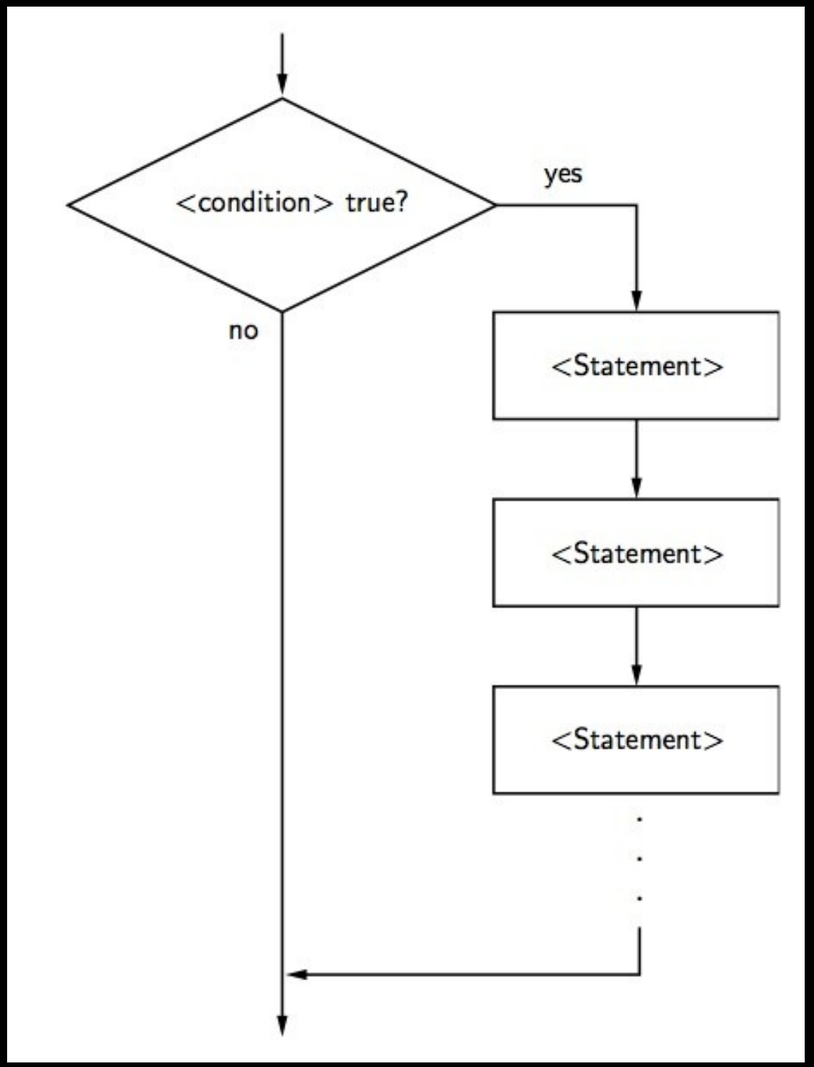
Decision Structures
Zelle - Chapter 7

Charles Severance - www.dr-chuck.com

Textbook: Python Programming: An Introduction to Computer Science, John Zelle

```
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"
```

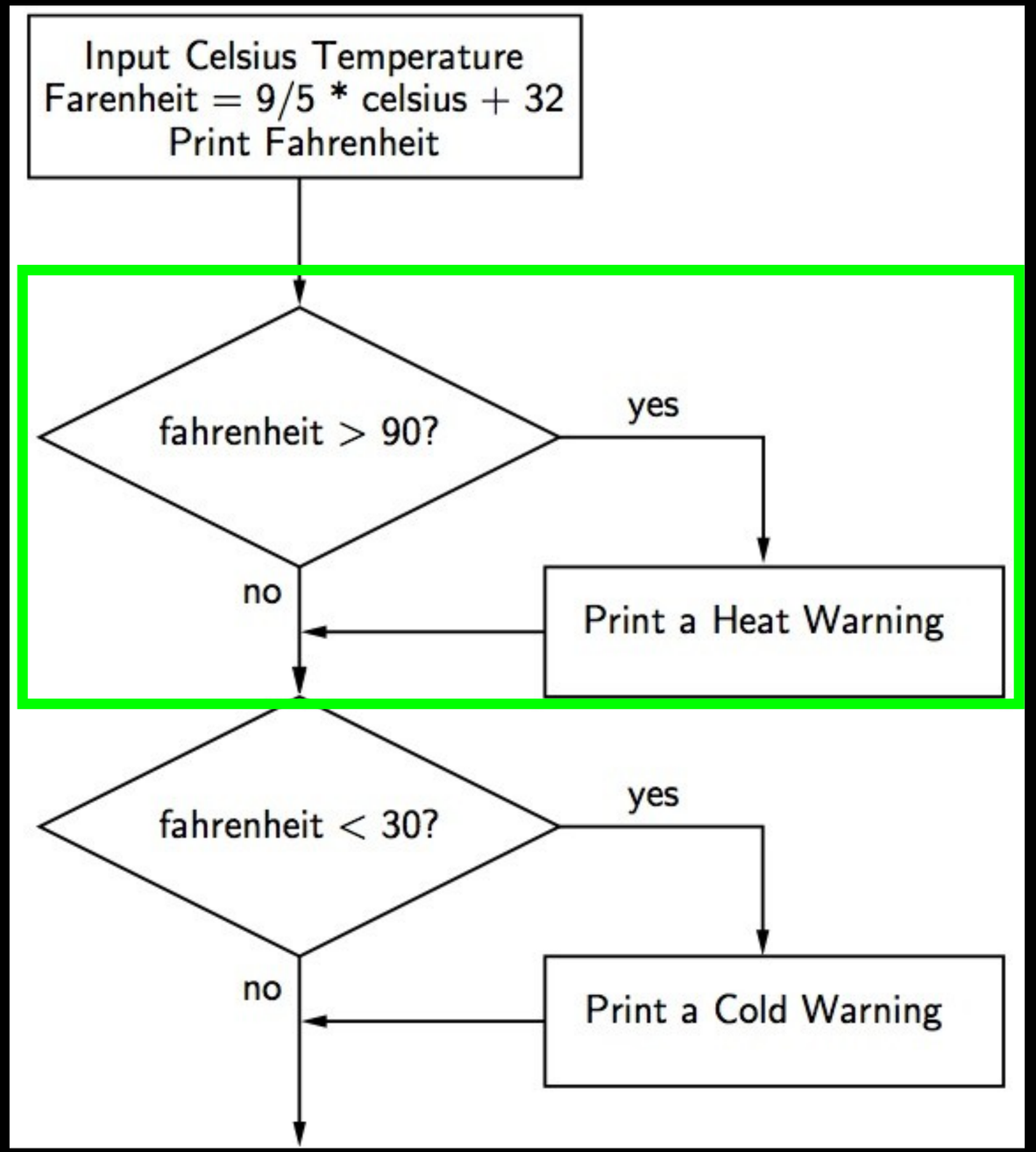
```
Before 5
Is 5
Is Still 5
Third 5
Afterwards 5
Before 6
Afterwards 6
```



