`while` and `break`

According to urban legend, saying "bloody mary" 13x into a mirror will summon a scary ghost. Here's one way of doing that.

```
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
print("Bloody Mary!")
```

Pretty annoying, right?

We can get around this by using a *loop*, which helps us repeat things as many times as we want.

The syntax `while` repeats a block of code until the test next to it evaluates `False`.

```
while input("Keep going?") == "yes":
  print("Bloody Mary!")
```

We can also make Python count to 13 for us. The variable `i` increases by one until it reaches 13.

```
i = 0
while i < 13:
  print("Bloody Mary!", i)
  i = i + 1
```

What if we want a way to back out of summoning the scary mirror ghost before the loop ends?

Python's `break` statement allows the programmer to end a loop. **Note:** the loop will automatically end when the condition is False; a break statement is NOT necessary. A break statement lets you *prematurely* end a loop, before the condition becomes `False`.

Let's put a `break` statement inside an `if` to stop the summoning if we get scared.

```
i = 0
while i < 13:
  print("Bloody Mary!", i)
  i = i + 1
  if input("Scared yet??? ") == "yes":
    print("ok let's stop")
    break
```

`for` and `range`

Programmers want to execute a code block `n` times so often that Python gives us a way to do just that: a `for` loop coupled with the function `range(0,n)`. Like so,

```
for i in range(0,13):
  print("Bloody Mary!", i)
```

Conceptually, `for` iterates over each element `in` a collection --- it puts each element in the collection one at a time into the variable `i`. The function `range(0,n-1)` creates a collection with the numbers `0` through `n-1`. (This way, when code inside `for i in range(n):` runs `n` times.) If we wanted the numbers 1 through 13 instead of 0 through 12, we would write `range(1,14)`.

Comprehension Question: How many times will "Boo!" get printed?

```
for i in range(3):
  for j in range(3):
    print("Boo!")
```

Challenge: There's one ghost in the first room. There's two ghosts in the second room. There's three ghosts in the third room... and so on and so forth. There's thirteen rooms in this haunted 80's penthouse. Can you use math and a loop to calculate how many ghosts we're going to run into total?