- Write a program that asks the user if they want to calculate the area of a square or a triangle. (The user will type in `square` or `triangle`.)
  - If they enter `square`, ask the user for the length of a side and print the area.
  - If they enter `triangle`, ask the user for the base and height and print the area.
- Write a program that lets the user type in two strings from the keyboard. The program will print which string comes first alphabetically. (Play around with this to figure out which sorts of strings come before other strings [i.e., letters, symbols, punctuation marks...])

Fact of the Day:

When programming languages were first developed, they usually included a statement called GOTO that permitted a program to jump from one line of code to another line of code at any time. This was used to implement things like if-else statements. In the 1960s-80s, using GOTO was strongly discouraged because it can lead to unmaintainable "spaghetti code."

Programming with if-else statements and other "higher-level" language constructs is known as ***structured programming.***

Multiple tests at once

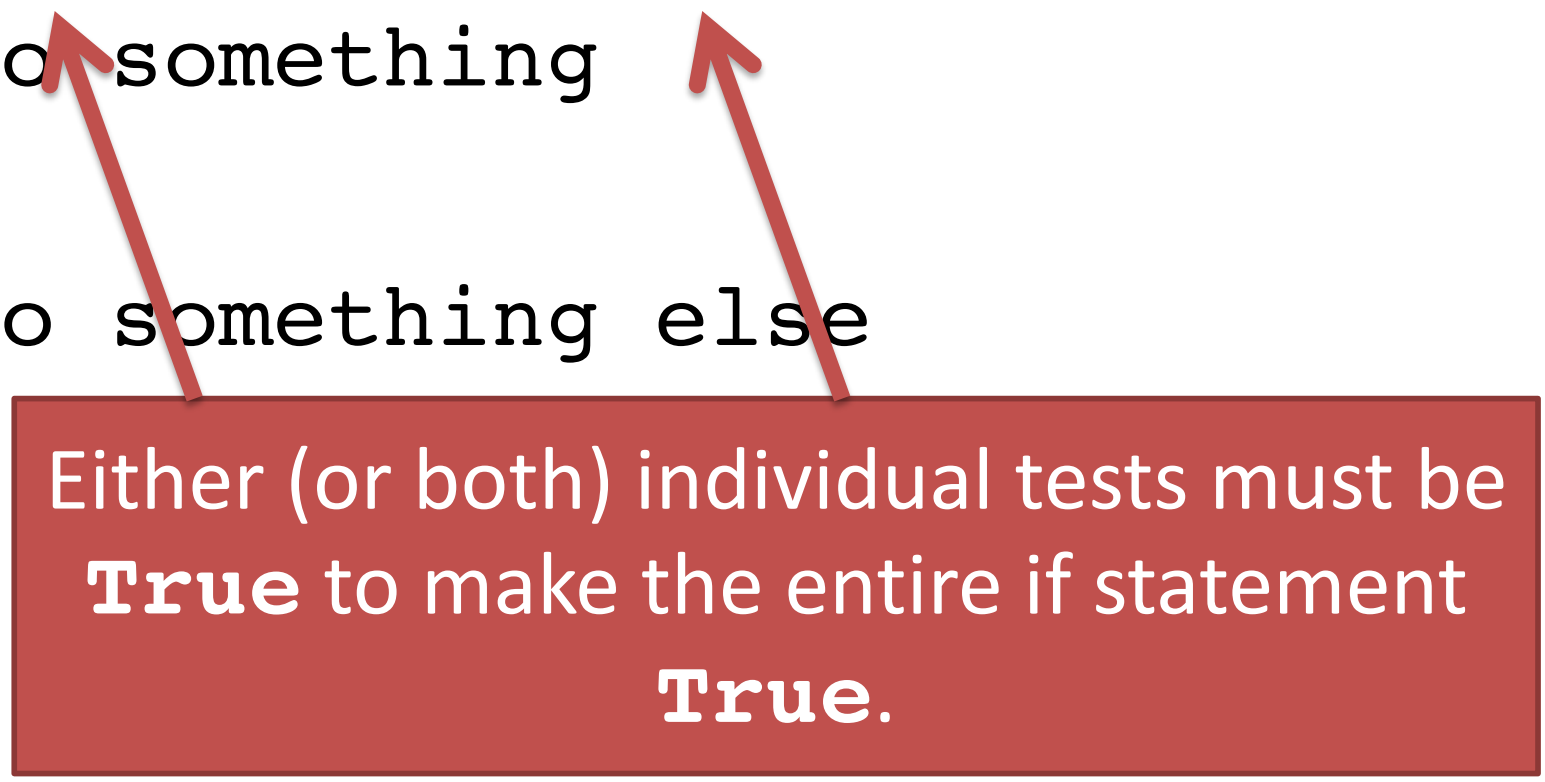
Multiple tests at once

```
if _________ and _________ :
    # do something
else:
    # do something else
```

Both individual tests must be **`True`** to make the entire if statement **`True`**.