One-Way Decisions

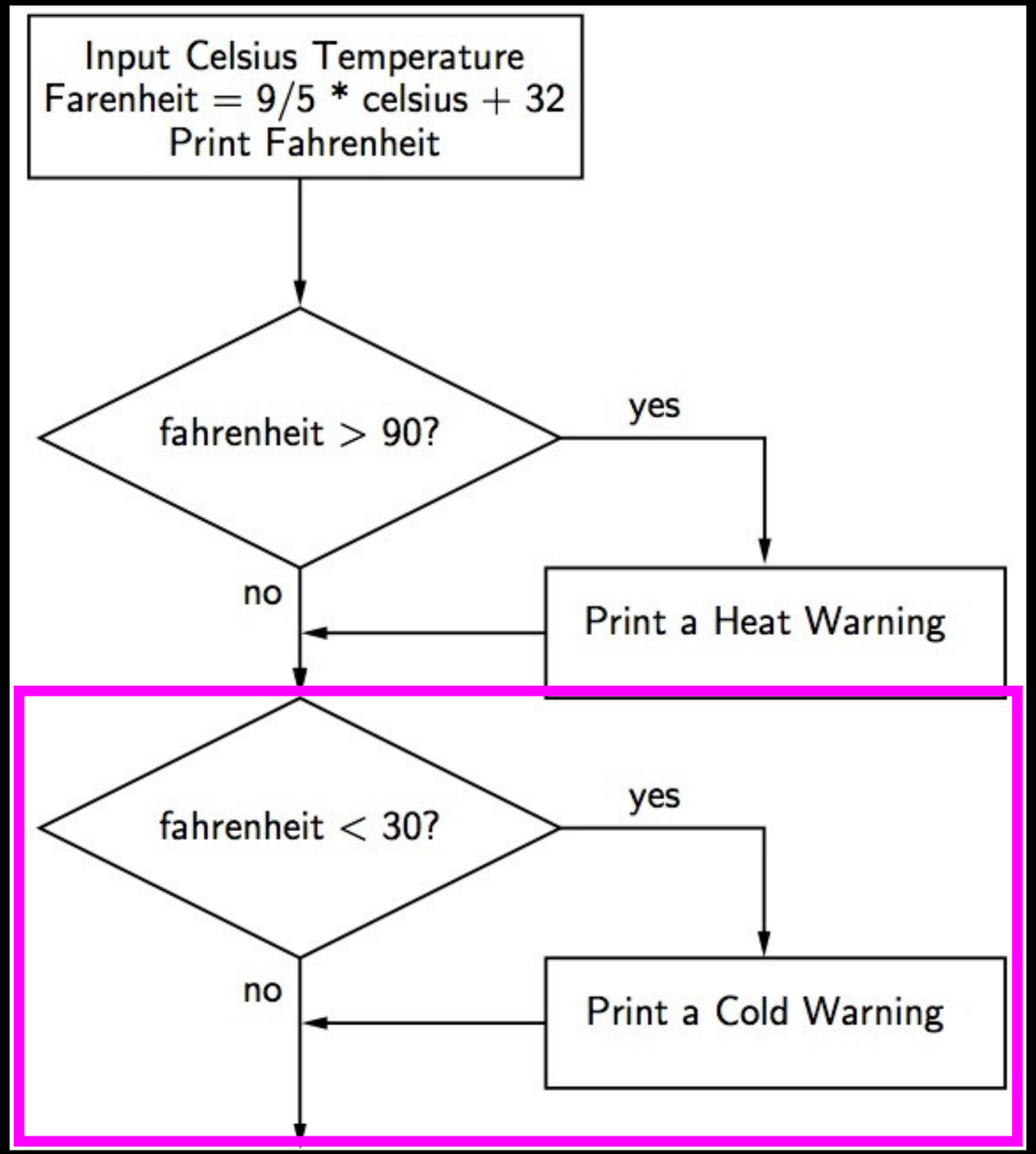
Several ifs

```
faren = 120
if ( faren > 90) :
   print "Heat Warning"

if ( faren < 32 ) :
    print "Cold Warning"
```



Several ifs

```
faren = 120
if ( faren > 90) :
   print "Heat Warning"

if ( faren < 32 ) :
    print "Cold Warning"
```



Comparison Operators

- Boolean expressions using comparison operators evaluate to - True / False - Yes / No
- Boolean expressions ask a question and produce a Yes or No result which we use to control program flow
- Comparison operators look at variables but do not change the variables

Python	Mathematics	Meaning
<	$<$	Less than
<=	$\leq$	Less than or equal to
==	$=$	Equal to
>=	$\geq$	Greater than or equal to
>	$>$	Greater than
!=	$\neq$	Not equal to

```
<expr> <relop> <expr>
```

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

Comparison Operators

```
x = 5
if ( x == 5 ) :    print "Equals 5“

if ( x > 4 ) :
    print "Greater than 4“
if ( x >= 5 ) :
    print "Greater than or Equal 5“

if ( x < 6 ) :print "Less than 6“

if ( x <= 5 ) :
    print "Less than or Equal 5“
if ( x != 6 ) :
    print "Not equal 6"
```

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

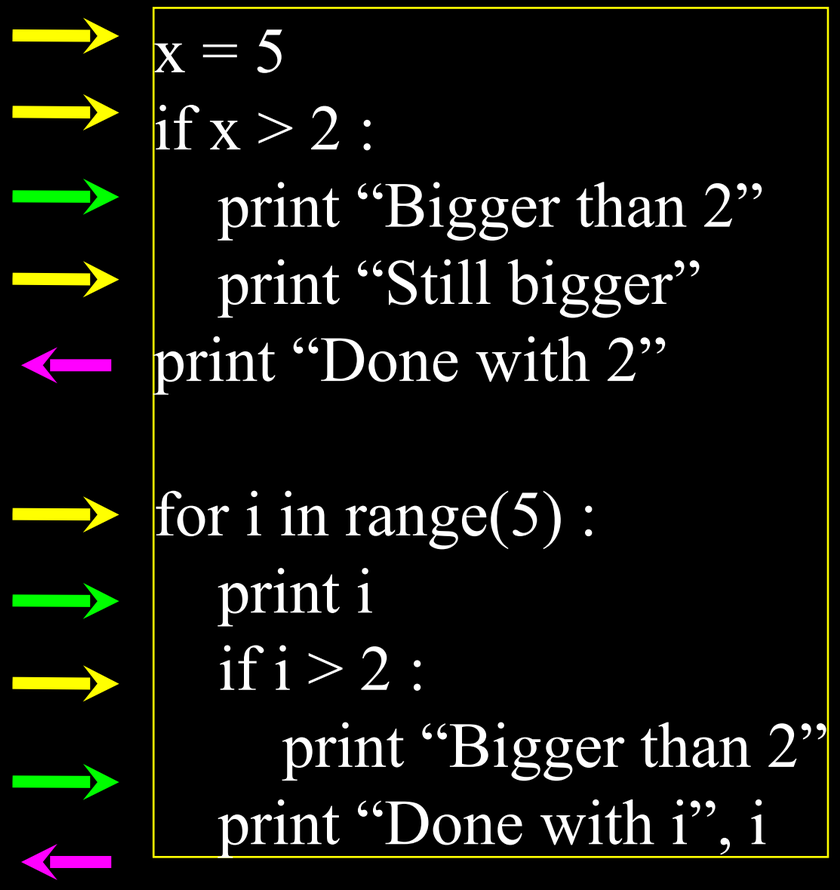
Review Indentation

- Must increase indent after an if statement or for statement (after :)
- Maintain indent to indicate the scope of the block (which lines are affected by the if/for)
- Reduce indent to back to the level of the if statement or for statement to indicate the end of the block
- Blank lines are ignored - they can appear anywhere
- Comments on a line by themselves are ignored

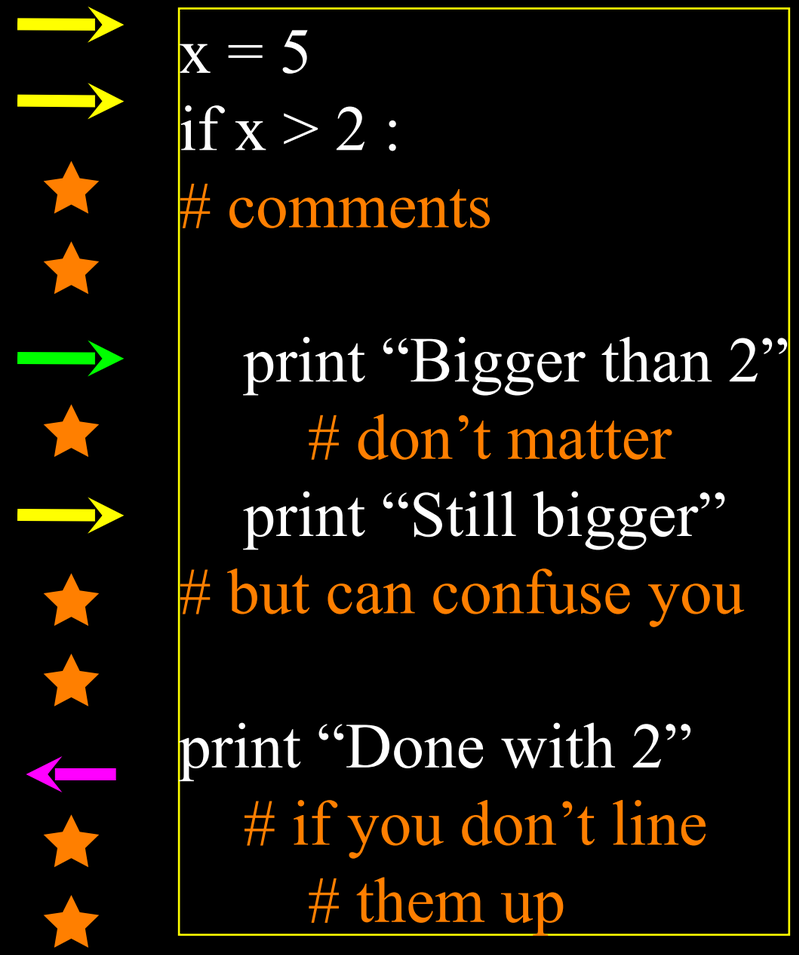
increase / maintain after if or for

decrease to indicate end of block

blank lines and comment lines ignored



```
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
```



```
x = 5
if x > 2 :
# comments


    print “Bigger than 2”
        # don’t matter
    print “Still bigger”
# but can confuse you


print “Done with 2”
    # if you don’t line
        # them up
```

