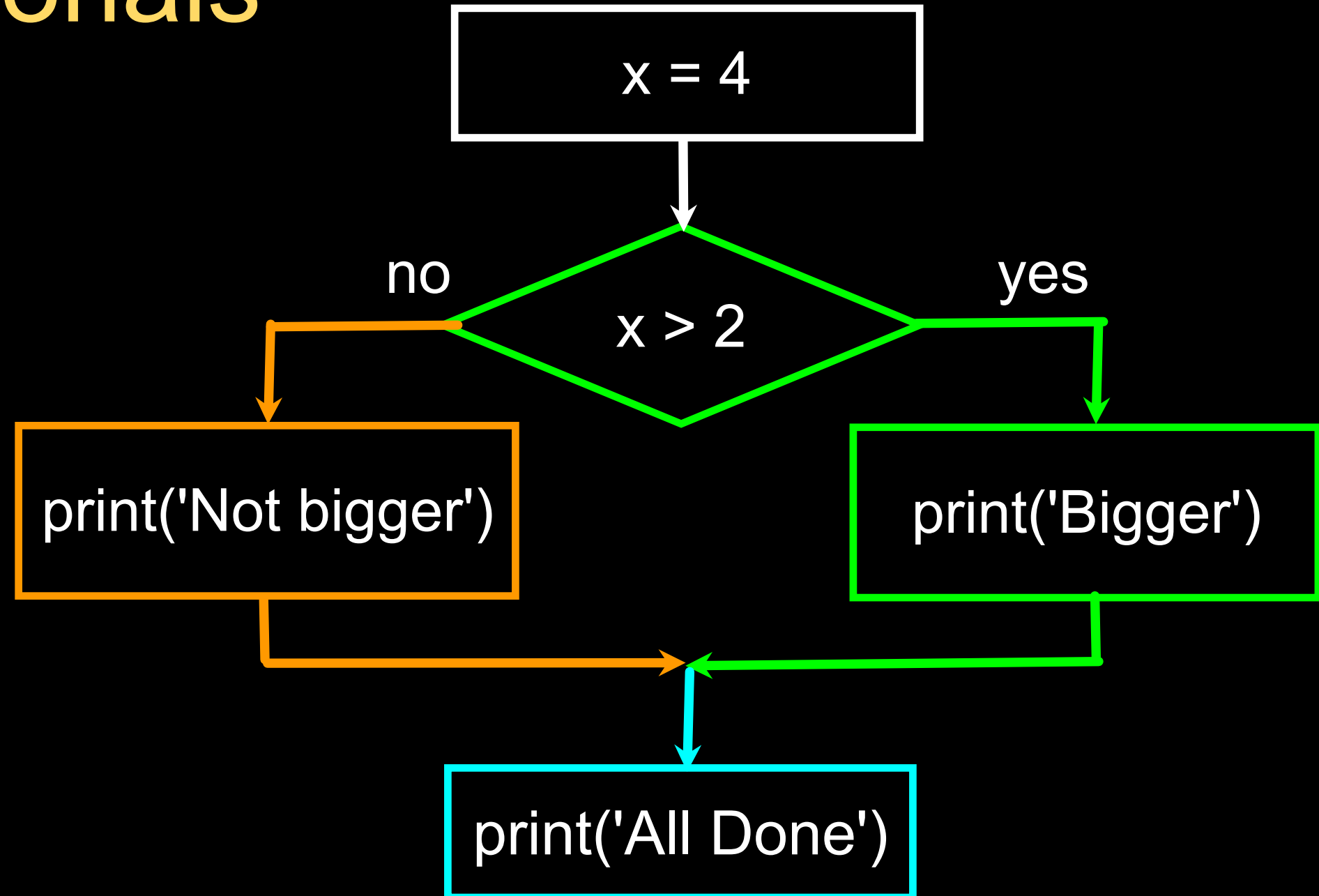


# Decisões bidirecionais com else:

```
x = 4
```

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

```
print('All done')
```

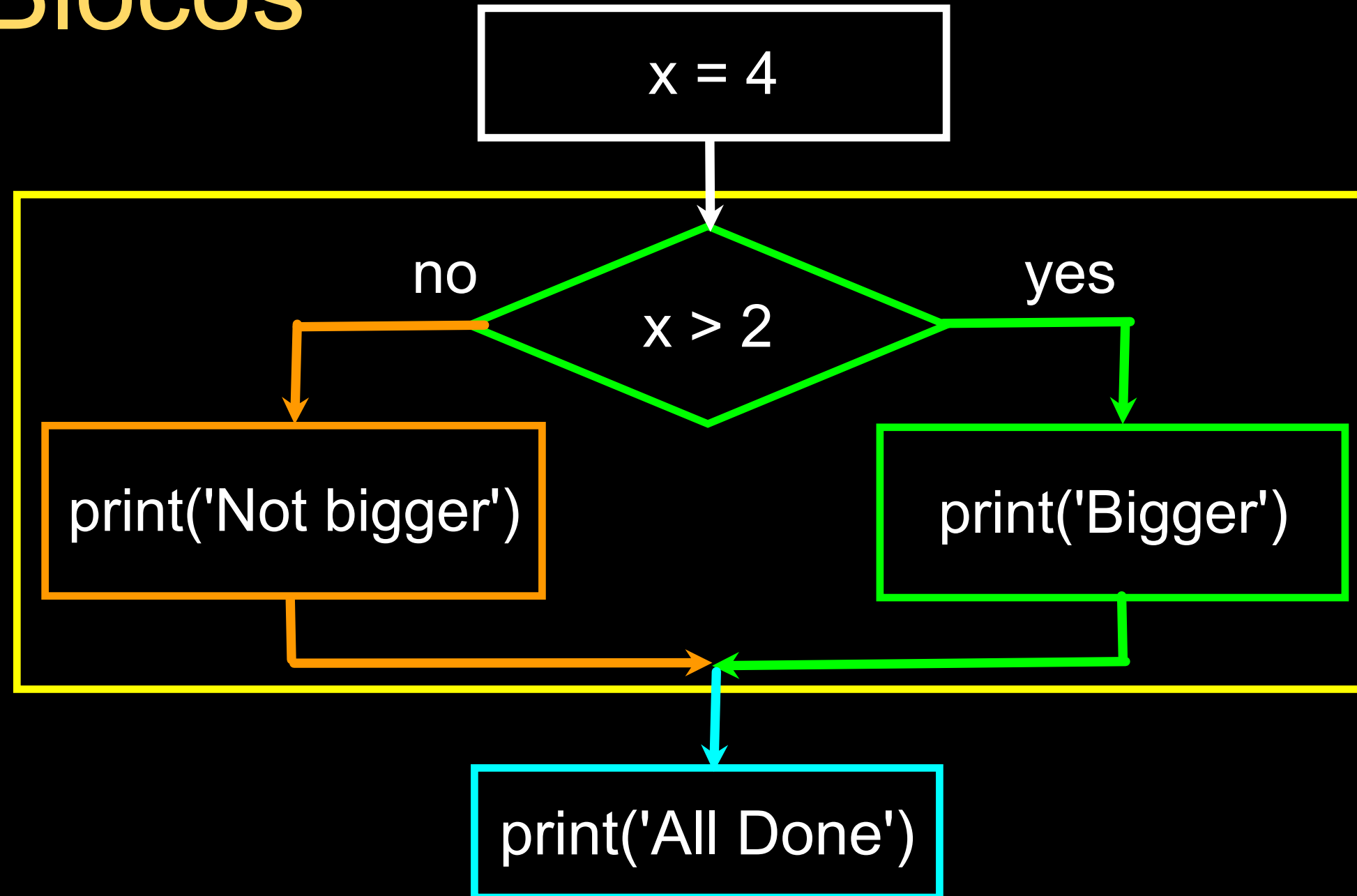