Multiple tests at once

```
if _________ or _________ :
    # do something
else:
    # do something else
```



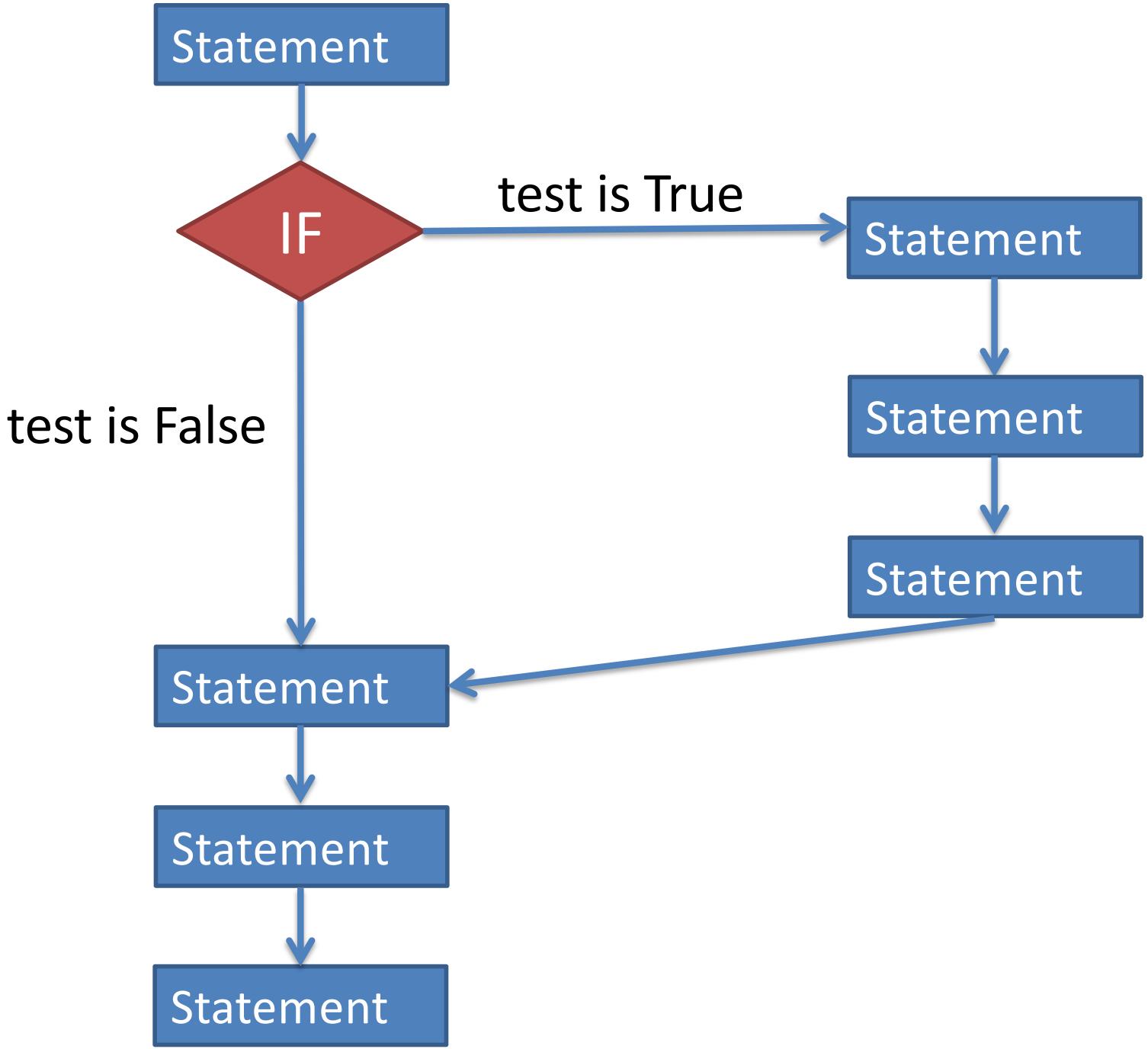
TN passes a new law that says you can't drink once you reach the age of 80.

```
age = int(input("What is your age? "))
if _____ :
  print("You may drink!")
else:
  print("You can't drink!")
```

You're writing an app that monitors the thermostat in your house and alerts you if the house temperature drops to 50 degrees or rises to 90 degrees.