Mental begin/end squares

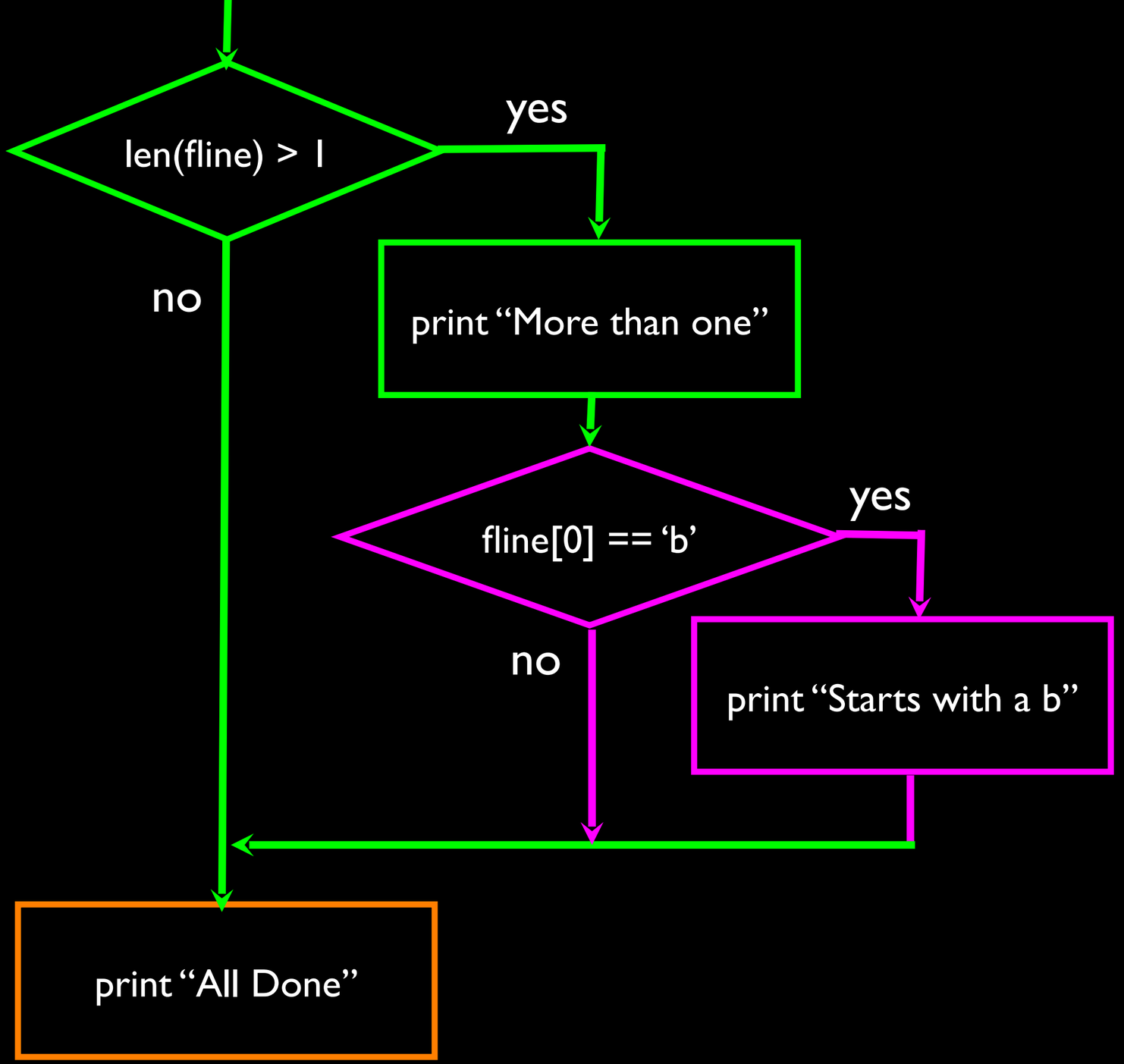
```
x = 5
if x > 2 :
    print “Bigger than 2”
    print “Still bigger”
print “Done with 2”
```

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

```
x = 5
if x > 2 :
# comments

    print “Bigger than 2”
        # don’t matter
    print “Still bigger”
# but can confuse you

print “Done with 2”
    # if you don’t line
        # them up
```

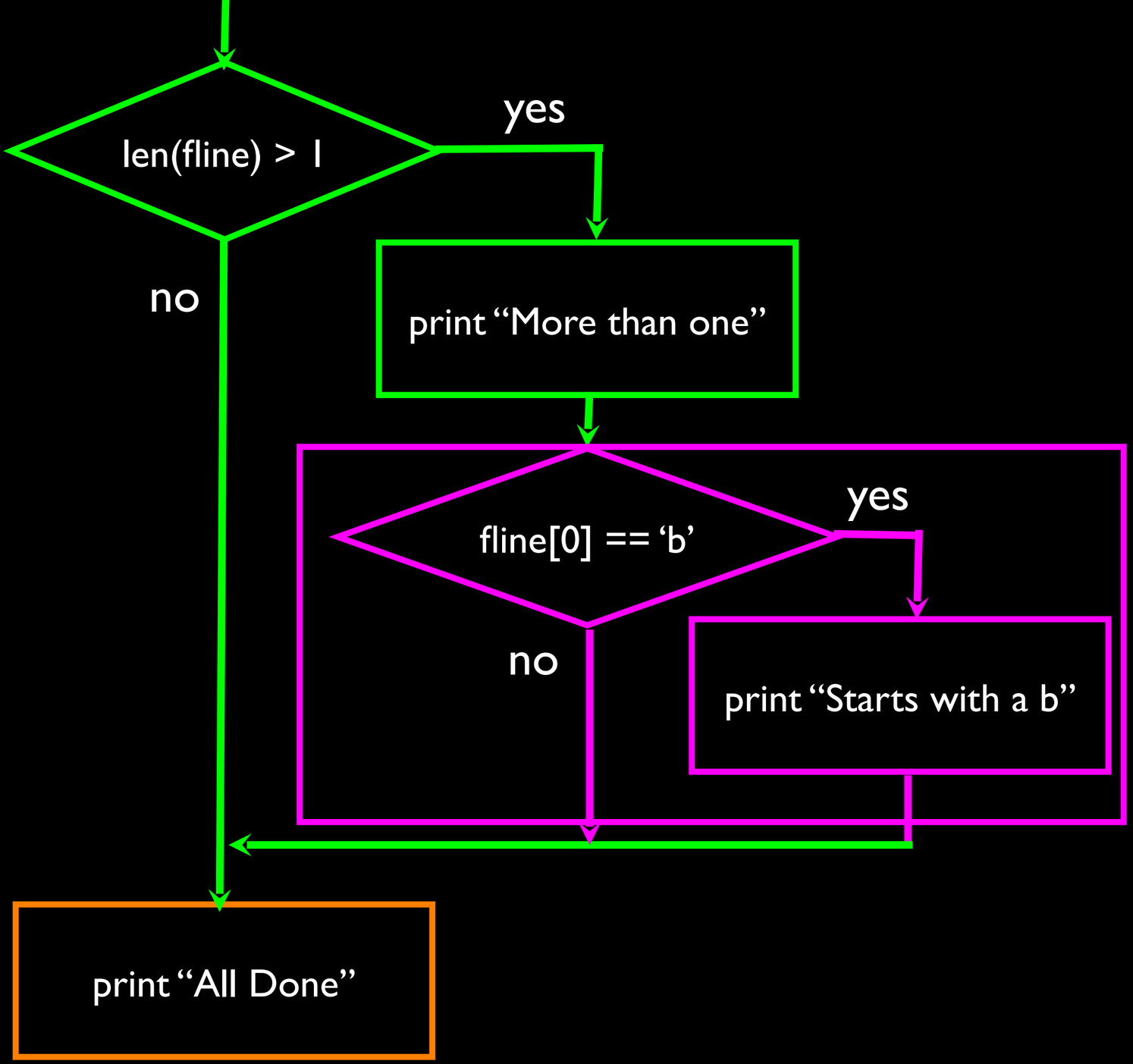
Nested Decisions

```
fline = "blah blah"

if len(fline) > 1 :
    print "More than one"
    if fline[0] == 'b' :
        print "Starts with a b"

print “All done”
```