# Visualizando os Blocos

x = 4

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

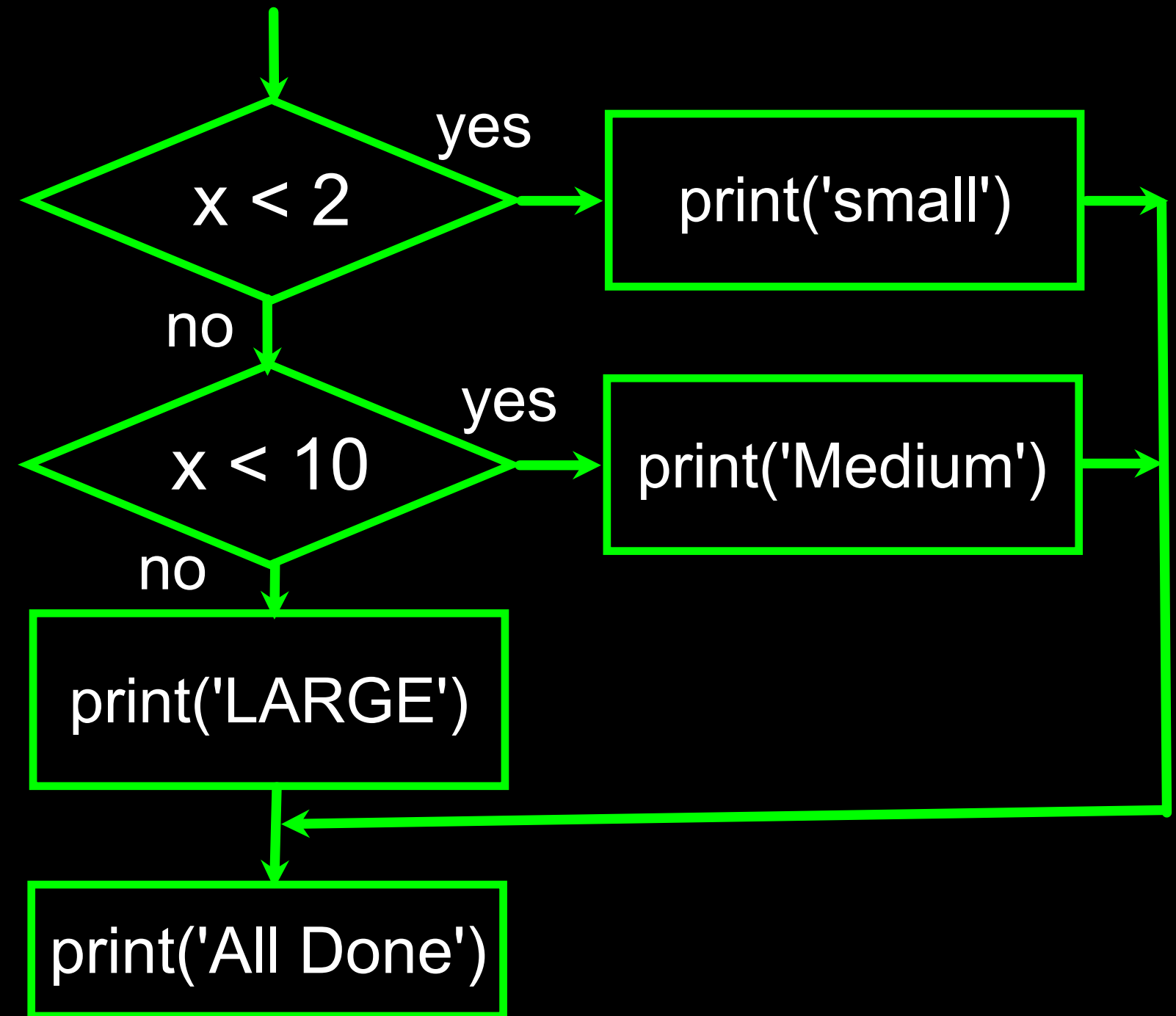
```
print('All done')
```



Mais Estruturas Condicionais...

