Unit 2

Computing Practice and Programming

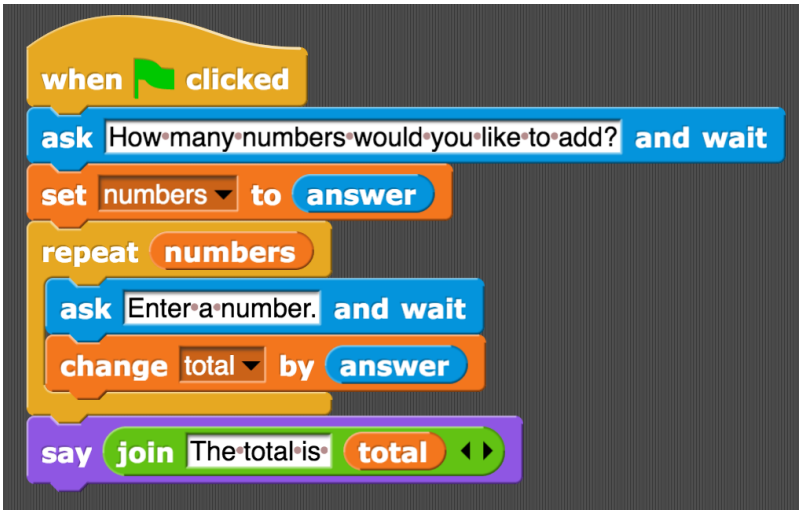
Part 4: Analyzing Programs

Debugging and Testing

After writing a program, it is important to run the program to ensure it produces the correct outcome in several different instances. This process is known as testing the program. If the program does not run as expected, you must find the problem with the code, known as debugging.

In SNAP, if a program is not working as expected you may need to edit your current blocks or add additional blocks to correct the issue.

The expected behavior of the program to the right is to ask the user how many numbers they would like to add, receive that input from the user, and then display the total.



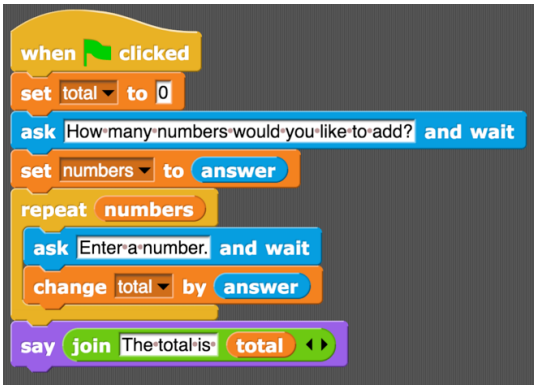
However, after running the program more than once, it displays an incorrect result.

A good starting point of debugging is to look at how the variables change as the program runs at the top left of the stage. Running the program adding two numbers, 15 and 10, provides the result to the right. This is correct, but to effectively test the program, you should always run it more than just once.

Next, we'll add two numbers again, 3 and 2. The result is not as expected because the sum of 3 and 2 is not 30. Looking at the totals variable, it appears the value was not cleared from the last time we ran the program.

Therefore, we now know that another block should be added to ensure the total is reset each time the program runs to receive the proper result.

To reset this total, we can always make sure is set to zero each time the program runs. After testing the program two more times with the same numbers, the correct result is shown both times. The program is functioning as expected and the debugging process is complete.

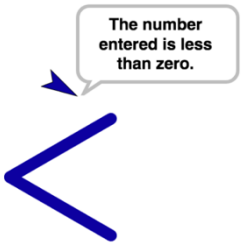


Correctness of Program

The correctness of a program can be evaluated by how well a program handles different scenarios. For example, if a program has three different outcomes, all three outcomes should be reached to ensure the program is functioning properly. In this simple program, the user is asked for a number to be compared to zero. As output, the program draws a less than, greater than, or equals sign and also prints the information to the user.

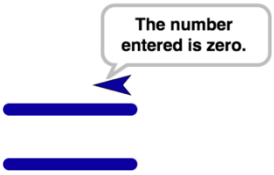
Test Cases

A test case outlines how a program should be run to test a specific outcome. This example contains a nested if-else condition and three possible outcomes. Therefore, three test cases can be created to verify each condition is working properly.



Test Case #1 – Less than Zero
For the first test case, we can enter any number less than zero and should see a less than sign along with a message that the number is less than zero. Entering -10 gives the result to the right. Therefore, the program works correctly when tested with a number less than zero.

Test Case #2 – Greater than Zero
Similarly, we can enter any greater than zero and should see a greater than sign along with a message that the number is greater than zero. Entering 25 gives the result to the right. Therefore, the program works correctly when tested with a number greater than zero.



Test Case #3 – Equal to Zero
For the last case, we must enter zero and should see an equals sign along with a message that the number is equal to zero. Entering 0 gives the result to the right. Therefore, the program works correctly when tested with zero.

Since our program was verified as working using the test cases of each possible outcome, we can assume that the program's logic is correct.