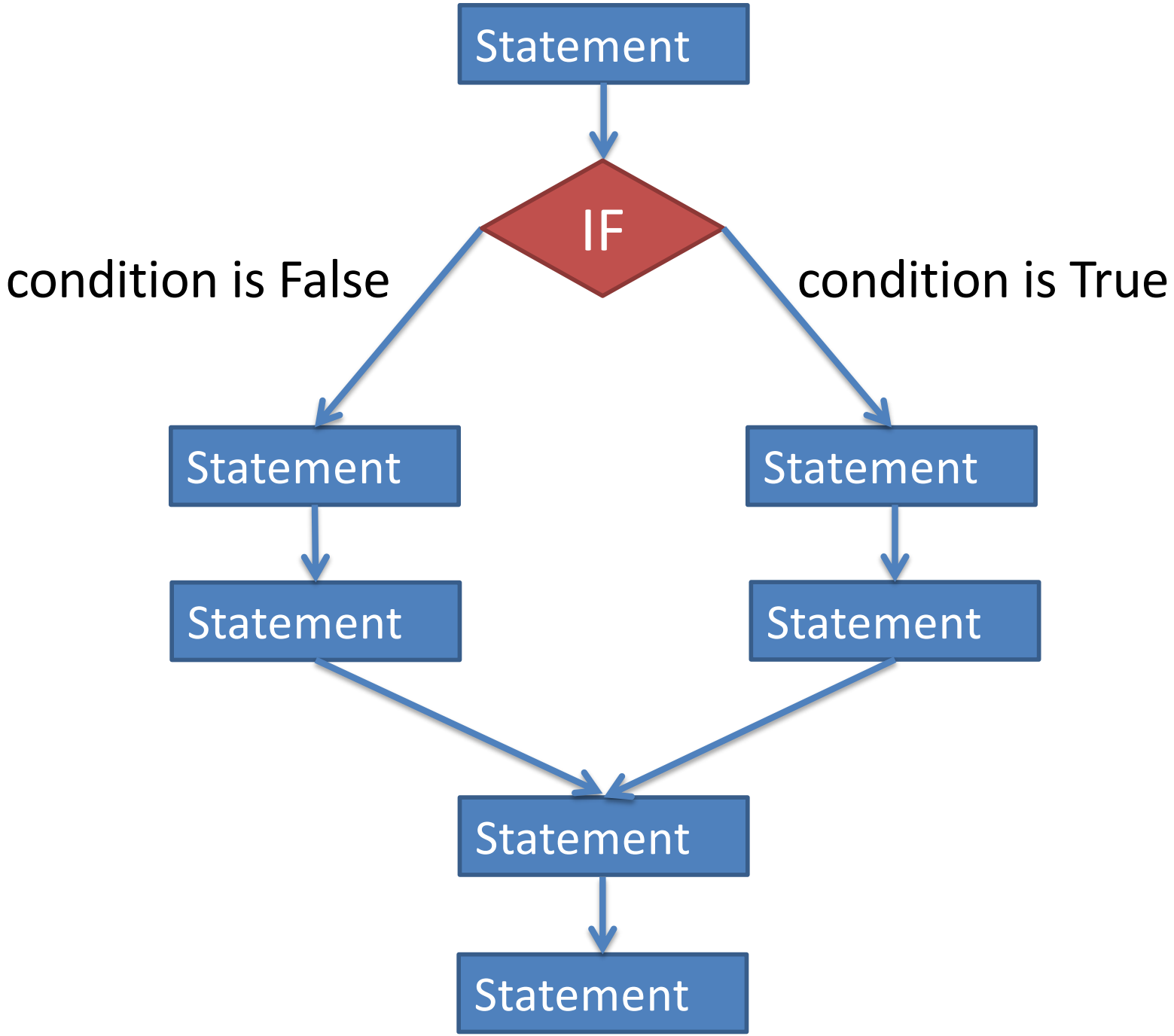
```
temp = read temperature somehow…
if _____ :
    # send a temperature alert here
    print("It's uncomfortable in here!")
```