# Multidirecionais

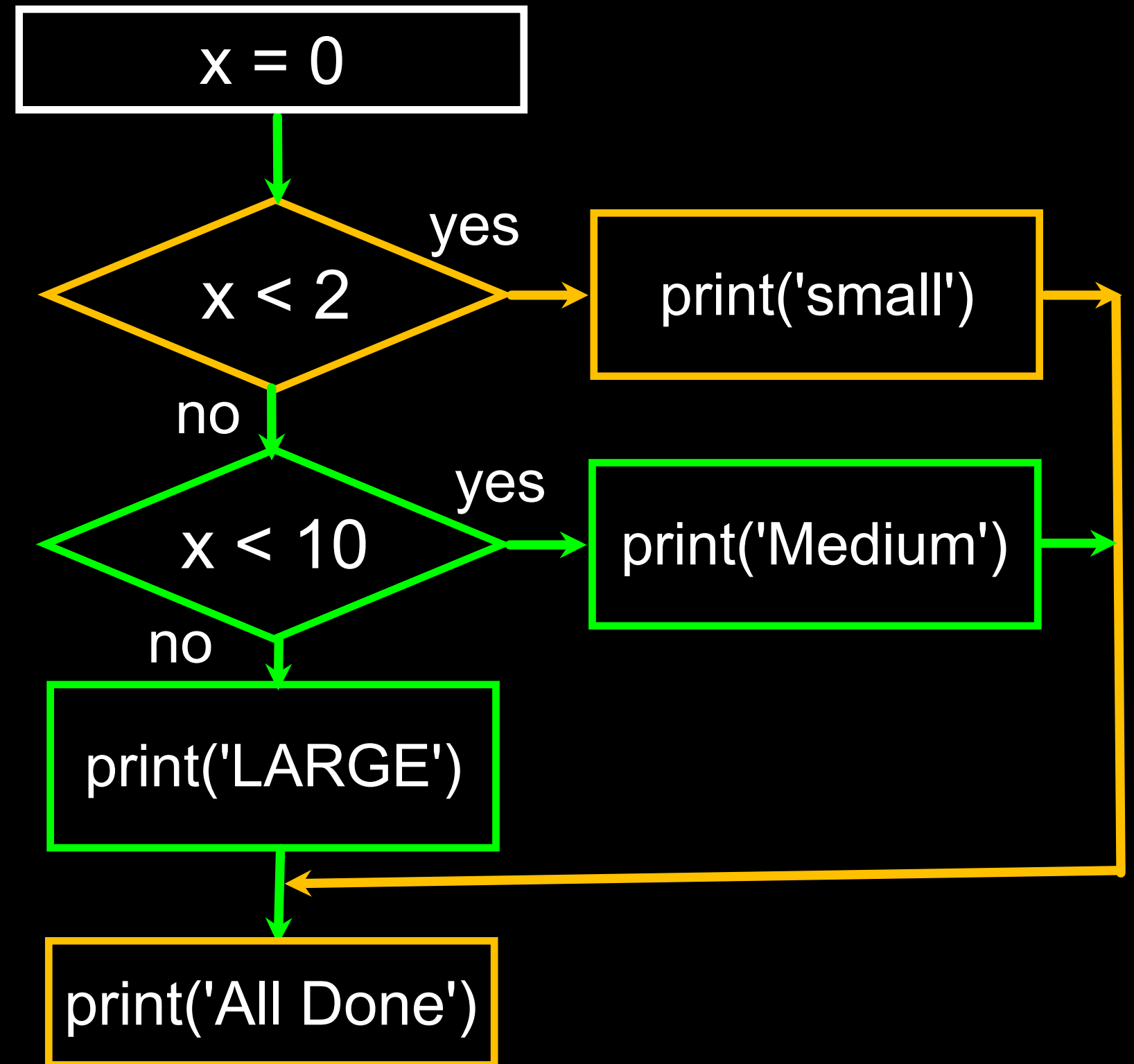
```
if x < 2 :  
    print('small')  
elif x < 10 :  
    print('Medium')  
else :  
    print('LARGE')  
print('All done')
```