Nested Decisions

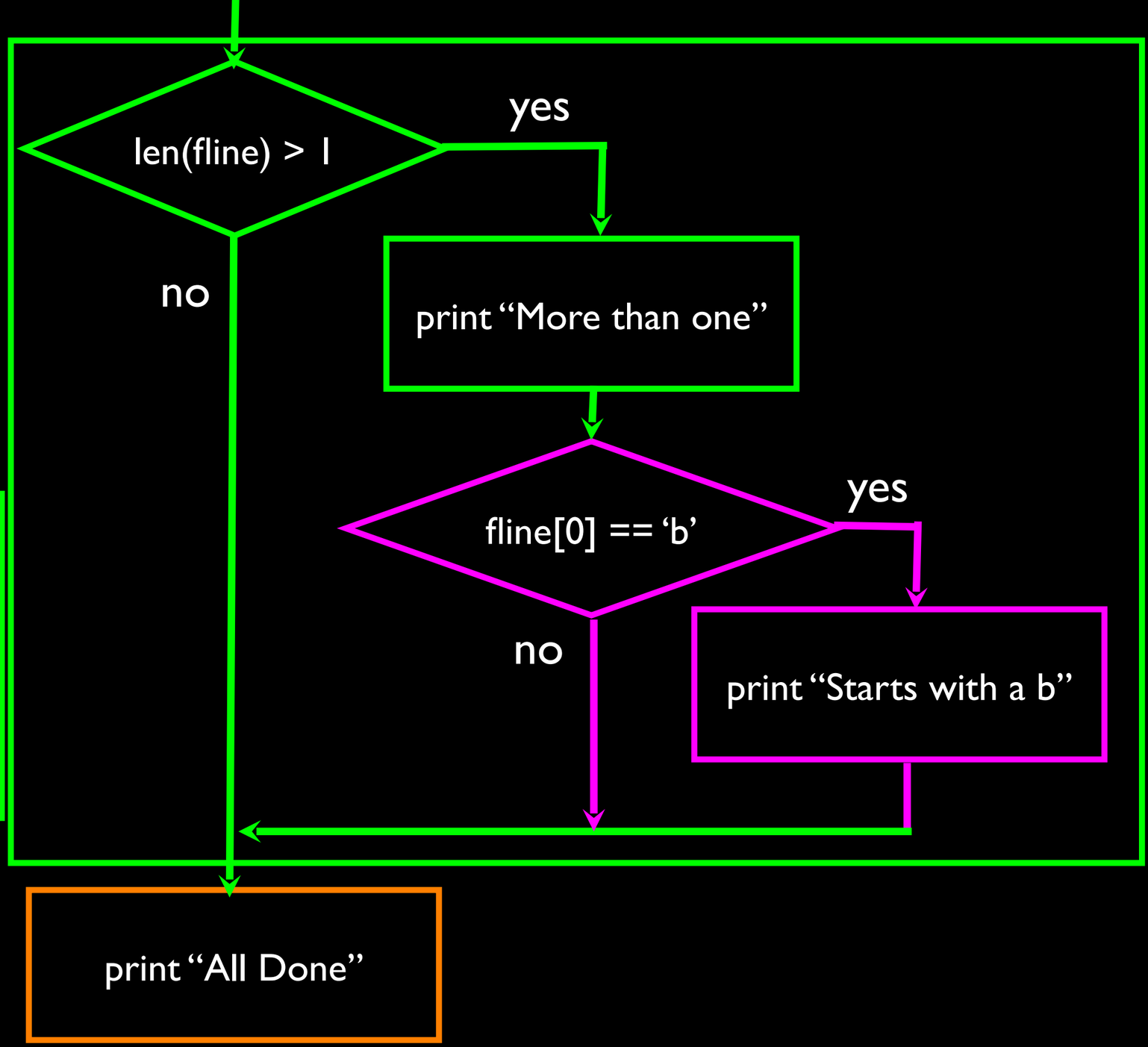
```
fline = "blah blah"

if len(fline) > 1 :
    print "More than one"
    if fline[0] == 'b' :
        print "Starts with a b"

print “All done”
```



Nested Decisions

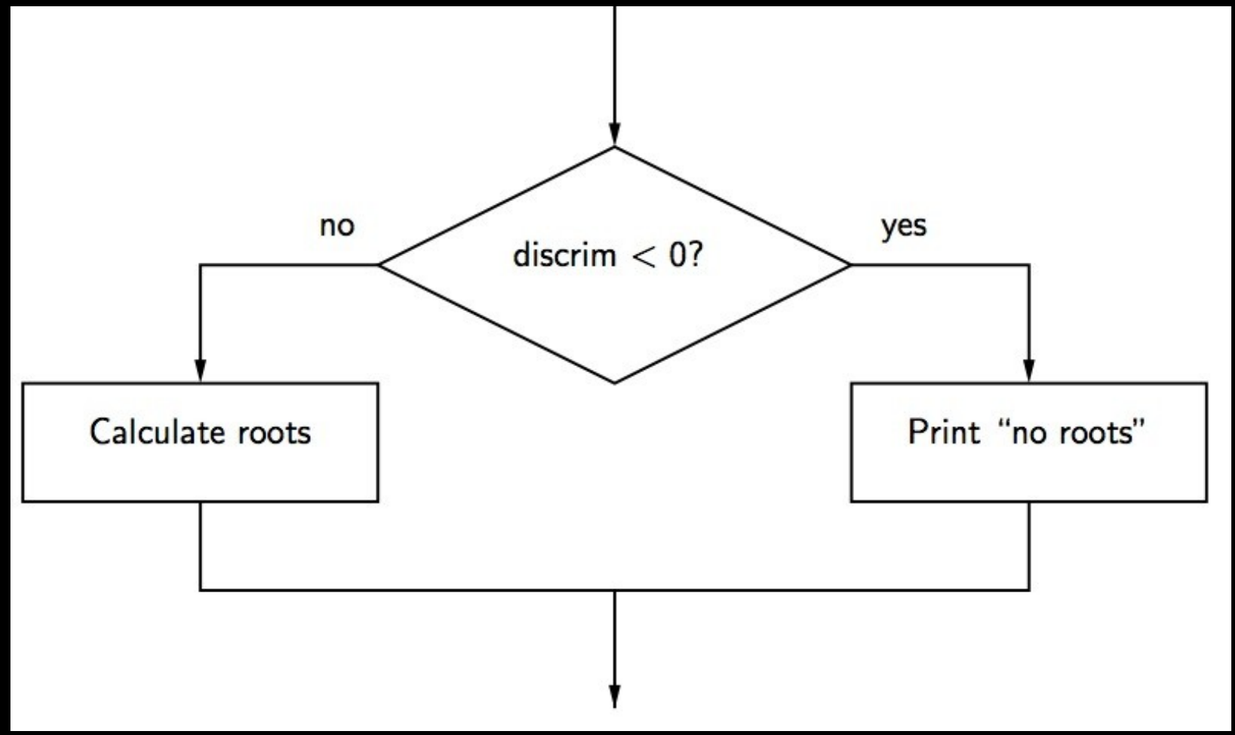
fline = "blah blah"

```
if len(fline) > 1 :
    print "More than one"
    if fline[0] == 'b' :
        print "Starts with a b"

print “All done”
```



