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

- Figure out what the steps of the loop are. Write them in pseudocode as numbered steps in an actual circle.
- Be sure to include ***exactly one step*** of the loop that has a ***condition that keeps the loop going***, and the opposite condition ***ends the loop***.
- Starting from step #1, turn each line of pseudocode into real Python code.
- When you get to the step of the loop that involves the condition that keeps the loop going, that is where you place the `while __condition__:` part, filling in the blank with the condition.
- Continue turning the pseudocode into Python code, but now place subsequent steps *inside* the while loop (indented). Keep writing code inside the loop until you get back to step #1. You must duplicate any pseudocode steps of the loop that have lower numbered steps than the loop condition step. In other words, if your loop condition step is #3, then steps #1 and #2 will appear before the loop **and** inside it.

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 looping when they type in a number greater than or equal to 100.

    a. Change the program so the number greater than or equal to 100 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 entered is even (any even number).

3. Write a program that asks the user to type in two integers. Print the sum of the two integers. Then ask for another pair of integers and print their sum. Keep this going in a loop, and have 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).