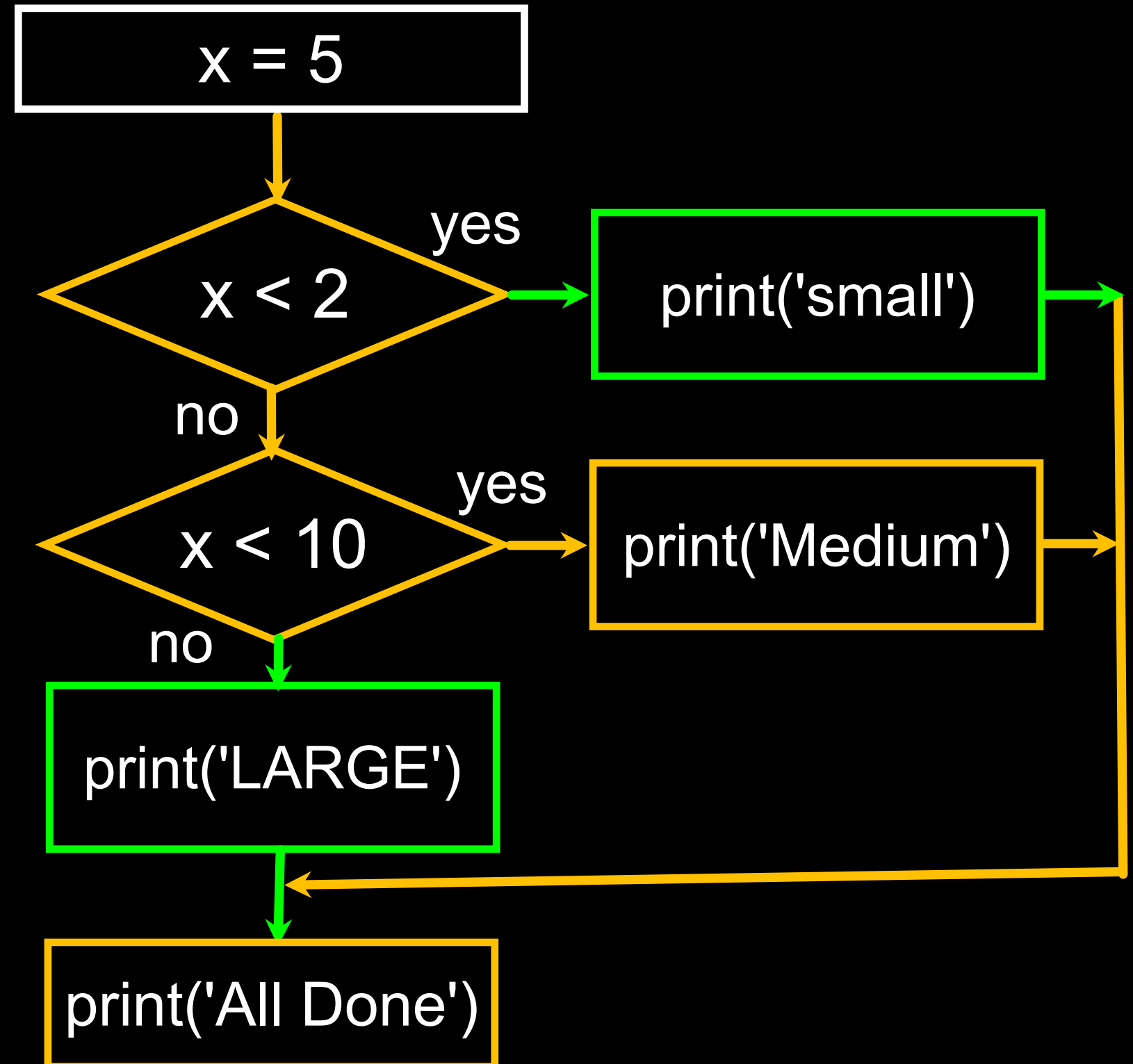
# Multidirecionais

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



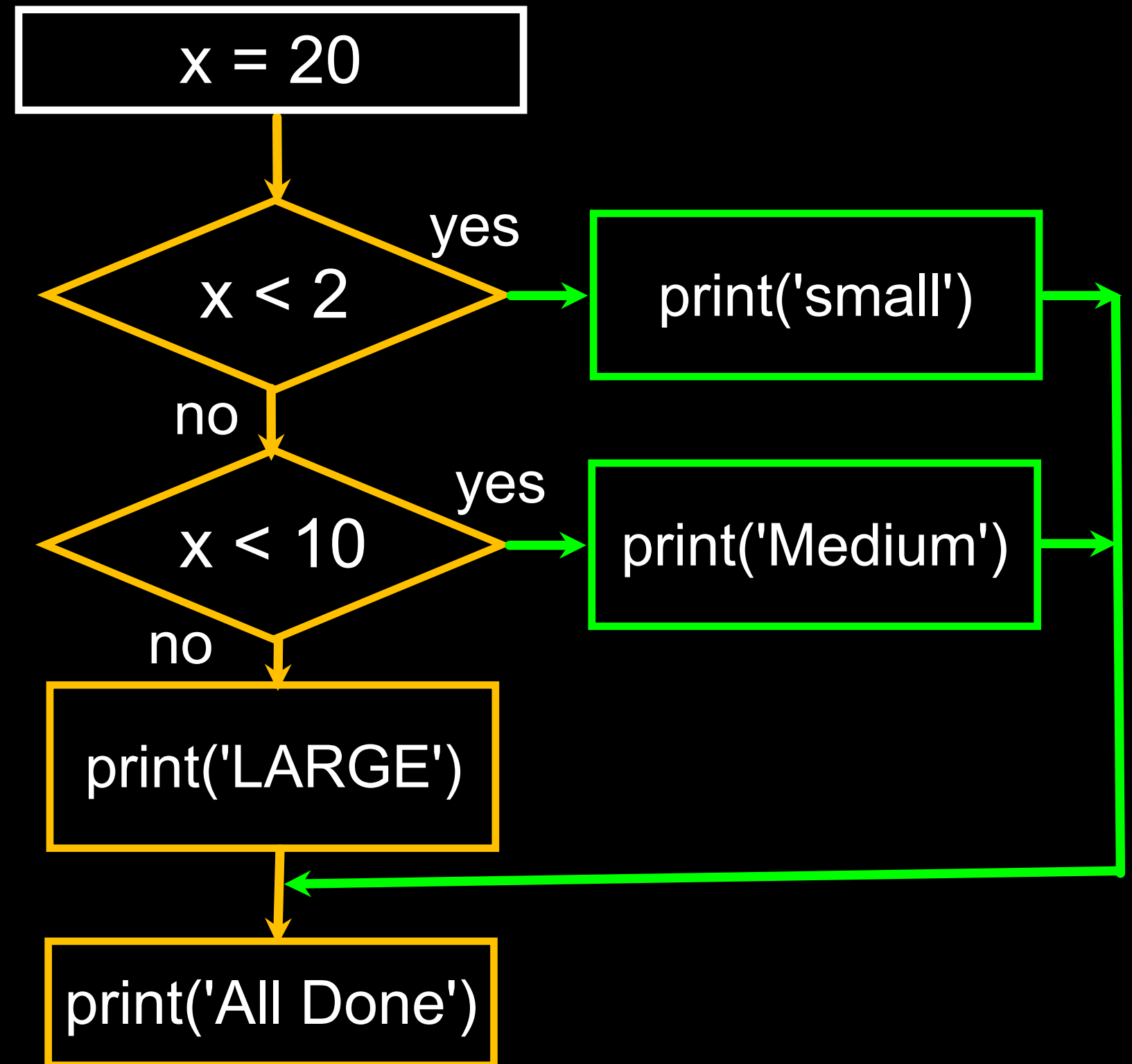
# Multidirecionais

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



# Multidirecionais

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



# Multidirecionais

```
# No 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')
```

# Quebra-cabeça Multidirecional

O que nunca será impresso independente do valor de x?

```
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')
```