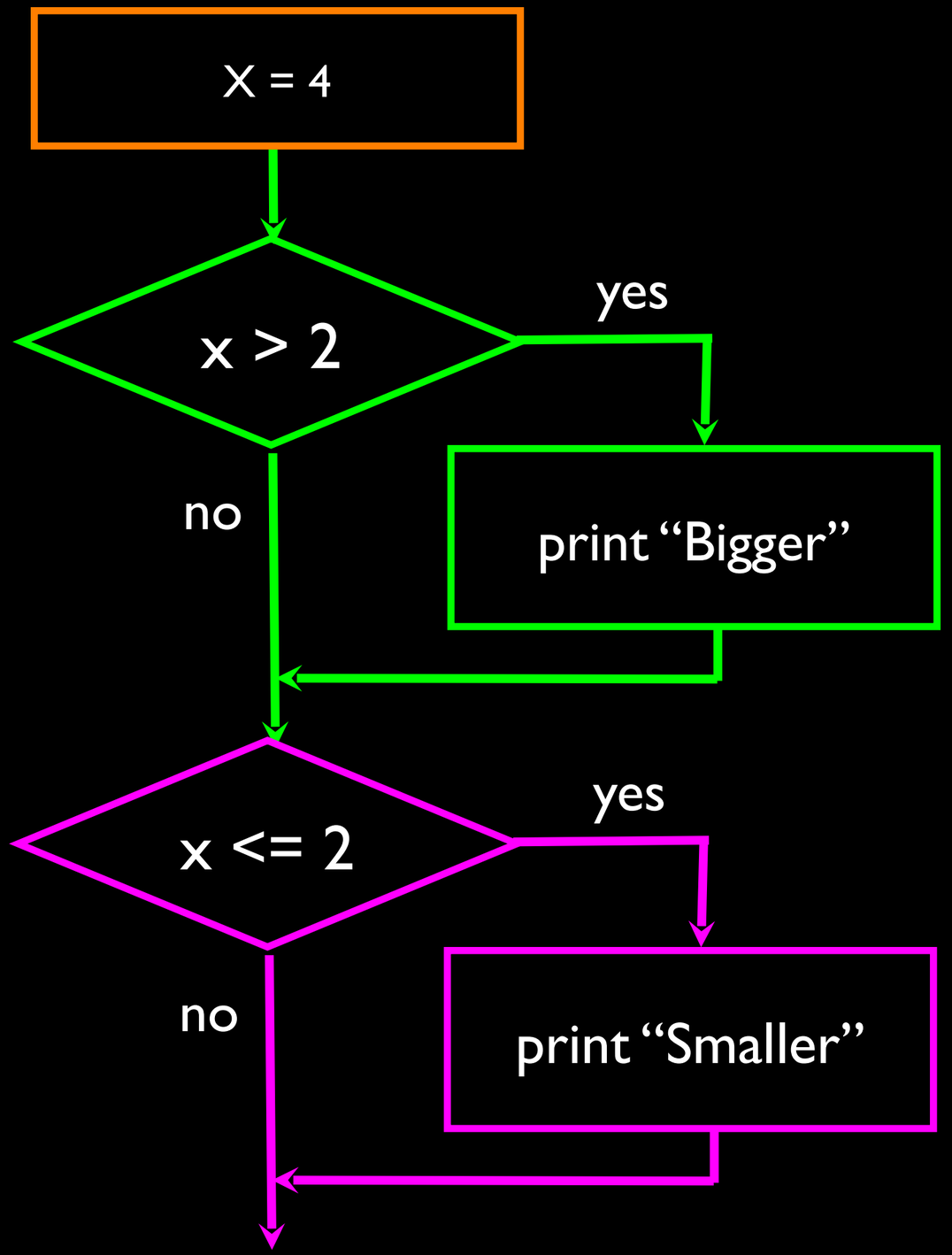
Two Way Decisions

- Sometimes we want to do one thing if a logical expression is true and something else if the expression is false
- It is like a fork in the road - we must choose one or the other path but not both



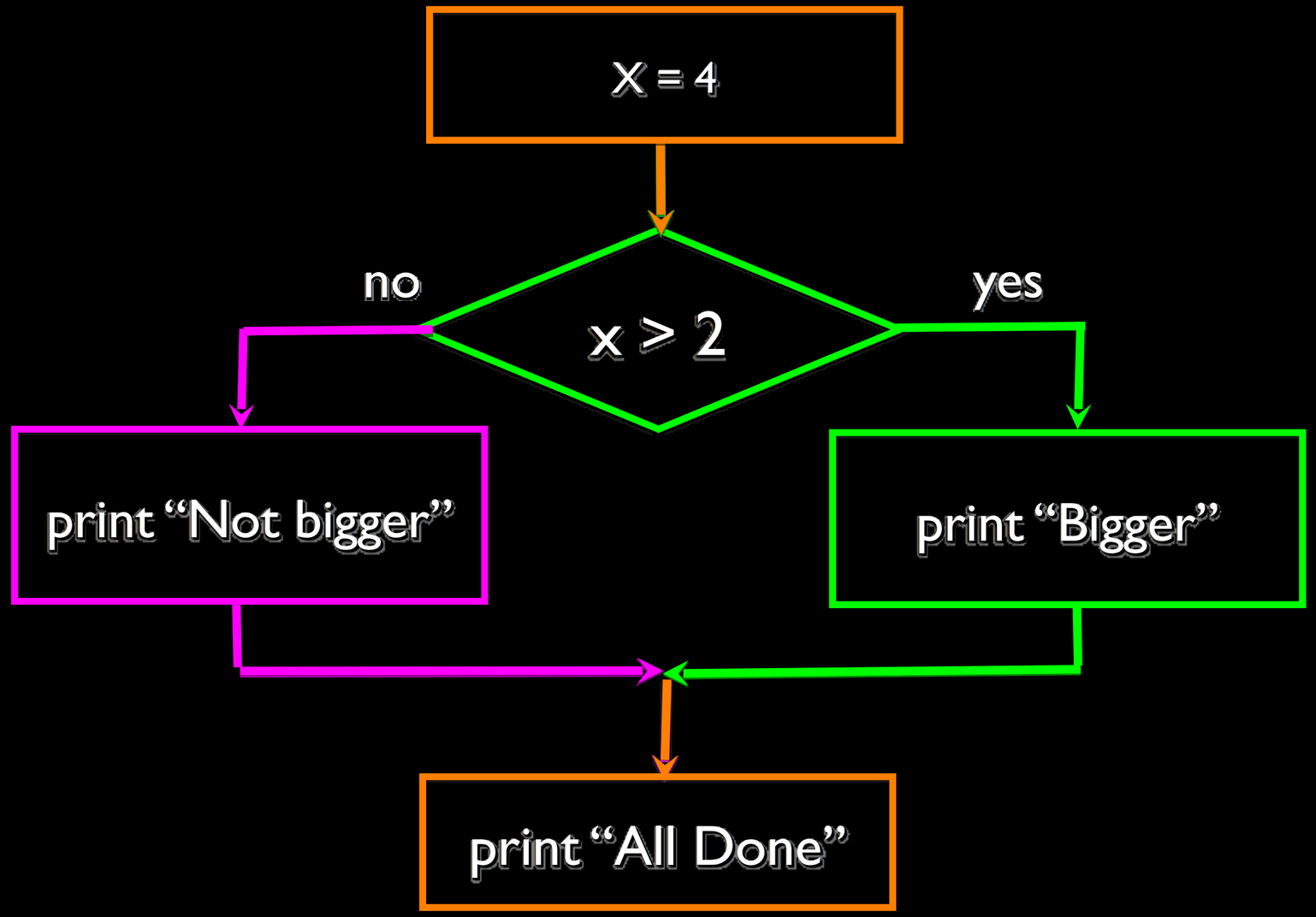
Two-way the hard way

```
x = 4
if x > 2 :
    print “Bigger”

if x <=2 :
    print “Smaller”
```



Two-way using else :

```
x = 4

if x > 2 :
    print "Bigger"
else :
    print "Not bigger"

print "All done"
```

