While loop syntax, pseudocode and how to write any while loop

```
while condition:
  statement                  # These indented statements form the body of the loop.
  statement
  [ more statements ... ]
statement                    # This statement is the first one run after the loop ends.
statement
[ more statements ... ]
```

A while loop runs a block of code as long as a condition is true. The first time the while statement is encountered, the condition is tested, and if it is true, all the statements in the body of the loop are run. Then the condition is tested again. If it is still true, then the body of the loop is run again. The condition is tested again, and this process continues---test condition, then run the body---over and over, until the condition becomes false, when the loop ends. At this point Python picks up with the next statement after the body of the loop.

Pseudocode is an informal description of an algorithm, written for a human, not a computer. It resembles computer code in its basic structure, in that usually retains structures like functions, if-else statements, and loops, but it may leave out technical details specific to a certain programming language. The goal of writing an algorithm in pseudocode is to allow a human to (1) read and understand the code and (2) easily translate the algorithm into any programming language.

For instance, in pseudocode, you can say:

```
x = get integer from keyboard
```

or

```
if x is between 0 and 100, then
  print x with two decimal places
```

whereas those are not true statements in Python. Pseudocode is not concerned with syntax, but with the semantics of the algorithm.

How to write a while loop

1. Write pseudocode for what the loop does by explicitly repeating lines of pseudocode until you've repeated the same code at least twice.
2. Include an "if" statement in your code that will be True if you want the loop to keep going, and False if you want the loop to stop.
3. Make sure the pseudocode repeats the "if" statement at least twice.
4. Find the statements between consecutive "if" statements. These statements will become the body of the loop.
5. The "if" test will become the "while" test.
6. If there is any pseudocode before the first "if" test, if will go immediately before the start of the while loop (outside of the body).

Using this idea will sometimes result in the same line(s) of code being placed both inside the loop and before the loop. This is not a mistake; sometimes this happens and is considered OK style. This is sometimes called "a loop and a half" because you have the entire loop plus some fraction of the statements inside the body of the loop repeated before it starts.

While Loop Day 2 Practice

Use the general loop-writing procedure from the previous page to write the following loops.

1. Write a program that asks the user to type in two names, over and over again. The program will print out which one is first alphabetically. Then the user will be asked if they would like to continue, and if they type yes, the program asks for another pair of names. This continues until the user types that they don't want to continue.

    a. Change the program so the loop automatically stops when the user types the word STOP for the first name. In other words, the user will no longer need to be explicitly asked if they want to continue.

    b. Change the program so the loop automatically stops when the user types the word STOP for *either* of the two names.

    c. Change the program so the loop automatically stops when the user types the word STOP for the first name, and when this happens, the user isn't even prompted for the second name (loop ends immediately after the first name being STOP).

2. Write a program so the user may enter integers from the keyboard over and over, in a loop. Stop when looping when they type in a number between 1 and 10.

    a. Change the program so the number between 1 and 10 that ends the loop isn't printed. (If your first program didn't print the number, then make it be printed instead).

    b. Change the program so the loop ends when the number is even.

3. Write a program that asks the user to type in two integers, over and over again. Print the sum of the two integers. Make the loop stop when the second integer is less than zero (and don't print the sum in this case).

    a. Change the loop so the sum is printed when the second integer is less than zero.

    b. Make the loop stop when the sum of the two integers is odd. (try it both with printing the sum and not printing).

    c. Make the loop stop when the first integer is less than zero (and then don't ask for the second one).