Comparison of if vs if-else

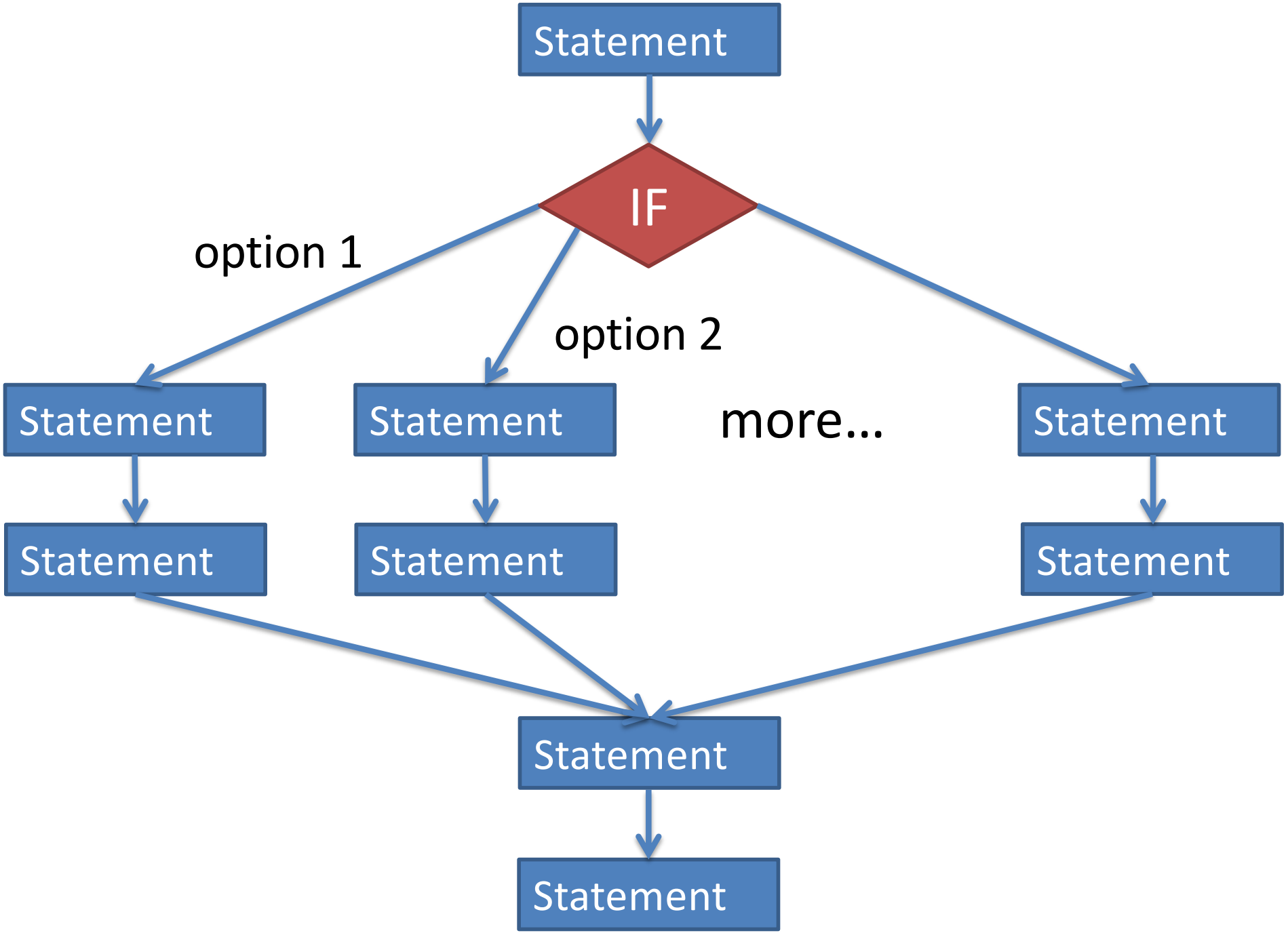

Statement
IF
test is True
Statement
Statement
Statement
test is False
Statement
Statement
Statement

```
if  condition :
    statement
    statement
    more statements…
statement
statement
more statements…
```

The *condition* must be something that is True or False.


Statement
IF
condition is False
condition is True
Statement
Statement
Statement
Statement
Statement
Statement

```
if  condition :
    statement
    more statements…
else:
    statement
    more statements…
more statements…
```


Statement
IF
option 1
option 2
Statement
Statement
more...
Statement
Statement
Statement
Statement
Statement
Statement

```
if   condition1   :
    statements…
elif    condition2     :
    statements…
elif    condition3     :
    statements…
(etc)
else:
    statements…
```

- Python runs each test in a group in order, top to bottom.
- Once a test is found that is `True`, the corresponding statements are run, ***and the rest of the tests and statements are ignored.***

```
x = 5
if x < 2:
   print("A")
elif x < 6:
   print("B")
elif x < 10:
   print("C")
```

- Any new "**`if`**" creates a new group, independent of any previous `if/elif/else`.

```
x = 5
if x < 2:
  print("A")
elif x < 6:
  print("B")
elif x < 10:
  print("C")
```

```
x = 5
if x < 2:
  print("A")
elif x < 6:
  print("B")
if x < 10:
  print("C")
```

Let's say a class has a grading scale of:

A = 90 and above

B = 80-89

C = 70-79

D = 60-69

F = below 60

Two more things...

OK to mix **`and`**s and **`or`**s, but you should parenthesize them:

```
if (x > 10 and  x < 100) or x < 0:
```

is different than

```
if  x > 10 and (x < 100  or x < 0):
```

Consider when `x = -1`.

If you ever want to negate a test, you can use not:

```
if x > 3:
  print("Yes!")

if not(x <= 3):
  print("Yes!")
```

and/or/not can be combined in any fashion, using parentheses to specify how they should be interpreted.

- See lab handout