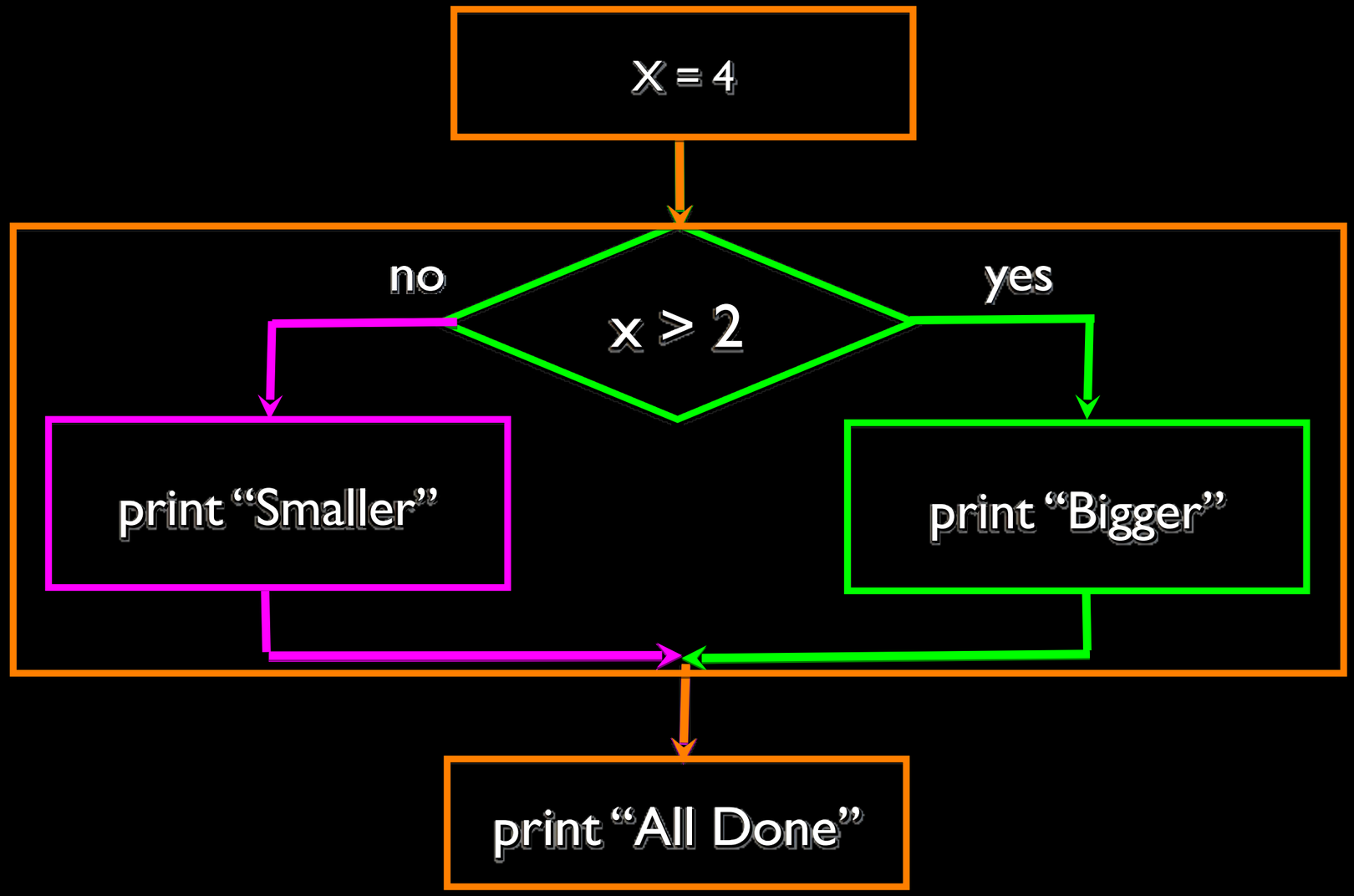


Two-way using else :

x = 4

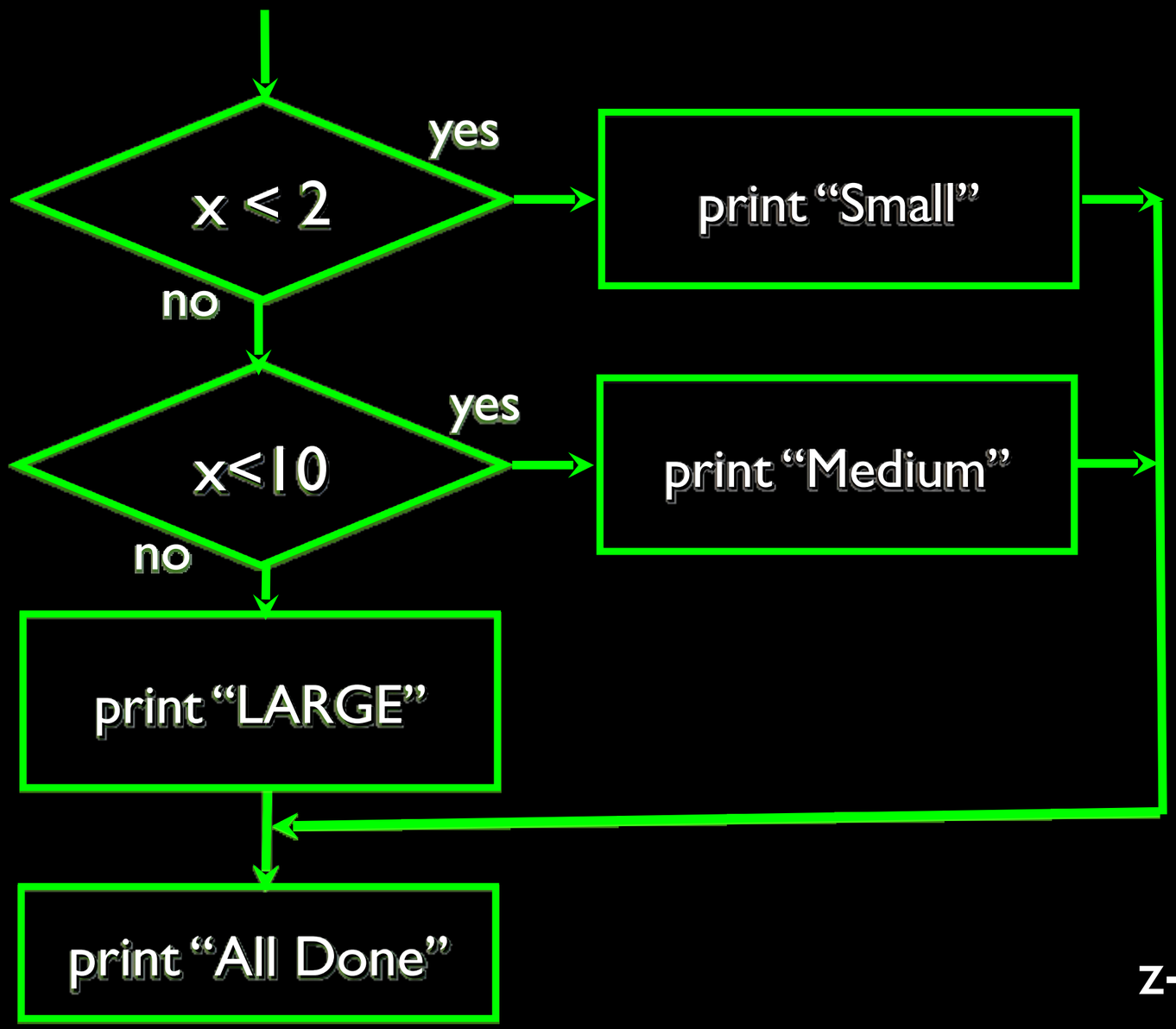
```
if x > 2 :
    print "Bigger"
else :
    print "Smaller"
```

print "All done"