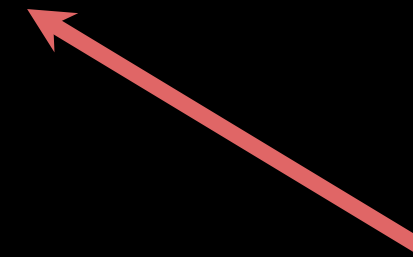
# A estrutura try / except

- You surround a dangerous section of code with **try** and **except**
- If the code in the **try** works - the **except** is skipped
- If the code in the **try** fails - it jumps to the **except** section

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



All  
Done

○  
programa  
para aqui

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

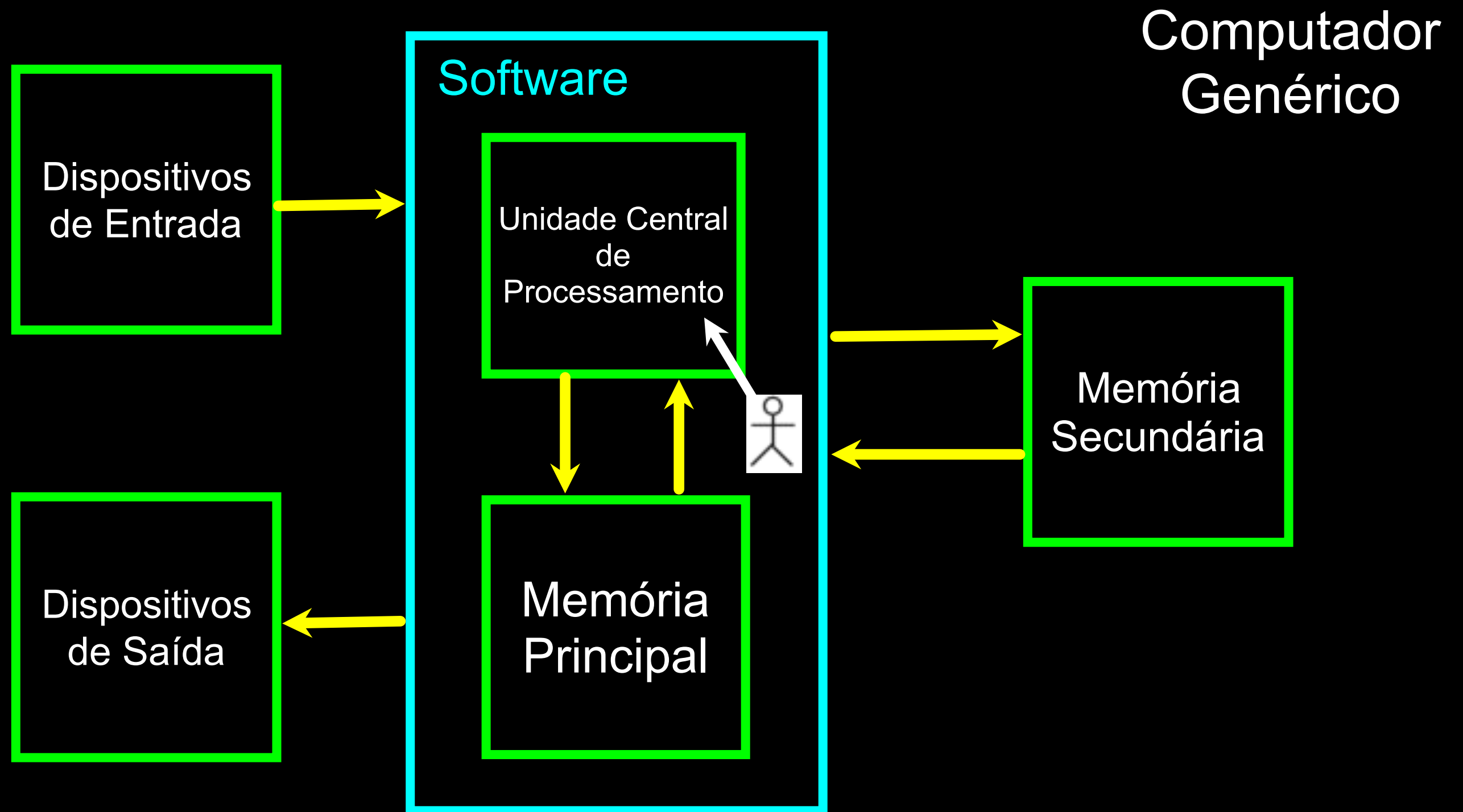


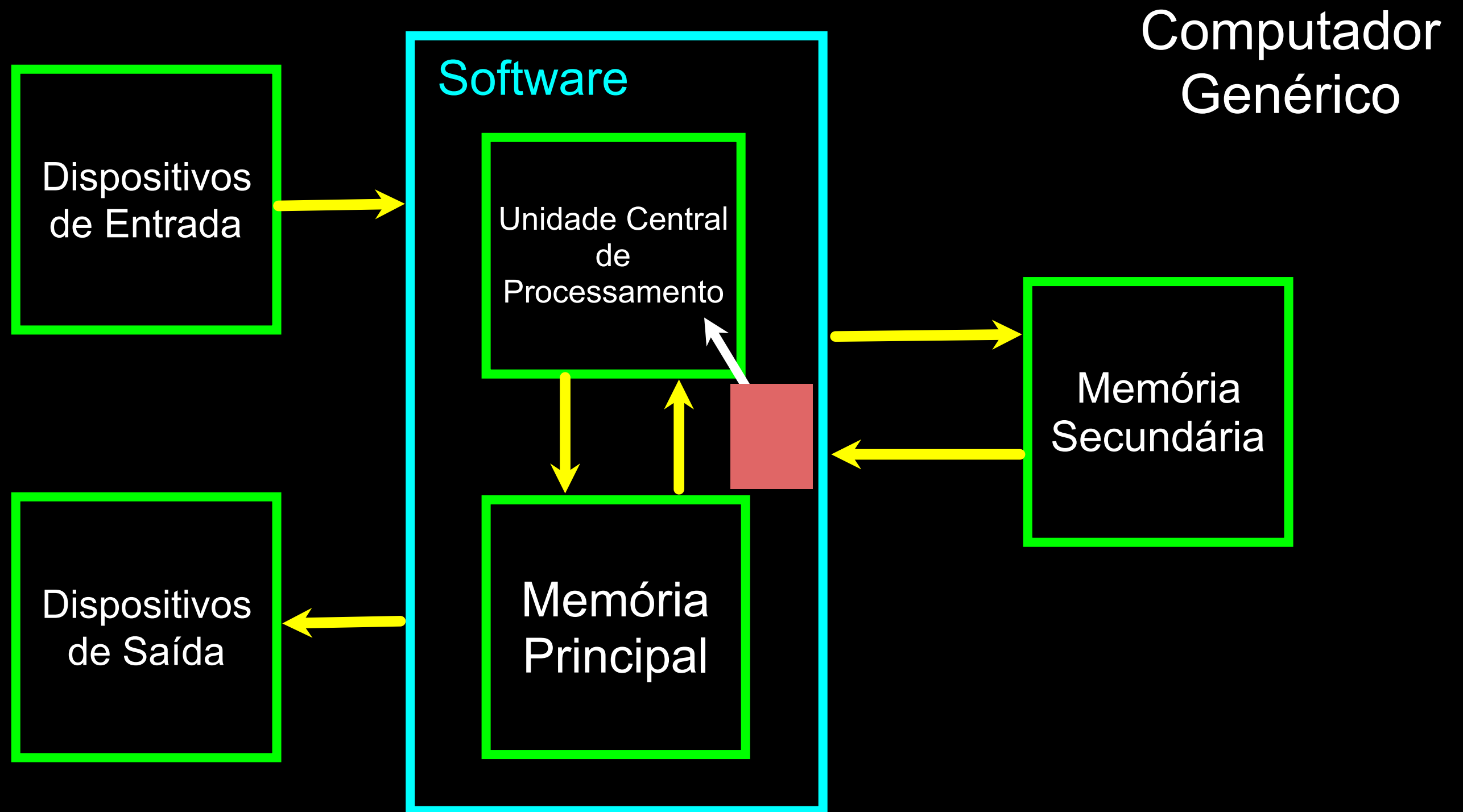
```
$ python3 notry.py
```

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

Terminado!







```
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)
```

Quando a primeira conversão falha - ela cai no “except”: e o programa continua.