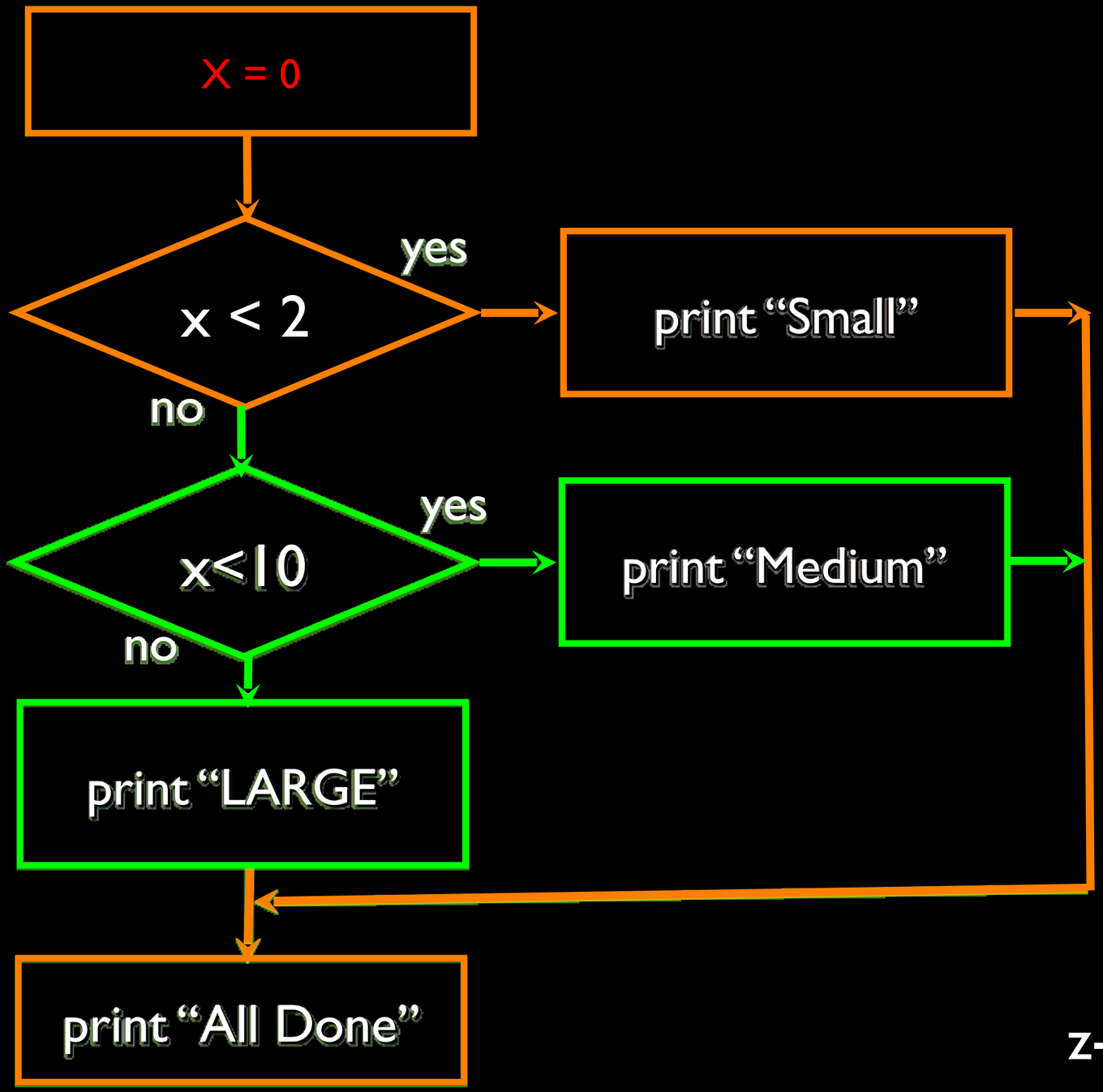
Multi-way

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



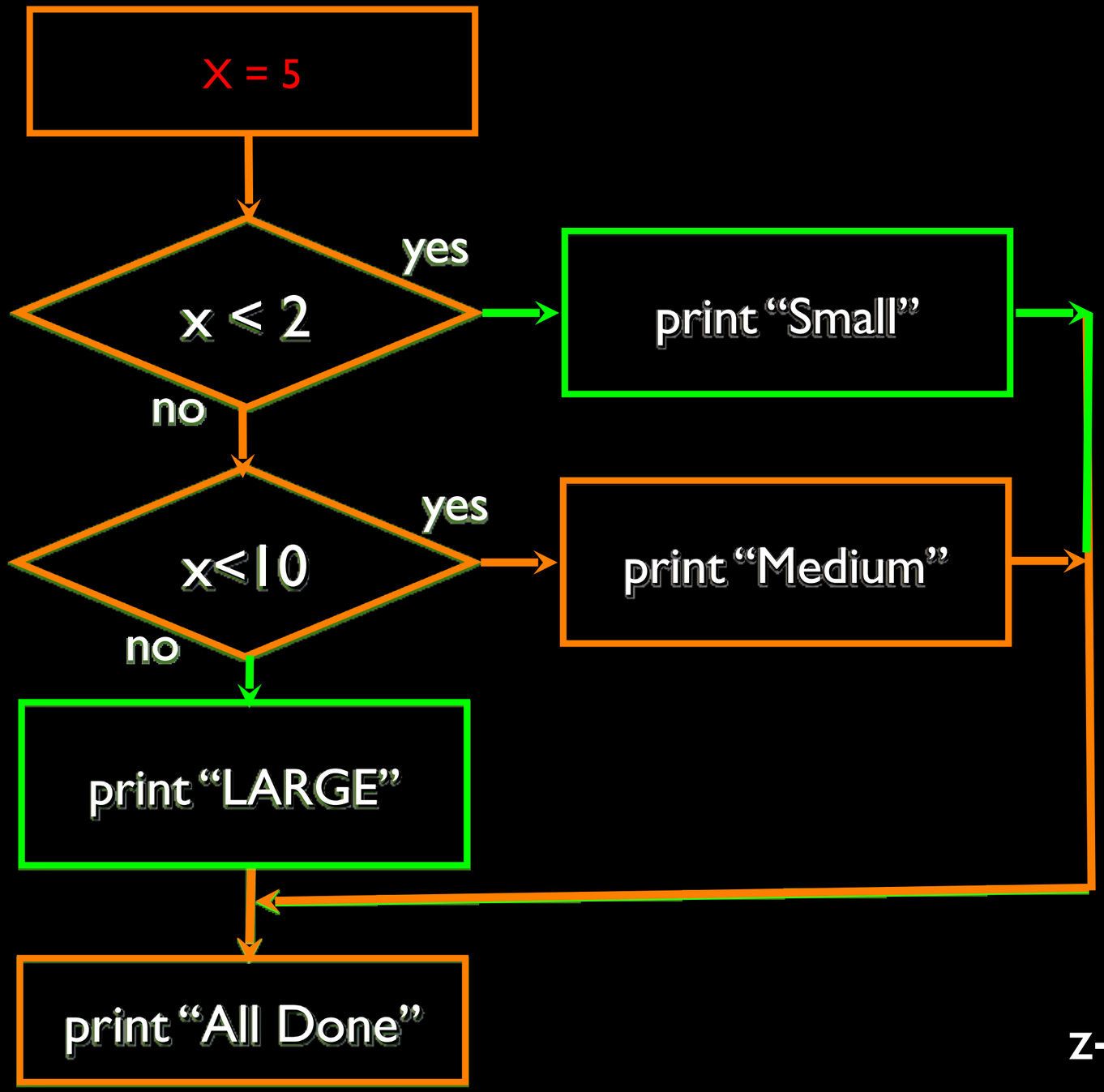
Multi-way

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



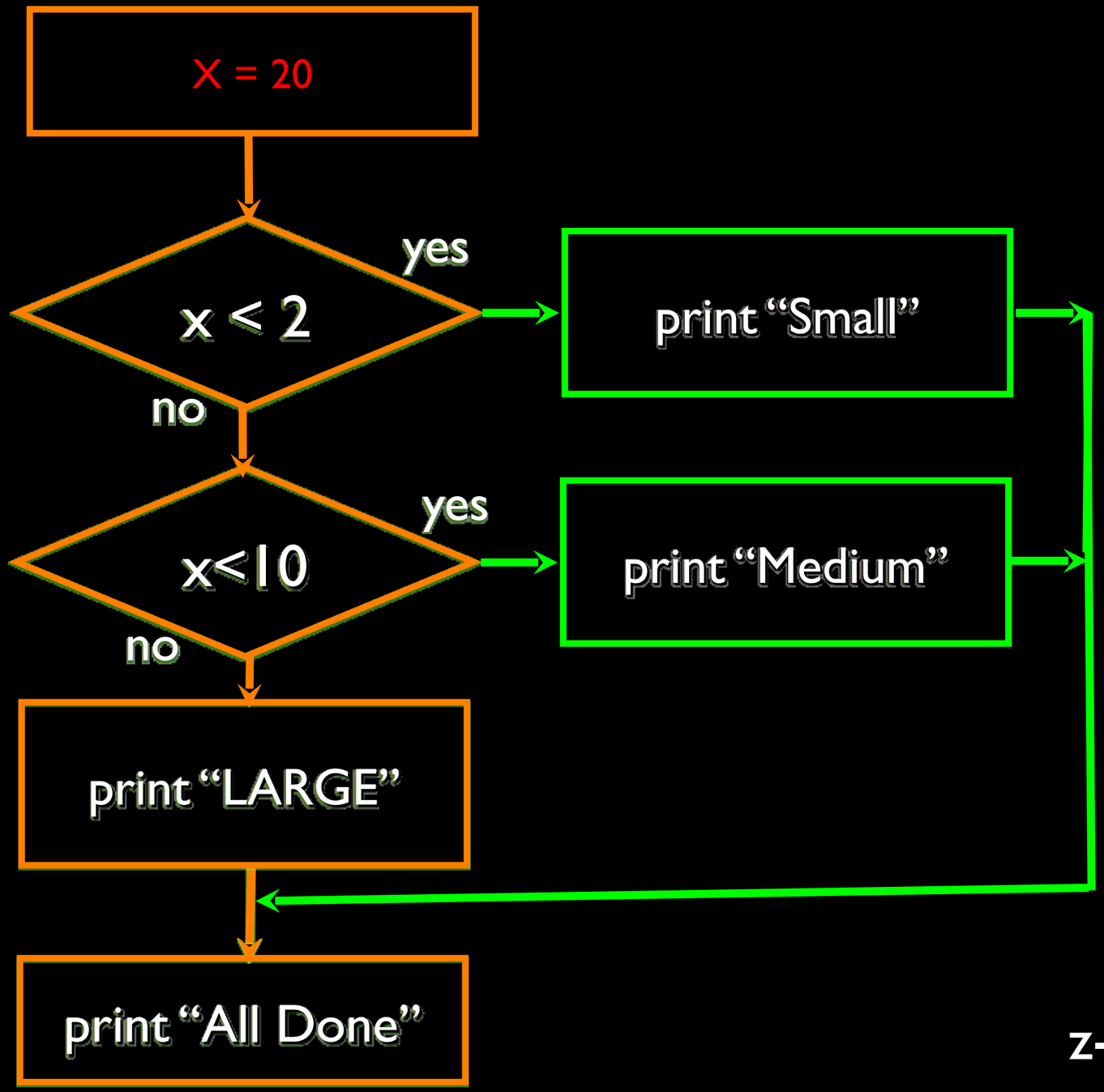
Multi-way

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



Multi-way

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



Multi-way

```
# 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"
```

Multi-way Puzzles

Which will never print?

```
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"
```

The try / except Structure

- 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)
```

The program stops here

```
$ python notry.py
Traceback (most recent call last):  File
"notry.py", line 6, in <module>
        istr = int(astr)
ValueError: invalid literal for int() with
base 10: 'Hello Bob'
```

All Done

```
$ 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
```

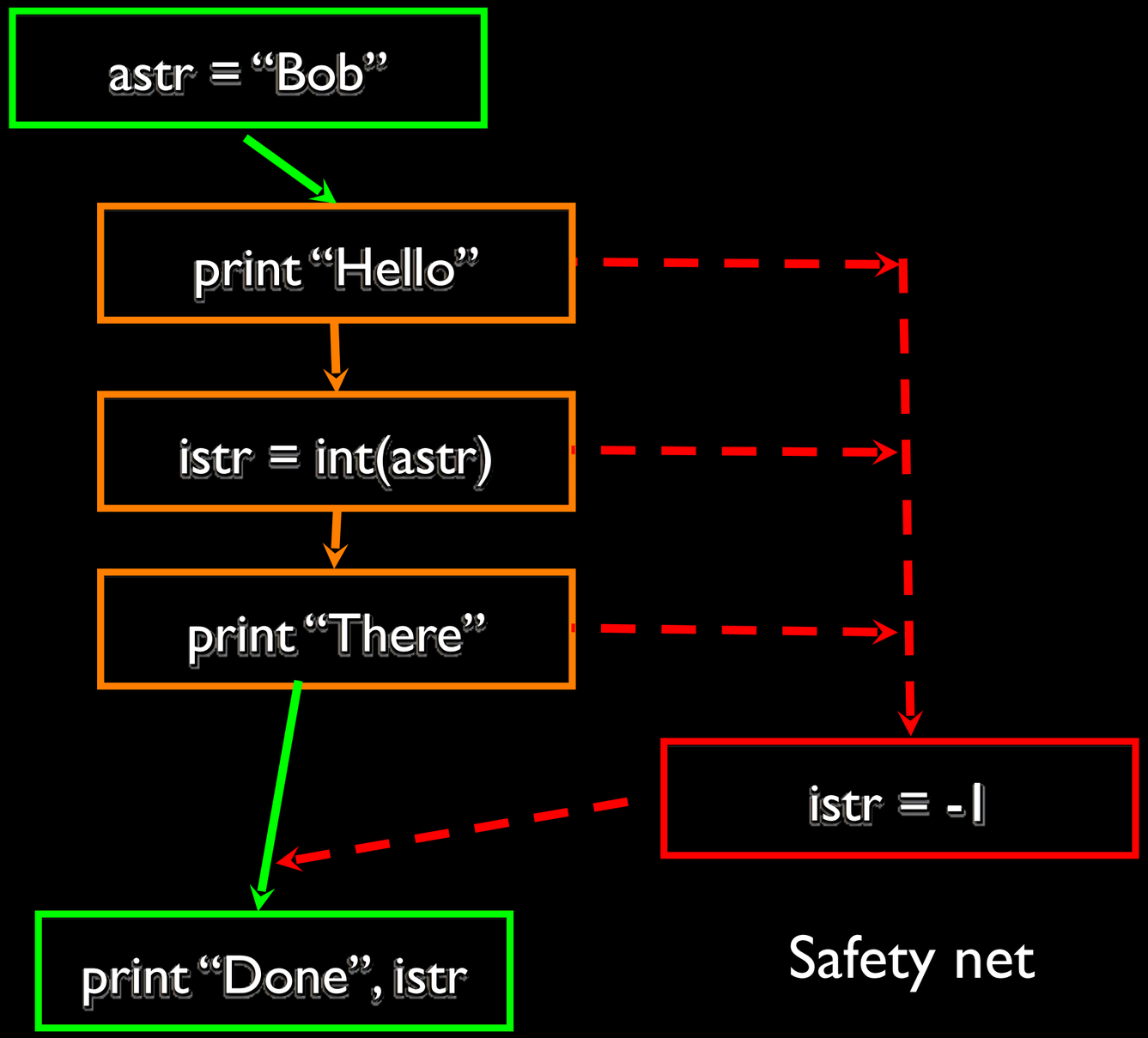
When the first conversion fails - it just drops into the except clause and the program continues.

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

When the second conversion succeeds - it just skips the except clause and the program continues.

try / except

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

print "Done", istr
```



Sample try/except

```
fname = raw_input("Enter a file name: ")
infile = open(fname, "r")
print “Blah...”
```

```
$ python frompart.py
Enter a file name: fred
Traceback (most recent call last):
  File "frompart.py", line 7, in <module>
    infile = open(fname, "r")
IOError: [Errno 2] No such file or directory: 'fred'
```

Sample try/except

```
fname = raw_input("Enter a file name: ")
try:
    infile = open(fname, "r")
except:
    print “File not found”,fname
    exit()
print “Blah...”
```

```
$ python frompart.py
Enter a file name: fred
File not found fred
$
```

Another try/except

```
fname = 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 file name:42
Nice work
$ python trynum.py
Enter a number:four
Not a number
$
```

Summary

- Indentation
- One Way Decisions
- Comparison operators == <= >= > < !=
- Nested Decisions
- Two way Decisions if : and else :
- Multiway decisions using elif
- Try / Except to compensate for errors