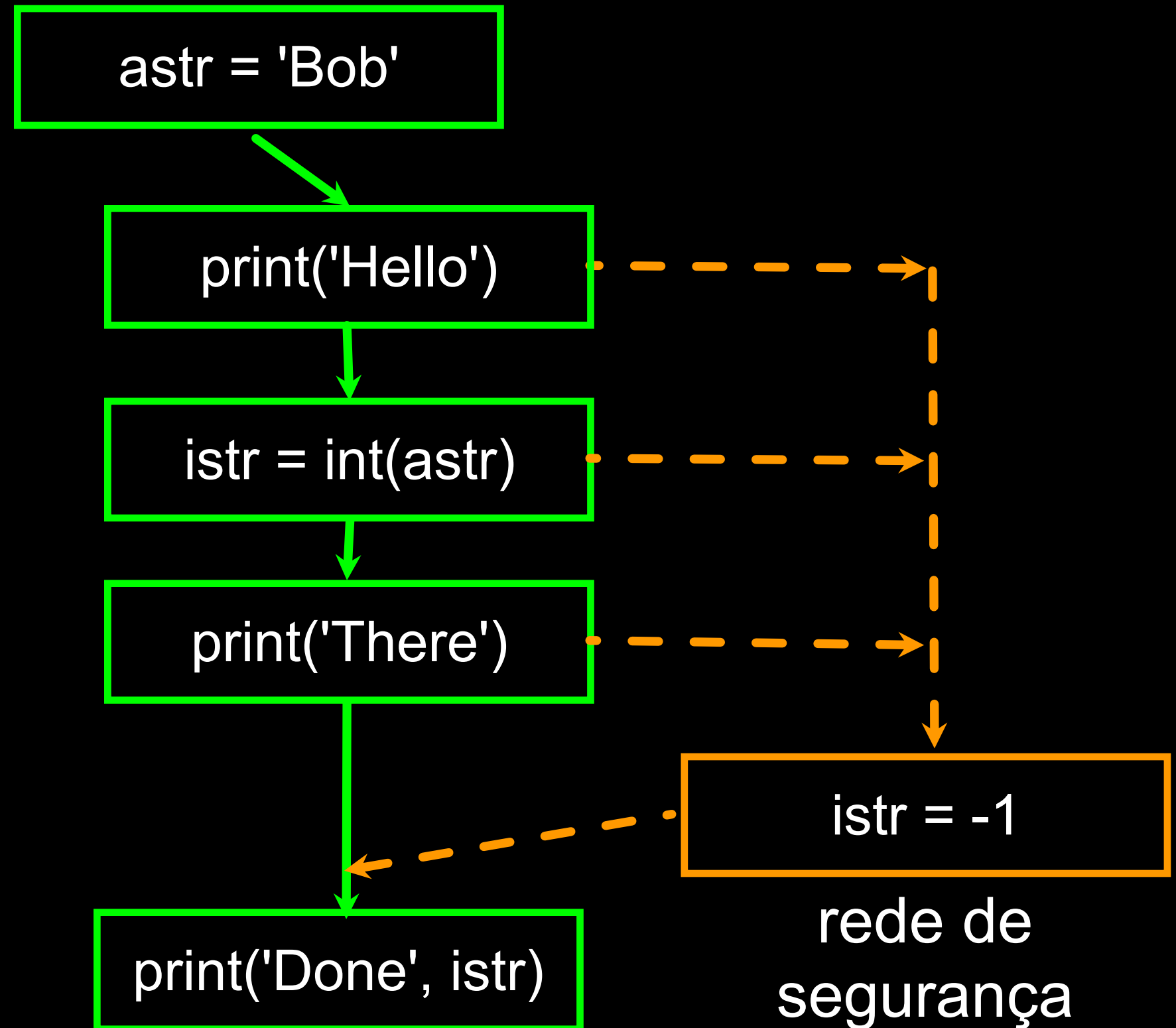
```
$ python tryexcept.py
First -1
Second 123
```

Quando a segunda conversão é bem-sucedida - ela simplesmente ignora o “except”: e o programa continua.

# try / except

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

print('Done', istr)
```



# Sample try / except

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

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

```
$ python3 trynum.py
Enter a number:42
Nice work
$ python3 trynum.py
Enter a number:forty-two
Not a number
$
```

# Resumo

- Operadores de comparação  
`==` `<=` `>=` `>` `<` `!=`
- Indentação
- Decisões Unidirecionais
- Decisões Bidirecionais:  
`if:` e `else:`
- Decisões aninhadas
- Decisões multidirecionais com  
`elif`
- `try` / `except` para tratar erros

## Exercício

Reescreva seu cálculo salarial para fornecer ao funcionário 1,5 vezes a taxa horária das horas trabalhadas acima de 40 horas.

Digite as horas: 45

Digite a taxa: 10

Pagamento: 475.0

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

## Exercício

Reescreva seu programa de pagamento usando “try” e “except” para que seu programa lide com entradas não numéricas normalmente.

Digite as horas: 20

Digite a taxa: nine

Error, please enter numeric input

Digite as horas: forty

Error, please enter numeric input





# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Traduzido para o Português Brasileiro por Filipe Calegario

...